SGS-THOMSON MICROELECTRONICS

ST9+ FAMILY

PROGRAMMING MANUAL

PRELIMINARY DATA

INTRODUCTION

The ST9 8/16 bit microcontroller family introduces a new generation of single-chip architecture. It offers fast program execution, efficient use of memory, sophisticated interrupt handling, input/output (I/O) flexibility and bit-manipulation capabilities, with easy system expansion. Virtually all of the ST9 configuration can be tailored to the needs of the user under program control. This enables the ST9 to serve as an I/O intensive microcontroller, as an intelligent peripheral controller within a larger system, or as a memory intensive microprocessor.

Programming of the ST9 is made easy in both high level languages such as C, or directly in assembler language, by the versatility of the 14 addressing modes coupled with the comprehensive instruction set operating on bits, BCD, 8 bit bytes and 16 bit words. The availability of the Register File, giving the programmer multiple 8 and 16 bit accumulators and index pointers, the fast interrupt response time, on-chip DMA and on-chip and external memory access capabilities give the ST9 a high efficiency for real-time control applications.

The ST9 has a range of family devices made up from various memory combinations (RAM, ROM/EPROM, EEPROM), powerful peripherals such as Multifunction Timers, Analog to Digital Converters, Serial Communications Interfaces and a standard Core.

The remainder of this section describes in more detail the ST9 features of primary interest to assembly language programmers. Please refer to the ST9 Technical Manual for detailed architectural and configuration information.

Note: This Programming Manual follows the syntax of the ST9 Software Tools (TR9 and GAS9, high-level Macro Assembler running under MS-DOS or Windows™, SUN-SPARC under SunOS or Solaris™). Register and bit names follow the recommendations of the symbols.inc file available with tools.

Table of Contents

Table of Contents

Table of Contents

1 SOFTWARE DESCRIPTION

1.1 ADDRESSING MODES

The ST9 offers a wide variety of established and new addressing modes and combinations to facilitate full and rapid access to the address spaces while reducing program length. The available addressing modes are shown in Table 1-1:

Single operand arithmetic, logic and shift byte instructions have direct register and indirect register addressing modes. For a full list of the possible combinations for each instruction type, please refer to the ST9 Programming Manual.

Table 1. Addressing Modes

Operand is In	Addressing Mode	Destination Location	Notation
Instruction	Immediate	Byte Word	#N #NN
Register File	Direct	Byte Word	r R rr RR
	Indirect	Byte/Word	(r) (R)
	Indexed	Byte/Word	N(r)
	Indirect Post-Increment	Byte	(r)+
Memory	Direct	Byte/Word	NN
	Indirect	Byte/Word	(rr)
	Indirect Post-Increment	Byte/Word	(rr)+
	Indirect Pre-Decrement	Byte/Word	-(rr)
	Short Indexed	Byte/Word	N(rrx)
	Long Indexed	Byte/Word	NN(rrx)
	Register Indexed	Byte/Word	rr(rrx)
Any bit of any working register	Direct	Bit	r.b
Any bit in program or data memory	Indirect	Bit	(rr).b
Program Memory	Direct		NN
	Relative		N
	Indirect		(rr) (RR)

ADDRESSING MODES (Continued)

Two Operands Arithmetic and Logic Instructions	
Destination	**Source**
Register Direct	Register Direct
Register Direct	Register Indirect
Register Direct	Memory Indirect
Register Direct	Memory Indexed
Register Direct	Memory Indirect with Post-Increment
Register Direct	Memory Indirect with Pre-Decrement
Register Direct	Memory Direct
Register Indirect	Register Direct
Memory Indirect	Register Direct
Memory Indexed	Register Direct
Memory Indirect with Post-Increment	Register Direct
Memory Indirect with Pre-Decrement	Register Direct
Memory Direct	Register Direct
Register Direct	Immediate
Memory Direct	Immediate
Memory Indirect	Immediate

ADDRESSING MODES (Continued)

Two Operands Load Instructions	
Destination	**Source**
Register Direct	Register Direct
Register Direct	Register Indirect
Register Direct	Register Indexed
Register Direct	Memory Indirect
Register Direct	Memory Indexed
Register Direct	Memory Indirect with Post-Increment
Register Direct	Memory Indirect with Pre-Decrement
Register Direct	Memory Direct
Register Indirect	Register Direct
Register Indexed	Register Direct
Memory Indirect	Register Direct
Memory Indexed	Register Direct
Memory Indirect with Post-Increment	Register Direct
Memory Indirect with Pre-Decrement	Register Direct
Memory Direct	Register Direct
Register Direct	Immediate
Memory Direct	Immediate
Memory Indirect	Immediate
Memory Indexed [1)]	Immediate

Two Operands Load Instructions[2)]	
Destination	**Source**
Register Indirect with Post-Increment	Memory Indirect with Post-Increment
Memory Indirect with Post-Increment	Register Indirect with Post-Increment
Memory Indirect with Post-Increment	Memory Indirect with Post-Increment

Notes:
1. Word Instructions Only
2. Load Byte Only

ADDRESSING MODES (Continued)

1.1.1 Register Addressing Modes

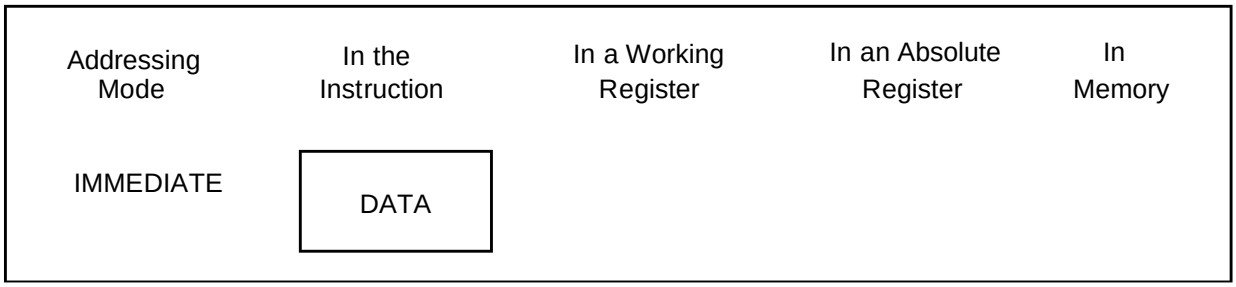
Immediate Addressing Mode

In the Immediate addressing mode, the data is found in the instruction. When using immediate data, a hash-mark (#) is used to distinguish it from an absolute address in memory.

Example: **`ldw RR42,#65535`**
loads the immediate value 65535 into the register pair R42 & R43. While the example shows decimal data, hexadecimal and binary values may also be used.

Example: **`ldw RR42, #0FFFFh`**

Figure 1. Immediate Register

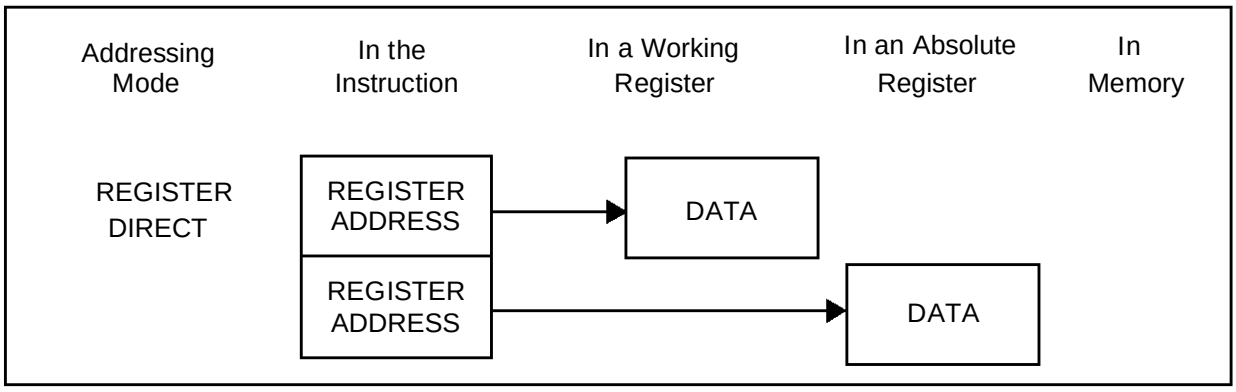


Direct Addressing Mode

In the direct addressing mode, a register can be addressed by using its absolute address in the Register File (in decimal, hexadecimal or binary form). Alternatively a register can be addressed directly as a working register;

Example: **`xch R0A2h,r4`**
exchanges the values in the register RA2h and working register number 4.

Figure 2. Direct Register



ADDRESSING MODES (Continued)

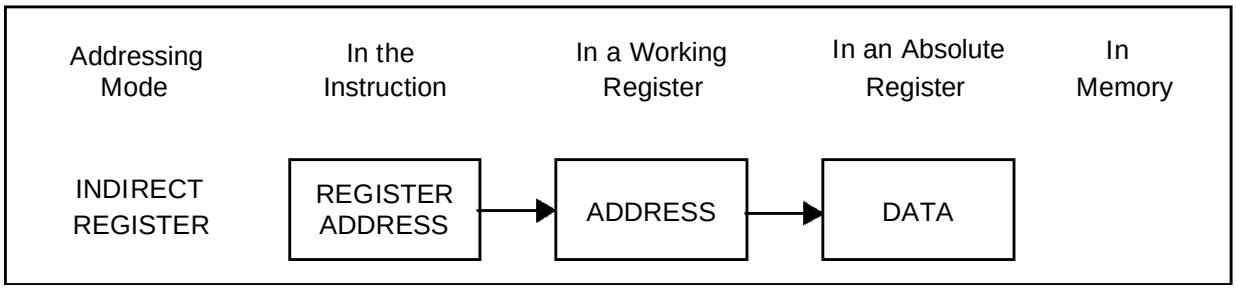
Indirect Addressing Mode

In the Indirect Register Addressing mode, the address of the data does not appear in the instruction but is located in a working register. The address of this register is given in the instruction. The indirect addressing mode is indicated by the use of parentheses.

Example: If register 200 contains 178 and working register 11 contains 86 then the instruction ld **`(r11),R200`** loads the value 178 into register 86.

Note: the indirect address can only be contained in a working register.

Figure 3. Indirect Register



Indexed Addressing Mode

To address a register using the Indexed mode, an offset value is used to add to an index value (which acts as a base or starting value). The offset value is the Immediate value given in the instruction while the index value is given by the contents of the working register.

Example: if working register 10 contains 55 then the instruction

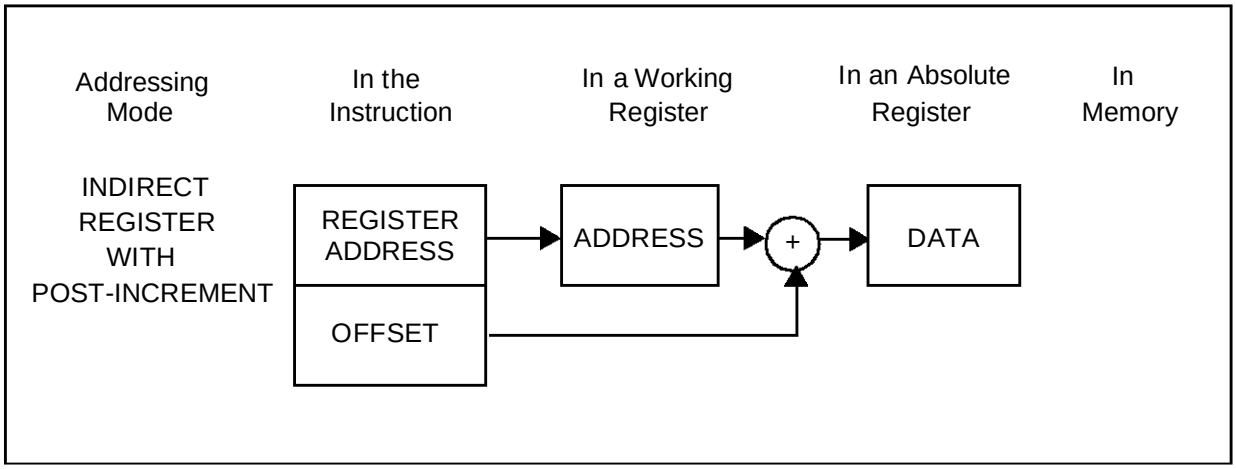
`ld 40(r10),r18`

loads register 95 (i.e.55+40) with the contents of working register 18.

The Register File never needs an absolute value requiring more than one byte and therefore only requires a short offset and a single register to contain the index.

Note: The index value can only be contained in a working register.

Figure 4. Indexed Register



ADDRESSING MODES (Continued)

Register Indirect Post-increment Addressing Mode

In this addressing mode, both destination and source addresses are given by the contents of working registers which are then post-incremented. The address of the memory location is contained in a working register pair, and the address of the register is contained into a single working register. Only working registers may be used to contain the addresses, this mode being indicated by both source and destination using parentheses followed by plus sign.

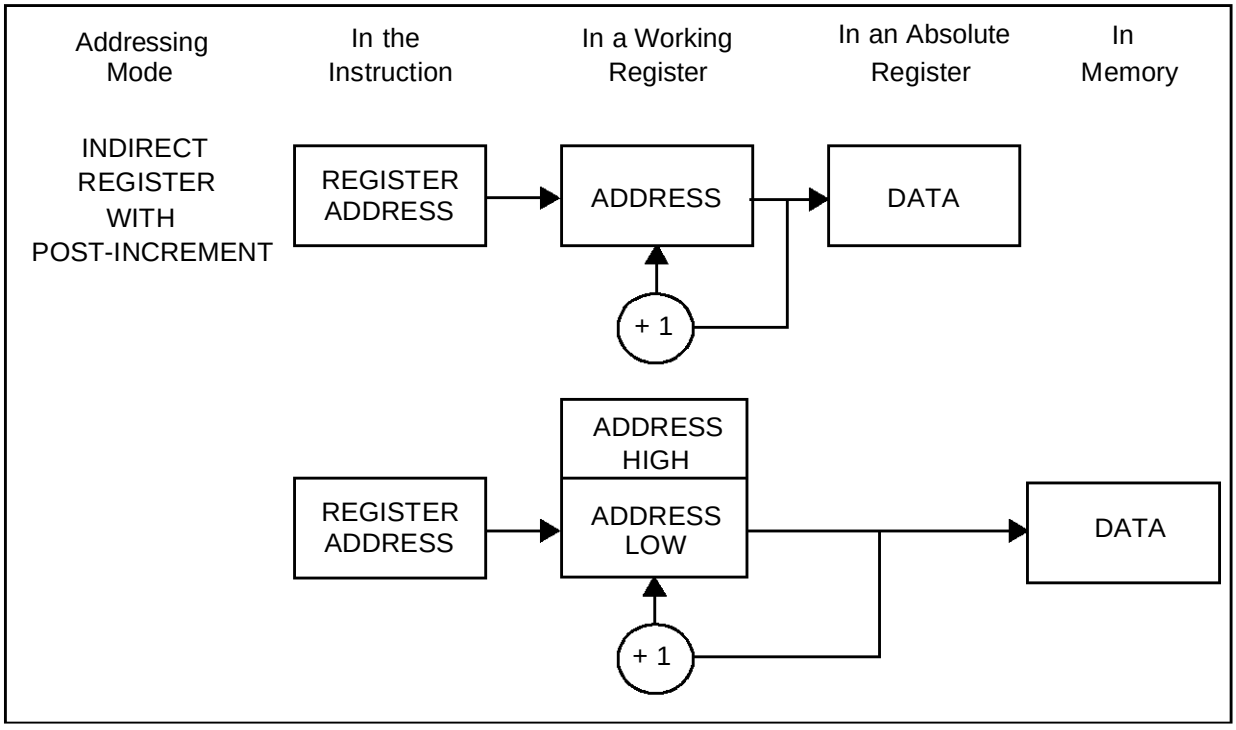
Example: if working register 8 contains the value 44, working register pair rr2 contains the value 2000, and register 44 contains the value 56, then by using the instruction

```
ld (rr2)+,(r8)+
```

the memory location 2000 will be loaded with the value 56. Immediately following this, the contents of r8 is incremented to 45 and the contents of rr2 is incremented to 2001.

This addressing mode is useful for moving blocks of data either from Register File to Memory or from Memory to Register File.

Figure 5. Register Indirect Post-increment



Direct Bit Addressing Mode

In the direct bit addressing mode, any bit in any working register can be addressed

Examples: bset r7.3

This instruction sets the bit 3 of the working register 7.

```
bld r7.3, r12.6
```

This instruction loads the bit 6 of the working register 12 in bit 3 of working register 7.

ADDRESSING MODES (Continued)

1.1.2 Memory Addressing Modes

The memory addressing modes described in this section are available to data and program memory. Thus before addressing the memory, it is necessary to indicate by use of the Set Program/Data Memory instructions, **spm** and **sdm**, in which memory the instructions are working. Since each memory space is 64K byte long, a word address is necessary to specify memory locations.

Direct Addressing Mode

The Memory Direct addressing mode requires the specific location within the memory. This only needs the absolute offset value which can be given in decimal, hex or binary form.

Thus the instruction:

```
ld 12345,r9
```

loads working register 9 data into memory location 12345

In the memory direct mode, it is possible to use an immediate addressing mode for the source operand.

Examples:

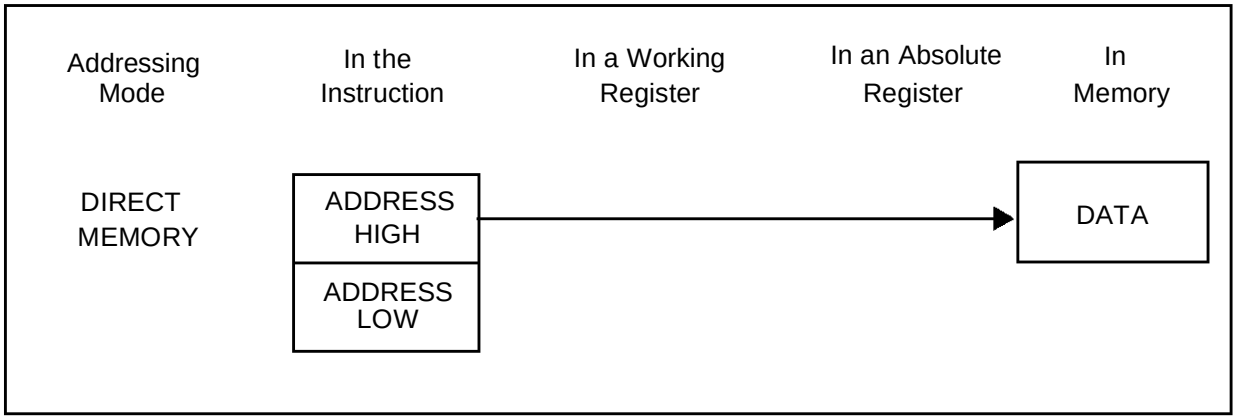
```
ld 12354,#34
```

will load the value 34 into the memory location 12354.

```
ldw 12354,#3457
```

will load the location pair 12354 and 12355 with the value 3457.

Figure 6. Memory Direct



ADDRESSING MODES (Continued)

Indirect Addressing Mode

When using the indirect addressing mode to access memory, the address is contained in a pair of working registers.

Example: if the working register pair r8 and r9 contains the value 2000 then the instruction

```
ld (rr8),#34
```

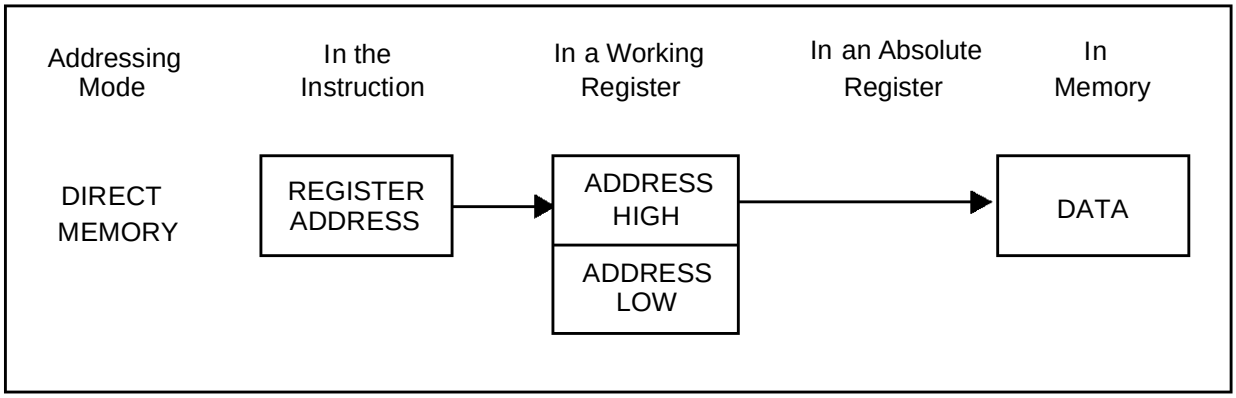
loads the value 34 into memory location 2000.

If the data to be stored is a word then the instruction ldw will automatically interpret the address as a pair of memory locations. So if rr8 contains 2000 then the instruction

```
ldw (rr8),#3 467
```

loads the memory locations 2000 and 2001 with the value 3467.

Figure 7. M emory Indirect



ADDRESSING MODES (Continued)

Indirect With Post-increment Adressing Mode

The indirect with post-increment addressing mode is similar to the memory indirect addressing mode but, in addition, after accessing the data in the currently pointed address, the value in the pointing working register pair is incremented. This mode is indicated by a plus sign following a working register pair in parentheses, e.g. (**rr4**)+.

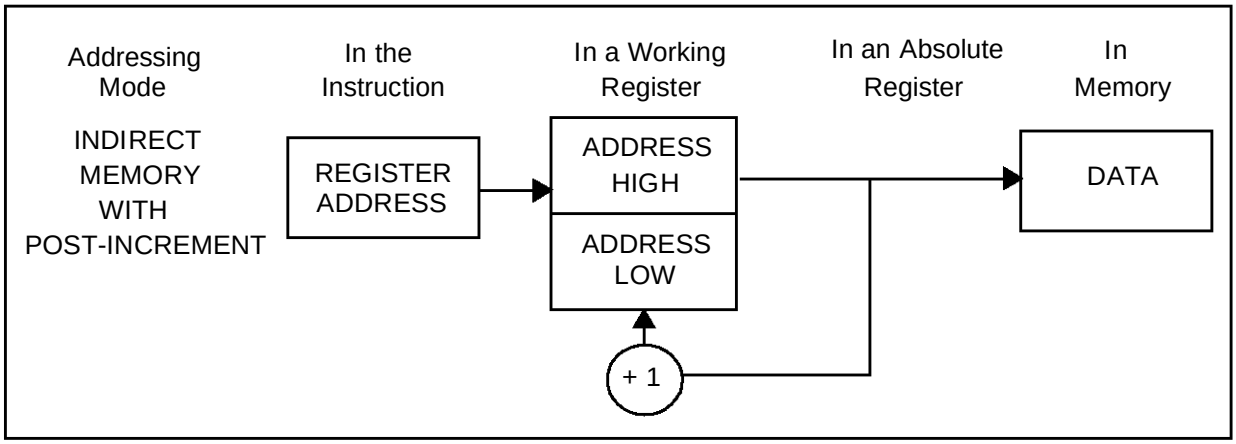
Example:If the working register pair rr4 (working registers r4 and r5) contains the value 3000 and memory location 3000 contains the value 88, then the instruction

```
ld R50,(rr4)+
```

loads register 50 with the value 88 and then the value in rr4 to be incremented to 3001.

This mode uses only working registers to contain the address. Thus the Indirect with Post-Increment addressing mode is most useful in repeated situations when a number of adjacent items of data are required in succession. The use of this addressing mode saves both time and program memory space since it cuts the usual increment instruction.

Figure 8. Memory Indirect Post-Increment



ADDRESSING MODES (Continued)

Indirect With Pre-decrement Addressing Mode

This indirect memory addressing mode has an automatic pre-decrement. The address can only be contained in working registers and the mode is indicated by a minus sign in front of the working registers which are in parentheses, e.g. -(rr6).

Example: if the working register pair rr6 contains the value 1111 and location 1110 contains the value 40 then the instruction

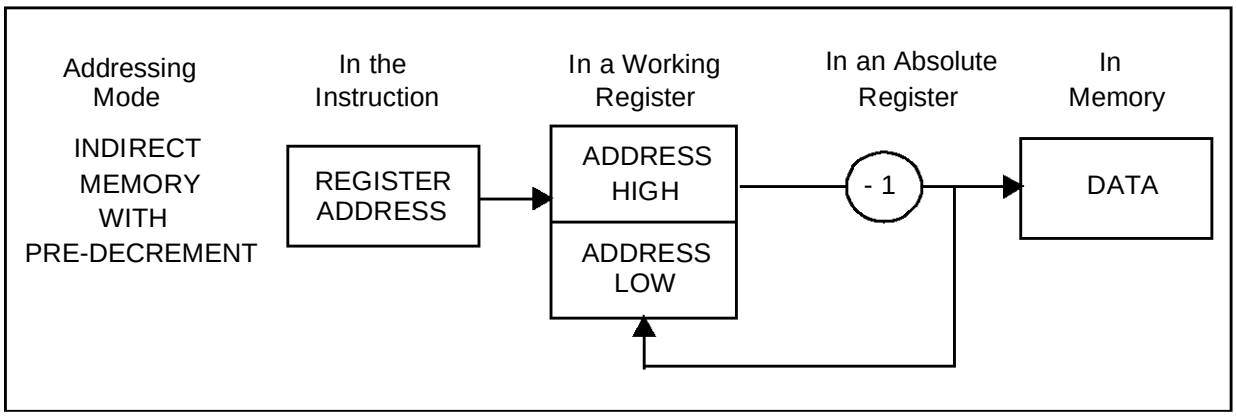
```
ld R56,-(rr6)
```

decrements the value in rr6 to 1110 and then loads the value 40 into register 56.

This addressing mode allows the ST9 to deal in the reverse order with data previously managed using the indirect post-increment mode without resetting the pointing registers (of the last post-increment).

The pre-decrement mode has the same benefits of time and program memory saving as the post-increment mode.

Figure 9. Memory Indirect Pre-Decrement



ADDRESSING MODES (Continued)

Indexed Addressing Modes

There are three indexed addressing modes, each using an indirect address plus offset format. The index address is given as an indirect address contained in a working register pair, while the offset can be long or short (a word or a byte). The address of the data required is given by the value of the working register pair indicated (the index), plus the value of the given offset.

Indexed with an Immediate Short and Long Offset

In these indexed modes the offset is a fixed and Immediate value included in the instruction. It may be either a short or long index as required, this immediate value being added to the address given by the working register pair. The short offset is signed-extensed to 16-bits before being added to the register pair.

Example: if the working register pair, rr6, contains the value 8000 and memory location 8034 contains the value 254 then the instruction

```
ld R55,34(rr6)
```

loads the value 254 into register 55.

Or, as another example, if the working register pair rr2 contains the value 2000 and register 78 contains the value 34 then the instruction.

```
ld 322(rr2),r78
```

loaded the value 34 into memory location 2322.

Figure 10. Memory Indexed with Immediate Short Offset

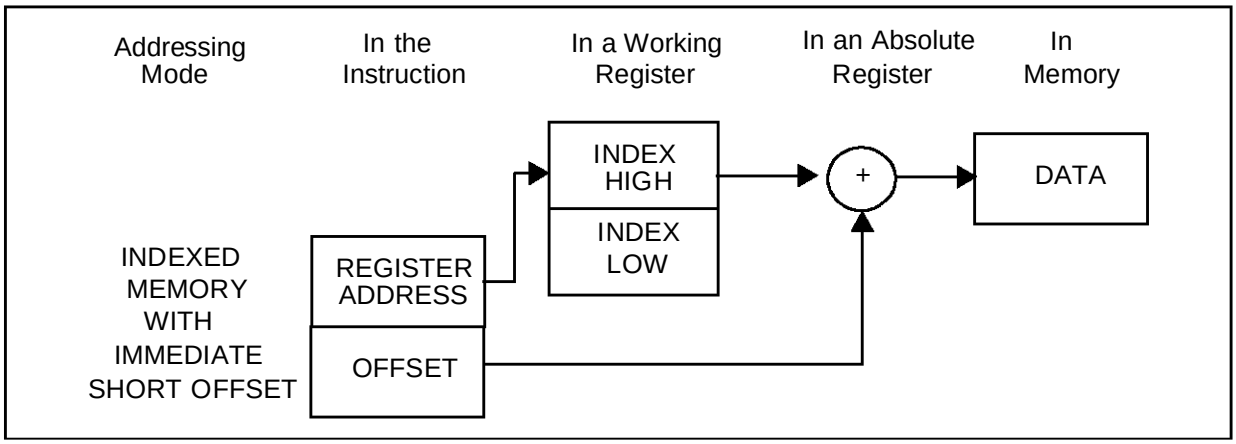
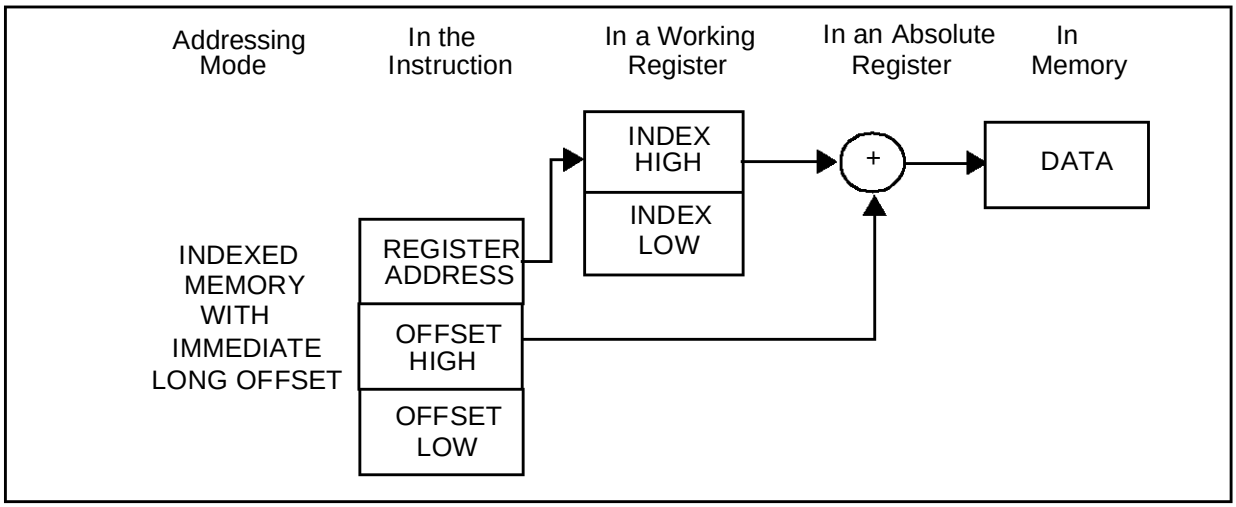


Figure 11. Memory Indexed with Immediate Long Offset



ADDRESSING MODES (Continued)

Indexed with a Register Offset

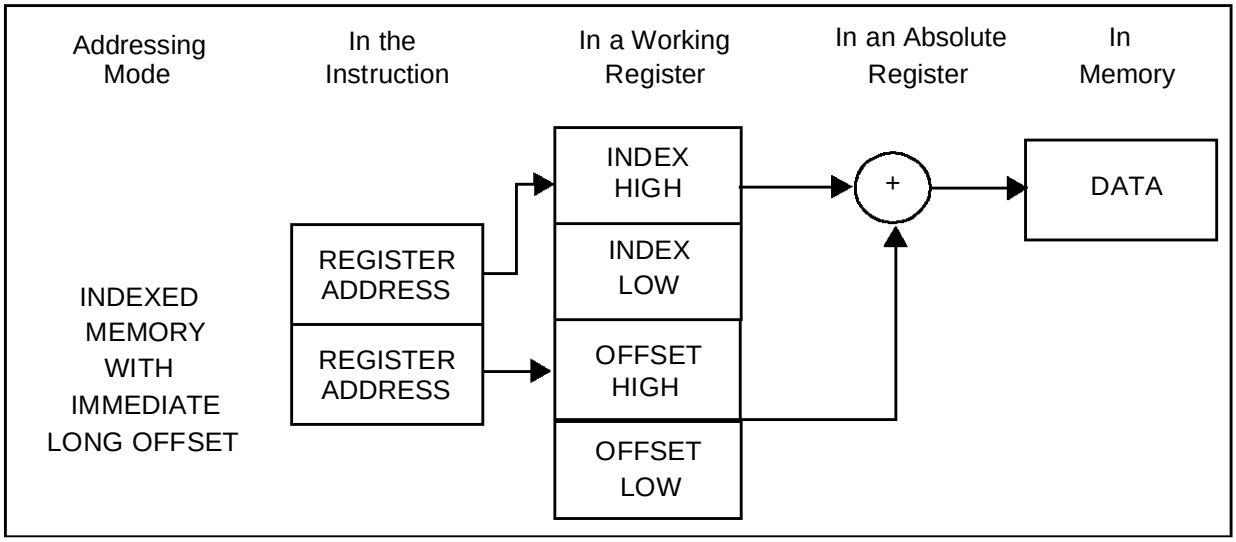
In this addressing mode, the index is supplied by one pair of working registers and the offset is supplied by a second pair of working registers. The format is rrx(rry), x and y being in the range 0,2,4...12,14.

Example: If working register pair rr0 contains the value 2222 and working register pair rr4 contains 3333 while register 45 contains the value 78 then the instruction

```
ld rr4(rr0),R45
```

loads the value 78 into memory location 5555.

Figure 12. Memory Indexed with Register Offset



Indirect Memory Bit Addressing Mode

In the indirect memory bit addressing mode, any bit of Program/Data memory location can be addressed with the **btset** (Bit Test and SET) instruction.

Example

```
btset (rr8).3
```

This instruction sets bit 3 of the memory location addressed by the working registers r8, r9 contents.

1.2 INSTRUCTION SET

The ST9+ instruction set consists of 94 instruction types which can be divided into eight groups:

– Load (two operands)
– Arithmetic & logic (two operands)
– Arithmetic Logic and Shift (one operand)
– Stack (one or two operands)
– Multiply & Divide (two or three operands)
– Boolean (one or two operands)
– Program Control (zero to two operands)
– Miscellaneous (zero to two operands)

The wide range of instructions eases use of the register file and memory, reducing operation times, while the register pointers mechanism allows an unmatched code efficiency and ultrafast context switching. A particularly notable feature is the comprehensive “Any Bit, Any Register” (ABAR) addressing capability of the Boolean instructions.

The ST9 can operate with a wide range of data lengths from single bits, 4-bit nibbles which can be in the form of Binary Coded Decimal (BCD) digits, 8-bit bytes, and 16-bit words.

The following summary shows the instructions belonging to each group and the number of operands required for each instruction. The source operand is “src”, “dst” is the destination operand, and “cc” is a condition code.

Table 2. Load Instructions (Two Operands)

Mnemonic	Operands	Instruction
LD LDW	dst,src dst,src	Load Load Word
LDPP LDPD LDDP LDDD	dst,src dst,src dst,src dst,src	Load (using CSR) -> (using CSR) Load (using DPRx) -> (using CSR) Load (using CSR) -> (using DPRx) Load (using DPRx) -> (using DPRx)

Table 3. .Arithmetic and Logic Instructions (Two Operands)

Mnemonic	Operands	Instruction
ADD ADDW	dst,src dst,src	Add Add Word
ADC ADCW	dst,src dst,src	Add With carry Add Word With Carry
SUB SUBW	dst,src dst,src	Substract Substract Word
SBC SBCW	dst,src dst,src	Substract With Carry Substract Word With Carry
AND ANDW	dst,src dst,src	Logical AND Logical Word AND
OR ORW	dst,src dst,src	Logical OR Logical Word OR
XOR XORW	dst,src dst,src	Logical Exclusive OR Logical Word Exclusive OR
CP CPW	dst,src dst,src	Compare Compare Word
TM TMW	dst,src dst,src	Test Under Mask Test Word Under Mask
TCM TCMW	dst,src dst,src	Test Complement Under Mask Test Word Complement Under Mask

INSTRUCTION SET (Continued)

Table 4. Arithmetic Logic and Shift Instructions (One Operand)

Mnemonic	Operands	Instruction
INC INCW	dst dst	Increment Increment Word
DEC DECW	dst dst	Decrement Decrement Word
SLA SLAW	dst dst	Shift Left Arithmetic Shift Word Left Arithmetic
SRA SRAW	dst dst	Shift Right Arithmetic Shift Word Right Arithmetic
RRC RRCW	dst dst	Rotate Right Through Carry Rotate Word Right Through Carry
RLC RLCW	dst dst	Rotate Left Through Carry Rotate Word Left Through Carry
ROR	dst	Rotate Right
ROL	dst	Rotate Left
CLR	dst	Clear Register
CPL	dst	Complement Register
SWAP	dst	Swap Nibbles
DA	dst	Decimal Adjust

Table 5. Stack Instructions (One or two Operands)

Mnemonic	Operands	Instruction
PUSH PUSHW PEA	src src src	Push on System Stack Push Word on System Stack Push Effective Address on System Stack
POP POPW	dst dst	Pop From System Stack Pop Word from System Stack
PUSHU PUSHUW PEAU	src src src	Push on User Stack Push Word on User Stack Push Effective Address on User Stack
POPU POPUW	dst dst	Pop From User Stack Pop Word From User Stack
LINK	Frame Pointer, Size (use system stack)	Move Stack Pointer upward; support for high-level language
UNLINK	Frame Pointer (use system stack)	Move Stack Pointer backward; support for high-level language
LINKU	Frame Pointer, Size (use user stack)	Move Stack Pointer upward; support for high-level language
UNLINKU	Frame Pointer (use user stack)	Move Stack Pointer backward; support for high-level language

Table 6. Multiply and Divide Instructions (Two or three Operands)

Mnemonic	Operands	Instruction
MUL	dst,src	Multiply 8x8
DIV DIVWS	dst,src dsth,dstl,src	Divide 16/8 Divide Word Stepped 32/16

INSTRUCTION SET (Continued)

Table 7. Boolean Instructions (One or Two Operands)

Mnemonic	Operands	Instruction
BSET	dst	Bit Set
BRES	dst	Bit Reset
BCPL	dst	Bit Complement
BTSET	dst	Bit Test and Set
BLD	dst,src	Bit Load
BAND	dst,src	Bit AND
BOR	dst,src	Bit OR
BXOR	dst,src	Bit XOR

Table 8. Program Control Instructions (Zero, One, or Two Operands)

Mnemonic	Operands	Instruction
RET		Return from Subroutine
RETS		Inter-segment Return to Subroutine
IRET		Return from Interrupt
JRcc	dst	Jump Relative If Condition is Met
JPcc	dst	Jump if Condition is Met
JP	dst	Unconditional Jump
JPS	dst [1)]	Unconditional Inter-segment Jump
CALL	dst	Unconditional Call
CALLS	dst [1)]	Inter-segment Call to Subroutine
BTJF	src,N	Bit Test and Jump if False
BTJT	src,N	Bit Test and Jump if True
DJNZ	dst,N	Decrement a Working Register and Jump if Non Zero
DWJNZ	dst,N	Decrement a Register Pair and Jump if Non Zero
CPJFI	src1, src2, N	Compare and Jump on False. Otherwise Post Increment
CPJTI	src1, src2, N	Compare and Jump on True. Otherwise Post Increment

1) There are two operands for JPS and CALLS:
-- the destination segment (1 byte)
- the destination address (2 bytes)

INSTRUCTION SET (Continued)

Table 9. Miscellaneous (None, One or Two Operands)

Mnemonic	Operands	Instruction
XCH	dst,src	Exchange Registers
SRP	src	Set Register Pointer Long (16 working registers)
SRP0	src	Set Register Pointer 0 (8 LSB working register)
SRP1	src	Set Register Pointer 1 (8 MSB working register)
SPP	src	Set Page Pointer
EXT	dst	Sign Extend
EI		Enable Interrupts
DI		Disable Interrupts
SCF		Set Carry Flag
RCF		Reset Carry Flag
CCF		Complement Carry Flag
SPM		Select Extended Memory addressing scheme through CSR Register
SDM		Select Extended Memory addressing scheme through DPRx Registers
NOP		No Operation
WFI		Stop Program Execution and Wait for the next Enabled Interrupt. If a DMA request is present, the CPU executes the DMA service routine and then automatically returns to the WFI
HALT		Stop Program Execution Until Next System Reset

INSTRUCTION SET (Continued)

1.2.1 ST9 Processor Flags

An important feature of a single chip microcomputer is the ability to test data and make the appropriate action based on the results. In order to provide this facility, FLAGR (register 231) in the register file is used as a flag register. Six bits of this register are used as the following flags:

Bit 7: C - Carry

Bit 6: Z - Zero

Bit 5: S - Sign

Bit 4: V - Overflow

Bit 3: D - Decimal Adjust

Bit 2: H - Half Carry

Bit 1 is reserved for emulation, and should be always written as 0.

Bit 0 selects extended memory addressing scheme through CSR or DPRx registers.

The Flag Register is further described in the Architecture Chapter.

1.2.2 Condition Codes

Flags C, Z, S, and OV control the operation of the “conditional” Jump instructions. The next table shows the condition codes and the flag settings.

Note : Some of the Status flags are used to indicate more than one condition e.g. Zero and Equal. In such cases the condition code is the same for both conditions.

Table 10. Condition Codes Table

Mnemonic code	Meaning	Flag setting	Hex. value	Binary value
F	Always False	_ _ _ _	0	0000
T	Always True	_ _ _ _	8	1000
C	Carry	C=1	7	0111
NC	Not carry	C=0	F	1111
Z	Zero	Z=1	6	0110
NZ	Not Zero	Z=0	E	1110
PL	Plus	S=0	D	1101
MI	Minus	S=1	5	0101
OV	Overflow	V=1	4	0100
NOV	No Overflow	V=0	C	1100
EQ	Equal	Z=1	6	0110
NE	Not Equal	Z=0	E	1110
GE	Greater Than or Equal	(S xor V)=0	9	1001
LT	Less Than	(S xor V)=1	1	0001
GT	Greater Than	(Z or(S xor V))=0	A	1010
LE	Less Than or Equal	(Z or(S xor V))=1	2	0010
UGE	Unsigned Greater Than or Equal	C=0	F	1111
ULT	Unsigned Less Than	C=1	7	0111
UGT	Unsigned Greater Than	(C=0 and Z=0)=1	B	1011
ULE	Unsigned Less Than or Equal	(C or Z)=1	3	0011

INSTRUCTION SET (Continued)

1.2.3 Notation

Operands and status flags are represented by a notational shorthand in the detailed instruction description (see programming manual).

The notation for operands (condition codes and address modes) and the actual operands they represent are as follows:

Table 11. Notation (Part 1)

Notation	Significance	Actual Operand/Range	
cc	Condition Code		
#N #NN	Immediate Byte Immediate Word	# data # data	where data is a byte expression where data is a word expression
r	Working Register	rn	where n=0-15
R	Direct Register	Rn	where n=0-255, except 208-223
rr	Direct Working Register Pair	rrn	where n is an even number in the range 0-14. (n=0,2,4,6....14)
RR	Direct Register Pair	RRn	where n is an even number in the range 0-254. (n=0,2,4,6....254) except 208-222
(r)	Indirect Working Register	(rn)	where n=0-15
(R)	Indirect register	(Rn)	where n=0-255
(r)+	Indirect working register post increment	(rn)+	where n=0-15
N(rx)	Indexed register	N(rx)	where x=0-15; N=0-255 (one byte)
N	Memory relative Short Address		Program label or expression in the range +127/-128 starting from the address of the next instruction
NN	Direct Memory Long Address		Program label or expression in the range 0-65535 in memory area
(rr)	Indirect Pair of Working Register Pointers	(rrn)	Where n is an even number in the range 0-14.(n=0,2,4,6....14)
(rr)+	Indirect Pair of Working Register Pointers with Post Increment	(rrn)+	where n is an even number in the range 0-14.(n=0,2,4,6....14)
-(rr)	Indirect Pair of Working Register Pointers with Pre Decrement	-(rrn)	where n is an even number in the range 0-14.(n=0,2,4,6....14)
N(rrx)	Indexed Pair of Working Register Pointers with Short Offset	N(rrx)	where x is an even number in the range 0-14.(x=0,2,4,6....14) and N is a signed one byte expression between +127/-128

INSTRUCTION SET (Continued)

Table 12. Notation (Part 2)

Notation	Significance	Actual Operand/Range	
NN(rrx)	Indexed Pair of Working Register Pointers with Long Offset	NN(rrx)	where x is an even number in the range 0-14.(x=0,2,4,6....14) and NN is word expression in the range between 0 and 65535
N(RRx)	Indexed Pair of Register Pointers with Short Offset	N(RRx)	where x is an even number in the range 0-254.(x=0,2,4,6...254) and N is a one byte signed expression in the range +127/-128
NN(RRx)	Indexed Pair of Register Pointers with Long Offset	NN(RRx)	where x is an even number in the range 0-254.(x=0,2,4,6....14) and NN is word expression in the range between 0 and 65535
rr(rrx)	Indexed Pair of Working Registers with a Pair of Working Registers used as Offset	rrn(rrx)	where n and x are two even numbers in the range 0-14. (n,x=0,2,4,6....14)
r.b	Bit pointer in a direct working register	rn.b	n=0.15 b is a number between 0-7; 0 is LSB, 7 is MSB
(rr).b	Bit pointer in a Memory Location using a Pair of Indirect Working Registers as Address Pointer	(rrn).b	where n is an even number in the range 0-14.(n=0,2,4,6....14) b is a number between 0-7; 0 is LSB, 7 is MSB
(RR)	Indirect pair of Register Pointer	(RRn)	where n is an even number in the range 0-255.(n=0,2,4,6....254)

Table 13. Symbols

Symbol	Meaning
dst	Destination Operand
src	Source Operand
OPC	Operation Code
XTN	Operation Code Extension
ofs	Source Offset
ofd	Destination Offset
r.b	Bit and Working Register
SSP	System Stack Pointer
USP	User Stack Pointer
PC	Program Counter
CC	Condition Code
C	Carry Flag
Z	Zero Flag
S	Sign Flag
V	Overflow Flag
D	Decimal Adjust Flag
CIC	Central Interrupt Control Register
btd	Source Bit of Working Register
⇐	Assignment of Result

1.3 INSTRUCTION SUMMARY

The following tables summarize the operation for each of the instructions which are listed with their corresponding mnemonic codes, addressing modes, byte counts, timing information, and affected flags.

GENERAL NOTES:

FLAGS STATUS:

– ^ : affected

– - : not affected

– 0 : reset to zero

– 1 : set to one

– ? : undefined

Note: for detailed information on the instruction set refer to the ST9+ programming manual.

– dst: destination operand

– src: source operand

– SSP: system stack pointer

– USP: user stack pointer

– PC: program counter

– cc: condition code

– C: carry flag

– CPR: code page register

– Z: zero flag

– S: sign flag

– V: overflow flag

– D: decimal adjust flag

– CIC: central interrupt control register

– DP : selects extended memory addressing scheme through CSR or DPRx registers.

TIMING INFORMATION:

The number of clock cycles given is valid when no wait states are added to memory accesses. In order to facilitate the evaluation of timings when wait states are added to memory access, two additional columns are given: P and D. P gives the number of accesses to program memory for instruction fetch: if wait states are added when accessing the memory containing the code, the number of these wait states, multiplied by the value of column P, must be added to the instruction duration. The same applies to column D, which gives the number of accesses needed for operands; these are typically in data memory, unless (except for stack operations, which are always performed with data memory) bit 0 of the FLAGS register is 0 (e.g. after executing the SPM instruction).

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
ADC : Addition of two bytes with carry								
ADC	r	r	2	4	2	0	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	R	R	3	6	3	0	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	r	R	3	6	3	0	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	R	r	3	6	3	0	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	r	(r)	2	6	2	0	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	R	(r)	3	6	3	0	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	r	(rr)	3	8	3	1	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	R	(rr)	3	8	3	1	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	r	NN	4	10	4	1	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	r	N(rrx)	4	12	4	1	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	R	N(rrx)	4	12	4	1	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	r	NN(rrx)	5	14	5	1	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	R	NN(rrx)	5	14	5	1	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	r	rr(rrx)	3	12	3	1	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	r	(rr)+	3	12	3	1	dst<-dst+src+C rr<-rr+1	^ ^ ^ ^ 0 ^
ADC	R	(rr)+	3	12	3	1	dst<-dst+src+C rr<-rr+1	^ ^ ^ ^ 0 ^
ADC	r	-(rr)	3	12	3	1	rr<-rr-1 dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	R	-(rr)	3	12	3	1	rr<-rr-1 dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	(r)	r	3	6	3	0	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	(r)	R	3	6	3	0	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	(rr)	r	3	12	3	2	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	(rr)	R	3	12	3	2	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	(rr)+	r	3	14	3	2	dst<-dst+src+C rr<-rr+1	^ ^ ^ ^ 0 ^
ADC	(rr)+	R	3	14	3	2	dst<-dst+src+C rr<-rr+1	^ ^ ^ ^ 0 ^
ADC	NN	r	4	12	4	2	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	N(rrx)	r	4	14	4	2	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	N(rrx)	R	4	14	4	2	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	NN(rrx)	r	5	16	5	2	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	NN(rrx)	R	5	16	5	2	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	rr(rrx)	r	3	14	3	2	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	-(rr)	r	3	14	3	2	rr<-rr-1 dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	-(rr)	R	3	14	3	2	rr<-rr-1 dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	r	#N	3	6	3	0	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	R	#N	3	6	3	0	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	(rr)	#N	3	10	3	2	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	NN	#N	5	16	5	2	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	(rr)	(rr)	3	14	3	3	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	(RR)	(rr)	3	14	3	3	dst<-dst+src+C	^ ^ ^ ^ 0 ^

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
ADCW : Add word with carry								
ADCW	rr	rr	2	8	2	0	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	RR	RR	3	8	3	0	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	rr	RR	3	8	3	0	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	RR	rr	3	8	3	0	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	rr	(r)	3	10	3	0	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	RR	(r)	3	10	3	0	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	rr	(rr)	2	12	2	2	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	RR	(rr)	3	12	3	2	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	rr	NN	4	14	4	2	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	rr	N(rrx)	4	14	4	2	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	RR	N(rrx)	4	14	4	2	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	rr	NN(rrx)	5	16	5	2	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	RR	NN(rrx)	5	16	5	2	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	rr	rr(rrx)	3	14	3	2	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	rr	(rr)+	3	14	3	2	dst<-dst+src+C rr<-rr+2	^ ^ ^ ^ ? ?
ADCW	RR	(rr)+	3	14	3	2	dst<-dst+src+C rr<-rr+2	^ ^ ^ ^ ? ?
ADCW	rr	-(rr)	3	14	3	2	rr<-rr-2 dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	RR	-(rr)	3	14	3	2	rr<-rr-2 dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	(r)	rr	3	10	3	0	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	(r)	RR	3	10	3	0	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	(rr)	rr	2	16	2	4	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	(rr)	RR	3	18	3	4	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	(rr)+	rr	3	18	3	4	dst<-dst+src+C rr<-rr+2	^ ^ ^ ^ ? ?
ADCW	(rr)+	RR	3	18	3	4	dst<-dst+src+C rr<-rr+2	^ ^ ^ ^ ? ?
ADCW	NN	rr	4	18	4	4	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	N(rrx)	rr	4	18	4	4	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	N(rrx)	RR	4	18	4	4	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	NN(rrx)	rr	5	20	5	4	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	NN(rrx)	RR	5	20	5	4	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	rr(rrx)	rr	3	18	3	4	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	-(rr)	rr	3	18	3	4	rr<-rr-2 dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	-(rr)	RR	3	18	3	4	rr<-rr-2 dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	rr	#NN	4	10	4	0	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	RR	#NN	4	10	4	0	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	(rr)	#NN	4	18	4	4	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	NN	#NN	6	22	6	4	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	N(rrx)	#NN	5	20	5	4	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	NN(rrx)	#NN	6	22	6	4	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	(rr)	(rr)	2	20	2	6	dst<-dst+src+C	^ ^ ^ ^ ? ?

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
ADD : Addition of two bytes without carry								
ADD	r	r	2	4	2	0	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	R	R	3	6	3	0	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	r	R	3	6	3	0	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	R	r	3	6	3	0	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	r	(r)	2	6	2	0	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	R	(r)	3	6	3	0	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	r	(rr)	3	8	3	1	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	R	(rr)	3	8	3	1	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	r	NN	4	10	4	1	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	r	N(rrx)	4	12	4	1	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	R	N(rrx)	4	12	4	1	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	r	NN(rrx)	5	14	5	1	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	R	NN(rrx)	5	14	5	1	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	r	rr(rrx)	3	12	3	1	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	r	(rr)+	3	12	3	1	dst<-dst+src rr<-rr+1	^ ^ ^ ^ 0 ^
ADD	R	(rr)+	3	12	3	1	dst<-dst+src rr<-rr+1	^ ^ ^ ^ 0 ^
ADD	r	-(rr)	3	12	3	1	rr<-rr-1 dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	R	-(rr)	3	12	3	1	rr<-rr-1 dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	(r)	r	3	6	3	0	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	(r)	R	3	6	3	0	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	(rr)	r	3	12	3	2	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	(rr)	R	3	12	3	2	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	(rr)+	r	3	14	3	2	dst<-dst+src rr<-rr+1	^ ^ ^ ^ 0 ^
ADD	(rr)+	R	3	14	3	2	dst<-dst+src rr<-rr+1	^ ^ ^ ^ 0 ^
ADD	NN	r	4	12	4	2	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	N(rrx)	r	4	14	4	2	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	N(rrx)	R	4	14	4	2	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	NN(rrx)	r	5	16	5	2	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	NN(rrx)	R	5	16	5	2	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	rr(rrx)	r	3	14	3	2	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	-(rr)	r	3	14	3	2	rr<-rr-1 dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	-(rr)	R	3	14	3	2	rr<-rr-1 dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	r	#N	3	6	3	0	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	R	#N	3	6	3	0	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	(rr)	#N	3	10	3	2	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	NN	#N	5	16	5	2	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	(rr)	(rr)	3	14	3	3	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	(RR)	(rr)	3	14	3	3	dst<-dst+src	^ ^ ^ ^ 0 ^

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
ADDW : Add words without carry								
ADDW	rr	rr	2	8	2	0	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	RR	RR	3	8	3	0	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	rr	RR	3	8	3	0	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	RR	rr	3	8	3	0	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	rr	(r)	3	10	3	0	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	RR	(r)	3	10	3	0	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	rr	(rr)	2	12	2	2	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	RR	(rr)	3	12	3	2	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	rr	NN	4	14	4	2	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	rr	N(rrx)	4	14	4	2	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	RR	N(rrx)	4	14	4	2	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	rr	NN(rrx)	5	16	5	2	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	RR	NN(rrx)	5	16	5	2	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	rr	rr(rrx)	3	14	3	2	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	rr	(rr)+	3	14	3	2	dst<-dst+src rr<-rr+2	^ ^ ^ ^ ? ?
ADDW	RR	(rr)+	3	14	3	2	dst<-dst+src rr<-rr+2	^ ^ ^ ^ ? ?
ADDW	rr	-(rr)	3	14	3	2	rr<-rr-2 dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	RR	-(rr)	3	14	3	2	rr<-rr-2 dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	(r)	rr	3	10	3	0	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	(r)	RR	3	10	3	0	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	(rr)	rr	2	16	2	4	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	(rr)	RR	3	18	3	4	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	(rr)+	rr	3	18	3	4	dst<-dst+src rr<-rr+2	^ ^ ^ ^ ? ?
ADDW	(rr)+	RR	3	18	3	4	dst<-dst+src rr<-rr+2	^ ^ ^ ^ ? ?
ADDW	NN	rr	4	18	4	4	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	N(rrx)	rr	4	18	4	4	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	N(rrx)	RR	4	18	4	4	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	NN(rrx)	rr	5	20	5	4	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	NN(rrx)	RR	5	20	5	4	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	rr(rrx)	rr	3	18	3	4	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	-(rr)	rr	3	18	3	4	rr<-rr-2 dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	-(rr)	RR	3	18	3	4	rr<-rr-2 dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	rr	#NN	4	10	4	0	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	RR	#NN	4	10	4	0	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	(rr)	#NN	4	18	4	4	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	NN	#NN	6	22	6	4	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	N(rrx)	#NN	5	20	5	4	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	NN(rrx)	#NN	6	22	6	4	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	(rr)	(rr)	2	20	2	6	dst<-dst+src	^ ^ ^ ^ ? ?

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
AND : Logical AND between two bytes								
AND	r	r	2	4	2	0	dst<-dst AND src	- ^ ^ 0 - -
AND	R	R	3	6	3	0	dst<-dst AND src	- ^ ^ 0 - -
AND	r	R	3	6	3	0	dst<-dst AND src	- ^ ^ 0 - -
AND	R	r	3	6	3	0	dst<-dst AND src	- ^ ^ 0 - -
AND	r	(r)	2	6	2	0	dst<-dst AND src	- ^ ^ 0 - -
AND	R	(r)	3	6	3	0	dst<-dst AND src	- ^ ^ 0 - -
AND	r	(rr)	3	8	3	1	dst<-dst AND src	- ^ ^ 0 - -
AND	R	(rr)	3	8	3	1	dst<-dst AND src	- ^ ^ 0 - -
AND	r	NN	4	10	4	1	dst<-dst AND src	- ^ ^ 0 - -
AND	r	N(rrx)	4	12	4	1	dst<-dst AND src	- ^ ^ 0 - -
AND	R	N(rrx)	4	12	4	1	dst<-dst AND src	- ^ ^ 0 - -
AND	r	NN(rrx)	5	14	5	1	dst<-dst AND src	- ^ ^ 0 - -
AND	R	NN(rrx)	5	14	5	1	dst<-dst AND src	- ^ ^ 0 - -
AND	r	rr(rrx)	3	12	3	1	dst<-dst AND src	- ^ ^ 0 - -
AND	r	(rr)+	3	12	3	1	dst<-dst AND src rr<-rr+1	- ^ ^ 0 - -
AND	R	(rr)+	3	12	3	1	dst<-dst AND src rr<-rr+1	- ^ ^ 0 - -
AND	r	-(rr)	3	12	3	1	rr<-rr-1 dst<-dst AND src	- ^ ^ 0 - -
AND	R	-(rr)	3	12	3	1	rr<-rr-1 dst<-dst AND src	- ^ ^ 0 - -
AND	(r)	r	3	6	3	0	dst<-dst AND src	- ^ ^ 0 - -
AND	(r)	R	3	6	3	0	dst<-dst AND src	- ^ ^ 0 - -
AND	(rr)	r	3	12	3	2	dst<-dst AND src	- ^ ^ 0 - -
AND	(rr)	R	3	12	3	2	dst<-dst AND src	- ^ ^ 0 - -
AND	(rr)+	r	3	14	3	2	dst<-dst AND src rr<-rr+1	- ^ ^ 0 - -
AND	(rr)+	R	3	14	3	2	dst<-dst AND src rr<-rr+1	- ^ ^ 0 - -
AND	NN	r	4	12	4	2	dst<-dst AND src	- ^ ^ 0 - -
AND	N(rrx)	r	4	14	4	2	dst<-dst AND src	- ^ ^ 0 - -
AND	N(rrx)	R	4	14	4	2	dst<-dst AND src	- ^ ^ 0 - -
AND	NN(rrx)	r	5	16	5	2	dst<-dst AND src	- ^ ^ 0 - -
AND	NN(rrx)	R	5	16	5	2	dst<-dst AND src	- ^ ^ 0 - -
AND	rr(rrx)	r	3	14	3	2	dst<-dst AND src	- ^ ^ 0 - -
AND	-(rr)	r	3	14	3	2	rr<-rr-1 dst<-dst AND src	- ^ ^ 0 - -
AND	-(rr)	R	3	14	3	2	rr<-rr-1 dst<-dst AND src	- ^ ^ 0 - -
AND	r	#N	3	6	3	0	dst<-dst AND src	- ^ ^ 0 - -
AND	R	#N	3	6	3	0	dst<-dst AND src	- ^ ^ 0 - -
AND	(rr)	#N	3	10	3	2	dst<-dst AND src	- ^ ^ 0 - -
AND	NN	#N	5	16	5	2	dst<-dst AND src	- ^ ^ 0 - -
AND	(rr)	(rr)	3	14	3	3	dst<-dst AND src	- ^ ^ 0 - -
AND	(RR)	(rr)	3	14	3	3	dst<-dst AND src	- ^ ^ 0 - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
ANDW : Logical AND between two words								
ANDW	rr	rr	2	8	2	0	dst<-dst AND src	- ^ ^ 0 - -
ANDW	RR	RR	3	8	3	0	dst<-dst AND src	- ^ ^ 0 - -
ANDW	rr	RR	3	8	3	0	dst<-dst AND src	- ^ ^ 0 - -
ANDW	RR	rr	3	8	3	0	dst<-dst AND src	- ^ ^ 0 - -
ANDW	rr	(r)	3	10	3	0	dst<-dst AND src	- ^ ^ 0 - -
ANDW	RR	(r)	3	10	3	0	dst<-dst AND src	- ^ ^ 0 - -
ANDW	rr	(rr)	2	12	2	2	dst<-dst AND src	- ^ ^ 0 - -
ANDW	RR	(rr)	3	12	3	2	dst<-dst AND src	- ^ ^ 0 - -
ANDW	rr	NN	4	14	4	2	dst<-dst AND src	- ^ ^ 0 - -
ANDW	rr	N(rrx)	4	14	4	2	dst<-dst AND src	- ^ ^ 0 - -
ANDW	RR	N(rrx)	4	14	4	2	dst<-dst AND src	- ^ ^ 0 - -
ANDW	rr	NN(rrx)	5	16	5	2	dst<-dst AND src	- ^ ^ 0 - -
ANDW	RR	NN(rrx)	5	16	5	2	dst<-dst AND src	- ^ ^ 0 - -
ANDW	rr	rr(rrx)	3	14	3	2	dst<-dst AND src	- ^ ^ 0 - -
ANDW	rr	(rr)+	3	14	3	2	dst<-dst AND src rr<-rr+2	- ^ ^ 0 - -
ANDW	RR	(rr)+	3	14	3	2	dst<-dst AND src rr<-rr+2	- ^ ^ 0 - -
ANDW	rr	-(rr)	3	14	3	2	rr<-rr-2 dst<-dst AND src	- ^ ^ 0 - -
ANDW	RR	-(rr)	3	14	3	2	rr<-rr-2 dst<-dst AND src	- ^ ^ 0 - -
ANDW	(r)	rr	3	10	3	0	dst<-dst AND src	- ^ ^ 0 - -
ANDW	(r)	RR	3	10	3	0	dst<-dst AND src	- ^ ^ 0 - -
ANDW	(rr)	rr	2	16	2	4	dst<-dst AND src	- ^ ^ 0 - -
ANDW	(rr)	RR	3	18	3	4	dst<-dst AND src	- ^ ^ 0 - -
ANDW	(rr)+	rr	3	18	3	4	dst<-dst AND src rr<-rr+2	- ^ ^ 0 - -
ANDW	(rr)+	RR	3	18	3	4	dst<-dst AND src rr<-rr+2	- ^ ^ 0 - -
ANDW	NN	rr	4	18	4	4	dst<-dst AND src	- ^ ^ 0 - -
ANDW	N(rrx)	rr	4	18	4	4	dst<-dst AND src	- ^ ^ 0 - -
ANDW	N(rrx)	RR	4	18	4	4	dst<-dst AND src	- ^ ^ 0 - -
ANDW	NN(rrx)	rr	5	20	5	4	dst<-dst AND src	- ^ ^ 0 - -
ANDW	NN(rrx)	RR	5	20	5	4	dst<-dst AND src	- ^ ^ 0 - -
ANDW	rr(rrx)	rr	3	18	3	4	dst<-dst AND src	- ^ ^ 0 - -
ANDW	-(rr)	rr	3	18	3	4	rr<-rr-2 dst<-dst AND src	- ^ ^ 0 - -
ANDW	-(rr)	RR	3	18	3	4	rr<-rr-2 dst<-dst AND src	- ^ ^ 0 - -
ANDW	rr	#NN	4	10	4	0	dst<-dst AND src	- ^ ^ 0 - -
ANDW	RR	#NN	4	10	4	0	dst<-dst AND src	- ^ ^ 0 - -
ANDW	(rr)	#NN	4	18	4	4	dst<-dst AND src	- ^ ^ 0 - -
ANDW	NN	#NN	6	22	6	4	dst<-dst AND src	- ^ ^ 0 - -
ANDW	N(rrx)	#NN	5	20	5	4	dst<-dst AND src	- ^ ^ 0 - -
ANDW	NN(rrx)	#NN	6	22	6	4	dst<-dst AND src	- ^ ^ 0 - -
ANDW	(rr)	(rr)	2	20	2	6	dst<-dst AND src	- ^ ^ 0 - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
BAND : Bit AND								
BAND	r.b	r.b	3	10	3	0	dst bit<-dst bit AND src bit	- - - - - -
BAND	r.b	r.!b	3	10	3	0	dst bit<-dst bit AND complemented src bit	- - - - - -
BCPL : Bit Complement								
BCPL	r.b		2	4	2	0	dst bit<-dst bit complemented	- - - - - -
BLD : Bit Load								
BLD	r.b	r.b	3	10	3	0	dst bit<-src bit	- - - - - -
BLD	r.b	r.!b	3	10	3	0	dst bit<-src bit complemented	- - - - - -
BOR : Bit OR								
BOR	r.b	r.b	3	10	3	0	dst bit<-dst bit OR src bit	- - - - - -
BOR	r.b	r.!b	3	10	3	0	dst bit<-dst bit OR complemented src bit	- - - - - -
BRES : Bit Reset								
BRES	r.b		2	4	2	0	dst bit<- 0	- - - - - -
BSET : Bit Set								
BSET	r.b		2	4	2	0	dst bit<- 1	- - - - - -
BTJF, BTJT : Bit test and jump								
BTJF	r.b	N	3	6/10	3/4	0	If test bit is 0, PC<-PC+N	- - - - - -
BTJT	r.b	N	3	6/10	3/4	0	If test bit is 1, PC<-PC+N	- - - - - -
BXOR : Bit Exclusive OR								
BXOR	r.b	r.b	3	10	3	0	dst bit<-dst bit XOR src bit	- - - - - -
BXOR	r.b	r.!b	3	10	3	0	dst bit<-dst bit XOR complemented src bit	- - - - - -
BTSET : Bit Test and Set								
BTSET	r.b		2	8	2	0	If test bit = 0, test bit <-1,Z<-1	- ^ ^ 0 - -
BTSET	(rr).b		2	14	2	2	If test bit = 0, test bit <-1,Z<-1	- ^ ^ 0 - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
CALL : Call a subroutine								
CALL	NN		3	12	3	2(1)	SSP<-SSP-2,(SSP)< PC, PC<-dst	- - - - - -
CALL	(rr)		2	12	3	2(1)	SSP<-SSP-2,(SSP)< PC, PC<-dst	- - - - - -
CALL	(RR)		2	12	3	2(1)	SSP<-SSP-2,(SSP)< PC, PC<-dst	- - - - - -
CALLS : Call a subroutine in another segment								
CALLS	N,NN		3	16	4	3(1)	SSP<-SSP-3, (SSP+1)<- PC, (SSP)<-CSR, CSR<-N, PC<-NN	- - - - - -
CALLS	(r),(rr)		2	16	3	3(1)	SSP<-SSP-3, (SSP+1)<- PC, (SSP)<-CSR, CSR<-r, PC<-rr	- - - - - -
CALLS	(R), (rr)		2	16	3	3(1)	SSP<-SSP-3, (SSP+1)<- PC, (SSP)<-CSR, CSR<-R, PC<-rr	- - - - - -
CCF : Complement Carry Flag								
CCF			1	4	1	0	C <- C complemented	^ - - - - -
CLR : Clear register								
CLR	r		2	4	2	0	dst<-0	- - - - - -
CLR	R		2	4	2	0	dst<-0	- - - - - -
CLR	(r)		2	4	2	0	dst<-0	- - - - - -
CLR	(R)		2	4	2	0	dst<-0	- - - - - -

Note 1. No data memory accesses are performed if the system stack is kept in the Register File.

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
CP : Compare bytes								
CP	r	r	2	4	2	0	dst-src	^ ^ ^ ^ - -
CP	R	R	3	6	3	0	dst-src	^ ^ ^ ^ - -
CP	r	R	3	6	3	0	dst-src	^ ^ ^ ^ - -
CP	R	r	3	6	3	0	dst-src	^ ^ ^ ^ - -
CP	r	(r)	2	6	2	0	dst-src	^ ^ ^ ^ - -
CP	R	(r)	3	6	3	0	dst-src	^ ^ ^ ^ - -
CP	r	(rr)	3	8	3	1	dst-src	^ ^ ^ ^ - -
CP	R	(rr)	3	8	3	1	dst-src	^ ^ ^ ^ - -
CP	r	NN	4	10	4	1	dst-src	^ ^ ^ ^ - -
CP	r	N(rrx)	4	12	4	1	dst-src	^ ^ ^ ^ - -
CP	R	N(rrx)	4	12	4	1	dst-src	^ ^ ^ ^ - -
CP	r	NN(rrx)	5	14	5	1	dst-src	^ ^ ^ ^ - -
CP	R	NN(rrx)	5	14	5	1	dst-src	^ ^ ^ ^ - -
CP	r	rr(rrx)	3	12	3	1	dst-src	^ ^ ^ ^ - -
CP	r	(rr)+	3	12	3	1	dst-src,rr<-rr+1	^ ^ ^ ^ - -
CP	R	(rr)+	3	12	3	1	dst-src,rr<-rr+1	^ ^ ^ ^ - -
CP	r	-(rr)	3	12	3	1	rr<-rr-1,dst-src	^ ^ ^ ^ - -
CP	R	-(rr)	3	12	3	1	rr<-rr-1,dst-src	^ ^ ^ ^ - -
CP	(r)	r	3	6	3	0	dst-src	^ ^ ^ ^ - -
CP	(r)	R	3	6	3	0	dst-src	^ ^ ^ ^ - -
CP	(rr)	r	3	10	3	1	dst-src	^ ^ ^ ^ - -
CP	(rr)	R	3	10	3	1	dst-src	^ ^ ^ ^ - -
CP	(rr)+	r	3	12	3	1	dst-src,rr<-rr+1	^ ^ ^ ^ - -
CP	(rr)+	R	3	12	3	1	dst-src,rr<-rr+1	^ ^ ^ ^ - -
CP	NN	r	4	10	4	1	dst-src	^ ^ ^ ^ - -
CP	N(rrx)	r	4	12	4	1	dst-src	^ ^ ^ ^ - -
CP	N(rrx)	R	4	12	4	1	dst-src	^ ^ ^ ^ - -
CP	NN(rrx)	r	5	14	5	1	dst-src	^ ^ ^ ^ - -
CP	NN(rrx)	R	5	14	5	1	dst-src	^ ^ ^ ^ - -
CP	rr(rrx)	r	3	12	3	1	dst-src	^ ^ ^ ^ - -
CP	-(rr)	r	3	12	3	1	rr<-rr-1,dst-src	^ ^ ^ ^ - -
CP	-(rr)	R	3	12	3	1	rr<-rr-1,dst-src	^ ^ ^ ^ - -
CP	r	#N	3	6	3	0	dst-src	^ ^ ^ ^ - -
CP	R	#N	3	6	3	0	dst-src	^ ^ ^ ^ - -
CP	(rr)	#N	3	8	3	1	dst-src	^ ^ ^ ^ - -
CP	NN	#N	5	14	5	1	dst-src	^ ^ ^ ^ - -
CP	(rr)	(rr)	3	12	3	2	dst-src	^ ^ ^ ^ - -
CP	(RR)	(rr)	3	12	3	2	dst-src	^ ^ ^ ^ - -
CPJFI, CPJTI : Compare with post-increment								
CPJFI	(rr), r	N	3	14/16	3	1	If compare not verified jump otherwise post-increment	- - - - - -
CPJTI	(rr), r	N	3	14/16	3	1	If compare verified jump otherwise post-increment	- - - - - -
CPL : Complement Register								
CPL	r		2	4	2	0	dst<- NOT dst	- ^ ^ 0 - -
CPL	R		2	4	2	0	dst<- NOT dst	- ^ ^ 0 - -
CPL	(r)		2	4	2	0	dst<- NOT dst	- ^ ^ 0 - -
CPL	(R)		2	4	2	0	dst<- NOT dst	- ^ ^ 0 - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
CPW : Compare word								
CPW	rr	rr	2	8	2	0	dst-src	^ ^ ^ ^ - -
CPW	RR	RR	3	8	3	0	dst-src	^ ^ ^ ^ - -
CPW	rr	RR	3	8	3	0	dst-src	^ ^ ^ ^ - -
CPW	RR	rr	3	8	3	0	dst-src	^ ^ ^ ^ - -
CPW	rr	(r)	3	10	3	0	dst-src	^ ^ ^ ^ - -
CPW	RR	(r)	3	10	3	0	dst-src	^ ^ ^ ^ - -
CPW	rr	(rr)	2	12	2	2	dst-src	^ ^ ^ ^ - -
CPW	RR	(rr)	3	14	3	2	dst-src	^ ^ ^ ^ - -
CPW	rr	NN	4	14	4	2	dst-src	^ ^ ^ ^ - -
CPW	rr	N(rrx)	4	14	4	2	dst-src	^ ^ ^ ^ - -
CPW	RR	N(rrx)	4	14	4	2	dst-src	^ ^ ^ ^ - -
CPW	rr	NN(rrx)	5	16	5	2	dst-src	^ ^ ^ ^ - -
CPW	RR	NN(rrx)	5	16	5	2	dst-src	^ ^ ^ ^ - -
CPW	rr	rr(rrx)	3	14	3	2	dst-src	^ ^ ^ ^ - -
CPW	rr	(rr)+	3	14	3	2	dst-src rr<-rr+2	^ ^ ^ ^ - -
CPW	RR	(rr)+	3	14	3	2	dst-src rr<-rr+2	^ ^ ^ ^ - -
CPW	rr	-(rr)	3	14	3	2	rr<-rr-2 dst-src	^ ^ ^ ^ - -
CPW	RR	-(rr)	3	14	3	2	rr<-rr-2 dst-src	^ ^ ^ ^ - -
CPW	(r)	rr	3	10	3	0	dst-src	^ ^ ^ ^ - -
CPW	(r)	RR	3	10	3	0	dst-src	^ ^ ^ ^ - -
CPW	(rr)	rr	2	14	2	2	dst-src	^ ^ ^ ^ - -
CPW	(rr)	RR	3	14	3	2	dst-src	^ ^ ^ ^ - -
CPW	(rr)+	rr	3	14	3	2	dst-src rr<-rr+2	^ ^ ^ ^ - -
CPW	(rr)+	RR	3	14	3	2	dst-src rr<-rr+2	^ ^ ^ ^ - -
CPW	NN	rr	4	16	4	2	dst-src	^ ^ ^ ^ - -
CPW	N(rrx)	rr	4	14	4	2	dst-src	^ ^ ^ ^ - -
CPW	N(rrx)	RR	4	14	4	2	dst-src	^ ^ ^ ^ - -
CPW	NN(rrx)	rr	5	16	5	2	dst-src	^ ^ ^ ^ - -
CPW	NN(rrx)	RR	5	16	5	2	dst-src	^ ^ ^ ^ - -
CPW	rr(rrx)	rr	3	14	3	2	dst-src	^ ^ ^ ^ - -
CPW	-(rr)	rr	3	14	3	2	rr<-rr-2 dst-src	^ ^ ^ ^ - -
CPW	-(rr)	RR	3	14	3	2	rr<-rr-2 dst-src	^ ^ ^ ^ - -
CPW	rr	#NN	4	10	4	0	dst-src	^ ^ ^ ^ - -
CPW	RR	#NN	4	10	4	0	dst-src	^ ^ ^ ^ - -
CPW	(rr)	#NN	4	14	4	2	dst-src	^ ^ ^ ^ - -
CPW	NN	#NN	6	20	6	2	dst-src	^ ^ ^ ^ - -
CPW	N(rrx)	#NN	5	16	5	2	dst-src	^ ^ ^ ^ - -
CPW	NN(rrx)	#NN	6	18	6	2	dst-src	^ ^ ^ ^ - -
CPW	(rr)	(rr)	2	16	2	4	dst-src	^ ^ ^ ^ - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
DA : Decimal adjust								
DA	r		2	4	2	0	dst<- DA dst	^ ^ ^ ? - -
DA	R		2	4	2	0	dst<- DA dst	^ ^ ^ ? - -
DA	(r)		2	6	2	0	dst<- DA dst	^ ^ ^ ? - -
DA	(R)		2	6	2	0	dst<- DA dst	^ ^ ^ ? - -
DEC : Decrement								
DEC	r		2	4	2	0	dst<- dst-1	- ^ ^ ^ - -
DEC	R		2	4	2	0	dst<- dst-1	- ^ ^ ^ - -
DEC	(r)		2	4	2	0	dst<- dst-1	- ^ ^ ^ - -
DEC	(R)		2	4	2	0	dst<- dst-1	- ^ ^ ^ - -
DECW : Decrement Word								
DECW	rr		2	6	2	0	dst<-dst-1	- ^ ^ ^ - -
DECW	RR		2	6	2	0	dst<-dst-1	- ^ ^ ^ - -
DI : Disable Interrupts								
DI			1	2	1	0	Bit 4 of the CIC Register is set to 0	- - - - - -
DIV : Divide 16 by 8								
DIV	rr	r	2	26/14	2/4	0/2	dst / src <- dst high=remainder 16/8 <- dst low=result	note 1
DIVWS : Divide Word Stepped 32 by 16								
DIVWS	rrhigh rrlow	rr	3	26	3	0	32/16	note 1
DJNZ : Decrement a working register and Jump if Non Zero								
DJNZ	r	N	2	6	2/3(3)	0	r <- r-1, If r=0 then PC<-PC+N	note 2
DWJNZ : Decrement a register pair and Jump if Non Zero								
DWJNZ	rr	N	3	8	3/4(3)	0	rr<-rr-1, If rr=0 then PC<-PC+N	note 2
DWJNZ	RR	N	3	8	3/4(3)	0	RR<-RR-1,If RR=0 then PC<-PC+N	

Note 1. Refer to the ST9+ Programming Manual for detailed information.

Note 2. Registers in groups E and F are not allowed, either directly or as working registers.

Note 3. Additional fetch when the jump is taken.

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
EI : Enable Interrupts								
EI			1	2	1	0	Bit 4 of the CICR register is set to 1	- - - - - -
EXT : Sign extend								
EXT	rr		2	6	2	0	r(7) —> r(n) n=8-15	- - - - - -
EXT	RR		2	6	2	0	R(7) —> R(n) n=8-15	- - - - - -
HALT : Halt Operation								
HALT			2	inf.	2	0	Stops all internal clocks until next system reset if not in Watchdog Mode	- - - - - -
INC : Increment								
INC	r		2	4	2	0	dst<- dst+1	- ^ ^ ^ - -
INC	R		2	4	2	0	dst<- dst+1	- ^ ^ ^ - -
INC	(r)		2	4	2	0	dst<- dst+1	- ^ ^ ^ - -
INC	(R)		2	4	2	0	dst<- dst+1	- ^ ^ ^ - -
INCW : Increment Word								
INCW	rr		2	6	2	0	dst<-dst+1	- ^ ^ ^ - -
INCW	RR		2	6	2	0	dst<-dst+1	- ^ ^ ^ - -
IRET : Return from Interrupt Routine								
IRET			1	12/14/ 16	1	3/4 [2]	FLAGS<-(SSP),SSP<-SSP+1, [CSR<-(SSP),SSP<SSP+1](2), PC<-(SSP), SSP<-SSP+2, CICR(4)<-1	note 1
JP : Jump to a Routine								
JP	NN		3	8	3	0	PC<-dst	- - - - - -
JP	(rr)		2	8	3	0	PC<-dst	- - - - - -
JP	(RR)		2	8	3	0	PC<-dst	- - - - - -
JPcc	NN		3	6/8	3	0	IF cc(condition code) is true, PC<-dst	- - - - - -
JPS : Jump to a routine in another segment								
JPS	N,NN		3	10	4	0	CSR<-N, PC<-NN	- - - - - -
JPS	(r),(rr)		2	10	3	0	CSR<-r, PC<-rr	- - - - - -
JPS	(R), (rr)		2	10	3	0	CSR<-R, PC<-rr	- - - - - -
JRcc : Conditional Relative Jump to a Routine								
JRcc	N		2	6	2/3(3)	0	IF cc(condition code)is true, PC<-PC+dst	- - - - - -

Note 1. All flags are restored to original setting (before interrupt occured).

Note 2. Performed only if register file is not used.

Note 3. Additional fetch when the jump is taken.

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
LD : Load byte instructions								
LD	r	r	2	4	2	0	dst<-src	- - - - - -
LD	R	R	3	6	3	0	dst<-src	- - - - - -
LD	r	R	2	4	2	0	dst<-src	- - - - - -
LD	R	r	2	4	2	0	dst<-src	- - - - - -
LD	r	(r)	2	6	2	0	dst<-src	- - - - - -
LD	R	(r)	3	6	3	0	dst<-src	- - - - - -
LD	r	(rr)	2	8	2	1	dst<-src	- - - - - -
LD	R	(rr)	3	8	3	1	dst<-src	- - - - - -
LD	r	NN	4	10	4	1	dst<-src	- - - - - -
LD	r	N(rx)	3	6	3	0	dst<-src	- - - - - -
LD	r	N(rrx)	4	12	4	1	dst<-src	- - - - - -
LD	R	N(rrx)	4	12	4	1	dst<-src	- - - - - -
LD	r	NN(rrx)	5	14	5	1	dst<-src	- - - - - -
LD	R	NN(rrx)	5	14	5	1	dst<-src	- - - - - -
LD	r	rr(rrx)	3	12	3	1	dst<-src	- - - - - -
LD	r	(rr)+	3	12	3	1	dst<-src,rr<-rr+1	- - - - - -
LD	R	(rr)+	3	12	3	1	dst<-src,rr<-rr+1	- - - - - -
LD	r	-(rr)	3	12	3	1	rr<-rr-1,dst<-src	- - - - - -
LD	R	-(rr)	3	12	3	1	rr<-rr-1,dst<-src	- - - - - -
LD	(r)	r	2	4	2	0	dst<-src	- - - - - -
LD	(r)	R	3	6	3	0	dst<-src	- - - - - -
LD	(rr)	r	2	8	2	1	dst<-src	- - - - - -
LD	(rr)	R	3	10	3	1	dst<-src	- - - - - -
LD	(rr)+	r	3	12	3	1	dst<-src,rr<-rr+1	- - - - - -
LD	(rr)+	R	3	12	3	1	dst<-src,rr<-rr+1	- - - - - -
LD	NN	r	4	10	4	1	dst-src	- - - - - -
LD	N(rx)	r	3	6	3	0	dst-src	- - - - - -
LD	N(rrx)	r	4	12	4	1	dst-src	- - - - - -
LD	N(rrx)	R	4	12	4	1	dst-src	- - - - - -
LD	NN(rrx)	r	5	14	5	1	dst-src	- - - - - -
LD	NN(rrx)	R	5	14	5	1	dst-src	- - - - - -
LD	rr(rrx)	r	3	12	3	1	dst-src	- - - - - -
LD	-(rr)	r	3	12	3	1	rr<-rr-1,dst<-src	- - - - - -
LD	-(rr)	R	3	12	3	1	rr<-rr-1,dst<-src	- - - - - -
LD	r	#N	2	4	2	0	dst<-src	- - - - - -
LD	R	#N	3	6	3	0	dst<-src	- - - - - -
LD	(rr)	#N	3	8	3	1	dst<-src	- - - - - -
LD	NN	#N	5	14	5	1	dst<-src	- - - - - -
LD	(rr)	(rr)	3	10	3	2	dst<-src	- - - - - -
LD	(RR)	(rr)	3	10	3	2	dst<-src	- - - - - -
LD	(r)+	(rr)+	2	12	2	1	dst<-src, rr<-rr+1,r<-r+1	- - - - - -
LD	(rr)+	(r)+	2	12	2	1	dst<-src, rr<-rr+1,r<-r+1	- - - - - -
LDPP,LDDP,LDPD, LDDD : Load from / to program / data memory								
LDPP	(rr)+	(rr)+	2	14	4	0	dst<-src (1),rr<-rr+1	- - - - - -
LDDP	(rr)+	(rr)+	2	14	3	1	dst<-src (2),rr<-rr+1	- - - - - -
LDPD	(rr)+	(rr)+	2	14	3	1	dst<-src (3),rr<-rr+1	- - - - - -
LDDD	(rr)+	(rr)+	2	14	2	2	dst<-src (4),rr<-rr+1	- - - - - -

Note 1. dst using CSR, src using CSR.
Note 2. dst using DPRx, src using CSR.
Note 3. dst using CSR, src using DPRx.
Note 4. dst using DPRx, src using DPRx.

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
LDW : Load word instructions								
LDW	rr	rr	2	6	2	0	dst<-src	- - - - - -
LDW	RR	RR	3	6	3	0	dst<-src	- - - - - -
LDW	rr	RR	3	6	3	0	dst<-src	- - - - - -
LDW	RR	rr	3	6	3	0	dst<-src	- - - - - -
LDW	rr	(r)	3	8	3	0	dst<-src	- - - - - -
LDW	RR	(r)	3	8	3	0	dst<-src	- - - - - -
LDW	rr	(rr)	2	10	2	2	dst<-src	- - - - - -
LDW	RR	(rr)	3	10	3	2	dst<-src	- - - - - -
LDW	rr	NN	4	12	4	2	dst<-src	- - - - - -
LDW	rr	N(rx)	3	8	3	0	dst<-src	- - - - - -
LDW	rr	N(rrx)	4	14	4	2	dst<-ssc	- - - - - -
LDW	RR	N(rrx)	4	14	4	2	dst<-src	- - - - - -
LDW	rr	NN(rrx)	5	16	5	2	dst<-src	- - - - - -
LDW	RR	NN(rrx)	5	16	5	2	dst<-src	- - - - - -
LDW	rr	rr(rrx)	3	14	3	2	dst<-src	- - - - - -
LDW	rr	(rr)+	3	14	3	2	dst<-src,rr<-rr+2	- - - - - -
LDW	RR	(rr)+	3	14	3	2	dst<-src,rr<-rr+2	- - - - - -
LDW	rr	-(rr)	3	14	3	2	rr<-rr-2,dst<-src	- - - - - -
LDW	RR	-(rr)	3	14	3	2	rr<-rr-2,dst<-src	- - - - - -
LDW	(r)	rr	3	6	3	0	dst<-src	- - - - - -
LDW	(r)	RR	3	6	3	0	dst<-src	- - - - - -
LDW	(rr)	rr	2	12	2	2	dst<-src	- - - - - -
LDW	(rr)	RR	3	12	3	2	dst<-src	- - - - - -
LDW	(rr)+	rr	3	14	3	2	dst<-src,rr<-rr+2	- - - - - -
LDW	(rr)+	RR	3	14	3	2	dst<-src,rr<-rr+2	- - - - - -
LDW	NN	rr	4	14	4	2	dst<-src	- - - - - -
LDW	N(rx)	rr	3	10	3	0	dst<-src	- - - - - -
LDW	N(rrx)	RR	4	14	4	2	dst<-src	- - - - - -
LDW	N(rrx)	rr	4	14	4	2	dst<-src	- - - - - -
LDW	NN(rrx)	RR	5	16	5	2	dst<-src	- - - - - -
LDW	NN(rrx)	rr	5	16	5	2	dst<-src	- - - - - -
LDW	rr(rrx)	rr	3	14	3	2	dst<-src	- - - - - -
LDW	-(rr)	rr	3	14	3	2	rr<-rr-2,dst<-src	- - - - - -
LDW	-(rr)	RR	3	14	3	2	rr<-rr-2,dst<-src	- - - - - -
LDW	rr	#NN	4	8	4	0	dst<-src	- - - - - -
LDW	RR	#NN	4	8	4	0	dst<-src	- - - - - -
LDW	(rr)	#NN	4	14	4	2	dst<-src	- - - - - -
LDW	N(rrx)	#NN	5	18	5	2	dst<-src	- - - - - -
LDW	NN(rrx)	#NN	6	20	6	2	dst<-src	- - - - - -
LDW	NN	#NN	6	18	6	2	dst<-src	- - - - - -
LDW	(rr)	(rr)	2	16	2	4	dst<-src	- - - - - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
LINK : create stack frame in system stack								
LINK	RR	#N	3	12/16	3	2/0	SSP<-SSP-1[2], (SSP)<-R[R], R[R]<-SSP, SSP<-SSP-N	- - - - - -
LINK	rr	#N	3	12/16	3	2/0	" "	- - - - - -
LINKU : create stack frame in system stack								
LINKU	RR	#N	3	12/16	3	2/0	USP<-USP-1[2], (USP)<-R[R], R[R]<-USP, USP<-USP-N	- - - - - -
LINKU	rr	#N	3	12/16	3	2/0	" "	- - - - - -
MUL : Multiply								
MUL	rr	r	2	22	2	0	dst <- dst x src, 8 x 8 multiply	note 1
NOP : No operation								
NOP			1	2	1	0	No Operation	- - - - - -

Note 1. Refer to ST9 programming manual for detailed information.
Note 2. The value inside [ ] is valid for external memory stack

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
OR : Logical OR between 2 bytes								
OR	r	r	2	4	2	0	dst<-dst OR src	- ^ ^ 0 - -
OR	R	R	3	6	3	0	dst<-dst OR src	- ^ ^ 0 - -
OR	r	R	3	6	3	0	dst<-dst OR src	- ^ ^ 0 - -
OR	R	r	3	6	3	0	dst<-dst OR src	- ^ ^ 0 - -
OR	r	(r)	2	6	2	0	dst<-dst OR src	- ^ ^ 0 - -
OR	R	(r)	3	6	3	0	dst<-dst OR src	- ^ ^ 0 - -
OR	r	(rr)	3	8	3	1	dst<-dst OR src	- ^ ^ 0 - -
OR	R	(rr)	3	8	3	1	dst<-dst OR src	- ^ ^ 0 - -
OR	r	NN	4	10	4	1	dst<-dst OR src	- ^ ^ 0 - -
OR	r	N(rrx)	4	12	4	1	dst<-dst OR src	- ^ ^ 0 - -
OR	R	N(rrx)	4	12	4	1	dst<-dst OR src	- ^ ^ 0 - -
OR	r	NN(rrx)	5	14	5	1	dst<-dst OR src	- ^ ^ 0 - -
OR	R	NN(rrx)	5	14	5	1	dst<-dst OR src	- ^ ^ 0 - -
OR	r	rr(rrx)	3	12	3	1	dst<-dst OR src	- ^ ^ 0 - -
OR	r	(rr)+	3	12	3	1	dst<-dst OR src, rr<-rr+1	- ^ ^ 0 - -
OR	R	(rr)+	3	12	3	1	dst<-dst OR src, rr<-rr+1	- ^ ^ 0 - -
OR	r	-(rr)	3	12	3	1	rr<-rr-1, dst<-dst OR src	- ^ ^ 0 - -
OR	R	-(rr)	3	12	3	1	rr<-rr-1, dst<-dst OR src	- ^ ^ 0 - -
OR	(r)	r	3	6	3	0	dst<-dst OR src	- ^ ^ 0 - -
OR	(r)	R	3	6	3	0	dst<-dst OR src	- ^ ^ 0 - -
OR	(rr)	r	3	12	3	2	dst<-dst OR src	- ^ ^ 0 - -
OR	(rr)	R	3	12	3	2	dst<-dst OR src	- ^ ^ 0 - -
OR	(rr)+	r	3	14	3	2	dst<-dst OR src, rr<-rr+1	- ^ ^ 0 - -
OR	(rr)+	R	3	14	3	2	dst<-dst OR src, rr<-rr+1	- ^ ^ 0 - -
OR	NN	r	4	12	4	2	dst<-dst OR src	- ^ ^ 0 - -
OR	N(rrx)	r	4	14	4	2	dst<-dst OR src	- ^ ^ 0 - -
OR	N(rrx)	R	4	14	4	2	dst<-dst OR src	- ^ ^ 0 - -
OR	NN(rrx)	r	5	16	5	2	dst<-dst OR src	- ^ ^ 0 - -
OR	NN(rrx)	R	5	16	5	2	dst<-dst OR src	- ^ ^ 0 - -
OR	rr(rrx)	r	3	14	3	2	dst<-dst OR src	- ^ ^ 0 - -
OR	-(rr)	r	3	14	3	2	rr<-rr-1, dst<-dst OR src	- ^ ^ 0 - -
OR	-(rr)	R	3	14	3	2	rr<-rr-1, dst<-dst OR src	- ^ ^ 0 - -
OR	r	#N	3	6	3	0	dst<-dst OR src	- ^ ^ 0 - -
OR	R	#N	3	6	3	0	dst<-dst OR src	- ^ ^ 0 - -
OR	(rr)	#N	3	10	3	2	dst<-dst OR src	- ^ ^ 0 - -
OR	NN	#N	5	16	5	2	dst<-dst OR src	- ^ ^ 0 - -
OR	(rr)	(rr)	3	14	3	3	dst<-dst OR src	- ^ ^ 0 - -
OR	(RR)	(rr)	3	14	3	3	dst<-dst OR src	- ^ ^ 0 - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
ORW : Logical OR between two words								
ORW	rr	rr	2	8	2	0	dst<-dst OR src	- ^ ^ 0 - -
ORW	RR	RR	3	8	3	0	dst<-dst OR src	- ^ ^ 0 - -
ORW	rr	RR	3	8	3	0	dst<-dst OR src	- ^ ^ 0 - -
ORW	RR	rr	3	8	3	0	dst<-dst OR src	- ^ ^ 0 - -
ORW	rr	(r)	3	10	3	0	dst<-dst OR src	- ^ ^ 0 - -
ORW	RR	(r)	3	10	3	0	dst<-dst OR src	- ^ ^ 0 - -
ORW	rr	(rr)	2	12	2	2	dst<-dst OR src	- ^ ^ 0 - -
ORW	RR	(rr)	3	12	3	2	dst<-dst OR src	- ^ ^ 0 - -
ORW	rr	NN	4	14	4	2	dst<-dst OR src	- ^ ^ 0 - -
ORW	rr	N(rrx)	4	14	4	2	dst<-dst OR src	- ^ ^ 0 - -
ORW	RR	N(rrx)	4	14	4	2	dst<-dst OR src	- ^ ^ 0 - -
ORW	rr	NN(rrx)	5	16	5	2	dst<-dst OR src	- ^ ^ 0 - -
ORW	RR	NN(rrx)	5	16	5	2	dst<-dst OR src	- ^ ^ 0 - -
ORW	rr	rr(rrx)	3	14	3	2	dst<-dst OR src	- ^ ^ 0 - -
ORW	rr	(rr)+	3	14	3	2	dst<-dst OR src rr<-rr+2	- ^ ^ 0 - -
ORW	RR	(rr)+	3	14	3	2	dst<-dst OR src rr<-rr+2	- ^ ^ 0 - -
ORW	rr	-(rr)	3	14	3	2	rr<-rr-2 dst<-dst OR src	- ^ ^ 0 - -
ORW	RR	-(rr)	3	14	3	2	rr<-rr-2 dst<-dst OR src	- ^ ^ 0 - -
ORW	(r)	rr	3	10	3	0	dst<-dst OR src	- ^ ^ 0 - -
ORW	(r)	RR	3	10	3	0	dst<-dst OR src	- ^ ^ 0 - -
ORW	(rr)	rr	2	16	2	4	dst<-dst OR src	- ^ ^ 0 - -
ORW	(rr)	RR	3	18	3	4	dst<-dst OR src	- ^ ^ 0 - -
ORW	(rr)+	rr	3	18	3	4	dst<-dst OR src rr<-rr+2	- ^ ^ 0 - -
ORW	(rr)+	RR	3	18	3	4	dst<-dst OR src rr<-rr+2	- ^ ^ 0 - -
ORW	NN	rr	4	18	4	4	dst<-dst OR src	- ^ ^ 0 - -
ORW	N(rrx)	rr	4	18	4	4	dst<-dst OR src	- ^ ^ 0 - -
ORW	N(rrx)	RR	4	18	4	4	dst<-dst OR src	- ^ ^ 0 - -
ORW	NN(rrx)	rr	5	20	5	4	dst<-dst OR src	- ^ ^ 0 - -
ORW	NN(rrx)	RR	5	20	5	4	dst<-dst OR src	- ^ ^ 0 - -
ORW	rr(rrx)	rr	3	18	3	4	dst<-dst OR src	- ^ ^ 0 - -
ORW	-(rr)	rr	3	18	3	4	rr<-rr-2 dst<-dst OR src	- ^ ^ 0 - -
ORW	-(rr)	RR	3	18	3	4	rr<-rr-2 dst<-dst OR src	- ^ ^ 0 - -
ORW	rr	#NN	4	10	4	0	dst<-dst OR src	- ^ ^ 0 - -
ORW	RR	#NN	4	10	4	0	dst<-dst OR src	- ^ ^ 0 - -
ORW	(rr)	#NN	4	18	4	4	dst<-dst OR src	- ^ ^ 0 - -
ORW	NN	#NN	6	22	6	4	dst<-dst OR src	- ^ ^ 0 - -
ORW	N(rrx)	#NN	5	20	5	4	dst<-dst OR src	- ^ ^ 0 - -
ORW	NN(rrx)	#NN	6	22	6	4	dst<-dst OR src	- ^ ^ 0 - -
ORW	(rr)	(rr)	2	20	2	6	dst<-dst OR src	- ^ ^ 0 - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
PEA : Push effective address on system stack								
PEA		N(rrx)	4	20	4	2/0(1)	SSP<-SSP-2, (SSP)<-rrx+N	- - - - - -
PEA		NN(rrx)	5	22	5	2/0(1)	SSP<-SSP-2, (SSP)<-rrx+NN	- - - - - -
PEA		N(RRx)	4	20	4	2/0(1)	SSP<-SSP-2, (SSP)<-RRx+N	- - - - - -
PEA		NN(RRx)	5	22	5	2/0(1)	SSP<-SSP-2, (SSP)<-RRx+NN	- - - - - -
PEAU : Push effective address on user stack								
PEAU		N(rrx)	4	20	4	2/0(1)	USP<-USP-2, (USP)<-rrx+N	- - - - - -
PEAU		NN(rrx)	5	22	5	2/0(1)	USP<-USP-2, (USP)<-rrx+NN	- - - - - -
PEAU		N(RRx)	4	20	4	2/0(1)	USP<-USP-2, (USP)<-RRx+N	- - - - - -
PEAU		NN(RRx)	5	22	5	2/0(1)	USP<-USP-2, (USP)<-RRx+NN	- - - - - -
POP : Pop system stack								
POP	r		2	8	2	1/0(1/0)	dst<-(SSP), SSP<-SSP+1	- - - - - -
POP	R		2	8	2	1/0(1/0)	dst<-(SSP), SSP<-SSP+1	- - - - - -
POP	(r)		2	8	2	1/0(1/0)	dst<-(SSP), SSP<-SSP+1	- - - - - -
POP	(R)		2	8	2	1/0(1/0	dst<-(SSP), SSP<-SSP+1	- - - - - -
POPU : Pop user stack								
POPU	r		2	8	2	1/0(1/0)	dst<-(USP), USP<-USP+1	- - - - - -
POPU	R		2	8	2	1/0(1/0)	dst<-(USP), USP<-USP+1	- - - - - -
POPU	(r)		2	8	2	1/0(1/0)	dst<-(USP), USP<-USP+1	- - - - - -
POPU	(R)		2	8	2	1/0(1/0)	dst<-(USP), USP<-USP+1	- - - - - -
POPUW : Pop word from user stack								
POPUW	rr		2	10	2	2/0(1/0)	dst<-(USP), USP<-USP+2	- - - - - -
POPUW	RR		2	10	2	2/0(1/0)	dst<-(USP), USP<-USP+2	- - - - - -
POPW : Pop word from system stack								
POPW	rr		2	10	2	2/0(1/0)	dst<-(SSP), SSP<-SSP+2	- - - - - -
POPW	RR		2	10	2	2/0(1/0)	dst<-(SSP), SSP<-SSP+2	- - - - - -
PUSH : Push system stack								
PUSH		r	2	8	2	1/0(1/0)	SSP<-SSP-1, (SSP)<-src	- - - - - -
PUSH		R	2	8	2	1/0(1/0)	SSP<-SSP-1, (SSP)<-src	- - - - - -
PUSH		(r)	2	8	2	1/0(1/0)	SSP<-SSP-1, (SSP)<-src	- - - - - -
PUSH		(R)	2	8	2	1/0(1/0)	SSP<-SSP-1, (SSP)<-src	- - - - - -
PUSH		#N	3	12	3	1/0(1/0)	SSP<-SSP-1, (SSP)<-src	- - - - - -
PUSHU : Push user stack								
PUSHU		r	2	8	2	1/0(1/0)	USP<-USP-1, (USP)<-src	- - - - - -
PUSHU		R	2	8	2	1/0(1/0)	USP<-USP-1, (USP)<-src	- - - - - -
PUSHU		(r)	2	8	2	1/0(1/0)	USP<-USP-1, (USP)<-src	- - - - - -
PUSHU		(R)	2	8	2	1/0(1/0)	USP<-USP-1, (USP)<-src	- - - - - -
PUSHU		#N	3	12	3	1/0(1/0)	USP<-USP-1, (USP)<-src	- - - - - -
PUSHUW : Push word on user stack								
PUSHUW		rr	2	8/10	3/2(2)	2/0(1)	USP<-USP-2, (USP)<-src	- - - - - -
PUSHUW		RR	2	8/10	3/2(2)	2/0(1)	USP<-USP-2, (USP)<-src	- - - - - -
PUSHUW		#NN	4	16	4	2/0(1)	USP<-USP-2, (USP)<-src	- - - - - -
PUSHW : Push Word on System Stack								
PUSHW		rr	2	8/10	3/2(2)	2/0(1)	SSP<-SSP-2, (SSP)<-src	- - - - - -
PUSHW		RR	2	8/10	3/2(2)	2/0(1)	SSP<-SSP-2, (SSP)<-src	- - - - - -
PUSHW		#NN	4	16	4	2/0(1)	SSP<-SSP-2, (SSP)<-src	- - - - - -

Note 1. No data memory accesses are done if the stack is kept in the Register File.
Note 2. Additional fetch when the stack is kept in Memory.

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
RCF : Reset carry flag								
RCF			1	4	1	0	C <- 0	0 - - - - -
RET : Return from subroutine								
RET			1	8/10	2	2(1)	PC <- (SSP), SSP <- SSP+2	- - - - - -
RETS : Return from subroutine in another segment								
RETS			2	12/14	2	3(1)	CSR<-(SSP), PC <- (SSP+1), SSP <- SSP+3	- - - - - -
RLC : Rotate left through carry								
RLC	r		2	4	2	0	dst(0)<-C, C<-dst(7) dst(n+1)<-dst(n) n=0-6	^ ^ ^ ^ - -
RLC	R		2	4	2	0	“ ”	^ ^ ^ ^ - -
RLC	(r)		2	6	2	0	“ ”	^ ^ ^ ^ - -
RLC	(R)		2	6	2	0	“ ”	^ ^ ^ ^ - -
RLCW : Rotate word left through carry								
RLCW	rr		2	8	2	0	dst(0)<-C, C<-dst(15) dst(n+1)<-dst(n) n=0-14	^ ? ^ ^ - -
RLCW	RR		2	8	2	0	“ ”	^ ? ^ ^ - -
ROL : Rotate left								
ROL	r		2	4	2	0	C<-dst(7), dst(0)<-dst(7) dst(n+1)<-dst(n) n=0-6	^ ^ ^ ^ - -
ROL	R		2	4	2	0	“ ”	^ ^ ^ ^ - -
ROL	(r)		2	6	2	0	“ ”	^ ^ ^ ^ - -
ROL	(R)		2	6	2	0	“ ”	^ ^ ^ ^ - -
ROR : Rotate right								
ROR	r		2	4	2	0	C<-dst(0), dst(7)<-dst(0) dst(n)<-dst(n+1) n=0-6	^ ^ ^ ^ - -
ROR	R		2	4	2	0	“ ”	^ ^ ^ ^ - -
ROR	(r)		2	6	2	0	“ ”	^ ^ ^ ^ - -
ROR	(R)		2	6	2	0	“ ”	^ ^ ^ ^ - -
RRC : Rotate right through carry								
RRC	r		2	4	2	0	dst(7)<-C, C<-dst(0) dst(n)<-dst(n+1) n=0-6	^ ^ ^ ^ - -
RRC	R		2	4	2	0	“ ”	^ ^ ^ ^ - -
RRC	(r)		2	6	2	0	“ ”	^ ^ ^ ^ - -
RRC	(R)		2	6	2	0	“ ”	^ ^ ^ ^ - -
RRCW : Rotate word right through carry								
RRCW	rr		2	8	2	0	dst(15)<-C, C<-dst(0) dst(n)<-dst(n+1) n=0-14	^ ^ ^ ^ - -
RRCW	RR		2	8	2	0	“ ”	^ ^ ^ ^ - -

Note 1. No data memory accesses are done if the stack is kept in the Register File.

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
SBC : Subtraction of 2 bytes with carry								
SBC	r	r	2	4	2	0	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	R	R	3	6	3	0	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	r	R	3	6	3	0	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	R	r	3	6	3	0	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	r	(r)	2	6	2	0	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	R	(r)	3	6	3	0	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	r	(rr)	3	8	3	1	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	R	(rr)	3	8	3	1	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	r	NN	4	10	4	1	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	r	N(rrx)	4	12	4	1	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	R	N(rrx)	4	12	4	1	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	r	NN(rrx)	5	14	5	1	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	R	NN(rrx)	5	14	5	1	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	r	rr(rrx)	3	12	3	1	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	r	(rr)+	3	12	3	1	dst<-dst-src-C rr<-rr+1	^ ^ ^ ^ 1 ^
SBC	R	(rr)+	3	12	3	1	dst<-dst-src-C rr<-rr+1	^ ^ ^ ^ 1 ^
SBC	r	-(rr)	3	12	3	1	rr<-rr-1 dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	R	-(rr)	3	12	3	1	rr<-rr-1 dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	(r)	r	3	6	3	0	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	(r)	R	3	6	3	0	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	(rr)	r	3	12	3	2	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	(rr)	R	3	12	3	2	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	(rr)+	r	3	14	3	2	dst<-dst-src-C rr<-rr+1	^ ^ ^ ^ 1 ^
SBC	(rr)+	R	3	14	3	2	dst<-dst-src-C rr<-rr+1	^ ^ ^ ^ 1 ^
SBC	NN	r	4	12	4	2	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	N(rrx)	r	4	14	4	2	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	N(rrx)	R	4	14	4	2	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	NN(rrx)	r	5	16	5	2	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	NN(rrx)	R	5	16	5	2	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	rr(rrx)	r	3	14	3	2	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	-(rr)	r	3	14	3	2	rr<-rr-1 dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	-(rr)	R	3	14	3	2	rr<-rr-1 dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	r	#N	3	6	3	0	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	R	#N	3	6	3	0	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	(rr)	#N	3	10	3	2	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	NN	#N	5	16	5	2	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	(rr)	(rr)	3	14	3	3	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	(RR)	(rr)	3	14	3	3	dst<-dst-src-C	^ ^ ^ ^ 1 ^

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
SBCW : Subtraction of word with carry								
SBCW	rr	rr	2	8	2	0	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	RR	RR	3	8	3	0	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	rr	RR	3	8	3	0	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	RR	rr	3	8	3	0	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	rr	(r)	3	10	3	0	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	RR	(r)	3	10	3	0	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	rr	(rr)	2	12	2	2	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	RR	(rr)	3	12	3	2	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	rr	NN	4	14	4	2	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	rr	N(rrx)	4	14	4	2	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	RR	N(rrx)	4	14	4	2	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	rr	NN(rrx)	5	16	5	2	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	RR	NN(rrx)	5	16	5	2	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	rr	rr(rrx)	3	14	3	2	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	rr	(rr)+	3	14	3	2	dst<-dst-src-C rr<-rr+2	^ ^ ^ ^ ? ?
SBCW	RR	(rr)+	3	14	3	2	dst<-dst+src+C rr<-rr+2	^ ^ ^ ^ ? ?
SBCW	rr	-(rr)	3	14	3	2	rr<-rr-2 dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	RR	-(rr)	3	14	3	2	rr<-rr-2 dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	(r)	rr	3	10	3	0	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	(r)	RR	3	10	3	0	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	(rr)	rr	2	16	2	4	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	(rr)	RR	3	18	3	4	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	(rr)+	rr	3	18	3	4	dst<-dst-src-C rr<-rr+2	^ ^ ^ ^ ? ?
SBCW	(rr)+	RR	3	18	3	4	dst<-dst-src-C rr<-rr+2	^ ^ ^ ^ ? ?
SBCW	NN	rr	4	18	4	4	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	N(rrx)	rr	4	18	4	4	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	N(rrx)	RR	4	18	4	4	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	NN(rrx)	rr	5	20	5	4	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	NN(rrx)	RR	5	20	5	4	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	rr(rrx)	rr	3	18	3	4	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	-(rr)	rr	3	18	3	4	rr<-rr-2 dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	-(rr)	RR	3	18	3	4	rr<-rr-2 dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	rr	#NN	4	10	4	0	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	RR	#NN	4	10	4	0	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	(rr)	#NN	4	18	4	4	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	NN	#NN	6	22	6	4	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	N(rrx)	#NN	5	20	5	4	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	NN(rrx)	#NN	6	22	6	4	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	(rr)	(rr)	2	20	2	6	dst<-dst-src-C	^ ^ ^ ^ ? ?

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
SCF : Set carry flag								
SCF			1	4	1	0	C <- 1	1 - - - - -
SDM : Set data memory								
SDM			1	4	1	0	Set Data Memory DP<-1 Note 1	- - - - - -
SLA : Shift left arithmetic								
SLA	r R (rr)		2 3 3	4 6 14	2 3 3	0 0 3	dst C<-dst(7), dst (0)<-0 dst(n+1)<-dst(n)n=0-6 “ ” “ ”	^ ^ ^ ^ 0 ^ ^ ^ ^ ^ 0 ^ ^ ^ ^ ^ 0 ^
SLAW : Shift word left arithmetic								
SLAW	rr RR (rr)		2 3 2	8 8 20	2 3 2	0 0 6	C<-dst(15), dst (0)<-0 dst(n+1)<-dst(n)n=1-14 “ ” “ ”	^ ^ ^ ^ ? ? ^ ^ ^ ^ ? ? ^ ^ ^ ^ ? ?
SPM : Set program memory								
SPM			1	4	1	0	Set Program Memory DP<-0 Note 2	- - - - - -
SPP : Set page pointer								
SPP		#N	2	4	2	0	Set Page Pointer	- - - - - -
SRA : Shift right arithmetic								
SRA SRA SRA SRA	r R (r) (R)		2 2 2 2	4 4 6 6	2 2 2 2	0 0 0 0	dst(7)<-dst(7), C<-dst(0) dst(n)<-dst(n+1)n=0-6 “ ” “ ” “ ”	^ ^ ^ 0 - - ^ ^ ^ 0 - - ^ ^ ^ 0 - - ^ ^ ^ 0 - -^
SRAW : Shift word right arithmetic								
SRAW SRAW	rr RR		2 2	8 8	2 2	0 0	dst(15)<-dst(15), C<-dst(0) dst(n)<-dst(n+1)n=0-14 “ ”	^ ?^ 0 - - ^ ?^ 0 - -

Note 1. Following this instruction, all addressing modes referring to address spaces will use DPRx registers.

Note 2. Following this instruction, all addressing modes referring to address spaces will use CSR register, except for the following instructions which operate with DPRx registers independently of the setting of the DP flag :CALLS, RETS, LINK/LINKU, UNLINK/UNLINKU, PUSH(W)/PUSHU(W), POP(W)/POPU(W), PEA/PEAU, and CALL, RET, IRET and interrupt execution (assuming the Stack Pointers are not pointing to the Register File).

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
SRP : Set register pointer								
SRP		#N	2	4	2	0	Set Register Pointer	- - - - - -
SRP0 : Set register pointer 0								
SRP0		#N	2	4	2	0	Set Register Pointer 0	- - - - - -
SRP1 : Set register pointer 1								
SRP1		#N	2	4	2	0	Set Register Pointer 1	- - - - - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
SUB : Subtraction of 2 bytes without carry								
SUB	r	r	2	4	2	0	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	R	R	3	6	3	0	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	r	R	3	6	3	0	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	R	r	3	6	3	0	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	r	(r)	2	6	2	0	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	R	(r)	3	6	3	0	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	r	(rr)	3	8	3	1	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	R	(rr)	3	8	3	1	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	r	NN	4	10	4	1	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	r	N(rrx)	4	12	4	1	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	R	N(rrx)	4	12	4	1	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	r	NN(rrx)	5	14	5	1	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	R	NN(rrx)	5	14	5	1	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	r	rr(rrx)	3	12	3	1	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	r	(rr)+	3	12	3	1	dst<-dst-src rr<-rr+1	^ ^ ^ ^ 1 ^
SUB	R	(rr)+	3	12	3	1	dst<-dst-src rr<-rr+1	^ ^ ^ ^ 1 ^
SUB	r	-(rr)	3	12	3	1	rr<-rr-1 dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	R	-(rr)	3	12	3	1	rr<-rr-1 dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	(r)	r	3	6	3	0	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	(r)	R	3	6	3	0	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	(rr)	r	3	12	3	2	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	(rr)	R	3	12	3	2	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	(rr)+	r	3	14	3	2	dst<-dst-src rr<-rr+1	^ ^ ^ ^ 1 ^
SUB	(rr)+	R	3	14	3	2	dst<-dst-src rr<-rr+1	^ ^ ^ ^ 1 ^
SUB	NN	r	4	12	4	2	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	N(rrx)	r	4	14	4	2	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	N(rrx)	R	4	14	4	2	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	NN(rrx)	r	5	16	5	2	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	NN(rrx)	R	5	16	5	2	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	rr(rrx)	r	3	14	3	2	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	-(rr)	r	3	14	3	2	rr<-rr-1 dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	-(rr)	R	3	14	3	2	rr<-rr-1 dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	r	#N	3	6	3	0	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	R	#N	3	6	3	0	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	(rr)	#N	3	10	3	2	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	NN	#N	5	16	5	2	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	(rr)	(rr)	3	14	3	3	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	(RR)	(rr)	3	14	3	3	dst<-dst-src	^ ^ ^ ^ 1 ^

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
SUBW : Subtraction of words								
SUBW	rr	rr	2	8	2	0	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	RR	RR	3	8	3	0	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	rr	RR	3	8	3	0	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	RR	rr	3	8	3	0	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	rr	(r)	3	10	3	0	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	RR	(r)	3	10	3	0	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	rr	(rr)	2	12	2	2	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	RR	(rr)	3	12	3	2	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	rr	NN	4	14	4	2	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	rr	N(rrx)	4	14	4	2	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	RR	N(rrx)	4	14	4	2	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	rr	NN(rrx)	5	16	5	2	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	RR	NN(rrx)	5	16	5	2	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	rr	rr(rrx)	3	14	3	2	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	rr	(rr)+	3	14	3	2	dst<-dst-src rr<-rr+2	^ ^ ^ ^ ? ?
SUBW	RR	(rr)+	3	14	3	2	dst<-dst-src rr<-rr+2	^ ^ ^ ^ ? ?
SUBW	rr	-(rr)	3	14	3	2	rr<-rr-2 dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	RR	-(rr)	3	14	3	2	rr<-rr-2 dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	(r)	rr	3	10	3	0	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	(r)	RR	3	10	3	0	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	(rr)	rr	2	16	2	4	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	(rr)	RR	3	18	3	4	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	(rr)+	rr	3	18	3	4	dst<-dst-src rr<-rr+2	^ ^ ^ ^ ? ?
SUBW	(rr)+	RR	3	18	3	4	dst<-dst-src rr<-rr+2	^ ^ ^ ^ ? ?
SUBW	NN	rr	4	18	4	4	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	N(rrx)	rr	4	18	4	4	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	N(rrx)	RR	4	18	4	4	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	NN(rrx)	rr	5	20	5	4	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	NN(rrx)	RR	5	20	5	4	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	rr(rrx)	rr	3	18	3	4	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	-(rr)	rr	3	18	3	4	rr<-rr-2 dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	-(rr)	RR	3	18	3	4	rr<-rr-2 dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	rr	#NN	4	10	4	0	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	RR	#NN	4	10	4	0	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	(rr)	#NN	4	18	4	4	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	NN	#NN	6	22	6	4	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	N(rrx)	#NN	5	20	5	4	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	NN(rrx)	#NN	6	22	6	4	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	(rr)	(rr)	2	20	2	6	dst<-dst-src	^ ^ ^ ^ ? ?
SWAP : Swap nibbles								
SWAP	r		2	8	2	0	dst(0-3)<--->dst(4-7)	? ^ ^ ? - -
SWAP	R		2	8	2	0	dst(0-3)<--->dst(4-7)	? ^ ^ ? - -
SWAP	(r)		2	8	2	0	dst(0-3)<--->dst(4-7)	? ^ ^ ? - -
SWAP	(R)		2	8	2	0	dst(0-3)<--->dst(4-7)	? ^ ^ ? - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
TCM : Test and complement byte under mask								
TCM	r	r	2	4	2	0	NOT dst AND src	- ^ ^ 0 - -
TCM	R	R	3	6	3	0	NOT dst AND src	- ^ ^ 0 - -
TCM	r	R	3	6	3	0	NOT dst AND src	- ^ ^ 0 - -
TCM	R	r	3	6	3	0	NOT dst AND src	- ^ ^ 0 - -
TCM	r	(r)	2	6	2	0	NOT dst AND src	- ^ ^ 0 - -
TCM	R	(r)	3	6	3	0	NOT dst AND src	- ^ ^ 0 - -
TCM	r	(rr)	3	8	3	1	NOT dst AND src	- ^ ^ 0 - -
TCM	R	(rr)	3	8	3	1	NOT dst AND src	- ^ ^ 0 - -
TCM	r	NN	4	10	4	1	NOT dst AND src	- ^ ^ 0 - -
TCM	r	N(rrx)	4	12	4	1	NOT dst AND src	- ^ ^ 0 - -
TCM	R	N(rrx)	4	12	4	1	NOT dst AND src	- ^ ^ 0 - -
TCM	r	NN(rrx)	5	14	5	1	NOT dst AND src	- ^ ^ 0 - -
TCM	R	NN(rrx)	5	14	5	1	NOT dst AND src	- ^ ^ 0 - -
TCM	r	rr(rrx)	3	12	3	1	NOT dst AND src	- ^ ^ 0 - -
TCM	r	(rr)+	3	12	3	1	NOT dst AND src rr<-rr+1	- ^ ^ 0 - -
TCM	R	(rr)+	3	12	3	1	NOT dst AND src rr<-rr+1	- ^ ^ 0 - -
TCM	r	-(rr)	3	12	3	1	rr<-rr-1 NOT dst AND src	- ^ ^ 0 - -
TCM	R	-(rr)	3	12	3	1	rr<-rr-1 NOT dst AND src	- ^ ^ 0 - -
TCM	(r)	r	3	6	3	0	NOT dst AND src	- ^ ^ 0 - -
TCM	(r)	R	3	6	3	0	NOT dst AND src	- ^ ^ 0 - -
TCM	(rr)	r	3	10	3	1	NOT dst AND src	- ^ ^ 0 - -
TCM	(rr)	R	3	10	3	1	NOT dst AND src	- ^ ^ 0 - -
TCM	(rr)+	r	3	12	3	1	NOT dst AND src rr<-rr+1	- ^ ^ 0 - -
TCM	(rr)+	R	3	12	3	1	dst<-dst AND src rr<-rr+1	- ^ ^ 0 - -
TCM	NN	r	4	10	4	1	NOT dst AND src	- ^ ^ 0 - -
TCM	N(rrx)	r	4	12	4	1	NOT dst AND src	- ^ ^ 0 - -
TCM	N(rrx)	R	4	12	4	1	NOT dst AND src	- ^ ^ 0 - -
TCM	NN(rrx)	r	5	14	5	1	NOT dst AND src	- ^ ^ 0 - -
TCM	NN(rrx)	R	5	14	5	1	NOT dst AND src	- ^ ^ 0 - -
TCM	rr(rrx)	r	3	12	3	1	NOT dst AND src	- ^ ^ 0 - -
TCM	-(rr)	r	3	12	3	1	rr<-rr-1 NOT dst AND src	- ^ ^ 0 - -
TCM	-(rr)	R	3	12	3	1	rr<-rr-1 NOT dst AND src	- ^ ^ 0 - -
TCM	r	#N	3	6	3	0	NOT dst AND src	- ^ ^ 0 - -
TCM	R	#N	3	6	3	0	NOT dst AND src	- ^ ^ 0 - -
TCM	(rr)	#N	3	8	3	1	NOT dst AND src	- ^ ^ 0 - -
TCM	NN	#N	5	14	5	1	NOT dst AND src	- ^ ^ 0 - -
TCM	(rr)	(rr)	3	12	3	2	NOT dst AND src	- ^ ^ 0 - -
TCM	(RR)	(rr)	3	12	3	2	NOT dst AND src	- ^ ^ 0 - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
TCMW : Test and complement word under mask								
TCMW	rr	rr	2	8	2	0	NOT dst AND src	- ^ ^ 0 - -
TCMW	RR	RR	3	8	3	0	NOT dst AND src	- ^ ^ 0 - -
TCMW	rr	RR	3	8	3	0	NOT dst AND src	- ^ ^ 0 - -
TCMW	RR	rr	3	8	3	0	NOT dst AND src	- ^ ^ 0 - -
TCMW	rr	(r)	3	10	3	0	NOT dst AND src	- ^ ^ 0 - -
TCMW	RR	(r)	3	10	3	0	NOT dst AND src	- ^ ^ 0 - -
TCMW	rr	(rr)	2	12	2	2	NOT dst AND src	- ^ ^ 0 - -
TCMW	RR	(rr)	3	14	3	2	NOT dst AND src	- ^ ^ 0 - -
TCMW	rr	NN	4	14	4	2	NOT dst AND src	- ^ ^ 0 - -
TCMW	rr	N(rrx)	4	14	4	2	NOT dst AND src	- ^ ^ 0 - -
TCMW	RR	N(rrx)	4	14	4	2	NOT dst AND src	- ^ ^ 0 - -
TCMW	rr	NN(rrx)	5	16	5	2	NOT dst AND src	- ^ ^ 0 - -
TCMW	RR	NN(rrx)	5	16	5	2	NOT dst AND src	- ^ ^ 0 - -
TCMW	rr	rr(rrx)	3	14	3	2	NOT dst AND src	- ^ ^ 0 - -
TCMW	rr	(rr)+	3	14	3	2	NOT dst AND src rr<-rr+2	- ^ ^ 0 - -
TCMW	RR	(rr)+	3	14	3	2	NOT dst AND src rr<-rr+2	- ^ ^ 0 - -
TCMW	rr	-(rr)	3	14	3	2	rr<-rr-2 NOT dst AND src	- ^ ^ 0 - -
TCMW	RR	-(rr)	3	14	3	2	rr<-rr-2 NOT dst AND src	- ^ ^ 0 - -
TCMW	(r)	rr	3	10	3	0	NOT dst AND src	- ^ ^ 0 - -
TCMW	(r)	RR	3	10	3	0	NOT dst AND src	- ^ ^ 0 - -
TCMW	(rr)	rr	2	14	2	2	NOT dst AND src	- ^ ^ 0 - -
TCMW	(rr)	RR	3	14	3	2	NOT dst AND src	- ^ ^ 0 - -
TCMW	(rr)+	rr	3	14	3	2	NOT dst AND src rr<-rr+2	- ^ ^ 0 - -
TCMW	(rr)+	RR	3	14	3	2	NOT dst AND src rr<-rr+2	- ^ ^ 0 - -
TCMW	NN	rr	4	16	4	2	NOT dst AND src	- ^ ^ 0 - -
TCMW	N(rrx)	rr	4	14	4	2	NOT dst AND src	- ^ ^ 0 - -
TCMW	N(rrx)	RR	4	14	4	2	NOT dst AND src	- ^ ^ 0 - -
TCMW	NN(rrx)	rr	5	16	5	2	NOT dst AND src	- ^ ^ 0 - -
TCMW	NN(rrx)	RR	5	16	5	2	NOT dst AND src	- ^ ^ 0 - -
TCMW	rr(rrx)	rr	3	14	3	2	NOT dst AND src	- ^ ^ 0 - -
TCMW	-(rr)	rr	3	14	3	2	rr<-rr-2 NOT dst AND src	- ^ ^ 0 - -
TCMW	-(rr)	RR	3	14	3	2	rr<-rr-2 NOT dst AND src	- ^ ^ 0 - -
TCMW	rr	#NN	4	10	4	0	NOT dst AND src	- ^ ^ 0 - -
TCMW	RR	#NN	4	10	4	0	NOT dst AND src	- ^ ^ 0 - -
TCMW	(rr)	#NN	4	14	4	2	NOT dst AND src	- ^ ^ 0 - -
TCMW	NN	#NN	6	20	6	2	NOT dst AND src	- ^ ^ 0 - -
TCMW	N(rrx)	#NN	5	16	5	2	NOT dst AND src	- ^ ^ 0 - -
TCMW	NN(rrx)	#NN	6	18	6	2	NOT dst AND src	- ^ ^ 0 - -
TCMW	(rr)	(rr)	2	16	2	4	NOT dst AND src	- ^ ^ 0 - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
TM : Test byte under mask								
TM	r	r	2	4	2	0	dst AND src	- ^ ^ 0 - -
TM	R	R	3	6	3	0	dst AND src	- ^ ^ 0 - -
TM	r	R	3	6	3	0	dst AND src	- ^ ^ 0 - -
TM	R	r	3	6	3	0	dst AND src	- ^ ^ 0 - -
TM	r	(r)	2	6	2	0	dst AND src	- ^ ^ 0 - -
TM	R	(r)	3	6	3	0	dst AND src	- ^ ^ 0 - -
TM	r	(rr)	3	8	3	1	dst AND src	- ^ ^ 0 - -
TM	R	(rr)	3	8	3	1	dst AND src	- ^ ^ 0 - -
TM	r	NN	4	10	4	1	dst AND src	- ^ ^ 0 - -
TM	r	N(rrx)	4	12	4	1	dst AND src	- ^ ^ 0 - -
TM	R	N(rrx)	4	12	4	1	dst AND src	- ^ ^ 0 - -
TM	r	NN(rrx)	5	14	5	1	dst AND src	- ^ ^ 0 - -
TM	R	NN(rrx)	5	14	5	1	dst AND src	- ^ ^ 0 - -
TM	r	rr(rrx)	3	12	3	1	dst AND src	- ^ ^ 0 - -
TM	r	(rr)+	3	12	3	1	dst AND src rr<-rr+1	- ^ ^ 0 - -
TM	R	(rr)+	3	12	3	1	dst AND -src rr<-rr+1	- ^ ^ 0 - -
TM	r	-(rr)	3	12	3	1	rr<-rr-1 dst AND src	- ^ ^ 0 - -
TM	R	-(rr)	3	12	3	1	rr<-rr-1 dst AND src	- ^ ^ 0 - -
TM	(r)	r	3	6	3	0	dst AND src	- ^ ^ 0 - -
TM	(r)	R	3	6	3	0	dst AND src	- ^ ^ 0 - -
TM	(rr)	r	3	10	3	1	dst AND src	- ^ ^ 0 - -
TM	(rr)	R	3	10	3	1	dst AND src	- ^ ^ 0 - -
TM	(rr)+	r	3	12	3	1 1	dst AND src rr<-rr+1	- ^ ^ 0 - -
TM	(rr)+	R	3	12	3	1 1	dst AND src rr<-rr+1	- ^ ^ 0 - -
TM	NN	r	4	10	4	1	dst AND src	- ^ ^ 0 - -
TM	N(rrx)	r	4	12	4	1	dst AND src	- ^ ^ 0 - -
TM	N(rrx)	R	4	12	4	1	dst AND src	- ^ ^ 0 - -
TM	NN(rrx)	r	5	14	5	1	dst AND src	- ^ ^ 0 - -
TM	NN(rrx)	R	5	14	5	1	dst AND src	- ^ ^ 0 - -
TM	rr(rrx)	r	3	12	3	1	dst AND src	- ^ ^ 0 - -
TM	-(rr)	r	3	12	3	1	rr<-rr-1 dst AND src	- ^ ^ 0 - -
TM	-(rr)	R	3	12	3	1	rr<-rr-1 dst AND src	- ^ ^ 0 - -
TM	r	#N	3	6	3	0	dst AND src	- ^ ^ 0 - -
TM	R	#N	3	6	3	0	dst AND src	- ^ ^ 0 - -
TM	(rr)	#N	3	8	3	1	dst AND src	- ^ ^ 0 - -
TM	NN	#N	5	14	5	1	dst AND src	- ^ ^ 0 - -
TM	(rr)	(rr)	3	12	3	2	dst AND src	- ^ ^ 0 - -
TM	(RR)	(rr)	3	12	3	2	dst AND src	- ^ ^ 0 - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
TMW : Test word under mask								
TMW	rr	rr	2	8	2	0	dst AND src	- ^ ^ 0 - -
TMW	RR	RR	3	8	3	0	dst AND src	- ^ ^ 0 - -
TMW	rr	RR	3	8	3	0	dst AND src	- ^ ^ 0 - -
TMW	RR	rr	3	8	3	0	dst AND src	- ^ ^ 0 - -
TMW	rr	(r)	3	10	3	0	dst AND src	- ^ ^ 0 - -
TMW	RR	(r)	3	10	3	0	dst AND src	- ^ ^ 0 - -
TMW	rr	(rr)	2	12	2	2	dst AND src	- ^ ^ 0 - -
TMW	RR	(rr)	3	14	3	2	dst AND src	- ^ ^ 0 - -
TMW	rr	NN	4	14	4	2	dst AND src	- ^ ^ 0 - -
TMW	rr	N(rrx)	4	14	4	2	dst AND src	- ^ ^ 0 - -
TMW	RR	N(rrx)	4	14	4	2	dst AND src	- ^ ^ 0 - -
TMW	rr	NN(rrx)	5	16	5	2	dst AND src	- ^ ^ 0 - -
TMW	RR	NN(rrx)	5	16	5	2	dst AND src	- ^ ^ 0 - -
TMW	rr	rr(rrx)	3	14	3	2	dst AND src	- ^ ^ 0 - -
TMW	rr	(rr)+	3	14	3	2	dst AND src rr<-rr+2	- ^ ^ 0 - -
TMW	RR	(rr)+	3	14	3	2	dst AND src rr<-rr+2	- ^ ^ 0 - -
TMW	rr	-(rr)	3	14	3	2	rr<-rr-2 dst AND src	- ^ ^ 0 - -
TMW	RR	-(rr)	3	14	3	2	rr<-rr-2 dst AND src	- ^ ^ 0 - -
TMW	(r)	rr	3	10	3	0	dst AND src	- ^ ^ 0 - -
TMW	(r)	RR	3	10	3	0	dst AND src	- ^ ^ 0 - -
TMW	(rr)	rr	2	14	2	2	dst AND src	- ^ ^ 0 - -
TMW	(rr)	RR	3	14	3	2	dst AND src	- ^ ^ 0 - -
TMW	(rr)+	rr	3	14	3	2	dst AND src rr<-rr+2	- ^ ^ 0 - -
TMW	(rr)+	RR	3	14	3	2	dst AND src rr<-rr+2	- ^ ^ 0 - -
TMW	NN	rr	4	16	4	2	dst AND src	- ^ ^ 0 - -
TMW	N(rrx)	rr	4	14	4	2	dst AND src	- ^ ^ 0 - -
TMW	N(rrx)	RR	4	14	4	2	dst AND src	- ^ ^ 0 - -
TMW	NN(rrx)	rr	5	16	5	2	dst AND src	- ^ ^ 0 - -
TMW	NN(rrx)	RR	5	16	5	2	dst AND src	- ^ ^ 0 - -
TMW	rr(rrx)	rr	3	14	3	2	dst AND src	- ^ ^ 0 - -
TMW	-(rr)	rr	3	14	3	2	rr<-rr-2 dst AND src	- ^ ^ 0 - -
TMW	-(rr)	RR	3	14	3	2	rr<-rr-2 dst AND src	- ^ ^ 0 - -
TMW	rr	#NN	4	10	4	0	dst AND src	- ^ ^ 0 - -
TMW	RR	#NN	4	10	4	0	dst AND src	- ^ ^ 0 - -
TMW	(rr)	#NN	4	14	4	2	dst AND src	- ^ ^ 0 - -
TMW	NN	#NN	6	20	6	2	dst AND src	- ^ ^ 0 - -
TMW	N(rrx)	#NN	5	16	5	2	dst AND src	- ^ ^ 0 - -
TMW	NN(rrx)	#NN	6	18	6	2	dst AND src	- ^ ^ 0 - -
TMW	(rr)	(rr)	2	16	2	4	dst AND src	- ^ ^ 0 - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
UNLINK: remove stack frame from sysyem stack								
UNLINK	RR		2	6/10	2	2/0	SSP<-RR, RR<-(SSP), SSP<-SSP+1[2]	- - - - - -
UNLINK	rr		2	6/10	2	2/0	“ “	- - - - - -
UNLINKU: remove stack frame from user stack								
UNLINKU	RR		2	6/10	2	2/0	USP<-RR, RR<-(USP), USP<-USP+1[2]	- - - - - -
UNLINKU	rr		2	6/10	2	2/0	“ “	- - - - - -
WFI : Wait for Interrupt								
WFI			2	4+...	2	0	wait for interrupt	- - - - - -

Note. The value inside [ ] is valid for external memory stack

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
XCH : Exchange Register								
XCH	r	r	3	6	3	0	dst <-> src	- - - - - -
XCH	R	R	3	6	3	0	dst <-> src	- - - - - -
XCH	r	R	3	6	3	0	dst <-> src	- - - - - -
XCH	R	r	3	6	3	0	dst <-> src	- - - - - -
XOR : Logical exclusive OR								
XOR	r	r	2	4	2	0	dst<-dst XOR src	- ^ ^ 0 - -
XOR	R	R	3	6	3	0	dst<-dst XOR src	- ^ ^ 0 - -
XOR	r	R	3	6	3	0	dst<-dst XOR src	- ^ ^ 0 - -
XOR	R	r	3	6	3	0	dst<-dst XOR src	- ^ ^ 0 - -
XOR	r	(r)	2	6	2	0	dst<-dst XOR src	- ^ ^ 0 - -
XOR	R	(r)	3	6	3	0	dst<-dst XOR src	- ^ ^ 0 - -
XOR	r	(rr)	3	8	3	1	dst<-dst XOR src	- ^ ^ 0 - -
XOR	R	(rr)	3	8	3	1	dst<-dst XOR src	- ^ ^ 0 - -
XOR	r	NN	4	10	4	1	dst<-dst XOR src	- ^ ^ 0 - -
XOR	r	N(rrx)	4	12	4	1	dst<-dst XOR src	- ^ ^ 0 - -
XOR	R	N(rrx)	4	12	4	1	dst<-dst XOR src	- ^ ^ 0 - -
XOR	r	NN(rrx)	5	14	5	1	dst<-dst XOR src	- ^ ^ 0 - -
XOR	R	NN(rrx)	5	14	5	1	dst<-dst XOR src	- ^ ^ 0 - -
XOR	r	rr(rrx)	3	12	3	1	dst<-dst XOR src	- ^ ^ 0 - -
XOR	r	(rr)+	3	12	3	1	dst<-dst XOR src rr<-r+1	- ^ ^ 0 - -
XOR	R	(rr)+	3	12	3	1	dst<-dst XOR src rr<-rr+1	- ^ ^ 0 - -
XOR	r	-(rr)	3	12	3	1	rr<-rr-1 dst<-dst XOR src	- ^ ^ 0 - -
XOR	R	-(rr)	3	12	3	1	rr<-rr-1 dst<-dst XOR src	- ^ ^ 0 - -
XOR	(r)	r	3	6	3	0	dst<-dst XOR src	- ^ ^ 0 - -
XOR	(r)	R	3	6	3	0	dst<-dst XOR src	- ^ ^ 0 - -
XOR	(rr)	r	3	12	3	2	dst<-dst XOR src	- ^ ^ 0 - -
XOR	(rr)	R	3	12	3	2	dst<-dst XOR src	- ^ ^ 0 - -
XOR	(rr)+	r	3	14	3	2	dst<-dst XOR src rr<-rr+1	- ^ ^ 0 - -
XOR	(rr)+	R	3	14	3	2	dst<-dst XOR src rr<-rr+1	- ^ ^ 0 - -
XOR	NN	r	4	12	4	2	dst<-dst XOR src	- ^ ^ 0 - -
XOR	N(rrx)	r	4	14	4	2	dst<-dst XOR src	- ^ ^ 0 - -
XOR	N(rrx)	R	4	14	4	2	dst<-dst XOR src	- ^ ^ 0 - -
XOR	NN(rrx)	r	5	16	5	2	dst<-dst XOR src	- ^ ^ 0 - -
XOR	NN(rrx)	R	5	16	5	2	dst<-dst XOR src	- ^ ^ 0 - -
XOR	rr(rrx)	r	3	14	3	2	dst<-dst XOR src	- ^ ^ 0 - -
XOR	-(rr)	r	3	14	3	2	rr<-rr-1 dst<-dst XOR src	- ^ ^ 0 - -
XOR	-(rr)	R	3	14	3	2	rr<-rr-1 dst<-dst XOR src	- ^ ^ 0 - -
XOR	r	#N	3	6	3	0	dst<-dst XOR src	- ^ ^ 0 - -
XOR	R	#N	3	6	3	0	dst<-dst XOR src	- ^ ^ 0 - -
XOR	(rr)	#N	3	10	3	2	dst<-dst XOR src	- ^ ^ 0 - -
XOR	NN	#N	5	16	5	2	dst<-dst XOR src	- ^ ^ 0 - -
XOR	(rr)	(rr)	3	14	3	3	dst<-dst XOR src	- ^ ^ 0 - -
XOR	(RR)	(rr)	3	14	3	3	dst<-dst XOR src	- ^ ^ 0 - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	P	D	Operation	Flags C Z S V D H
XORW : Logical exclusive OR between words								
XORW	rr	rr	2	8	2	0	dst<-dst XOR src	- ^ ^ 0 - -
XORW	RR	RR	3	8	3	0	dst<-dst XOR src	- ^ ^ 0 - -
XORW	rr	RR	3	8	3	0	dst<-dst XOR src	- ^ ^ 0 - -
XORW	RR	rr	3	8	3	0	dst<-dst XOR src	- ^ ^ 0 - -
XORW	rr	(r)	3	10	3	0	dst<-dst XOR src	- ^ ^ 0 - -
XORW	RR	(r)	3	10	3	0	dst<-dst XOR src	- ^ ^ 0 - -
XORW	rr	(rr)	2	12	2	2	dst<-dst XOR src	- ^ ^ 0 - -
XORW	RR	(rr)	3	12	3	2	dst<-dst XOR src	- ^ ^ 0 - -
XORW	rr	NN	4	14	4	2	dst<-dst XOR src	- ^ ^ 0 - -
XORW	rr	N(rrx)	4	14	4	2	dst<-dst XOR src	- ^ ^ 0 - -
XORW	RR	N(rrx)	4	14	4	2	dst<-dst XOR src	- ^ ^ 0 - -
XORW	rr	NN(rrx)	5	16	5	2	dst<-dst XOR src	- ^ ^ 0 - -
XORW	RR	NN(rrx)	5	16	5	2	dst<-dst XOR src	- ^ ^ 0 - -
XORW	rr	rr(rrx)	3	14	3	2	dst<-dst XOR src	- ^ ^ 0 - -
XORW	rr	(rr)+	3	14	3	2	dst<-dst XOR src rr<-rr+2	- ^ ^ 0 - -
XORW	RR	(rr)+	3	14	3	2	dst<-dst XOR src rr<-rr+2	- ^ ^ 0 - -
XORW	rr	-(rr)	3	14	3	2	rr<-rr-2 dst<-dst XOR src	- ^ ^ 0 - -
XORW	RR	-(rr)	3	14	3	2	rr<-rr-2 dst<-dst XOR src	- ^ ^ 0 - -
XORW	(r)	rr	3	10	3	0	dst<-dst XOR src	- ^ ^ 0 - -
XORW	(r)	RR	3	10	3	0	dst<-dst XOR src	- ^ ^ 0 - -
XORW	(rr)	rr	2	16	2	4	dst<-dst XOR src	- ^ ^ 0 - -
XORW	(rr)	RR	3	18	3	4	dst<-dst XOR src	- ^ ^ 0 - -
XORW	(rr)+	rr	3	18	3	4	dst<-dst XOR src rr<-rr+2	- ^ ^ 0 - -
XORW	(rr)+	RR	3	18	3	4	dst<-dst XOR src rr<-rr+2	- ^ ^ 0 - -
XORW	NN	rr	4	18	4	4	dst<-dst XOR src	- ^ ^ 0 - -
XORW	N(rrx)	rr	4	18	4	4	dst<-dst XOR src	- ^ ^ 0 - -
XORW	N(rrx)	RR	4	18	4	4	dst<-dst XOR src	- ^ ^ 0 - -
XORW	NN(rrx)	rr	5	20	5	4	dst<-dst XOR src	- ^ ^ 0 - -
XORW	NN(rrx)	RR	5	20	5	4	dst<-dst XOR src	- ^ ^ 0 - -
XORW	rr(rrx)	rr	3	18	3	4	dst<-dst XOR src	- ^ ^ 0 - -
XORW	-(rr)	rr	3	18	3	4	rr<-rr-2 dst<-dst XOR src	- ^ ^ 0 - -
XORW	-(rr)	RR	3	18	3	4	rr<-rr-2 dst<-dst XOR src	- ^ ^ 0 - -
XORW	rr	#NN	4	10	4	0	dst<-dst XOR src	- ^ ^ 0 - -
XORW	RR	#NN	4	10	4	0	dst<-dst XOR src	- ^ ^ 0 - -
XORW	(rr)	#NN	4	18	4	4	dst<-dst XOR src	- ^ ^ 0 - -
XORW	NN	#NN	6	22	6	4	dst<-dst XOR src	- ^ ^ 0 - -
XORW	N(rrx)	#NN	5	20	5	4	dst<-dst XOR src	- ^ ^ 0 - -
XORW	NN(rrx)	#NN	6	22	6	4	dst<-dst XOR src	- ^ ^ 0 - -
XORW	(rr)	(rr)	2	20	2	6	dst<-dst XOR src	- ^ ^ 0 - -

2 OPCODE MAP

R, Rd, Rs: general registers (8 bits), if the first 4 bits are Dh then the last 4 bits specify a working register.

rr,rrd,rrs: working register pair (3 bits), a single bit (0 or 1) follows.

RR,RRd,RRs: general register pair (7 bits), if the first 4 bits are Dh then the last 3 bits specify a working register; a single bit (0 or 1) follows.

bd,bs: destination and source bit numbers (3 bits followed by a 0 or 1), associated to rd and rs for r.b bit addressing modes.

N: 8-bit quantity.

NN: 16-bit quantity (divided in Nh & Nl) nnnnnn: 6-bit segment number.

wwwww: 5-bit register window number (for SRP...) pppppp: 6-bit register page number (for SPP).

K: ALU parametrization, 4 bits (corresponding instruction is ALD or ALDW, value of K gives instruction to be substituted for "ALD":

K=	0	1	2	3	4	5	6	8	9	A	F
ALD=	OR	AND	SBC	ADC	ADD	SUB	XOR	CP	CP	TM	LD

Table 14. Opcode Map

OPCODE	BYTE1	BYTE2	BYTE3	BYTE4	BYTE5	OPCODE MNEMONIC	ADDRESSING MODE
00						EI	
01						SCF	
02	rd,rs					OR	rd,rs
03	rd,rs					OR	rd,(rs)
04	Rs	Rd				OR	Rd,Rs
05	Rd	N				OR	Rd,#N
06	K,rr1	Nd	Nhs	Nls		ALDW	Nd(rr),#NNs
06	K,rr0	Nhd	Nld	Nhs	Nls	ALDW	NNd(rr),#NNs
07	RRs0	RRd0				ORW	RRd,RRs
07	RRd1	Nhs	Nls			ORW	RRd,#NN
08	Rs					LD	r0,Rs
09	Rd					LD	Rd,r0
0A	N					DJNZ	r0,N
0B	N					JR	F,N
0C	N					LD	r0,#N
0D	Nh	Nl				JP	F,NN
0E	rrd0,rrs0					ORW	rrd,rrs
0E	rrd1,rrs0					ORW	(rrd),rrs
0E	rrd0,rrs1					ORW	rrd,(rrs)
0E	rrd1,rrs1					ORW	(rrd),(rrs)
0F	bd1,rd	bs0,rs				BOR	rd.b,rs.b
0F	bd1,rd	bs1,rs				BOR	rd.b,!rs.b
0F	bd0,rd					BSET	rd.b
10						DI	
11						RCF	
12	rd,rs					AND	rd,rs
13	rd,rs					AND	rd,(rs)
14	Rs	Rd				AND	Rd,Rs

OPCODE	BYTE1	BYTE2	BYTE3	BYTE4	BYTE5	OPCODE MNEMONIC	ADDRESSING MODE
15	Rd	N				AND	Rd,#N
16	Rs	Rd				XCH	Rs,Rd
17	RRs0	RRd0				ANDW	RRd,RRs
17	RRd1	Nhs	Nls			ANDW	RRd,#NN
18	Rs					LD	r1,Rs
19	Rd					LD	Rd,r1
1A	N					DJNZ	r1,N
1B	N					JR	LT,N
1C	N					LD	r1,#N
1D	Nh	Nl				JP	LT,NN
1E	rrd0,rrs0					ANDW	rrd,rrs
1E	rrd1,rrs0					ANDW	(rrd),rrs
1E	rrd0,rrs1					ANDW	rrd,(rrs)
1E	rrd1,rrs1					ANDW	(rrd),(rrs)
1F	bd1,rd	bs0,rs				BAND	rd.b,rs.b
1F	bd1,rd	bs1,rs				BAND	rd.b,rs.b
1F	bd0,rd					BRES	rd.b
20	Rd					POPU	Rd
21	Rd					POPU	(Rd)
22	rd,rs					SBC	rd,rs
23	rd,rs					SBC	rd,(rs)
24	Rs	Rd				SBC	Rd,Rs
25	Rd	N				SBC	Rd,#N
26	K,rr1	N	R			ALD	N(rr),R
26	K,rr0	N	N	R		ALD	NN(rr),R
27	RRs0	RRd0				SBCW	RRd,RRs
27	RRd1	Nhs	Nls			SBCW	RRd,#NN
28	Rs					LD	r2,Rs
29	Rd					LD	Rd,r2
2A	N					DJNZ	r2,N
2B	N					JR	LE,N
2C	N					LD	r2,#N
2D	Nh	Nl				JP	LE,NN
2E	rrd0,rrs0					SBCW	rrd,rrs
2E	rrd1,rrs0					SBCW	(rrd),rrs
2E	rrd0,rrs1					SBCW	rrd,(rrs)
2E	rrd1,rrs1					SBCW	(rrd),(rrs)
2F	RR0					SRAW	RR
2F	K,0001	Ns	Ndh	Ndl		ALD	NNd,#Ns
30	Rd					PUSHU	Rd
31	Rd					PUSHU	(Rd)
32	rd,rs					ADC	rd,rs
33	rd,rs					ADC	rd,(rs)
34	Rs	Rd				ADC	Rd,Rs
35	Rd	N				ADC	Rd,#N
36	RR0					RRCW	RR
36	K,0001	Nsh	Nsl	Ndh	Ndl	ALDW	NNd,#NNs

OPCODE	BYTE1	BYTE2	BYTE3	BYTE4	BYTE5	OPCODE MNEMONIC	ADDRESSING MODE
37	RRs0	RRd0				ADCW	RRd,RRs
37	RRd1	Nhs	Nls			ADCW	RRd,#NN
38	Rs					LD	r3,Rs
39	Rd					LD	Rd,r3
3A	N					DJNZ	r3,N
3B	N					JR	ULE,N
3C	N					LD	r3,#N
3D	Nh	Nl				JP	ULE,NN
3E	rrd0,rrs0					ADCW	rrd,rrs
3E	rrd1,rrs0					ADCW	(rrd),rrs
3E	rrd0,rrs1					ADCW	rrd,(rrs)
3E	rrd1,rrs1					ADCW	(rrd),(rrs)
3F	01nnnnnn	Nh	Nl			CALLS	nnnnnn,NN
3F	11nnnnnn	Nh	Nl			JPS	nnnnnn,NN
40	Rd					DEC	Rd
41	Rd					DEC	(Rd)
42	rd,rs					ADD	rd,rs
43	rd,rs					ADD	rd,(rs)
44	Rs	Rd				ADD	Rd,Rs
45	Rd	N				ADD	Rd,#N
46						RET	
47	RRs0	RRd0				ADDW	RRd,RRs
47	RRd1	Nhs	Nls			ADDW	RRd,#NN
48	Rs					LD	r4,Rs
49	Rd					LD	Rd,r4
4A	N					DJNZ	r4,N
4B	N					JR	OV,N
4C	N					LD	r4,#N
4D	Nh	Nl				JP	OV,NN
4E	rrd0,rrs0					ADDW	rrd,rrs
4E	rrd1,rrs0					ADDW	(rrd),rrs
4E	rrd0,rrs1					ADDW	rrd,(rrs)
4E	rrd1,rrs1					ADDW	(rrd),(rrs)
4F	rrd0,rs					MUL	rrd,rs
50	Rd					INC	Rd
51	Rd					INC	(Rd)
52	rd,rs					SUB	rd,rs
53	rd,rs					SUB	rd,(rs)
54	Rs	Rd				SUB	Rd,Rs
55	Rd	N				SUB	Rd,#N
56	RRs0	rrh0,rrl0				DIVWS	rrh,rrl,RRs
57	RRs0	RRd0				SUBW	RRd,RRs
57	RRd1	Nhs	Nls			SUBW	RRd,#NN
58	Rs					LD	r5,Rs
59	Rd					LD	Rd,r5
5A	N					DJNZ	r5,N
5B	N					JR	MI,N

OPCODE	BYTE1	BYTE2	BYTE3	BYTE4	BYTE5	OPCODE MNEMONIC	ADDRESSING MODE
5C	N					LD	r5,#N
5D	Nh	Nl				JP	MI,NN
5E	rrd0,rrs0					SUBW	rrd,rrs
5E	rrd1,rrs0					SUBW	(rrd),rrs
5E	rrd0,rrs1					SUBW	rrd,(rrs)
5E	rrd1,rrs1					SUBW	(rrd),(rrs)
5F	rrd0,rs					DIV	rrd,rs
60	rrs1,rrx0	K,rd				ALD	rd,rrs(rrx)
60	rrd1,rrx1	K,rs				ALD	rrd(rrx),rs
60	rrs0,rrx0	K,rrde				ALDW	rrd,rrs(rrx)
60	rrd0,rrx1	K,rrse				ALDW	rrd(rrx),rrs
61						CCF	
62	rd,rs					XOR	rd,rs
63	rd,rs					XOR	rd,(rs)
64	Rs	Rd				XOR	Rd,Rs
65	Rd	N				XOR	Rd,#N
66	Rs					PUSH	Rs
67	RRs0	RRd0				XORW	RRd,RRs
67	RRd1	Nhs	Nls			XORW	RRd,#NN
68	Rs					LD	r6,Rs
69	Rd					LD	Rd,r6
6A	N					DJNZ	r6,N
6B	N					JR	EQ,N
6C	N					LD	r6,#N
6D	Nh	Nl				JP	EQ,NN
6E	rrd0,rrs0					XORW	rrd,rrs
6E	rrd1,rrs0					XORW	(rrd),rrs
6E	rrd0,rrs1					XORW	rrd,(rrs)
6E	rrd1,rrs1					XORW	(rrd),(rrs)
6F	bd1,rd	bs0,rs				BXOR	rd.bd,rs.bs
6F	bd1,rd	bs1,rs				BXOR	rd.bd, rs.bs
6F	bd0,rd					BCPL	rd.bd
70	Rd					DA	Rd
71	Rd					DA	(Rd)
72	K,rrs1	Rd				ALD	Rd,(rrs)
72	K,rrd0	Rs				ALD	(rrd),Rs
73	K,rrs0	RRd0				ALD	(RRd),(rrs)
73	0100,rr1	R				CALLS	(R),(rr)
73	1100,rr1	R				JPS	(R),(rr)
74	RRd1					CALL	(RRd)
74	RRs0					PUSHW	RRs
75	RRd0					POPW	RRd
75	RRd1					UNLINK	RRd
76	Rd					POP	Rd
77	Rd					POP	(Rd)
78	Rs					LD	r7,Rs
79	Rd					LD	Rd,r7

OPCODE	BYTE1	BYTE2	BYTE3	BYTE4	BYTE5	OPCODE MNEMONIC	ADDRESSING MODE
7A	N					DJNZ	r7,N
7B	N					JR	UL,N
7C	N					LD	r7,#N
7D	Nh	Nl				JP	UL,NN
7E	K,rrs0	RRd0				ALDW	RRd,(rrs)
7F	K,rrx1	N	Rd			ALD	Rd,N(rrx)
7F	K,rrx0	Nh	Nl	Rd		ALD	Rd,NN(rrx)
80	Rd					CPL	Rd
81	Rd					CPL	(Rd)
82	rd,rs					CP	rd,rs
83	rd,rs					CP	rd,(rs)
84	Rs	Rd				CP	Rd,Rs
85	Rd	N				CP	Rd,#N
86	K,rrx1	N	RRs1			ALDW	N(rrx),RRs
86	K,rrx1	N	RRd0			ALDW	RRd,N(rrx)
86	K,rrx0	Nh	Nl	RRs1		ALDW	NN(rrx),RRs
86	K,rrx0	Nh	Nl	RRd0		ALDW	RRd,NN(rrx)
87	RRs0	RRd0				CPW	RRd,RRs
87	RRd1	Nhs	Nls			CPW	RRd,#NN
88	Rs					LD	r8,Rs
89	Rd					LD	Rd,r8
8A	N					DJNZ	r8,N
8B	N					JR	T,N
8C	N					LD	r8,#N
8D	Nh	Nl				JP	T,NN
8E	rrd0,rrs0					CPW	rrd,rrs
8E	rrd1,rrs0					CPW	(rrd),rrs
8E	rrd0,rrs1					CPW	rrd,(rrs)
8E	rrd1,rrs1					CPW	(rrd),(rrs)
8F	RRd0					RLCW	RRd
8F	F1	N				PUSH	N
8F	F3	N				PUSHU	N
8F	C1	Nh	Nl			PUSH	NN
8F	C3	Nh	Nl			PUSHU	NN
8F	01	RRx0	N			PEA	N(RRx)
8F	01	RRx1	Nl	Nh		PEA	NN(RRx)
8F	03	RRx0	N			PEAU	N(RRx)
8F	03	RRx1	Nl	Nh		PEAU	NN(RRx)
90	Rd					CLR	Rd
91	Rd					CLR	(Rd)
92	rd,rs					CP	rd,rs
93	rd,rs					CP	rd,(rs)
94	Rs	Rd				CP	Rd,Rs
95	Rd	N				CP	Rd,#N
96	RRs0	K,rd				ALDW	(rd),RRs
97	RRs0	RRd0				CPW	RRd,RRs
97	RRd1	Nhs	Nls			CPW	RRd,#NN

OPCODE	BYTE1	BYTE2	BYTE3	BYTE4	BYTE5	OPCODE MNEMONIC	ADDRESSING MODE
98	Rs					LD	r9,Rs
99	Rd					LD	Rd,r9
9A	N					DJNZ	r9,N
9B	N					JR	GE,N
9C	N					LD	r9,#N
9D	Nh	Nl				JP	GE,NN
9E	rrd0,rrs0					CPW	rrd,rrs
9E	rrd1,rrs0					CPW	(rrd),rrs
9E	rrd0,rrs1					CPW	rrd,(rrs)
9E	rrd1,rrs1					CPW	(rrd),(rrs)
9F	rrs0,rd	N				CPJF	rd,(rrs),N
9F	rrs1,rd	N				CPJT	rd,(rrs),N
A0	Rd					ROL	Rd
A1	Rd					ROL	(Rd)
A2	rd,rs					TM	rd,rs
A3	rd,rs					TM	rd,(rs)
A4	Rs	Rd				TM	Rd,Rs
A5	Rd	N				TM	Rd,#N
A6	K,rs	RRd0				ALDW	RRd,(rs)
A7	RRs0	RRd0				TMW	RRd,RRs
A7	RRd1	Nhs	Nls			TMW	RRd,#NN
A8	Rs					LD	r10,Rs
A9	Rd					LD	Rd,r10
AA	N					DJNZ	r10,N
AB	N					JR	GT,N
AC	N					LD	r10,#N
AD	Nh	Nl				JP	GT,NN
AE	rrd0,rrs0					TMW	rrd,rrs
AE	rrd1,rrs0					TMW	(rrd),rrs
AE	rrd0,rrs1					TMW	rrd,(rrs)
AE	rrd1,rrs1					TMW	(rrd),(rrs)
AF	b0,rd	N				BTJT	b.rd,N
AF	b1,rd	N				BTJF	b.rd,N
B0	Rd					RLC	Rd
B1	Rd					RLC	(Rd)
B2	rs,rx	N				LD	N(rx),rs
B3	rd,rx	N				LD	rd,N(rx)
B4	K,rrs1	Rd				ALD	Rd,(rrs)+
B4	K,rrd0	Rs				ALD	(rrd)+,Rs
B5	rd,rrs0					LD	rd,(rrs)
B5	rs,rrd1					LD	(rrd),rs
B6	RR0					PUSHUW	RR
B6	RR1	N				LINKU	RR,#N
B7	RR0					POPUW	RR
B7	RR1					UNLINKU	RR
B8	Rs					LD	r11,Rs
B9	Rd					LD	Rd,r11

OPCODE	BYTE1	BYTE2	BYTE3	BYTE4	BYTE5	OPCODE MNEMONIC	ADDRESSING MODE
BA	N					DJNZ	r11,N
BB	N					JR	UGT,N
BC	N					LD	r11,#N
BD	Nh	Nl				JP	UGT,NN
BE	K,rrd1	RRs0				ALDW	(rr),RR
BE	K,rrd0	Nh	Nl			ALDW	(rr),#NN
BF	RRd0	Nh	Nl			LDW	RRd,#NN
BF	01					HALT	
C0	Rd					ROR	Rd
C1	Rd					ROR	(Rd)
C2	K,rrs1	Rd				ALD	Rd,-(rrs)
C2	K,rrd0	Rs				ALD	-(rrd),Rs
C3	K,rrs1	RRd				ALDW	
C3	K,rrd0	RRs				ALDW	RRd,-(rrs)
C4	K,rd	Nh	Nl			ALD	-(rrd),RRs
C5	K,rs	Nh	Nl			ALD	NN,rs
C6	RR0	N				DWJNZ	RR,N
C6	RR1					EXT	RR
C7	wwwww000					SRP	wwwww
C7	wwwww100					SRP0	wwwww
C7	wwwww101					SRP1	wwwww
C7	pppppp10					SPP	pppppp
C8	Rs					LD	r12,Rs
C9	Rd					LD	Rd,r12
CA	N					DJNZ	r12,N
CB	N					JR	NOV,N
CC	N					LD	r12,#N
CD	Nh	Nl				JP	NOV,NN
CE						ETRAP	
CF	RRd0					DECW	RRd
D0	Rd					RRC	Rd
D1	Rd					RRC	(Rd)
D2	Nh	Nl				CALL	NN
D3						IRET	
D4	RR0					JP	(RR)
D4	RR1	N				LINK	RR,#N
D5	K,rrs1	RRd0				ALDW	RRd,(rrd)+
D5	K,rrd0	RRs0				ALDW	(rrd)+,RRs
D6	rrd0,rrs0					LDPP	(rrd)+,(rrs)+
D6	rrd1,rrs0					LDDP	(rrd)+,(rrs)+
D6	rrd0,rrs1					LDPD	(rrd)+,(rrs)+
D6	rrd1,rrs1					LDDD	(rrd)+,(rrs)+
D7	rd,rrs1					LD	(rd)+,(rrs)+
D7	rs,rrd0					LD	(rrd)+,(rs)+
D8	Rs					LD	r13,Rs
D9	Rd					LD	Rd,r13
DA	N					DJNZ	r13,N

OPCODE	BYTE1	BYTE2	BYTE3	BYTE4	BYTE5	OPCODE MNEMONIC	ADDRESSING MODE
DB	N					JR	PL,N
DC	N					LD	r13,#N
DD	Nh	Nl				JP	PL,NN
DE	rrs1,rx	N				LDW	N(rx),rrs
DE	rrd0,rx	N				LDW	rrd,N(rx)
DF	RRd0					INCW	RRd
E0	Rd					SRA	Rd
E1	Rd					SRA	(Rd)
E2	K,rrd0	Nh	Nl			ALDW	rrd,NN
E2	K,rrs1	Nh	Nl			ALDW	NN,rrs
E3	rrd0,rrs0					LDW	rrd,rrs
E3	rrd1,rrs0					LDW	(rrd),rrs
E3	rrd0,rrs1					LDW	rrd,(rrs)
E3	rrd1,rrs1					LDW	(rrd),(rrs)
E4	rd,rs					LD	rd,(rs)
E5	rd,rs					LD	(rd),rs
E6	Rs	K,rd				ALD	(rd),Rs
E7	K,rs	Rd				ALD	Rd,(rs)
E8	Rs					LD	r14,Rs
E9	Rd					LD	Rd,r14
EA	N					DJNZ	r14,N
EB	N					JR	NE,N
EC	N					LD	r14,#N
ED	Nh	Nl				JP	NE,NN
EE						SPM	
EF	RRs0	RRd0				LDW	RRd,RRs
EF	01					WFI	
EF	31					ERET	
F0	Rd					SWAP	Rd
F1	Rd					SWAP	(Rd)
F2	bs1,rs	bd0,rd				BLD	rd.bd,rs.bs
F2	bs1,rs	bd1,rd				BLD	rd.bd,rs.bs
F2	bd0,rd					BTSET	rd.bd
F3	K,rrd0	N				ALD	(rrd),#N
F4	Rs	Rd				LD	Rd,Rs
F5	Rd	N				LD	Rd,#N
F6	bd0,rr0					BTSET	(rr).bd
F6	01					RETS	
F7	Rs					PUSH	(Rs)
F8	Rs					LD	r15,Rs
F9	Rd					LD	Rd,r15
FA	N					DJNZ	r15,N
FB	N					JR	NC,N
FC	N					LD	r15,#N
FD	Nh	Nl				JP	NC,NN
FE						SDM	
FF						NOP	

INSTRUCTIONS

ADC ADC

Add with carry (byte) Register, Register

ADC dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycle	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src
[OPC] [dst | src]	2	4	32	-	r	r
	2	6	33	-	r	(r)
[OPC] [src] [dst]	3	6	34	-	R	R
	3	6	34	-	r	R
	3	6	34	-	R	r
[OPC] [src] [XTN | dst]	3	6	E6	3	(r)	R
	3	6	E6	3	(r)	r
[OPC] [XTN | src] [dst]	3	6	E7	3	R	(r)

OPERATION: dst ⇐ dst + src + C

The source byte, along with the carry flag, is added to the destination byte and the result is stored in the destination byte. The source and destination byte can be addressed either directly or indirectly.

FLAGS:
C: Set if carry from MSB of result, otherwise cleared.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Always reset to zero.
H: Set if carry from low-order nibble occurred.

EXAMPLE:

Instruction	HEX	Binary
ADC r8,R64	34 40 D8	0011 0100 0100 0000 1101 1000

If the carry flag is set, working register 8 contains 35 (decimal) and register 64 contains 22 (decimal), after this instruction working register 8 will contain 58.

ADC ADC

Add with carry (byte) Register, Memory

ADC dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [XTN |src,1] [dst]	3	8	72	3	R	(rr)	a
	3	8	72	3	r	(rr)	a
	3	12	B4	3	R	(rr)+	b
	3	12	B4	3	r	(rr)+	b
	3	12	C2	3	R	-(rr)	c
	3	12	C2	3	r	-(rr)	c
[OPC] [ofs,1|src,0] [XTN | dst]	3	12	60	3	r	rr(rrx)	a
[OPC] [XTN |src,1] [ofs] [dst]	4	12	7F	3	R	N(rr)	a
	4	12	7F	3	r	N(rr)	a
[OPC] [XTN | dst] [src h] [src l]	4	10	C4	3	r	NN	a
[OPC] [XTN |src,0] [ofs h] [ofs l] [dst]	5	14	7F	3	R	NN(rr)	a
	5	14	7F	3	r	NN(rr)	a

OPERATION a: dst ⇐ dst + src + C

The source byte, along with the carry flag, is added to the destination byte and the result is stored in the destination byte. The destination byte is held in the destination register. The source byte can be addressed directly, indirectly or by indexing.

OPERATION b: dst ⇐ dst + src + C
rr ⇐ rr + 1

The source byte, along with the carry flag, is added to the destination byte and the result is stored in the destination byte. The source byte is in the memory location addressed by the source register pair, the destination byte is in the destination register. The contents of the source register pair are incremented after the ADC has been carried out.

OPERATION c: rr ⇐ rr - 1
dst ⇔ dst + src + C

The source byte, along with the carry flag, is added to the destination byte and the result is stored in the destination byte. The source byte is in the memory location addressed by the source register pair, the destination byte is in the destination register. The contents of the source register pair are decremented before the ADC is carried out.

ADC ADC

Add with carry (byte) Register, Memory

ADC dst,src (Cont'd)

FLAGS:
C: Set if carry from MSB of result, otherwise cleared.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Always reset to zero.
H: Set if carry from low-order nibble occurred.

EXAMPLE:

Instruction	HEX	Binary
ADC r8,(rr4)+	B4 35 D8	1011 0100 0011 0101 1101 1000

If the carry flag is reset, working register 8 contains 11 (decimal), working register pair 4 contains 4200 (decimal) and memory location 4200 contains 11 (decimal), after this instruction working register 8 will contain 22 and working register pair 4 will contain 4201

ADC ADC

Add with carry (byte) Memory, Register

ADC dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [XTN |dst,0] [src]	3	12	72	3	(rr)	R	a
	3	12	72	3	(rr)	r	a
	3	14	B4	3	(rr)+	R	b
	3	14	B4	3	(rr)+	r	b
	3	14	C2	3	-(rr)	R	c
	3	14	C2	3	-(rr)	r	c
[OPC] [ofd,1|dst,1] [XTN | src]	3	14	60	3	rr(rrx)	r	a
[OPC] [XTN |dst,1] [ofd] [src]	4	14	26	3	N(rr)	R	a
	4	14	26	3	N(rr)	r	a
[OPC] [XTN | src] [dst h] [dst l]	4	12	C5	3	NN	r	a
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src]	5	16	26	3	NN(rr)	R	a
	5	16	26	3	NN(rr)	r	a

OPERATION a: dst ⇐ dst + src + C

The source byte, along with the carry flag, is added to the destination byte and the result is stored in the destination byte. The source byte is held in the source register. The destination byte can be addressed directly, indirectly or by indexing.

OPERATION b: dst ⇐ dst + src + C
rr ⇔ rr + 1

The source byte, along with the carry flag, is added to the destination byte and the result is stored in the destination byte. The source byte is in the source register, the destination byte is in the memory location addressed by the destination register pair. The contents of the destination register pair are incremented after the ADC has been carried out.

OPERATION c: rr ⇐ rr - 1
dst ⇔ dst + src + C

The source byte, along with the carry flag, is added to the destination byte and the result is stored in the destination byte. The source byte is in the source register, the destination byte is in the memory location addressed by the destination register pair. The contents of the destination register pair are decremented before the ADC is carried out.

ADC ADC

Add with carry (byte) Memory, Register

ADC dst,src (Cont'd)

FLAGS:
C: Set if carry from MSB of result, otherwise cleared.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Always reset to zero.
H: Set if carry from low-order nibble occurred.

.

EXAMPLE:

Instruction	HEX	Binary
ADC 4028,r8	C5 38 0F BC	1100 0101 0011 1000 0000 1111 1011 1100

If the carry flag is set, memory location 4028 contains 200 (decimal) and working register 8 contains 32 (decimal), after this instruction memory location 4028 will contain 233.

ADC ADC

Add with carry (byte) Memory, Memory

ADC dst,src

INSTRUCTION FORMAT:

[OPC] [XTN |src,0] [dst,0]	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
	3	14	73	3	(RR)	(rr)
	3	14	73	3	(rr)*	(rr)

OPERATION: dst ⇐ dst + src + C

The source byte, along with the carry flag, is added to the destination byte and the result is stored in the destination byte. The source byte is in the memory location addressed by the source register pair, the destination byte is in the memory location addressed by the destination register pair.

FLAGS:
C: Set if carry from MSB of result, otherwise cleared.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Always reset to zero.
H: Set if carry from low-order nibble occurred.

EXAMPLE:

Instruction	HEX	Binary
ADC (rr4),(rr8)	73 38 D4	0111 0011 0011 1000 1101 0100

If the carry flag is set, working register pair 4 contains 2800 (decimal), memory location 2800 contains 46 (decimal), working register pair 8 contains 4200 (decimal) and memory location 4200 contains 45 (decimal), after this instruction memory location 2800 will contain 92.

ADC ADC

Add with carry (byte) All, Immediate

ADC dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst] [src]	3	6	35	-	R	#N
	3	6	35	-	r	#N
[OPC] [XTN |dst,0] [src]	3	10	F3	3	(rr)	#N
[OPC] [XTN] [src] [dst h] [dst l]	5	16	2F	31	NN	#N

OPERATION: dst ⇐ dst + src + C

The source byte, along with the carry flag, is added to the destination byte and the result is stored in the destination byte. The source byte is the immediate value in the operand, the destination byte can be in memory or in the register file.

FLAGS:
C: Set if carry from MSB of result, otherwise cleared.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Always reset to zero.
H: Set if carry from low-order nibble occurred.

EXAMPLE:

Instruction	HEX	Binary
ADC (rr8),#32	F3 38 20	1111 0011 0010 1000 0010 0000

If the carry flag is set, working register pair 8 contains 4028 (decimal) and memory location 4028 contains 74 (decimal), after this instruction memory location 4028 will contain 107.

ADCW ADCW

Add With Carry (Word) - Register, Register

ADCW dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst,0|src,0]	2	8	3E	-	rr	rr
[OPC] [src,0] [dst,0]	3	8	37	-	RR	RR
	3	8	37	-	rr	RR
	3	8	37	-	RR	rr
[OPC] [src,0] [XTN | dst]	3	10	96	3	(r)	RR
	3	10	96	3	(r)	rr
[OPC] [XTN | src] [dst,0]	3	10	A6	3	RR	(r)
	3	10	A6	3	rr	(r)

OPERATION: dst ⇐ dst + src + C

The source word, along with the carry flag, is added to the destination word and the result is stored in the destination word. The source and destination word can be addressed either directly or indirectly.

FLAGS:
C: Set if carry from MSB of result, otherwise cleared.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Undefined.
H: Undefined.

EXAMPLE:

Instruction	HEX	Binary
ADCW (r8),RR64	96 40 38	1001 0110 0100 0000 0011 1000

If the carry flag is zero, register pair 64 contains 1102 (decimal), working register 8 contains 200 (decimal), and register pair 200 contains 2550 (decimal), after this instruction register pair 200 will hold 3652.

ADCW ADCW

Add With Carry (Word) - Register, Memory

ADCW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [dst,0|src,1]	2	12	3E	-	rr	(rr)	a
[OPC] [XTN |src,0] [dst,0]	3	12	7E	3	RR	(rr)	a
[OPC] [XTN |src,1] [dst,0]	3	14	D5	3	RR	(rr)+	b
	3	14	D5	3	rr	(rr)+	b
	3	14	C3	3	RR	-(rr)	c
	3	14	C3	3	rr	-(rr)	c
[OPC] [ofs,0|src,0] [XTN |dst,0]	3	14	60	3	rr	rr(rrx)	a
[OPC] [XTN |src,1] [ofs] [dst,0]	4	14	86	3	RR	N(rr)	a
	4	14	86	3	rr	N(rr)	a
[OPC] [XTN |dst,0] [src h] [src l]	4	14	E2	3	rr	NN	a
[OPC] [XTN |src,0] [ofs h] [ofs l] [dst,0]	5	16	86	3	RR	NN(rr)	a
	5	16	86	3	rr	NN(rr)	a

OPERATION a: dst ⇐ dst + src + C

The source word, along with the carry flag, is added to the destination word and the result is stored in the destination word. The destination word is held in the destination register. The source word can be addressed directly, indirectly or by indexing.

OPERATION b: dst ⇐ dst + src + C
rr ⇔ rr + 2

The source word, along with the carry flag, is added to the destination word and the result is stored in the destination word. The source word is in the memory location addressed by the source register pair, the destination word is in the destination register. The contents of the source register pair are incremented after the ADD has been carried out.

OPERATION c: rr ⇐ rr - 2
dst ⇔ dst + src + C

The source word, along with the carry flag, is added to the destination word and the result is stored in the destination word. The source word is in the memory location addressed by the source register pair, the destination word is in the destination register. The contents of the source register pair are decremented before the ADD is carried out.

ADCW ADCW

Add With Carry (Word) - Register, Memory

ADCW dst,src (Cont'd)

FLAGS:

C: Set if carry from MSB of result, otherwise cleared.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Undefined.
H: Undefined.

EXAMPLE:

Instruction	HEX	Binary
ADCW RR64,-(rr4)	C3 35 40	1100 0011 0011 0101 0100 0000

If the carry flag is set, working register pair 4 contains 1184 (decimal), register pair 64 contains 5000 (decimal) and memory location 1182 contains 1100 (decimal), after this instruction working register pair 64 will contain 6101 and register pair 4 will contain 1182.

ADCW ADCW

Add With Carry (Word) - Memory, Register

ADCW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode		Oper
					dst	src	
[OPC] [dst,1|src,0]	2	16	3E	-	(rr)	rr	a
[OPC] [XTN |dst,1] [src,0]	3	18	BE	3	(rr)	RR	a
[OPC] [XTN |dst,0] [src,0]	3	18	D5	3	(rr)+	RR	b
	3	18	D5	3	(rr)+	rr	b
	3	18	C3	3	-(rr)	RR	c
	3	18	C3	3	-(rr)	rr	c
[OPC] [ofd,0|dst,1] [XTN |src,0]	3	18	60	3	rr(rrx)	rr	a
[OPC] [XTN |dst,1] [ofd] [src,1]	4	18	86	3	N(rr)	RR	a
	4	18	86	3	N(rr)	rr	a
[OPC] [XTN |src,1] [dst h] [dst l]	4	22	E2	3	NN	rr	a
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src,1]	5	20	86	3	NN(rr)	RR	a
	5	20	86	3	NN(rr)	rr	a

OPERATION a: dst ⇐ dst + src + C

The source word, along with the carry flag, is added to the destination word and the result is stored in the destination word. The source word is held in the source register. The destination word can be addressed directly, indirectly or by indexing.

OPERATION b: dst ⇐ dst + src + C
rr ⇐ rr + 2

The source word, along with the carry flag, is added to the destination word and the result is stored in the destination word. The source word is in the source register, the destination word is in the memory location addressed by the destination register pair. The contents of the destination register pair are incremented after the ADD has been carried out.

OPERATION c: rr ⇐ rr - 2
dst ⇔ dst + src + C

The source word, along with the carry flag, is added to the destination word and the result is stored in the destination word. The source word is in the source register, the destination word is in the memory location addressed by the destination register pair. The contents of the destination register pair are decremented before the ADD is carried out.

ADCW ADCW

Add With Carry (Word) - Memory, Register

ADCW dst,src (Cont'd)

FLAGS:

C: Set if carry from MSB of result, otherwise cleared.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Undefined.
H: Undefined.

EXAMPLE:

Instruction	HEX	Binary
ADCW (rr4)+,RR64	D5 34 40	1101 0101 0011 0100 0100 0000

If the carry flag is set, register pair 64 contains 1250 (decimal), working register pair 4 contains 1064 (decimal), and memory location 1064 contains 1750, after this instruction is carried out memory pair 1064 will contain 3001 and working register pair 4 will contain 1066.

ADCW ADCW

Add With Carry (Word) - Memory, Memory

ADCW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [dst,1|src,1]	2	20	3E	-	(rr)	(rr)

OPERATION: dst ⇐ dst + src + C

The source word, along with the carry flag, is added to the destination word and the result is stored in the destination word. The source word is in the memory location addressed by the source register pair, the destination word is in the memory location addressed by the destination register pair.

FLAGS:
C: Set if carry from MSB of result, otherwise cleared.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Undefined.
H: Undefined.

EXAMPLE:

Instruction	HEX	Binary
ADCW (rr4),(rr6)	3E 57	0011 1110 0101 0111

If the carry flag is set, working register pair 6 contains 1002 (decimal), memory pair 1002 contains 2300 (decimal), working register pair 4 contains 1060 (decimal) and memory pair 1060 contains 2700 (decimal), after this instruction memory pair 1060 will contain 5001.

ADCW ADCW

Add With Carry (Word) - All, Immediate

ADCW dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [dst,1] [src h] [src l]	4	10	37	-	RR	#NN
	4	10	37	-	rr	#NN
[OPC] [XTN |dst,0] [src h] [src l]	4	18	BE	3	(rr)	#NN
[OPC] [XTN |dst,1] [ofd] [src h] [src l]	5	20	06	3	N(rr)	#NN
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src h] [src l]	6	22	06	3	NN(rr)	#NN
[OPC] [XTN] [src h] [src l] [dst h] [dst l]	6	22	36	31	NN	#NN

OPERATION: dst ⇐ dst + src + C

The source word, along with the carry flag, is added to the destination word and the result is stored in the destination word. The source word is the immediate value in the operand, the destination word can be in memory or in the register file.

FLAGS:
C: Set if carry from MSB of result, otherwise cleared.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Undefined.
H: Undefined.

EXAMPLE:

Instruction	HEX	Binary
ADCW RR64,#4268	37 41 10 AC	0011 0111 0100 0001 0001 0000 1010 1100

If the carry flag is zero and register pair 64 contains 2000 (decimal), after this instruction has been carried out register pair 64 will contain the decimal value 6268.

ADD ADD

Add (byte) Register, Register

ADD dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst | src]	2	4	42	-	r	r
	2	6	43	-	r	(r)
[OPC] [src] [dst]	3	6	44	-	R	R
	3	6	44	-	r	R
	3	6	44	-	R	r
[OPC] [src] [XTN | dst]	3	6	E6	4	(r)	R
	3	6	E6	4	(r)	r
[OPC] [XTN | src] [dst]	3	6	E7	4	R	(r)

OPERATION: dst ⇐ dst + src

The source byte is added to the destination byte and the result is stored in the destination byte. The source and destination byte can be addressed either directly or indirectly.

FLAGS:
C: Set if carry from MSB of result, otherwise cleared.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Always reset to zero.
H: Set if carry from low-order nibble occurred.

EXAMPLE:

Instruction	HEX	Binary
ADD (r8),R255	E6 FF 48	1110 0110 1111 1111 0100 1000

If working register 8 contains 28 (decimal), register 28 contains 43 (decimal) and register 255 contains 21 (decimal) after this instruction register 28 will contain 64.

ADD ADD

Add (byte) Register, Memory

ADD dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [XTN |src,1] [dst]	3	8	72	4	R	(rr)	a
	3	8	72	4	r	(rr)	a
	3	12	B4	4	R	(rr)+	b
	3	12	B4	4	r	(rr)+	b
	3	12	C2	4	R	-(rr)	c
	3	12	C2	4	r	-(rr)	c
[OPC] [ofs,1|src,0] [XTN | dst]	3	12	60	4	r	rr(rrx)	a
[OPC] [XTN |src,1] [ofs] [dst]	4	12	7F	4	R	N(rr)	a
	4	12	7F	4	r	N(rr)	a
[OPC] [XTN | dst] [src h] [src l]	4	10	C4	4	r	NN	a
[OPC] [XTN |src,0] [ofs h] [ofs l] [dst]	5	14	7F	4	R	NN(rr)	a
	5	14	7F	4	r	NN(rr)	a

OPERATION a: dst ⇐ dst + src

The source byte is added to the destination byte and the result is stored in the destination byte. The destination byte is held in the destination register. The source byte can be addressed directly, indirectly or by indexing.

OPERATION b: dst ⇐ dst + src
rr ⇔ rr + 1

The source byte is added to the destination byte and the result is stored in the destination byte. The source byte is in the memory location addressed by the source register pair, the destination byte is in the destination register. The contents of the source register pair are incremented after the ADD has been carried out.

OPERATION c: rr ⇐ rr - 1
dst ⇔ dst + src

The source byte is added to the destination byte and the result is stored in the destination byte. The source byte is in the memory location addressed by the source register pair, the destination byte is in the destination register. The contents of the source register pair are decremented before the ADD is carried out.

ADD ADD

Add (byte) Register, Memory

ADD dst,src (Cont'd)

FLAGS:
C: Set if carry from MSB of result, otherwise cleared.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Always reset to zero.
H: Set if carry from low-order nibble occurred.

EXAMPLE:

Instruction	HEX	Binary
ADD R32,-(rr4)	C2 45 20	1100 0010 0100 0101 0010 0000

If register 32 contains 207 (decimal), working register pair 4 contains 4200 (decimal) and memory location 4199 contains 27 (decimal), after this instruction register 32 will contain 234 and working register pair 4 will contain 4199.

ADD ADD

Add (byte) Memory, Register

ADD dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode		Oper
					dst	src	
[OPC] [XTN |dst,0] [src]	3	12	72	4	(rr)	R	a
	3	12	72	4	(rr)	r	a
	3	14	B4	4	(rr)+	R	b
	3	14	B4	4	(rr)+	r	b
	3	14	C2	4	-(rr)	R	c
	3	14	C2	4	-(rr)	r	c
[OPC] [ofd,1|dst,1] [XTN | src]	3	14	60	4	rr(rrx)	r	a
[OPC] [XTN |dst,1] [ofd] [src]	4	14	26	4	N(rr)	R	a
	4	14	26	4	N(rr)	r	a
[OPC] [XTN | src] [dst h] [dst l]	4	12	C5	4	NN	r	a
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src]	5	16	26	4	NN(rr)	R	a
	5	16	26	4	NN(rr)	r	a

OPERATION a: dst ⇐ dst + src

The source byte is added to the destination byte and the result is stored in the destination byte. The source byte is held in the source register. The destination byte can be addressed directly, indirectly or by indexing.

OPERATION b: dst ⇐ dst + src
rr ⇔ rr + 1

The source byte is added to the destination byte and the result is stored in the destination byte. The source byte is in the source register, the destination byte is in the memory location addressed by the destination register pair. The contents of the destination register pair are incremented after the ADD has been carried out.

OPERATION c: rr ⇐ rr - 1
dst ⇔ dst + src

The source byte is added to the destination byte and the result is stored in the destination byte. The source byte is in the source register, the destination byte is in the memory location addressed by the destination register pair. The contents of the destination register pair are decremented before the ADD is carried out.

ADD ADD

Add (byte) Memory, Register

ADD dst,src (Cont'd)

FLAGS:
C: Set if carry from MSB of result, otherwise cleared.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Always reset to zero.
H: Set if carry from low-order nibble occurred.

EXAMPLE:

Instruction	HEX	Binary
ADD 6(rr8),R255	26 49 06 FF	0010 0110 0100 1001 0000 0110 1111 1111

If working register pair 8 contains 4028 (decimal), memory location 4034 contains 110 (decimal) and register 255 contains 100 (decimal), after this instruction memory location 4034 will contain 210.

ADD ADD

Add (byte) Memory, Memory

ADD dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [XTN |src,0] [dst,0]	3	14	73	4	(RR)	(rr)
	3	14	73	4	(rr)*	(rr)

OPERATION: dst ⇐ dst + src

The source byte is added to the destination byte and the result is stored in the destination byte. The source byte is in the memory location addressed by the source register pair, the destination byte is in the memory location addressed by the destination register pair.

FLAGS:
C: Set if carry from MSB of result, otherwise cleared.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Always reset to zero.
H: Set if carry from low-order nibble occurred.

EXAMPLE:

Instruction	HEX	Binary
ADD (rr4),(rr8)	73 48 D4	0111 0011 0100 1000 1101 0100

If working register pair 4 contains 2800 (decimal) and memory location 2800 contains 46 (decimal), working register pair 8 contains 4200 (decimal) and memory location 4200 contains 45 (decimal), after this instruction memory location 2800 will contain 91.

ADD ADD

Add (byte) All, Immediate

ADD dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [dst] [src]	3	6	45	-	R	#N
	3	6	45	-	r	#N
[OPC] [XTN |dst,0] [src]	3	10	F3	4	(rr)	#N
[OPC] [XTN] [src] [dst h] [dst l]	5	16	2F	41	NN	#N

OPERATION: dst ⇐ dst + src

The source byte is added to the destination byte and the result is stored in the destination byte. The source byte is the immediate value in the operand, the destination byte can be in memory or in the register file.

FLAGS:
C: Set if carry from MSB of result, otherwise cleared.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Always reset to zero.
H: Set if carry from low-order nibble occurred.

EXAMPLE:

Instruction	HEX	Binary
ADD (rr8),#32	F3 48 20	1111 0011 0100 1000 0010 0000

If working register pair 8 contains 4028 (decimal) and memory location 4028 contains 74 (decimal), after this instruction memory location 4028 will contain 106.

ADDW ADDW

Add (Word) - Register, Register

ADDW dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst,0|src,0]	2	8	4E	-	rr	rr
[OPC] [src,0] [dst,0]	3	8	47	-	RR	RR
	3	8	47	-	rr	RR
	3	8	47	-	RR	rr
[OPC] [src,0] [XTN | dst]	3	10	96	4	(r)	RR
	3	10	96	4	(r)	rr
[OPC] [XTN | src] [dst,0]	3	10	A6	4	RR	(r)
	3	10	A6	4	rr	(r)

OPERATION: dst ⇐ dst + src

The source word is added to the destination word and the result is stored in the destination word. The source and destination words, held in register pairs, can be addressed either directly or indirectly.

FLAGS:
C: Set if carry from MSB of result, otherwise cleared.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Undefined.
H: Undefined.

EXAMPLE:

Instruction	HEX	Binary
ADDW RR64,(r8)	A6 48 40	1010 0110 0100 1000 0100 0000

If working register 8 contains 124, register pair 124 contains 1300 (decimal) and register pair 64 contains 800 (decimal) after this instruction register pair 64 will contain 2100.

ADDW ADDW

Add (Word) - Register, Memory

ADDW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [dst,0|src,1]	2	12	4E	-	rr	(rr)	a
[OPC] [XTN |src,0] [dst,0]	3	12	7E	4	RR	(rr)	a
[OPC] [XTN |src,1] [dst,0]	3	14	D5	4	RR	(rr)+	b
	3	14	D5	4	rr	(rr)+	b
	3	14	C3	4	RR	-(rr)	c
	3	14	C3	4	rr	-(rr)	c
[OPC] [ofs,0|src,0] [XTN |dst,0]	3	14	60	4	rr	rr(rrx)	a
[OPC] [XTN |src,1] [ofs] [dst,0]	4	14	86	4	RR	N(rr)	a
	4	14	86	4	rr	N(rr)	a
[OPC] [XTN |dst,0] [src h] [src l]	4	14	E2	4	rr	NN	a
[OPC] [XTN |src,0] [ofs h] [ofs l] [dst,0]	5	16	86	4	RR	NN(rr)	a
	5	16	86	4	rr	NN(rr)	a

OPERATION a: dst ⇐ dst + src

The source word is added to the destination word and the result is stored in the destination word. The destination word is held in the destination register. The source word can be addressed directly, indirectly or by indexing.

OPERATION b: dst ⇐ dst + src
rr ⇔ rr + 2

The source word is added to the destination word and the result is stored in the destination word. The source word is in the memory location addressed by the source register pair, the destination word is in the destination register. The contents of the source register pair are incremented after the ADD has been carried out.

OPERATION c: rr ⇐ rr - 2
dst ⇔ dst + src

The source word is added to the destination word and the result is stored in the destination word. The source word is in the memory location addressed by the source register pair, the destination word is in the destination register. The contents of the source register pair are decremented before the ADD is carried out.

ADDW ADDW

Add (Word) - Register, Memory

ADDW dst,src (Cont'd)

FLAGS:
C: Set if carry from MSB of result, otherwise cleared.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Undefined.
H: Undefined.

EXAMPLE:

Instruction	HEX	Binary
ADDW RR64,(rr8)	7E 48 40	0111 1110 0100 1000 0100 0000

If working register pair 8 contains 1240 (decimal), memory pair 1240 contains 3000 (decimal) and register pair 64 contains 1000 (decimal), after this instruction working register pair 64 will contain 4000.

ADDW ADDW

Add (Word) - Memory, Register

ADDW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [dst,1|src,0]	2	16	4E	-	(rr)	rr	a
[OPC] [XTN |dst,1] [src,0]	3	18	BE	4	(rr)	RR	a
[OPC] [XTN |dst,0] [src,0]	3	18	D5	4	(rr)+	RR	b
	3	18	D5	4	(rr)+	rr	b
	3	18	C3	4	-(rr)	RR	c
	3	18	C3	4	-(rr)	rr	c
[OPC] [ofd,0|dst,1] [XTN |src,0]	3	18	60	4	rr(rrx)	rr	a
[OPC] [XTN |dst,1] [ofd] [src,1]	4	18	86	4	N(rr)	RR	a
	4	18	86	4	N(rr)	rr	a
[OPC] [XTN |src,1] [dst h] [dst l]	4	18	E2	4	NN	rr	a
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src,1]	5	20	86	4	NN(rr)	RR	a
	5	20	86	4	NN(rr)	rr	a

OPERATION a: dst ⇐ dst + src

The source word is added to the destination word and the result is stored in the destination word. The source word is held in the source register. The destination word can be addressed directly, indirectly or by indexing.

OPERATION b: dst ⇐ dst + src
rr ⇔ rr + 2

The source word is added to the destination word and the result is stored in the destination word. The source word is in the source register, the destination word is in the memory location addressed by the destination register pair. The contents of the destination register pair are incremented after the ADD has been carried out.

OPERATION c: rr ⇐ rr - 2
dst ⇔ dst + src

The source word is added to the destination word and the result is stored in the destination word. The source word is in the source register, the destination word is in the memory location addressed by the destination register pair. The contents of the destination register pair are decremented before the ADD is carried out.

ADDW ADDW

Add (Word) - Memory, Register

ADDW dst,src (Cont'd)

FLAGS:
C: Set if carry from MSB of result, otherwise cleared.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Undefined.
H: Undefined.

EXAMPLE:

Instruction	HEX	Binary
ADDW (rr4)+,RR64	D5 44 40	1101 0101 0100 0100 0100 0000

If register pair 64 contains 1250 (decimal), working register pair 4 contains 1064 (decimal), and memory pair 1064 contains 1750, after this instruction is carried out memory pair 1064 will contain 3000 and working register pair 4 will contain 1066.

ADDW ADDW

Add (Word) - Memory, Memory

ADDW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [dst,1|src,1]	2	20	4E	-	(rr)	(rr)

OPERATION: dst ⇐ dst + src

The source word is added to the destination word and the result is stored in the destination word. The source word is in the memory location addressed by the source register pair, the destination word is in the memory location addressed by the destination register pair.

FLAGS:
C: Set if carry from MSB of result, otherwise cleared.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Undefined.
H: Undefined.

EXAMPLE:

Instruction	HEX	Binary
ADDW (rr4),(rr6)	4E 57	0100 1110 0101 0111

If working register pair 6 contains 1002 (decimal), memory pair 1002 contains 2300 (decimal), working register pair 4 contains 1060 (decimal) and memory pair 1060 contains 2700 (decimal), after this instruction memory pair 1060 will contain 5000.

ADDW ADDW

Add (Word) - All, Immediate

ADDW dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst,1] [src h] [src l]	4	10	47	-	RR	#NN
	4	10	47	-	rr	#NN
[OPC] [XTN |dst,0] [src h] [src l]	4	18	BE	4	(rr)	#NN
[OPC] [XTN |dst,1] [ofd] [src h] [src l]	5	20	06	4	N(rr)	#NN
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src h] [src l]	6	22	06	4	NN(rr)	#NN
[OPC] [XTN] [src h] [src l] [dst h] [dst l]	6	22	36	41	NN	#NN

OPERATION: dst ⇐ dst + src

The source word is added to the destination word and the result is stored in the destination word. The source word is the immediate value in the operand, the destination word can be in memory or in the register file.

FLAGS:
C: Set if carry from MSB of result, otherwise cleared.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Undefined.
H: Undefined.

EXAMPLE:

Instruction	HEX	Binary
ADDW RR64,#4268	47 41 10 AC	0100 0111 0100 0001 0001 0000 1010 1100

If register pair 64 contains 2000 (decimal), after this instruction has been carried out register pair 64 will contain the decimal value 6268.

AND AND

AND (byte) Register, Register

AND dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst | src]	2	4	12	-	r	r
	2	6	13	-	r	(r)
[OPC] [src] [dst]	3	6	14	-	R	R
	3	6	14	-	r	R
	3	6	14	-	R	r
[OPC] [src] [XTN | dst]	3	6	E6	1	(r)	R
	3	6	E6	1	(r)	r
[OPC] [XTN | src] [dst]	3	6	E7	1	R	(r)

OPERATION: dst ⇐ dst AND src

The contents of the source are ANDed with the destination byte and the results stored in the destination byte. The contents of the source are not affected.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 7 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
AND r8,R64	14 40 D8	0001 0100 0100 0000 1101 1000

If working register 8 contains 11001100 and register 64 contains 10000101, after this instruction working register 8 will contain 10000100.

AND AND

AND (byte) Register, Memory

AND dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode		Oper
					dst	src	
[OPC] [XTN |src,1] [dst]	3	8	72	1	R	(rr)	a
	3	8	72	1	r	(rr)	a
	3	12	B4	1	R	(rr)+	b
	3	12	B4	1	r	(rr)+	b
	3	12	C2	1	R	-(rr)	c
	3	12	C2	1	r	-(rr)	c
[OPC] [ofs,1|src,0] [XTN | dst]	3	12	60	1	r	rr(rrx)	a
[OPC] [XTN |src,1] [ofs] [dst]	4	12	7F	1	R	N(rr)	a
	4	12	7F	1	r	N(rr)	a
[OPC] [XTN | dst] [src h] [src l]	4	10	C4	1	r	NN	a
[OPC] [XTN |src,0] [ofs h] [ofs l] [dst]	5	14	7F	1	R	NN(rr)	a
	5	14	7F	1	r	NN(rr)	a

OPERATION a: dst ⇐ dst AND src

The source byte is ANDed with the destination byte and the result stored in the destination byte. The destination register is addressed directly, the memory location is addressed either directly, indirectly or by indexing.

OPERATION b: dst ⇐ dst AND src
rr ⇐ rr + 1

The contents of the memory location addressed by the source register pair are ANDed with the contents of the directly addressed destination register the result stored in the destination byte. The contents of the source register pair are incremented after the AND has been carried out.

OPERATION c: rr ⇐ rr - 1
dst ⇐ dst AND src

The contents of the source register pair are decremented and then the contents of the memory location addressed by the source register pair are ANDed with the contents of the directly addressed destination register. The result is stored in the destination byte.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 7 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

AND AND

AND (byte) Register, Memory

AND dst,src (Cont'd)

EXAMPLE:

Instruction	HEX	Binary
AND r8,4028	C4 18 0F BC	1100 0100 0001 1000 0000 1111 1011 1100

If working register 8 contains 11001100 and memory location 4028 contains 10000101, after this instruction working register 8 will contain 10000100.

AND AND

AND (byte) Memory, Register

AND dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [XTN |dst,0] [src]	3	12	72	1	(rr)	R	a
	3	12	72	1	(rr)	r	a
	3	14	B4	1	(rr)+	R	b
	3	14	B4	1	(rr)+	r	b
	3	14	C2	1	-(rr)	R	c
	3	14	C2	1	-(rr)	r	c
[OPC] [ofd,1|dst,1] [XTN | src]	3	14	60	1	rr(rrx)	r	a
[OPC] [XTN |dst,1] [ofd] [src]	4	14	26	1	N(rr)	R	a
	4	14	26	1	N(rr)	r	a
[OPC] [XTN | src] [dst h] [dst l]	4	12	C5	1	NN	r	a
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src]	5	16	26	1	NN(rr)	R	a
	5	16	26	1	NN(rr)	r	a

OPERATION a: dst ⇐ dst AND src

The source byte is ANDed with the destination byte and the result stored in the destination byte. The source registers are addressed directly, the memory location are addressed either directly, indirectly or by indexing.

OPERATION b: dst ⇐ dst AND src
rr ⇐ rr + 1

The contents of the memory location addressed by the destination register pair (destination byte) are ANDed with the contents of the directly addressed source register the result stored in the destination byte. The contents of the destination register pair are incremented after the AND has been carried out.

OPERATION c: rr ⇐ rr - 1
dst ⇐ dst AND src

The contents of the destination register pair are decremented and then the contents of the memory location addressed by the destination register pair (destination byte) are ANDed with the contents of the directly addressed source register. The result is stored in the destination byte.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 7 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

AND AND

AND (byte) Memory, Register

AND dst,src (Cont'd)

EXAMPLE:

Instruction	HEX	Binary
AND 4028,r8	C5 18 0F BC	1100 0101 0001 1000 0000 1111 1011 1100

If working register 8 contains 11001100 and memory location 4028 contains 10000101, after this instruction memory location 4028 will contain 10000100.

AND AND

AND (byte) Memory, Memory

AND dst,src

INSTRUCTION FORMAT:

[OPC] [XTN |src,0] [dst,0]	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
	3	14	73	1	(RR)	(rr)
	3	14	73	1	(rr)*	(rr)

OPERATION: dst ⇐ dst AND src

The contents of the memory location addressed by the source register pair are ANDed with the content of the memory location addressed by the destination register pair. The source and destination addresses are for the word high order byte.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 7 is set, otherwise cleared.
V: Always reset to zero
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
AND (rr4),(rr8)	73 18 D4	0111 0011 0001 1000 1101 0100

If working register pair 4 contains 2800 (decimal), memory location 2800 contains 11001100, working register pair 8 contains 4200 (decimal) and memory location 4200 contains 11000011, after this instruction memory location 2800 will contain 11000000.

AND AND

AND (byte) All, Immediate

AND dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [dst] [src]	3	6	15	-	R	#N
	3	6	15	-	r	#N
[OPC] [XTN |dst,0] [src]	3	10	F3	1	(rr)	#N
[OPC] [XTN] [src] [dst h] [dst l]	5	16	2F	11	NN	#N

OPERATION: dst ⇐ dst AND src

The value #N is ANDed with the content of the destination register or memory location (destination byte) and stored in the destination byte.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 7 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
AND (rr8),#32	F3 18 20	1111 0011 0001 1000 0010 0000

If working register pair 8 contains 4028 (decimal) and memory location 4028 contains 11101100, after this instruction memory location 4028 will contain 00100000.

ANDW ANDW

AND (Word) - Register, Register

ANDW dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst,0|src,0]	2	8	1E	-	rr	rr
[OPC] [src,0] [dst,0]	3	8	17	-	RR	RR
	3	8	17	-	rr	RR
	3	8	17	-	RR	rr
[OPC] [src,0] [XTN | dst]	3	10	96	1	(r)	RR
	3	10	96	1	(r)	rr
[OPC] [XTN | src] [dst,0]	3	10	A6	1	RR	(r)
	3	10	A6	1	rr	(r)

OPERATION: dst ⇐ dst AND src

The source word is ANDed with the destination word. The result is left in the destination. The source and destination words can be addressed either directly or indirectly.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 15 is set, otherwise cleared.
V: Always reset to zero.
D: Undefined.
H: Undefined.

EXAMPLE:

Instruction	HEX	Binary
ANDW RR32,RR64	17 40 20	0001 0111 0100 0000 0010 0000

If register pair 64 contains 11001100/11001100B and register pair 32 contains 10101010/10101010B, after this instruction register pair 32 will contain 10001000/10001000B.

ANDW ANDW

AND (Word) - Register, Memory

ANDW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [dst,0|src,1]	2	12	1E	-	rr	(rr)	a
[OPC] [XTN |src,0] [dst,0]	3	12	7E	1	RR	(rr)	a
[OPC] [XTN |src,1] [dst,0]	3	14	D5	1	RR	(rr)+	b
	3	14	D5	1	rr	(rr)+	b
	3	14	C3	1	RR	-(rr)	c
	3	14	C3	1	rr	-(rr)	c
[OPC] [ofs,0|src,0] [XTN |dst,0]	3	14	60	1	rr	rr(rrx)	a
[OPC] [XTN |src,1] [ofs] [dst,0]	4	14	86	1	RR	N(rr)	a
	4	14	86	1	rr	N(rr)	a
[OPC] [XTN |dst,0] [src h] [src l]	4	14	E2	1	rr	NN	a
[OPC] [XTN |src,0] [ofs h] [ofs l] [dst,0]	5	16	86	1	RR	NN(rr)	a
	5	16	86	1	rr	NN(rr)	a

OPERATION a: dst ⇐ dst AND src

The source word is ANDed with the destination word. The result is left in the destination word. The destination word is held in the destination register. The source word can be addressed directly, indirectly or by indexing. The contents of the source are not affected.

OPERATION b: dst ⇐ dst AND src
rr ⇐ rr + 2

The source word is ANDed with the destination word. The result is left in the destination word. The source word is in the memory location addressed by the source register pair, the destination word is in the destination register. The contents of the source register pair are incremented after the AND has been carried out.

OPERATION c: rr ⇐ rr - 2
dst ⇐ dst AND src

The source word is ANDed with the destination word. The result is left in the destination word. The source word is in the memory location addressed by the source register pair, the destination word is in the destination register. The contents of the source register pair are decremented before the AND is carried out.

ANDW ANDW

AND (Word) - Register, Memory

ANDW dst,src (Cont'd)

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 15 is set, otherwise cleared.
V: Always reset to zero.
D: Undefined.
H: Undefined.

EXAMPLE:

Instruction	HEX	Binary
ANDW RR64,(rr4)+	D5 15 40	1101 0101 0001 0101 0100 0000

If working register pair 4 contains 1184 (decimal), register pair 64 contains 10101010/10101010B and memory pair 1184 contains 11001100/11001100B, after this instruction register pair 64 will contain 10001000/10001000B and register pair 4 will contain 1186.

ANDW ANDW

AND (Word) - Memory, Register

ANDW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [dst,1|src,0]	2	16	1E	-	(rr)	rr	a
[OPC] [XTN |dst,1] [src,0]	3	18	BE	1	(rr)	RR	a
[OPC] [XTN |dst,0] [src,0]	3	18	D5	1	(rr)+	RR	b
	3	18	D5	1	(rr)+	rr	b
	3	18	C3	1	-(rr)	RR	c
	3	18	C3	1	-(rr)	rr	c
[OPC] [ofd,0|dst,1] [XTN |src,1]	3	18	60	1	rr(rrx)	rr	a
[OPC] [XTN |dst,1] [ofd] [src,1]	4	18	86	1	N(rr)	RR	a
	4	18	86	1	N(rr)	rr	a
[OPC] [XTN |src,1] [dst h] [dst l]	4	18	E2	1	NN	rr	a
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src,1]	5	20	86	1	NN(rr)	RR	a
	5	20	86	1	NN(rr)	rr	a

OPERATION a: dst ⇐ dst AND src

The source word is ANDed with the destination word. The result is stored in the destination word. The source word is held in the source register pair. The destination word can be addressed directly, indirectly or by indexing.

OPERATION b: dst ⇐ dst AND src
rr ⇐ rr + 2

The source word is ANDed with the destination word. The result is stored in the destination word. The source word is in the source register pair, the destination word is in the memory location addressed by the destination register pair. The contents of the destination register pair pair are incremented after the AND has been carried out.

OPERATION c: rr ⇐ rr - 2
dst ⇐ dst AND src

The source word is ANDed with the destination word. The result is stored in the destination word. The source word is in the source register pair , the destination word is in the memory pair addressed by the destination register pair. The contents of the destination register pair pair are decremented before the AND is carried out.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 15 is set, otherwise cleared.
V: Always reset to zero.
D: Undefined.
H: Undefined.

ANDW ANDW

AND (Word) - Memory, Register

ANDW dst,src (Cont'd)

EXAMPLE:

Instruction	HEX	Binary
ANDW (rr8),RR64	BE 19 40	1011 1110 0001 1001 0100 0000

If register pair 64 contains 11001100/11001100B, working register pair 8 contains 2000 (decimal) and memory pair 2000 contains 10101010/10101010B, after this instruction memory pair 2000 will hold 10001000/10001000B.

ANDW ANDW

AND (Word) - Memory, Memory

ANDW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
`[ OPC ] [dst,1|src,1]`	2	20	1E	-	(rr)	(rr)

OPERATION: dst ⇐ dst AND src

The source word is ANDed with the destination word. The result is stored in the destination word. The source word is in the memory pair addressed by the source register pair, the destination word is in the memory pair addressed by the destination register pair.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 15 is set, otherwise cleared.
V: Always reset to zero.
D: Undefined.
H: Undefined.

EXAMPLE:

Instruction	HEX	Binary
ANDW (rr4),(rr6)	1E 57	0001 1110 0101 0111

If working register pair 6 contains register 1002 (decimal), memory pair 1002 contains 11001100/11001100B, working register pair 4 contains 1060 (decimal), and memory pair 1060 contains 10101010/10101010B, after this instruction memory pair 1060 will contain 10001000/10001000B.

ANDW ANDW

AND (Word) - All, Immediate

ANDW dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst,1] [src h] [src l]	4	10	17	-	RR	#NN
	4	10	17	-	rr	#NN
[OPC] [XTN |dst,0] [src h] [src l]	4	18	BE	1	(rr)	#NN
[OPC] [XTN |dst,1] [ofd] [src h] [src l]	5	20	06	1	N(rr)	#NN
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src h] [src l]	6	22	06	1	NN(rr)	#NN
[OPC] [XTN] [src h] [src l] [dst h] [dst l]	6	22	36	11	NN	#NN

OPERATION: dst ⇐ dst AND src

The source word is ANDed with the destination word. The result is stored in the destination word. The source word is the immediate value in the operand, the destination word can be in memory or in the register file.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 15 is set, otherwise cleared.
V: Always reset to zero.
D: Undefined.
H: Undefined.

EXAMPLE:

Instruction	HEX	Binary
ANDW RR64,#52428	17 41 CC CC	0001 0111 0100 0001 1100 1100 1100 1100

If register pair 64 contains 10101010/10101010B, after this instruction has been carried out register pair 64 will contain 10001000/10001000B.

BAND BAND

Bit AND

BAND dst.b,src.b

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode		Oper
					dst	src	
`[ OPC ] [btd,1| dst ] [bts,1| src ]`	3	10	1F	-	r.b	r.b	a
`[ OPC ] [btd,1| dst ] [bts,0| src ]`	3	10	1F	-	r.b	r.!b	b

OPERATION a: dst bit ⇐ dst bit AND src bit

The selected bit in the source working register is ANDed with the selected bit of the destination working register and the result left in the destination bit. All other bits in the destination register remain unaffected. Both source and destination are directly addressed.

OPERATION b: dst bit ⇐ dst bit AND NOT src bit

The complement of the selected bit in the source working register is ANDed with the selected bit of the destination working register and the result left in the destination bit. All other bits in the destination register remain unaffected. Both source and destination are directly addressed.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
BAND r4.5,r8.2	1F B4 58	0001 1111 1011 0100 0101 1000

If bit 2 of working register 8 is 0 and bit 5 of working register 4 is 1, after this instruction bit 5 of working register 4 will be 0.

NOTE: A bit AND can use the same or different nibbles of the same register as both source and destination.

BCPL BCPL

Bit Complement

BCPL dst.b

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [btd,0| dst]	2	4	6F	-	r.b	-

OPERATION: dst bit ⇐ NOT dst bit

The selected bit in the destination working register is set to its own complement. All other bits in the destination register remain unaffected. The destination is directly addressed.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
BCPL r4.5	6F A4	0110 1111 1010 0100

If bit 5 of working register 4 was 1, after this instruction it will be 0.

BLD BLD

Bit Load

BLD dst.b,src.b

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode		Oper
					dst	src	
[OPC] [bts,1| src] [btd,0| dst]	3	10	F2	-	r.b	r.b	a
[OPC] [bts,1| src] [btd,1| dst]	3	10	F2	-	r.b	r.!b	b

OPERATION a: dst bit ⇐ src bit

The selected bit in the source working register is loaded into the selected bit of the destination working register. All other bits in the destination register remain unaffected. Both source and destination are directly addressed.

OPERATION b: dst bit ⇐ NOT src bit

The complement of the selected bit in the source working register is loaded into the selected bit of the destination working register. All other bits in the destination register remain unaffected. Both source and destination are directly addressed.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
BLD r4.5,r8.!2	F2 58 B4	1111 0010 0101 1000 1011 0100

If bit 2 of working register 8 is 1, after this instruction bit 5 of working register 4 will be 0.

NOTE: A bit load can use the same or different nibbles of the same register as both source and destination.

BOR BOR

Bit OR

BOR dst.b,src.b

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode		Oper
					dst	src	
[OPC] [btd,1| dst] [bts,0| src]	3	10	0F	-	r.b	r.b	a
[OPC] [btd,1| dst] [bts,1| src]	3	10	0F	-	r.b	r.!b	b

OPERATION a: dst bit ⇐ dst bit OR src bit

The selected bit in the source working register is ORed with the selected bit of the destination register and the result left in the destination bit. All other bits in the destination register remain unaffected. Both source and destination are directly addressed.

OPERATION b: dst bit ⇐ dst bit OR NOT src bit

The complement of the selected bit in the source working register is ORed with the selected bit of the destination working register and the result left in the destination bit. All other bits in the destination register remain unaffected. Both source and destination are directly addressed.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
BOR r4.5,r8.2	0F B4 48	0000 1111 1011 0100 0100 1000

If bit 2 of working register 8 is 1 and bit 5 of working register 4 is 0, after this instruction bit 5 of working register 4 will be 1.

NOTE: A bit OR can use the same or different nibbles of the same register as both source and destination.

BRES BRES

Bit Reset

BRES dst.b

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [btd,0| dst]	2	4	1F	-	r.b	-

OPERATION: dst bit ⇐ 0

The selected bit in the destination working register is reset to 0. All other bits in the destination register remain unaffected. The destination is directly addressed.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
BRES r4.5	1F A4	0001 1111 1010 0100

After this instruction bit 5 of working register 4 will be 0.

BSET BSET

Bit Set

BSET dst.b

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [btd,0| dst]	2	4	0F	-	r.b	-

OPERATION: dst bit ⇐ 1

The selected bit in the destination working register is set to 1. All other bits in the destination register remain unaffected. The destination is directly addressed.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
BSET r4.5	0F A4	0000 1111 1010 0100

After this instruction bit 5 of working register 4 will be 1.

BTJF BTJF

Bit Test And Jump If False

BTJF dst,src

INSTRUCTION FORMAT:

	No. Bytes	No.Cycl		OPC (HEX)	Addr Mode		PC offs.
		No Jmp	Jmp		dst	src	
[OPC] [btd,1| dst] [N]	3	6	10	AF	-	r.b	N

OPERATION: If tested bit is zero then PC⇐ PC + N where -128≥ N ≥ 127

The specified bit in the destination working register is tested and if it is found to be equal to zero, the source value N is added to the program counter and control passes to the statement whose address is now in the PC. If the tested bit is one, the instruction following BTJF is executed. N is a relative value in the range +127/-128.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
BTJF r10.2,-40	AF 5A D8	1010 1111 0101 1010 1101 1000

If bit 2 of working register 10 is zero and the program counter holds 200, after this instruction the program counter will jump to address 160.

BTJT BTJT

Bit Test And Jump If True

BTJT dst,src

INSTRUCTION FORMAT:

	No. Bytes	No.Cycl		OPC (HEX)	Addr Mode		PC offs.
		No Jmp	Jmp		dst	src	
[OPC] [btd,0| dst] [N]	3	6	10	AF	-	r.b	N

OPERATION: If tested bit is one then PC⇐ PC + N where -128 ≥ N ≥ 127

The specified bit in the destination working register is tested and if it is found to be equal to one, the source value N is added to the program counter and control passes to the statement whose address is now in the PC. If the tested bit is zero the instruction following BTJT is executed. N is a relative value in the range +127/-128.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
BTJT r10.2,+40	AF 4A 28	1010 1111 0100 1010 0010 1000

If bit 2 of working register 10 is a one and the program counter holds 200, after this instruction the program counter will jump to address 240.

BTSET BTSET

Bit Test and Set

BTSET dst.b

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode		Oper
					dst	src	
[OPC] [btd,0| dst]	2	8	F2	-	r.b	-	
[OPC] [btd,0| dst,0]	2	14	F6	-	(rr).b	-	

OPERATION: If dst.b = 0 then dst.b ⇐ 1

The selected bit in the destination is tested; if zero it is set to one and the zero flag set. The destination is addressed either by working register direct or memory indirect.

FLAGS:
C: Unaffected.
Z: Set if bit was zero, otherwise cleared.
S: Set if bit 7 is tested and was set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
BTSET r4.5	F2 A4	1111 0010 1010 0100

If bit 5 of working register 4 is 0, after this instruction it is set to 1 and the zero flag is also set to 1.

BXOR BXOR

Bit XOR

BXOR dst.b,src.b

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode		Oper
					dst	src	
`[ OPC ] [btd,1| dst ] [bts,0| src ]`	3	10	6F	-	r.b	r.b	a
`[ OPC ] [btd,1| dst ] [bts,1| src ]`	3	10	6F	-	r.b	r.!b	b

OPERATION a: dst bit ⇐ dst bit XOR src bit

The selected bit in the source working register is XORed with the selected bit of the destination register and the result left in the destination bit. All other bits in the destination register remain unaffected. Both source and destination are directly addressed.

OPERATION b: dst bit ⇐ dst bit XOR NOT src bit

The complement of the selected bit in the source working register is XORed with the selected bit of the destination working register and the result left in the destination bit. All other bits in the destination register remain unaffected. Both source and destination are directly addressed.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
BXOR r4.5,r8.2	6F B4 48	0110 1111 1011 0100 0100 1000

If bit 2 of working register 8 is 1 and bit 5 of working register 4 is 0, after this instruction bit 5 of working register 4 will be 1.

NOTE: A bit XOR can use the same or different nibbles of the same register as both source and destination.

CALL CALL

Unconditional Call Subroutine

CALL dst

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst,1]	2	12	74	-	(RR)	-
	2	12	74	-	(rr)*	-
[OPC] [dst h] [dst l]	3	12	D2	-	NN	-

OPERATION:

SSP ⇐ SSP - 2
(SSP) ⇐ PC
PC ⇐ dst

The current contents of the program counter (PC) are pushed onto the top of the system stack. (The program counter value used is the address of the first instruction byte following the CALL instruction). The specified destination address is then loaded into the PC and points to the first instruction of the CALL procedure.

In direct memory addressing mode the destination is in the memory location addressed by the absolute value in the operand.

In the indirect memory addressing mode the destination is in the memory location addressed by the contents of the destination register pair.

FLAGS:

No flags affected.

EXAMPLE:

Instruction	HEX	Binary
CALL 3521H	D2 35 21	1101 0010 0011 0101 0010 0001

If the content of the program counter is 1A47 (hex) and the content of the system stack pointer is 3002 (hex) the above instruction will cause the stack pointer to be decremented to 3000 (hex), 1A4A (the address following the instruction) is stored in external data memory 3000 (hex) and 3001 (hex), and the program counter is loaded with 3521 (hex). The program counter now points to the address of the first statement in the procedure to be executed.

CALLS CALLS

Unconditional Far Call Subroutine

CALLS seg, dst

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	seg
[OPC] [seg] [dst h] [dst l]	4	16	3F	01xx xxxx	NN	N
[OPC] [dst,1] [seg]	3	16	73	-	(rr)	(R)
	3	16	73	-	(rr)	(r)

OPERATION:

SSP ⇐ SSP - 1
(SSP) ⇐ CSR
SSP ⇐ SSP - 2
(SSP) ⇐ PC
CSR ⇐ seg
PC ⇐ dst

This instruction uses a different stack frame, saving both the PC and the CSR on the stack.

The current contents of the CSR and program counter (PC) are pushed onto the top of the system stack. (The program counter value used is the address of the first instruction byte following the CALLS instruction). The specified destination address is then loaded into the CSR and PC and points to the first instruction of the CALLS procedure.

In direct memory addressing mode the destination is in the memory location addressed by the absolute value in the operands.

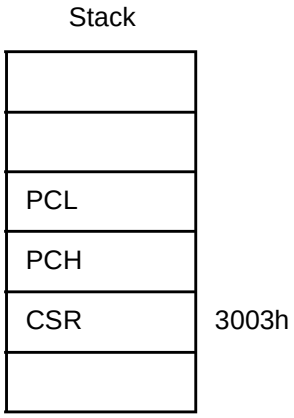
In the indirect memory addressing mode the destination is in the memory location addressed by the contents of the destination registers.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
CALLS 12H, 3521H	3F 52 35 21	0011 1111 0101 0010 0011 0101 0010 0001

If the content of the PC is 1A47 (hex), the content of CSR is 6 and the content of the system stack pointer is 3003 (hex) the above instruction will cause the stack pointer to be decremented to 3000 (hex), 1A4B (the address following the instruction) is stored in external data memory 3000 (hex) and 3001 (hex), the value 6 (CSR) is stored in 3002 (hex), CSR and PC are loaded with 12, 3521 (hex). The program counter now points to the address of the first statement in the procedure to be executed.



CCF CCF

Complement Carry Flag

CCF

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC]	1	4	61	-	-	-

OPERATION: C ⇐ NOT C

The carry flag is complemented; if C=1 it is changed to C=0 and vice-versa.

FLAGS: C: Complemented.
No other flags affected.

EXAMPLE:

Instruction	HEX	Binary
CCF	61	0110 0001

If the carry flag is set to one, after this instruction it will be reset to zero.

CLR CLR

Clear Register

CLR dst

INSTRUCTION FORMAT:

[OPC] [dst]	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
	2	4	90	-	R	-
	2	4	90	-	r	-
	2	4	91	-	(R)	-
	2	4	91	-	(r)	-

OPERATION: dst ⇐ 0

The contents of destination register, directly or indirectly addressed, is cleared to zero.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
CLR (R32)	91 20	1001 0001 0010 0000

If register 32 holds 142, after this instruction register 142 will contain 0.

CP CP

Compare (byte) Register, Register

CP dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst | src]	2	4	92	-	r	r
	2	6	93	-	r	(r)
[OPC] [src] [dst]	3	6	94	-	R	R
	3	6	94	-	r	R
	3	6	94	-	R	r
[OPC] [src] [XTN | dst]	3	6	E6	9	(r)	R
	3	6	E6	9	(r)	r
[OPC] [XTN | src] [dst]	3	6	E7	9	R	(r)

OPERATION: dst - src

The source byte is compared with (subtracted from) the destination byte and C, Z, S and V flags are affected. The destination byte remains unaffected. The source and destination byte can be addressed either directly or indirectly.

FLAGS:
C: Cleared if carry from MSB, otherwise set indicating a borrow.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
CP (r8),R255	E6 FF 98	1110 0110 1111 1111 1001 1000

If working register 8 contains 28 (decimal), register 28 contains 11001100 and register 255 contains 10000101, after this instruction: Z, C , S and V flags will be reset to zero.

CP CP

Compare (byte) Register, Memory

CP dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode		Oper
					dst	src	
[OPC] [XTN |src,1] [dst]	3	8	72	9	R	(rr)	a
	3	8	72	9	r	(rr)	a
	3	12	B4	9	R	(rr)+	b
	3	12	B4	9	r	(rr)+	b
	3	12	C2	9	R	-(rr)	c
	3	12	C2	9	r	-(rr)	c
[OPC] [ofs,1|src,0] [XTN | dst]	3	12	60	9	r	rr(rrx)	a
[OPC] [XTN |src,1] [ofs] [dst]	4	12	7F	9	R	N(rr)	a
	4	12	7F	9	r	N(rr)	a
[OPC] [XTN | dst] [src h] [src l]	4	10	C4	9	r	NN	a
[OPC] [XTN |src,0] [ofs h] [ofs l] [dst]	5	14	7F	9	R	NN(rr)	a
	5	14	7F	9	r	NN(rr)	a

OPERATION a: dst - src

The source byte is compared with (subtracted from) the destination byte and C, Z, S and V flags are affected The destination byte remains unaffected. The destination byte is held in the destination register. The source byte can be addressed directly, indirectly or by indexing.

OPERATION b: dst - src
rr ⇐ rr + 1

The source byte is compared with (subtracted from) the destination byte and C, Z, S and V flags are affected. The destination byte remains unaffected. The source byte is in the memory location addressed by the source register pair, the destination byte is in the destination register. The contents of the source register pair are incremented after the CP has been carried out.

OPERATION c: rr ⇐ rr - 1
dst - src

The source byte is compared with (subtracted from) the destination byte and C, Z, S and V flags are affected. The destination byte remains unaffected. The source byte is in the memory location addressed by the source register pair, the destination byte is in the destination register. The contents of the source register pair are decremented before the CP is carried out.

CP CP

Compare (byte) Register, Memory

CP dst,src (Cont'd)

FLAGS:
C: Cleared if carry from MSB, otherwise set indicating a borrow.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
CP R32,-(rr4)	C2 95 20	1100 0010 1001 0101 0010 0000

If register 32 contains 11001100, working register pair 4 contains 4200 (decimal) and memory location 4199 contains 11001100, after this instruction, C, S, V flags will be reset to zero, the zero flag will be set to one and working register pair 4 will contain 4199.

CP CP

Compare (byte) Memory, Register

CP dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode		Oper
					dst	src	
[OPC] [XTN |dst,0] [src]	3	10	72	9	(rr)	R	a
	3	10	72	9	(rr)	r	a
	3	12	B4	9	(rr)+	R	b
	3	12	B4	9	(rr)+	r	b
	3	12	C2	9	-(rr)	R	c
	3	12	C2	9	-(rr)	r	c
[OPC] [ofd,1|dst,1] [XTN | src]	3	12	60	9	rr(rrx)	r	a
[OPC] [XTN |dst,1] [ofd] [src]	4	12	26	9	N(rr)	R	a
	4	12	26	9	N(rr)	r	a
[OPC] [XTN | src] [dst h] [dst l]	4	10	C5	9	NN	r	a
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src]	5	14	26	9	NN(rr)	R	a
	5	14	26	9	NN(rr)	r	a

OPERATION a: dst - src

The source byte is compared with (subtracted from) the destination byte and C, Z, S and V flags are affected. The destination byte remains unaffected. The source byte is held in the source register. The destination byte can be addressed directly, indirectly or by indexing.

OPERATION b: dst - src
rr ⇐ rr + 1

The source byte is compared with (subtracted from) the destination byte and C, Z, S and V flags are affected. The destination byte remains unaffected. The source byte is in the source register, the destination byte is in the memory location addressed by the destination register pair. The contents of the destination register pair are incremented after the CP has been carried out.

OPERATION c: rr ⇐ rr - 1
dst - src

The source byte is compared with (subtracted from) the destination byte and C, Z, S and V flags are affected. The destination byte remains unaffected. The source byte is in the source register , the destination byte is in the memory location addressed by the destination register pair. The contents of the destination register pair are decremented before the CP is carried out.

CP CP

Compare (byte) Memory, Register

CP dst,src (Cont'd)

FLAGS:
C: Cleared if carry from MSB, otherwise set indicating a borrow.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
CP 4028,R8	C5 98 0F BC	1100 0101 1001 1000 0000 1111 1011 1100

If memory location 4028 contains 11001100 and working register 8 contains 10000101, after this instruction the zero flag will be reset to zero.

CP CP

Compare (byte) Memory, Memory

CP dst,src

INSTRUCTION FORMAT:

[OPC] [XTN |src,0] [dst,0]	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
	3	12	73	9	(RR)	(rr)
	3	12	73	9	(rr)*	(rr)

OPERATION: dst - src

The source byte is compared with (subtracted from) the destination byte and C, Z, S and V flags are affected. The destination byte remains unaffected. The source byte is in the memory location addressed by the source register pair, the destination byte is in the memory location addressed by the destination register pair.

FLAGS:
C: Cleared if carry from MSB, otherwise set indicating a borrow.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
CP (rr4),(rr8)	73 98 D4	0111 0011 1001 1000 1101 0100

If working register pair 4 contains 2800 (decimal), memory location 2800 contains 11001100, working register pair 8 contains 4200 (decimal) and memory location 4200 contains 11001100, after this instruction the zero flag will be set to one.

CP CP

Compare (byte) All, Immediate

CP dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [dst] [src]	3	6	95	-	R	#N
	3	6	95	-	r	#N
[OPC] [XTN |dst,0] [src]	3	8	F3	9	(rr)	#N
[OPC] [XTN] [src] [dst h] [dst l]	5	14	2F	91	NN	#N

OPERATION: dst - src

The source byte is compared with (subtracted from) the destination byte and C, Z, S and V flags are affected. The destination byte remains unaffected. The source byte is the immediate value in the operand, the destination byte can be in memory or in the register file.

FLAGS:
C: Cleared if carry from MSB, otherwise set indicating a borrow.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
CP (rr8),#32	F3 28 20	1111 0011 0010 1000 0010 0000

If working register pair 8 contains 4028 (decimal) and memory location 4028 contains 11101100, after this instruction the zero flag will be reset to zero.

CPJFI CPJFI

Compare And Jump If False Otherwise Post-Increment

CPJFI dst,src,N

INSTRUCTION FORMAT:

	No. Bytes	No.Cycl		OPC (HEX)	Addr Mode		PC offs.
		No Jmp	Jmp		dst	src	
[OPC] [src,0| dst] [PC Offset]	3	14	16	9F	r	(rr)	N

OPERATION: If compare not verified jump, otherwise increment source register pair.

The source operand is compared to (subtracted from) the destination operand. If the result is different from zero the offset N (where N is in the range -128/+127) is added to the program counter and control passes to the statement whose address is now in the PC, otherwise the source pointer is incremented by one and the instruction following the CPJFI is executed.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
CPJFI r2,(rr14),+100	9F E2 64	1001 1111 1110 0010 0110 0100

If the current value of the program counter is 340 (decimal) and working register 2 contains 11001100B, working register pair 14 contains 3000 (decimal) and memory location 3000 holds 10000100B the program counter will now point at program location 440 (decimal).

NOTE : The source value must exist within the destination area (or limit checks must be included).

CPJTI CPJTI

Compare And Jump If True Otherwise Post-Increment

CPJTI dst,src,N

INSTRUCTION FORMAT:

	No. Bytes	No.Cycl		OPC (HEX)	Addr Mode		PC offs.
		No Jmp	Jmp		dst	src	
[OPC] [src,1| dst] [PC Offset]	3	14	16	9F	r	(rr)	N

OPERATION: If compare verified jump, otherwise increment source registers pair.

The source operand is compared to (subtracted from) the destination operand. If the result is zero the offset N (where N is in the range -128/+127) is added to the program counter and control passes to the statement whose address is now in the PC, otherwise the source pointer is incremented by one and the instruction following the CPJTI is executed.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
CPJTI r2,(rr14),+100	9F F2 64	1001 1111 1111 0010 0110 0100

If the current value of the program counter is 340 (decimal) and working register 2 contains 11001100B, working register pair 14 contains 3000 (decimal) and memory location 3000 holds 11001100B the program counter will now point at program location 440 (decimal).

NOTE : The source value must exist within the destination area (or limit checks must be included).

CPL CPL

Complement Register

CPL dst

INSTRUCTION FORMAT:

[OPC] [dst]	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
	2	4	80	-	R	-
	2	4	80	-	r	-
	2	4	81	-	(R)	-
	2	4	81	-	(r)	-

OPERATION: dst ⇐ NOT dst

The contents of the destination register, directly or indirectly addressed, are one complemented (1 becomes 0 and 0 becomes 1).

FLAGS:
C: Unaffected.
Z: Set if result is zero, otherwise cleared.
S: Set if result bit 7 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
CPL (R32)	81 20	1000 0001 0010 0000

If register 32 contains 142 and register 142 holds 10101010B, after this instruction the contents of register 142 become 01010101B.

CPW CPW

Compare (Word) - Register, Register

CPW dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst,0|src,0]	2	8	9E	-	rr	rr
[OPC] [src,0] [dst,0]	3	8	97	-	RR	RR
	3	8	97	-	rr	RR
	3	8	97	-	RR	rr
[OPC] [src,0] [XTN | dst]	3	10	96	9	(r)	RR
	3	10	96	9	(r)	rr
[OPC] [XTN | src] [dst,0]	3	10	A6	9	RR	(r)
	3	10	A6	9	rr	(r)

OPERATION: dst - src

The source word is compared with (subtracted from) the destination word and the appropriate flags set. The destination remains unaltered. The source and destination word can be addressed either directly or indirectly.

FLAGS:
C: Cleared if carry from MSB of result, otherwise set.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
CPW (r8),RR64	96 40 98	1001 0110 0100 0000 1001 1000

If register pair 64 contains 11001100/11001100B, working register 8 contains 200 (decimal) and register pair 200 contains 01001000/01001000B, after this instruction the zero flag will be reset.

CPW CPW

Compare (Word) - Register, Memory

CPW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [dst,0|src,1]	2	12	9E	-	rr	(rr)	a
[OPC] [XTN |src,0] [dst,0]	3	14	7E	9	RR	(rr)	a
[OPC] [XTN |src,1] [dst,0]	3	14	D5	9	RR	(rr)+	b
	3	14	D5	9	rr	(rr)+	b
	3	14	C3	9	RR	-(rr)	c
	3	14	C3	9	rr	-(rr)	c
[OPC] [ofs,0|src,0] [XTN |dst,0]	3	14	60	9	rr	rr(rrx)	a
[OPC] [XTN |src,1] [ofs] [dst,0]	4	14	86	9	RR	N(rr)	a
	4	14	86	9	rr	N(rr)	a
[OPC] [XTN |dst,0] [src h] [src l]	4	14	E2	9	rr	NN	a
[OPC] [XTN |src,0] [ofs h] [ofs l] [dst,0]	5	16	86	9	RR	NN(rr)	a
	5	16	86	9	rr	NN(rr)	a

OPERATION a: dst - src

The source word is compared with (subtracted from) the destination word and the appropriate flags set. The destination remains unaltered. The destination word is held in the destination register. The source word can be addressed directly, indirectly or by indexing.

OPERATION b: dst - src
rr ⇐ rr + 2

The source word is compared with (subtracted from) the destination word and appropriate flags set. The destination remains unaltered. The source word is in the memory location addressed by the source register pair, the destination word is in the destination register. The contents of the source register pair are incremented after the compare has been carried out.

OPERATION c: rr ⇐ rr - 2
dst ⇐ dst - src

The source word is compared with (subtracted from) the destination word and the appropriate flags set. The destination remains unaltered. The source word is in the memory location addressed by the source register pair, the destination word is in the destination register. The contents of the source register pair are decremented before the compare is carried out.

CPW CPW

Compare (Word) - Register, Memory

CPW dst,src (Cont'd)

FLAGS:
C: Cleared if carry from MSB of result, otherwise set.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
CPW RR64,-(rr4)	C3 95 40	1100 0011 1001 0101 0100 0000

If working register pair 4 contains 1184 (decimal), register pair 64 contains 11001100/11001100B and memory pair 1182 contains 11001100/11001100B, after this instruction has been carried out the zero flag will be set and register pair 4 will contain 1182.

CPW CPW

Compare (Word) - Memory, Register

CPW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [dst,1|src,0]	2	14	9E	-	(rr)	rr	a
[OPC] [XTN |dst,1] [src,0]	3	14	BE	9	(rr)	RR	a
[OPC] [XTN |dst,0] [src,0]	3	14	D5	9	(rr)+	RR	b
	3	14	D5	9	(rr)+	rr	b
	3	14	C3	9	-(rr)	RR	c
	3	14	C3	9	-(rr)	rr	c
[OPC] [ofd,0|dst,1] [XTN |src,0]	3	14	60	9	rr(rrx)	rr	a
[OPC] [XTN |dst,1] [ofd] [src,1]	4	14	86	9	N(rr)	RR	a
	4	14	86	9	N(rr)	rr	a
[OPC] [XTN |src,1] [dst h] [dst l]	4	16	E2	9	NN	rr	a
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src,1]	5	16	86	9	NN(rr)	RR	a
	5	16	86	9	NN(rr)	rr	a

OPERATION a: dst - src

The source word is compared with (subtracted from) the destination word and the appropriate flags set. The destination remains unaltered. The source word is held in the source register. The destination word can be addressed directly, indirectly or by indexing.

OPERATION b: dst - src
rr ⇐ rr + 2

The source word is compared with (subtracted from) the destination word and the appropriate flags set. The destination remains unaltered. The source word is in the source register, the destination word is in the memory location addressed by the destination register pair. The contents of the destination register pair are incremented after the compare has been carried out.

OPERATION c: rr ⇐ rr - 2
dst - src

The source word is compared with (subtracted from) the destination word and the appropriate flags set. The destination remains unaltered. The source word is in the source register , the destination word is in the memory location addressed by the destination register pair. The contents of the destination register pair are decremented before the compare is carried out.

CPW CPW

Compare (Word) - Memory, Register

CPW dst,src (Cont'd)

FLAGS:
C: Cleared if carry from MSB of result, otherwise set.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
CPW (rr4)+,RR64	D5 94 40	1101 0101 1001 0100 0100 0000

If register pair 64 contains 11001100/11001100B, working register pair 4 contains 1064 (decimal) and memory pair 1064 contains 01001000/01001000B, after this instruction has been carried out the zero flag will be reset and working register pair 4 will contain 1066.

CPW CPW

Compare (Word) - Memory, Memory

CPW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [dst,1|src,1]	2	16	9E	-	(rr)	(rr)

OPERATION: dst - src

The source word is compared with (subtracted from) the destination word and the appropriate flags set. The destination remains unaltered. The source word is in the memory location addressed by the source register pair, the destination word is in the memory location addressed by the destination register pair.

FLAGS:
C: Cleared if carry from MSB of result, otherwise set.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
CPW (rr4),(rr6)	9E 57	1001 1110 0101 0111

If working register pair 6 contains 1002 (decimal), memory pair 1002 contains 11001100/11001100B, working register pair 4 contains 1060 (decimal) and memory pair 1060 contains 11001100/11001100B, after this instruction the zero flag will be set.

CPW CPW

Compare (word) - All, Immediate

CPW dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst,1] [src h] [src l]	4	10	97	-	RR	#NN
	4	10	97	-	rr	#NN
[OPC] [XTN |dst,0] [src h] [src l]	4	14	BE	9	(rr)	#NN
[OPC] [XTN |dst,1] [ofd] [src h] [src l]	5	16	06	9	N(rr)	#NN
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src h] [src l]	6	18	06	9	NN(rr)	#NN
[OPC] [XTN] [src h] [src l] [dst h] [dst l]	6	20	36	91	NN	#NN

OPERATION: dst - src

The source word is compared with (subtracted from) the destination word and the appropriate flags set. The destination remains unaltered. The source word is the immediate value in the operand, the destination word can be in memory or in the register file.

FLAGS:
C: Cleared if carry from MSB of result, otherwise set.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
CPW RR64,#52428	97 41 CC CC	1001 0111 0100 0001 1100 1100 1100 1100

If register pair 64 contains 01001000/01001000B, after this instruction has been carried out the zero flag will be reset to zero.

DA DA

Decimal Adjust

DA dst

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [src]	2	4	70	-	R	-
	2	4	70	-	r	-
	2	6	71	-	(R)	-
	2	6	71	-	(r)	-

OPERATION: dst ⇐ DA dst

After an addition (ADD, ADC) or subtraction (SUB, SBC), this instruction adds a number, determined by the binary result of the previous arithmetic operation, in order to convert the contents of the destination register into two 4-bit BCD digits. The following table indicates the operation performed:

Instruction	Carry before DA	Bits 4-7 value (Hex)	H Flag before DA	Bits 0-3 value (Hex)	Number added to byte	Carry after DA
	0	0-9	0	0-9	00	0
	0	0-8	0	A-F	06	0
	0	0-9	1	0-3	06	0
ADD	0	A-F	0	0-9	60	1
ADC	0	9-F	0	A-F	66	1
	0	A-F	1	0-3	66	1
	1	0-2	0	0-9	60	1
	1	0-2	0	A-F	66	1
	0	0-9	0	0-9	00	0
SUB	0	0-8	1	6-F	FA	0
SBC	1	7-F	0	0-9	A0	1
	1	6-F	1	6-F	9A	1

DA DA

Decimal Adjust

DA dst (Cont'd)

FLAGS:
C: Set if carry from MSB, otherwise cleared.(see table above)
Z: Set if result is zero, otherwise cleared.
S: Set if result bit 7 is set, otherwise clered.
V: Undefined.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
DA R32	70 20	0111 0000 0010 0000

If addition is performed using the BCD values 15 and 27, the result should be 42. The sum is incorrect, however, in the destination location when using standard binary arithmetic.

0001	0101		0011	1100	
0010	0111		0000	0110	
0011	1100	=3CH	0100	0010	=42H

The DA statement adjusts this result so that the correct BCD representation is obtained.

DEC DEC

Decrement Register

DEC dst

INSTRUCTION FORMAT:

[OPC] [dst]	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
	2	4	40	-	R	-
	2	4	40	-	r	-
	2	4	41	-	(R)	-
	2	4	41	-	(r)	-

OPERATION: dst ⇐ dst - 1

The content of destination register, directly or indirectly addressed, is decremented by 1.

FLAGS:
C: Unaffected.
Z: Set if result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
DEC (r2)	41 D2	0100 0001 1101 0010

If working register 2 holds 122 and register 122 contains 100 (decimal), after this instruction is executed register 122 will contain 99.

DECW DECW

Decrement Word Register

DECW dst

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [dst,0]	2	6	CF	-	RR	-
	2	6	CF	-	rr	-

OPERATION: dst ⇐ dst - 1

The destination register content is decremented by 1.

FLAGS:
C: Unaffected.
Z: Set if result is zero, otherwise cleared.
S: Set if result is negative, otherwise cleared.
V: Set if arithmetic overflow occurred, otherwise cleared.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
DECW rr2	CF D2	1100 1111 1101 0010

If working register pair 2 holds 2000 (decimal), after this instruction is executed it will contain 1999 (decimal).

DI DI

Disable Interrupts

DI

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC]	1	2	10	-	-	-

OPERATION: CIC.4 ⇐ 0

Bit 4 of the Central Interrupt Control register (R230) is reset to zero. All interrupts except NMI are then disabled.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
DI	10	0001 0000

After this instruction all interrupts (except NMI) are disabled.

NOTE: The NMI (Not Maskable Interrupt) can be disabled only with a general chip reset.

DIV DIV

Divide (16/8)

DIV dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [dst,0| src]	2	26/14	5F	-	rr	r

OPERATION: dst/src:dst (low) ⇐- result
dst (high) ⇐- remainder

The contents of the destination register pair are divided by the contents of the source register. The result is left in the destination register low byte and the remainder in the destination register high byte. This operation takes 26 clock cycles.
If the dividend high byte is greater than the divider, this operation takes 14 clock cycles.

Input	rr_dst	= dividend
	r_src	= divisor
Output	rr_dst high	= remainder
	rr_dst low	= result

The src byte holds the unmodified divisor.

FLAGS:
C: Set to one if divide performed correctly, reset in case of overflow.
Z: Set if result is zero, otherwise reset.
S: Set if reminder is zero, otherwise reset.
V: Undefined.
D: Always set to one.
H: Undefined.

EXAMPLE:

Instruction	HEX	Binary
DIV rr8,r6	5F 86	0101 1111 1000 0110

If working register 6 contains 30 (decimal) and working register pair 8 contains 500 (decimal), after this instruction working register 9 will contain 16 (decimal) and working register 8 will contain 20 (decimal).

DIV DIV

Divide (16/8)

DIV dst,src (Cont'd)

NOTE 1: If the dividend high is greater than or equal to the divisor the instruction is not carried out, the carry flag is reset to zero (D flag is always set to one), all other flags are undefined. This control takes 20 clock cycles and both destination and source register remain unmodified.

NOTE 2: If the divisor is zero, a trap is generated simulating a subroutine call. The current Program Counter is saved on the system stack and then the PC is set to the contents of memory locations 0002 and 0003 of the Program memory which contains the Divide-by-zero trap vector. This procedure takes 38 clock cycles.

Location 0002 ⇒ Interrupt Vector Pointer High
Location 0003 ⇒ Interrupt Vector Pointer Low

The "divide by zero attempted" subroutine should be written by the user.

Warning: The subroutine must be terminated by RET (not IRET).

DIVWS DIVWS

Divide Word Stepped (32/16)

DIVWS dsth, dstl, src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode		
					dsh	dsl	src
[OPC] [src,0] [dsh,0|dsl,0]	3	26	56	-	rr	rr	RR
	3	26	56	-	rr	rr	rr

OPERATION:

When executed 16 times and then followed by a RLCW on the destination low working register pair, this instruction carries out a 32 bit by 16 bit divide and leaves the result in the destination low working register pair and the remainder in the destination high working register pair. No automatic controls are carried out on the relationship between divisor and dividend before this instruction is carried out, nor is the divisor checked for zero, these should be supplied by the user.

FLAGS: All undefined.

EXAMPLE:

Instruction	HEX	Binary
DIVWS rr6,rr8,RR10	56 0A 68	0101 0110 0000 1010 0110 1000

Working register pair 6 will contain the 16 high order bits of the dividend, working register pair 8 will contain the 16 low order bits of the dividend and register pair 10 will contain the 16 bit divisor. After this instruction working register pair 8 will contain the result and working register pair 6 the remainder. See subroutine example.

NOTE: A typical example of a subroutine using the DIVWS instruction is shown below.

DIVWS DIVWS

Divide Word Stepped (32/16)

DIVWS dsth, dstl, src (Cont'd)

DIVSTEP SUBROUTINE EXAMPLE

This subroutine first checks that divisor is greater than the dividend high byte and that the divisor is greater than zero before carrying out the division.

```
d_len   = r0
dvsr    = RR10
dvd_hi  = rr6
dvd_low = rr8
;
;inputs:   RR10 = 16 bit divisor
;          rr6  = 32 bit dividend high
;          rr8  = 32 bit dividend low
;outputs:  RR10 = unmodified divisor
;          rr6  = remainder
;          rr8  = result
;
DIVSTEP:
       cpw dvd_hi,dvsr   ;check dividend higher than divisor
       jrug Out          ;if not leave subroutine
       cpw dvsr,#0000h   ;check divisor zero
       jrnz Defloop      ;if true start divide
Out:   ret
Defloop:
       pushu d_len          ;set 16 bit step divide loop
       ld d_len,#16
Loop:  divws dvd_hi,dvd_low,dvsr
                            ;carry out divws
       djnz d_len,Loop      ;16 times
       rlcw dvd_low
       popu d_len
       ret
```

DJNZ DJNZ

Decrement And Jump If Not Zero

DJNZ dst,N

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode		PC offs.
					dst	src	
[dst | OPC] [PC Offset]	2	6	A	-	r	-	N

OPERATION: dst ⇐ dst - 1
If dst not equal to 0 then PC⇐ PC + N

The destination working register being used as a counter is decremented. If the contents of the register are not zero after decrementing, the offset N (where N is in the range -128/+127) is added to the program counter. The original value of the program counter is taken to be the address of the instruction byte following the DJNZ instruction. When the working register counter reaches zero, control falls through to the statement following the DJNZ statement.

FLAGS: No flags affected.

EXAMPLE: DJNZ is typically used to control a “loop” of instructions. In the following example 12 bytes are moved from one area in the register file to another one. The steps involved are:

```
;load 12 into the counter (working register 6)
;set up the loop to perform the moves
;end the loop with djnz
pointer1 = oldbuf-1
pointer2 = newbuf-1

       ld   r6,#12          ;load counter
Loop:  ld   r9,pointer1(r6 ;move one byte to
       ld   pointer2(r6),r9;new location
       djnz r6,Loop         ;decrement and loop until
                            ;counter = 0
```

NOTE : Due to the ST9 architecture, the DJNZ instruction cannot be used with registers in group E or F accessed through working registers pointing to such groups, as the result of this test is undefined.

DWJNZ DWJNZ

Decrement Word And Jump If Not Zero

DWJNZ dst,N

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode		PC offs.
					dst	src	
[OPC] [dst,0] [PC Offset]	3	8	16	C6	RR	-	N
	3	8	16	C6	rr	-	N

OPERATION: dst ⇐ dst - 1
If dst not equal to 0 then PC⇐ PC + N

The destination register being used as a counter is decremented. If the contents of the register are not zero after decrementing, the offset N (where N is in the range -128/ +127) is added to the program counter. The original value of the program counter is taken to be the address of the instruction byte following the DWJNZ instruction. When the register counter reaches zero, control falls through to the statement following the DWJNZ statement.

FLAGS: No flags affected.

EXAMPLE: DWJNZ is typically used to control a “loop” of instructions. In the following example 300 bytes are moved from one area in the register file to another. The steps involved are:

```
;load 300 into the counter (working register pair 6)
;set up the loop to perform the moves
;end the loop with dwjnz
pointer1 = oldbuf-1
pointer2 = newbuf-1
       ld rr6,#300          ;load counter
Loop:  ld r9, pointer1(rr6) ;move one to byte
       ld pointer2(rr6),r9  ;new location
       dwjnz rr6,Loop       ;decrement and loop until
                            ;counter = 0
```

NOTE : Due to the ST9 architecture, the DWJNZ instruction cannot be used with registers in group E or F accessed through working registers pointing to such groups, as the result of this test is undefined.

EI EI

Enable Global Interrupts

EI

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC]	1	2	00	-	-	-

OPERATION: CIC.4 ⇐ 1

Bit 4 of the Central Interrupt Control register (R230) is set to one. All interrupts except NMI are then enabled.

FLAGS: No flag affected.

EXAMPLE:

Instruction	HEX	Binary
EI	00	0000 0000

After this instruction all interrupts (except NMI) are enabled.

NOTE: The NMI (Not Maskable Interrupt) must be separately enabled (see Technical Manual).

EXT EXT

Sign Extend

EXT dst

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [dst,1]	2	6	C6	-	RR	-
	2	6	C6	-	rr	-

OPERATION: dst(n) MSB ⇐ dst(7) LSB; where n=8,..,15

This instruction extends to the MSB register the sign bit (bit 7) of the LSB register. If bit 7 of the LSB is 1, all bits of the MSB register will be set to 1, if bit 7 of the LSB is 0, all bits of the MSB are reset. The destination is directly addressed.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
EXT RR10	C6 0B	1100 0110 0000 1011

If bit 7 of register R11 is 1, after this instruction all bits in register R10 will be 1.

HALT HALT

Halt

HALT

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [XTN]	2	inf.	BF	01	-	-

OPERATION: Stops program execution until next system reset.

FLAGS: No flags Affected.

EXAMPLE:

Instruction	HEX	Binary
HALT	BF 01	1011 1111 0000 0001

When the program encounters this instruction it is halted until a reset is executed.

INC INC

Increment Register

INC dst

INSTRUCTION FORMAT:

[OPC] [dst]	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
	2	4	50	-	R	-
	2	4	50	-	r	-
	2	4	51	-	(R)	-
	2	4	51	-	(r)	-

OPERATION: dst ⇐ dst + 1

The content of the destination register, directly or indirectly addressed, is incremented by 1.

FLAGS:
C: Unaffected.
Z: Set if result is zero, otherwise cleared.
S: Set if result is negative, otherwise cleared.
V: Set if arithmetic overflow occurred, otherwise cleared.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
INC (R32)	51 20	0101 0001 0010 0000

If register 32 holds 142 and register 142 contains 95 (decimal), after this instruction register 142 will contain 96.

INCW INCW

Register Increment Word

INCW dst

INSTRUCTION FORMAT:

[OPC] [dst,0]	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
	2	6	DF	-	RR	-
	2	6	DF	-	rr	-

OPERATION: dst ⇐ dst + 1

The destination register pair content is incremented by 1.

FLAGS:
C: Unaffected.
Z: Set if result is zero, otherwise cleared.
S: Set if result is negative, otherwise cleared.
V: Set if arithmetic overflow occurred, otherwise cleared.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
INCW RR32	DF 20	1101 1111 0010 0000

If register pair 32 contains 4000 (decimal) after this instruction it will contain 4001 (decimal).

IRET IRET

Interrupt Return

IRET

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC]	1	12/14/16	D3	-	-	-

OPERATION:

EMR2.EMCSV = 0 (ISR used):

FLAGS ⇐ (SSP)
SSP ⇐ SSP + 1
PC ⇐ (SSP)
SSP ⇐ SSP + 2
CICR.4 ⇐ 1

EMR2.EMCSV = 1 (CSR used):

FLAGS ⇐ (SSP)
SSP ⇐ SSP + 1
CSR ⇐ (SSP)
SSP ⇐ SSP + 1
PC ⇐ (SSP)
SSP ⇐ SSP + 2
CICR.4 ⇐ 1

Issued at the end of an interrupt service routine, this instruction restores the flag register and the program counter. It also re-enables any interrupts that are potentially enabled.

This instruction has a different operation if bit ENCSR of EMR2 is set. When bit ENCSR of register EMR2 is set, CSR is also pushed in the case of an interrupt, and is restored when IRET is executed.

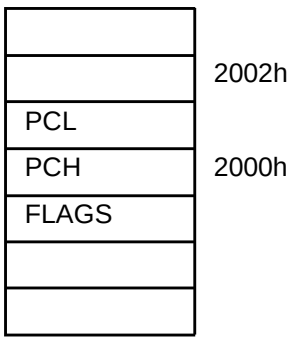
FLAGS: All flags are restored to original setting (before interrupt occurred).

EXAMPLE:

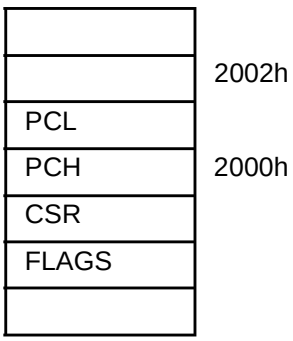
Instruction	HEX	Binary
IRET	D3	1101 0011

This instruction causes the program to resume execution exactly at the point it left when an interrupt service routine was initiated. All flags are set to the status they had when the interrupt service routine was started.

EMR2.EMCSV = 0 (ISR used):



EMR2.EMCSV = 1 (CSR used):



JP JP

Unconditional Jump

JP dst

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [dst,0]	2	8	D4	-	(RR)	-
	2	8	D4	-	(rr)*	-
[OPC] [dst h] [dst l]	3	8	8D	-	NN	-

OPERATION: PC ⇐ dst

The unconditional jump simply replaces the contents of the program counter with the destination contents. Control then passes to the statement addressed by the program counter.

The destination operand can be in a directly or indirectly addressed program memory location.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
JP 1024	8D 04 00	1000 1101 0000 0100 0000 0000

The instruction replaces the content of the program counter with 1024 (decimal) and transfers program control to that location.

JPS JPS

Unconditional Far Jump

JPS seg, dst

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	seg
[OPC] [dst,1] [seg]	3	10	73	C0	(rr)	(r)
	3	10	73	C0	(rr)	(R)
[OPC] [seg] [dst h] [dst l]	4	10	3F	C0	NN	N

OPERATION: CSR ⇐ src

PC ⇐ dst

The unconditional inter segment jump simply replaces the contents of the CSR and program counter with the destination contents. Control then passes to the statement addressed by the program counter.

The destination operand can be in a directly or indirectly addressed program memory location.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
JPS 0,1024	3F C0 04 00	0011 1111 1100 0000 0000 0100 0000 0000

The instruction replaces the content of the program counter with 1024 (decimal) and transfers program control to that location.

JPcc JPcc

Conditional Jump

JPcc dst

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[cc | OPC] [dst h] [dst l]	3	6/8	D	-	NN	-

OPERATION: If cc is true, PC ⇐ dst

The conditional jump transfers program control to the designated location if the condition code specified by “cc” is true. The destination operand is a directly addressed program memory location.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
JPEQ 1024	6D 04 00	0110 1101 0000 0100 0000 0000

If the result of the last mathematic or logic operation left the zero flag set, then the program counter is loaded with 1024 (decimal) and control is transferred to that location.

JRcc JRcc

Conditional Jump Relative

JRcc dst

INSTRUCTION FORMAT:	No. Bytes	No. Cycl		OPC (HEX)	OPC XTN	Addr Mode	
		No Jmp	Jmp			dst	src
[cc | OPC] [dst]	2	6	6	B	-	N	-

OPERATION: If cc is true, PC ⇐ PC + dst

The conditional jump adds the immediate data to the program counter and control is transferred to the new location if the condition code specified by “cc” is true. The range of the relative address is +127/-128, and the original value of the program counter is taken to be the address of the first instruction byte following the JRcc statement.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
JREQ 24	6B 18	0110 1011 0001 1000

If the result of the last mathematic or logic operation left the zero flag set then the program counter is loaded with the present value plus 24 and control is transferred to that location.

LD LD

Load (byte) Register, Register

LD dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[dst | OPC] [src]	2	4	8	-	r	R
	2	6	8	-	r	r
[src | OPC] [dst]	2	4	9	-	R	r
[OPC] [dst | src]	3	4	E6	F	(r)	r
	2	6	E4	-	r	(r)
[OPC] [src] [XTN | dst]	3	6	E6	F	(r)	R
[OPC] [XTN | src] [dst]	3	6	E7	F	R	(r)
[OPC] [src | dst] [ofd]	3	6	B2	-	N(r)	r
[OPC] [dst | src] [ofs]	3	6	B3	-	r	N(r)
[OPC] [src] [dst]	3	6	F4	-	R	R

OPERATION: dst ⇐ src

The contents of the source are loaded into the destination. The contents of the source are not affected. The source and destination can both be addressed directly, indirectly or by indexing.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
LD r8,72(r5)	B3 85 48	1011 0011 1000 0101 0100 1000

If register 5 contains 183 (decimal) and register 255 (i.e. 183+72) contains 131 (decimal), after this instruction working register 8 will contain 131.

LD LD

Load (byte) Register, Memory

LD dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode		Oper
					dst	src	
[OPC] [dst |src,0]	2	8	B5	-	r	(rr)	a
[OPC] [dst |src,1]	2	12	D7	-	(r)+	(rr)+	d
[OPC] [XTN |src,1] [dst]	3	12	B4	F	R	(rr)+	b
	3	12	B4	F	r	(rr)+	b
	3	12	C2	F	R	-(rr)	c
	3	12	C2	F	r	-(rr)	c
	3	8	72	F	R	(rr)	a
[OPC] [ofs,1|src,0] [XTN | dst]	3	12	60	F	r	rr(rrx)	a
[OPC] [XTN |src,1] [ofs] [dst]	4	12	7F	F	R	N(rr)	a
	4	12	7F	F	r	N(rr)	a
[OPC] [XTN | dst] [src h] [src l]	4	10	C4	F	r	NN	a
[OPC] [XTN |src,0] [ofs h] [ofs l] [dst]	5	14	7F	F	R	NN(rr)	a
	5	14	7F	F	r	NN(rr)	a

OPERATION a: dst ⇐ src

The destination register will be loaded with the contents of the memory location addressed either directly, indirectly or by indexing.

OPERATION b: dst ⇐ src
rr ⇐ rr + 1

The contents of the memory location addressed by the source register pair are loaded into the directly addressed destination register. The contents of the source register pair are incremented after the load has been carried out.

OPERATION c: rr ⇐ rr - 1
dst ⇐ src

The contents of the source register pair are decremented and then the contents of the memory location addressed by the source register pair are loaded into the directly addressed destination register.

OPERATION d: dst ⇐ src
r ⇐ r + 1
rr ⇐ rr + 1

The contents of the memory location addressed by the source register pair are loaded into the register addressed by the destination register. The source and destination register are incremented after the load has been carried out.

LD LD

Load (byte) Register, Memory

LD dst,src (Cont'd)

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
LD (r4)+,(rr6)+	D7 47	1101 0111 0100 0111

If working register 4 contains 100 (decimal), working register pair 6 contains 1242 (decimal) and memory location 1242 contains 132, after this instruction register 100 will contain 132, working register 4 will contain 101 and working register 6 will contain 1243.

LD LD

Load (byte) Memory, Register

LD dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode		Oper
					dst	src	
[OPC] [src |dst,0]	2	12	D7	-	(rr)+	(r)+	d
[OPC] [src |dst,1]	2	8	B5	-	(rr)	(r)	a
[OPC] [XTN |dst,0] [src]	3	12	B4	F	(rr)+	R	b
	3	12	B4	F	(rr)+	r	b
	3	12	C2	F	-(rr)	R	c
	3	12	C2	F	-(rr)	r	c
	3	10	72	F	(rr)	R	a
[OPC] [ofd,1|dst,1] [XTN | src]	3	12	60	F	rr(rrx)	r	a
[OPC] [XTN |dst,1] [ofd] [src]	4	12	26	F	N(rr)	R	a
	4	12	26	F	N(rr)	r	a
[OPC] [XTN | src] [dst h] [dst l]	4	10	C5	F	NN	r	a
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src]	5	14	26	F	NN(rr)	R	a
	5	14	26	F	NN(rr)	r	a

OPERATION a: dst ⇐ src

The data in the source register is loaded into the memory location addressed either directly, indirectly or by indexing.

OPERATION b: dst ⇐ src
rr ⇐ rr + 1

The memory location addressed by the destination register pair is loaded with the contents of the directly addressed source register. The contents of the destination register pair are incremented after the load has been carried out.

OPERATION c: rr ⇐ rr - 1
dst ⇐ src

The contents of the destination register pair are decremented and then the memory location addressed by the the destination register pair is loaded with the contents of the directly addressed source register.

OPERATION d: dst ⇐ src
r ⇐ r + 1
rr ⇐ rr + 1

The memory location addressed by the destination register pair is loaded with the contents of the register addressed by the source register. The source and destination register are incremented after the load has been carried out.

LD LD

Load (byte) Memory, Register

LD dst,src (Cont'd)

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
LD (rr4)+,(r6)+	D7 64	1101 0111 0110 0100

If working register pair 4 contains 1000 (decimal), working register 6 contains 242 (decimal) and register 242 contains 132, after this instruction memory location 1000 will contain 132, working register pair 4 will contain 1001 and working register 6 will contain 243.

LD LD

Load (Byte) Memory, Memory

LD dst,src

INSTRUCTION FORMAT:

[OPC] [XTN |src,0] [dst,0]	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
	3	10	73	F	(RR)	(rr)
	3	10	73	F	(rr)*	(rr)

OPERATION: dst ⇐ src

The contents of the memory location addressed by the source register pair are loaded into the memory location addressed by the destination register pair.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
LD (rr4),(rr6)	73 F6 D4	0111 0011 1111 0110 1101 0100

If working register pair 4 contains 1000 (decimal), working register pair 6 contains 1242 (decimal) and memory location 1242 contains 132, after this instruction memory location 1000 will contain 132.

LD LD

Load (Byte) All, Immediate

LD dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[dst | OPC] [src]	2	4	C	-	r	#N
[OPC] [dst] [src]	3	6	F5	-	R	#N
[OPC] [XTN |dst,0] [src]	3	8	F3	F	(rr)	#N
[OPC] [XTN] [src] [dst h] [dst l]	5	14	2F	F1	NN	#N

OPERATION: dst ⇐ src

The value #N is loaded into the destination register or memory location.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
LD r8,#242	8C F2	1000 1100 1111 0010

After this instruction has been carried out working register 8 contains the decimal value 242.

LDPP LDDP LDPD LDDD

Load (Byte) Data/Program Memory, Data/Program Memory

LDPP dst,src LDDP dst,src LDPD dst,src LDDD dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
LDPP: [OPC] [dst,0|src,0]	2	14	D6	-	(rr)+	(rr)+
LDDP: [OPC] [dst,1|src,0]	2	14	D6	-	(rr)+	(rr)+
LDPD: [OPC] [dst,0|src,1]	2	14	D6	-	(rr)+	(rr)+
LDDD: [OPC] [dst,1|src,1]	2	14	D6	-	(rr)+	(rr)+

OPERATION: dst ⇐ src
rrd ⇐ rrd + 1
rrs ⇐ rrs + 1

The data in the indirectly addressed memory source byte is loaded into the indirectly addressed memory destination byte. The contents of the working register pairs used to address both source and destination are incremented after the instruction has been carried out. Source and destination can be both in the data memory, both in the program memory or one can be in the data memory while the other is in the program memory.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
LDDD (rr8)+,(rr12)+	D6 9D	1101 0110 1001 1101

If working register pair 8 contains 1131 (decimal), working register pair 12 contains 2400 (decimal) and the memory location 2400 contains 100 (decimal), after this instruction memory location 1131 will contain 100, working register pair 8 will contain 1132 and working register pair 12 will contain 2401.

LDW LDW

Load (Word) Register, Register

LDW dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst,0|src,0]	2	6	E3	-	rr	rr
[OPC] [src,0] [XTN | dst]	3	6	96	F	(r)	RR
	3	6	96	F	(r)	rr
[OPC] [XTN | src] [dst,0]	3	8	A6	F	RR	(r)
	3	8	A6	F	rr	(r)
[OPC] [src,1| dst] [ofd]	3	10	DE	-	N(r)	rr
[OPC] [dst,0| src] [ofs]	3	8	DE	-	rr	N(r)
[OPC] [src,0] [dst,0]	3	6	EF	-	RR	RR
	3	6	EF	-	rr	RR
	3	6	EF	-	RR	rr

OPERATION: dst ⇐ src

The contents of the source are loaded into the destination. The contents of the source are not affected. The source and destination can be addressed directly, indirectly or by indexing.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
LDW rr8,RR254	EF FE D8	1110 1111 1111 1110 1101 1000

If register pair 254 contains 3F C1 (hex), after this instruction the working register pair 8 will contains 3F C1 (hex).

LDW LDW

Load (Word) Register, Memory

LDW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [dst,0|src,1]	2	10	E3	-	rr	(rr)	a
[OPC] [XTN |src,1] [dst,0]	3	14	D5	F	RR	(rr)+	b
	3	14	D5	F	rr	(rr)+	b
	3	14	C3	F	RR	-(rr)	c
	3	14	C3	F	rr	-(rr)	c
[OPC] [XTN |src,0] [dst,0]	3	10	7E	F	RR	(rr)	a
[OPC] [ofs,0|src,0] [XTN |dst,0]	3	14	60	F	rr	rr(rrx)	a
[OPC] [XTN |src,1] [ofs] [dst,0]	4	14	86	F	RR	N(rr)	a
	4	14	86	F	rr	N(rr)	a
[OPC] [XTN |dst,0] [src h] [src l]	4	12	E2	F	rr	NN	a
[OPC] [XTN |src,0] [ofs h] [ofs l] [dst,0]	5	16	86	F	RR	NN(rr)	a
	5	16	86	F	rr	NN(rr)	a

OPERATION a: dst ⇐ src

In the destination register pair will be loaded the contents of the memory location addressed either directly, indirectly or by indexing.

OPERATION b: dst ⇐ src
rr ⇐ rr + 2

The word in the memory pair addressed by the source register pair is loaded into the destination register pair The source address is for the word high order byte. The contents of the source register pair are incremented by two after the load has been carried out.

OPERATION c: rr ⇐ rr - 2
dst ⇐ src

The contents of the source register pair are decremented twice and then the word in the memory pair addressed by the source register pair is loaded into the destination register pair. The source address is for the word high order byte.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
LD rr8,(rr4)+	D5 F5 D8	1101 0101 1111 0101 1101 1000

If working register 4 contains 2400 (decimal) and memory pair 2400 contains 56 ED (Hex), after this instruction working register pair 8 will contain 56 ED and working register pair 4 will contain 2402.

LDW LDW

Load (Word) Memory, Register

LDW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [dst,1|src,0]	2	12	E3	-	(rr)	rr	a
[OPC] [XTN |dst,0] [src,0]	3	14	D5	F	(rr)+	RR	b
	3	14	D5	F	(rr)+	rr	b
	3	14	C3	F	-(rr)	RR	c
	3	14	C3	F	-(rr)	rr	c
[OPC] [XTN |dst,1] [src,0]	3	12	BE	F	(rr)	RR	a
[OPC] [ofd,0|dst,1] [XTN |src,0]	3	14	60	F	rr(rrx)	rr	a
[OPC] [XTN |dst,1] [ofd] [src,1]	4	14	86	F	N(rr)	RR	a
	4	14	86	F	N(rr)	rr	a
[OPC] [XTN |src,1] [dst h] [dst l]	4	14	E2	F	NN	rr	a
[OPC] [XTN |src,0] [ofd h] [ofd l] [src,1]	5	16	86	F	NN(rr)	RR	a
	5	16	86	F	NN(rr)	rr	a

OPERATION a: dst ⇐ src

The contents of the source register pair are loaded into the memory pair addressed either directly, indirectly or by indexing. The destination address is for the word high order byte.

OPERATION b: dst ⇐ src
rr ⇐ rr + 2

The contents of the source register pair are loaded into the memory pair addressed by the contents of the destination register pair. The destination address is for the word high order byte. The contents of the destination register pair are incremented twice after the load has been carried out.

OPERATION c: rr ⇐ rr - 2
dst ⇐ src

The contents of the destination register pair are decremented twice and then the contents of the source register pair are loaded into the memory pair addressed by the contents of the destination register pair. The destination address is for the word high order byte.

LDW LDW

Load (Word) Memory, Register

LDW dst,src (Cont'd)

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
LDW (rr4)+,RR64	D5 F4 40	1101 0101 1111 0100 0100 0000

If working register pair 4 contains 1024 (decimal) and register pair 64 contains 8F E3 (Hex), after this instruction memory pair 1024 will contain 8F E3 and register pair 4 will contain 1026.

LDW LDW

Load (Word) Memory, Memory

LDW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [dst,1|src,1]	2	16	E3	-	(rr)	(rr)

OPERATION: dst ⇐ src

The contents of the memory pair addressed by the source register pair are loaded into the memory pair addressed by the destination register pair. The source and destination addresses are for the word high order byte.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
LDW (rr4),(rr6)	E3 57	1110 0011 0101 0111

If working register pair 4 contains 1024 (decimal), working register pair 6 contains 2042 (decimal) and memory pair 2042 contains CB ED (Hex), after this instruction memory pair 1024 will contain CB ED.

LDW LDW

Load (Word) All, Immediate

LDW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst,0] [src h] [src l]	4	8	BF	-	RR	#NN
	4	8	BF	-	rr	#NN
[OPC] [XTN |dst,0] [src h] [src l]	4	14	BE	F	(rr)	#NN
[OPC] [XTN |dst,1] [ofd] [src h] [src l]	5	18	06	F	N(rr)	#NN
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src h] [src l]	6	20	06	F	NN(rr)	#NN
[OPC] [XTN] [src h] [src l] [dst h] [dst l]	6	18	36	F1	NN	#NN

OPERATION: dst ⇐ src

The value #NN is loaded into the destination register pair or memory pair.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
LDW RR100,#4268	BF 64 10 AC	1011 1111 0110 0100 0001 0000 1010 1100

After this instruction has been carried out register pair 100 contains the decimal value 4268 (10 AC Hex.).

LINK LINK

Link code

LINK

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst,1] [src]	3	16/12	D4	-	RR	#N
[OPC] [XTN |dst,1] [src]	3	16/12	D4	D	rr	#N

OPERATION:

Stack in memory (16 cycles)	Stack in the register file (12 cycles)
SSP = SSP - 2	SSP(low) = SSP(low) - 1
(SSP) = RR	(SSP(low)) = RR(low)
RR = SSP	RR(low) = SSP(low)
SSP = SSP - N	SSP(low) = SSP(low) - N
	SSP(high) = undefined

In C functions, the compiler needs to push variables in the system stack and to keep the return address location of the function inside the stack.

Therefore, a frame pointer is used, and 2 pieces of code named prologue and epilogue need to be added at the beginning and at the end of the function.

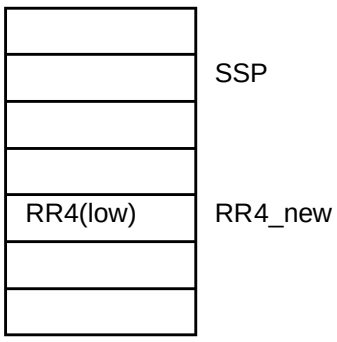
The ”Link” instruction is used to reduce the code overhead generated by the compiler inside the function.

FLAGS: No flags affected.

EXAMPLES: LINK RR4, #3.

Stack in memory (16 cycles)

Stack in the register file (12 cycles)



After the instruction, RR4 points to the location where previous RR4 has been stored.

LINKU LINKU

Link code

LINKU

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [XTN |dst,1] [src]	3	16/12	B6	D	rr	#N
[OPC] [dst,1] [src]	3	16/12	B6	-	RR	#N

OPERATION:

Stack in memory(16 cycles)	Stack in the register file (12 cycles)
USP = USP - 2	USP(low) = USP(low) - 1
(USP) = RR	(USP(low)) = RR(low)
RR = USP	RR(low) = USP(low)
USP = USP - N	USP(low) = USP(low) - N
	USP(high) = undefined

In C functions, the compiler needs to push variables in the user stack and to keep the return address location of the function inside the stack.

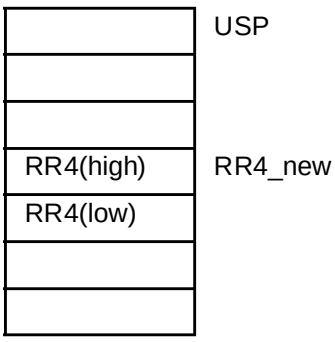
Therefore, a frame pointer is used, and 2 pieces of code named prologue and epilogue need to be added at the beginning and at the end of the function.

The ”Linku” instruction is used to reduce the code overhead generated by the compiler inside the function.

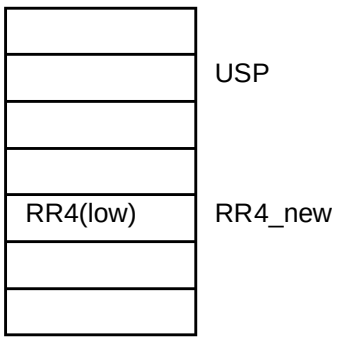
FLAGS: No flags affected.

EXAMPLES: LINKU RR4, #3.

Stack in memory (16 cycles)



Stack in the register file (12 cycles)



After the instruction, RR4 points to the location where previous RR4 has been stored.

MUL MUL

Multiply (8x8)

MUL dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [dst,0| src]	2	22	4F	-	rr	r

OPERATION: dst ⇐ dst(low) * src

The contents of the source register are multiplied by the low order byte of the destination register pair. The 16 bit result is left in the destination register pair.

Input rr dst high (even address) = don't care
rr dst low (odd address) = first operand
rr src = second operand

Output rr dst high= MSB of the result
rr dst low = LSB of the result

The src byte holds the unmodified second operand.

FLAGS:
C: Contains a copy of result bit 0.
Z: Set if result MSB is zero, otherwise cleared.
S: Contains a copy of result bit 15.
V: Set if result LSB is zero, otherwise reset.
D: Always reset to zero.
H: Undefined.

EXAMPLE:

Instruction	HEX	Binary
MUL rr6,r8	4F 68	0100 1111 0110 1000

If working register 7 contains 35 and working register 8 contains 220, after this instruction working register pair 6 will contain 7700 (decimal), i.e. working register 6 will contain 1E (Hex) and register 7 will contain 14 (Hex).

NOP NOP

No Operation

NOP

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC]	1	2	FF	-	-	-

OPERATION: No Operation is carried out. This instruction is often used in timing or delay loops.

FLAGS:No flags affected.

OR OR

OR (byte) Register, Register

OR dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst | src]	2	4	02	-	r	r
	2	6	03	-	r	(r)
[OPC] [src] [dst]	3	6	04	-	R	R
	3	6	04	-	r	R
	3	6	04	-	R	r
[OPC] [src] [XTN | dst]	3	6	E6	0	(r)	R
	3	6	E6	0	(r)	r
[OPC] [XTN | src] [dst]	3	6	E7	0	R	(r)

OPERATION: dst ⇐ dst OR src

The contents of the source are ORed with the destination byte and the results stored in the destination byte. The contents of the source are not affected.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 7 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
OR r8,R64	04 40 D8	0000 0100 0100 0000 1101 1000

If working register 8 contains 11001100 and register 64 contains 10000101, after this instruction working register 8 will contain 11001101.

OR OR

OR (byte) Register, Memory

OR dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [XTN |src,1] [dst]	3	8	72	0	R	(rr)	a
	3	8	72	0	r	(rr)	a
	3	12	B4	0	R	(rr)+	b
	3	12	B4	0	r	(rr)+	b
	3	12	C2	0	R	-(rr)	c
	3	12	C2	0	r	-(rr)	c
[OPC] [ofs,1|src,0] [XTN | dst]	3	12	60	0	r	rr(rrx)	a
[OPC] [XTN |src,1] [ofs] [dst]	4	12	7F	0	R	N(rr)	a
	4	12	7F	0	r	N(rr)	a
[OPC] [XTN | dst] [src h] [src l]	4	10	C4	0	r	NN	a
[OPC] [XTN |src,0] [ofs h] [ofs l] [dst]	5	14	7F	0	R	NN(rr)	a
	5	14	7F	0	r	NN(rr)	a

OPERATION a: dst ⇐ dst OR src

The source byte is ORed with the destination byte and the result stored in the destination byte. The destination register is addressed directly, the memory location (source byte) addressed either directly, indirectly or by indexing.

OPERATION b: dst ⇐ dst OR src
rr ⇐ rr + 1

The contents of the memory location addressed by the source register pair are ORed with the contents of the directly addressed destination register. The result is stored in the destination register. The contents of the source register pair are incremented after the OR has been carried out.

OPERATION c: rr ⇐ rr - 1
dst ⇐ dst OR src

The contents of the source register pair are decremented and then the contents of the memory location addressed by the source register pair are ORed with the contents of the directly addressed destination register. The result is stored in the destination register.

OR OR

OR (byte) Register, Memory

OR dst,src (Cont'd)

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 7 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
OR r8,4028	C4 08 0F BC	1100 0100 0000 1000 0000 1111 1011 1100

If working register 8 contains 11001100 and memory location 4028 contains 10000101, after this instruction working register 8 will contain 11001101.

OR OR

OR (byte) Memory, Register

OR dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [XTN |dst,0] [src]	3	12	72	0	(rr)	R	a
	3	12	72	0	(rr)	r	a
	3	14	B4	0	(rr)+	R	b
	3	14	B4	0	(rr)+	r	b
	3	14	C2	0	-(rr)	R	c
	3	14	C2	0	-(rr)	r	c
[OPC] [ofd,1|dst,1] [XTN | src]	3	14	60	0	rr(rrx)	r	a
[OPC] [XTN |dst,1] [ofd] [src]	4	14	26	0	N(rr)	R	a
	4	14	26	0	N(rr)	r	a
[OPC] [XTN | src] [dst h] [dst l]	4	12	C5	0	NN	r	a
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src]	5	16	26	0	NN(rr)	R	a
	5	16	26	0	NN(rr)	r	a

OPERATION a: dst ⇐ dst OR src

The source byte is ORed with the destination byte and the result stored in the destination byte. The source registers are addressed directly, the memory location are addressed either directly, indirectly or by indexing.

OPERATION b: dst ⇐ dst OR src
rr ⇐ rr + 1

The contents of the memory location addressed by the destination register pair (destination byte) are ORed with the contents of the directly addressed source register. The result is stored in the destination byte. The contents of the destination register pair are incremented after the OR has been carried out.

OPERATION c: rr ⇐ rr - 1
dst ⇐ dst OR src

The contents of the destination register pair are decremented and then the contents of the memory location addressed by the destination register pair (destination byte) are ORed with the contents of the directly addressed source register. The result is stored in the destination byte.

OR OR

OR (byte) Memory, Register

OR dst,src (Cont'd)

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 7 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
OR 4028,r8	C5 08 0F BC	1100 0101 0000 1000 0000 1111 1011 1100

If working register 8 contains 11001100 and memory location 4028 contains 10000101, after this instruction memory location 4028 will contain 11001101.

OR OR

OR (byte) Memory, Memory

OR dst,src

INSTRUCTION FORMAT:

[OPC] [XTN |src,0] [dst,0]	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
	3	14	73	0	(RR)	(rr)
	3	14	73	0	(rr)	(rr)

OPERATION: dst ⇐ dst OR src

The contents of the memory location addressed by the source register pair are ORed with the content of the memory location addressed by the destination register pair. The source and destination addresses are for the word high order byte.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 7 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
OR (rr4),(rr8)	73 08 D4	0111 0011 0000 1000 1101 0100

If working register pair 4 contains 2800 (decimal), memory location 2800 contains 11001100, working register pair 8 contains 4200 (decimal) and memory location 4200 contains 00001100, after this instruction memory location 2800 will contain 11001100.

OR OR

OR (byte) All, Immediate

OR dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst] [src]	3	6	05	-	R	#N
	3	6	05	-	r	#N
[OPC] [XTN |dst,0] [src]	3	10	F3	0	(rr)	#N
[OPC] [XTN] [src] [dst h] [dst l]	5	16	2F	01	NN	#N

OPERATION: dst ⇐ dst OR src

The value #N is ORed with the content of the destination register or memory location.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 7 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
OR (rr8),#32	F3 18 20	1111 0011 0001 1000 0010 0000

If working register pair 8 contains 4028 (decimal) and memory location 4028 contains 11101101, after this instruction memory location 4028 will contain 11101101.

ORW ORW

OR (Word) - Register, Register

ORW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [dst,0|src,0]	2	8	0E	-	rr	rr
[OPC] [src,0] [dst,0]	3	8	07	-	RR	RR
	3	8	07	-	rr	RR
	3	8	07	-	RR	rr
[OPC] [src,0] [XTN | dst]	3	10	96	0	(r)	RR
	3	10	96	0	(r)	rr
[OPC] [XTN | src] [dst,0]	3	10	A6	0	RR	(r)
	3	10	A6	0	rr	(r)

OPERATION: dst ⇐ dst OR src

The source word is ORed with the destination word and the result is stored in the destination word. The source and destination word can be addressed either directly or indirectly.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 15 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
ORW (r8),RR64	96 40 08	1001 0110 0100 0000 0000 1000

If register pair 64 contains 11001100/11001100B, working register 8 contains 200 (decimal) and register pair 200 contains 10101010/10101010B, after this instruction register pair 200 will hold 11101110/11101110B.

ORW ORW

OR (Word) - Register, Memory

ORW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [dst,0|src,1]	2	12	0E	-	rr	(rr)	a
[OPC] [XTN |src,0] [dst,0]	3	12	7E	0	RR	(rr)	a
[OPC] [XTN |src,1] [dst,0]	3	14	D5	0	RR	(rr)+	b
	3	14	D5	0	rr	(rr)+	b
	3	14	C3	0	RR	-(rr)	c
	3	14	C3	0	rr	-(rr)	c
[OPC] [ofs,0|src,0] [XTN |dst,0]	3	14	60	0	rr	rr(rrx)	a
[OPC] [XTN |src,1] [ofs] [dst,0]	4	14	86	0	RR	N(rr)	a
	4	14	86	0	rr	N(rr)	a
[OPC] [XTN |dst,0] [src h] [src l]	4	14	E2	0	rr	NN	a
[OPC] [XTN |src,0] [ofs h] [ofs l] [dst,0]	5	16	86	0	RR	NN(rr)	a
	5	16	86	0	rr	NN(rr)	a

OPERATION a: dst ⇐ dst OR src

The source word is ORed with the destination word and the result is stored in the destination word. The destination word is held in the destination register. The source word can be addressed directly, indirectly or by indexing.

OPERATION b: dst ⇐ dst OR src
rr ⇐ rr + 2

The source word is ORed with the destination word and the result is stored in the destination word. The source word is in the memory location addressed by the source register pair, the destination word is in the destination register. The contents of the source register pair are incremented after the OR has been carried out.

OPERATION c: rr ⇐ rr - 2
dst ⇐ dst OR src

The source word is ORed with the destination word and the result is stored in the destination word. The source word is in the memory location addressed by the source register pair, the destination word is in the destination register. The contents of the source register pair are decremented before the OR is carried out.

ORW ORW

OR (Word) - Register, Memory

ORW dst,src (Cont'd)

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 15 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
ORW RR64,-(rr4)	C3 05 40	1100 0011 0000 0101 0100 0000

If working register pair 4 contains 1184 (decimal), register pair 64 contains 10101010/10101010B and memory pair 1182 contains 11001100/11001100B, after this instruction register pair 64 will contain 11101110/11101110B and register pair 4 will contain 1182.

ORW ORW

OR (Word) - Memory, Register

ORW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [dst,1|src,0]	2	16	0E	-	(rr)	rr	a
[OPC] [XTN |dst,1] [src,0]	3	18	BE	0	(rr)	RR	a
[OPC] [XTN |dst,0] [src,0]	3	18	D5	0	(rr)+	RR	b
	3	18	D5	0	(rr)+	rr	b
	3	18	C3	0	-(rr)	RR	c
	3	18	C3	0	-(rr)	rr	c
[OPC] [ofd,0|dst,1] [XTN |src,0]	3	18	60	0	rr(rrx)	rr	a
[OPC] [XTN |dst,1] [ofd] [src,1]	4	18	86	0	N(rr)	RR	a
	4	18	86	0	N(rr)	rr	a
[OPC] [XTN |src,1] [dst h] [dst l]	4	18	E2	0	NN	rr	a
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src,1]	5	20	86	0	NN(rr)	RR	a
	5	20	86	0	NN(rr)	rr	a

OPERATION a: dst ⇐ dst OR src

The source word is ORed with the destination word and the result is stored in the destination word. The source word is held in the source register. The destination word can be addressed directly, indirectly or by indexing.

OPERATION b: dst ⇐ dst OR src
rr ⇐ rr + 2

The source word is ORed with the destination word and the result is stored in the destination word. The source word is in the source register, the destination word is in the memory location addressed by the destination register pair. The contents of the destination register pair are incremented after the OR has been carried out.

OPERATION c: rr ⇐ rr - 2
dst ⇐ dst OR src

The source word is ORed with the destination word and the result is stored in the destination word. The source word is in the source register , the destination word is in the memory location addressed by the destination register pair. The contents of the destination register pair are decremented before the OR is carried out.

ORW ORW

OR (Word) - Memory, Register

ORW dst,src (Cont'd)

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 15 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
ORW (rr4)+,RR64	D5 04 40	1101 0101 0000 0100 0100 0000

If register pair 64 contains 11001100/1101100B, working register pair 4 contains 1064 (decimal) and memory pair 1064 contains 10101010/10101010B, after this instruction has been carried out memory pair 1064 will contain 11101110/ 11101110B and working register pair 4 will contain 1066.

ORW ORW

OR (Word) - Memory, Memory

ORW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [dst,1|src,1]	2	20	0E	-	(rr)	(rr)

OPERATION: dst ⇐ dst OR src

The source word is ORed with the destination word and the result is stored in the destination word. The source word is in the memory location addressed by the source register pair, the destination word is in the memory location addressed by the destination register pair.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 15 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
ORW (rr4),(rr6)	0E 57	0000 1110 0101 0111

If working register pair 6 contains 1002 (decimal), memory pair 1002 contains 11001100/11001100B, working register pair 4 contains 1060 (decimal) and memory pair 1060 contains 10101010/10101010B, after this instruction memory pair 1060 will contain 11101110/11101110B.

ORW ORW

OR (Word) - All, Immediate

ORW dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst,1] [src h] [src l]	4	10	07	-	RR	#NN
	4	10	07	-	rr	#NN
[OPC] [XTN |dst,0] [src h] [src l]	4	18	BE	0	(rr)	#NN
[OPC] [XTN |dst,1] [ofd] [src h] [src l]	5	20	06	0	N(rr)	#NN
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src h] [src l]	6	22	06	0	NN(rr)	#NN
[OPC] [XTN] [src h] [src l] [dst h] [dst l]	6	22	36	01	NN	#NN

OPERATION: dst ⇐ dst OR src

The source word is ORed with the destination word and the result is stored in the destination word. The source word is the immediate value in the operand, the destination word can be in memory or in the register file.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 15 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
ORW RR64,#52428	07 41 CC CC	0000 0111 0100 0001 1100 1100 1100 1100

If register pair 64 contains 10101010/10101010B, after this instruction has been carried out register pair 64 will contain 11101110/11101110B.

PEA PEA

Push Effective Address on System Stack

PEA src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [XTN] [src,0] [ofs]	4	20	8F	01	-	N(RR)
	4	20	8F	01	-	N(rr)*
[OPC] [XTN] [src,1] [ofs l] [ofs h]	5	22	8F	01	-	NN(RR)
	5	22	8F	01	-	NN(rr)*

OPERATION: SSP ⇐ SSP - 2
(SSP) ⇐ RR + “a” (Where “a” is the immediate value N or NN)

The present value of the SSP is decremented by 2 and the content of the source register pair summed with the offset is pushed onto the system stack.

FLAGS: No flag affected.

EXAMPLE:

Instruction	HEX	Binary
PEA 16(RR32)	8F 01 20 10	1000 1111 0000 0001 0010 0000 0001 0000

The content of register pair RR32 is 1024, to this value is added the immediate value 16 and the result is pushed into the stack location pointed by the pre-decremented system stack pointer.

PEAU PEAU

Push Effective Address on User Stack

PEAU src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [XTN] [src,0] [ofs]	4	20	8F	03	-	N(RR)
	4	20	8F	03	-	N(rr)*
[OPC] [XTN] [src,1] [ofs l] [ofs h]	5	22	8F	03	-	NN(RR)
	5	22	8F	03	-	NN(rr)*

OPERATION: USP ⇐ USP - 2
(USP) ⇐ RR + “a” (Where “a” is the immediate value N or NN)

The present value of the USP is decremented by 2 and the contents of the source register pair summed with the offset is pushed into the user stack.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
PEAU 16(RR32)	8F 03 20 10	1000 1111 0000 0011 0010 0000 0001 0000

The content of register pair RR32 is 1024, to this value is added the immediate value 16 and the result is pushed into the stack location pointed by the pre-decremented user stack pointer.

POP POP

Pop Byte from System Stack

POP dst

INSTRUCTION FORMAT:

[OPC] [dst]	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
	2	8	76	-	R	-
	2	8	76	-	r	-
	2	8	77	-	(R)	-
	2	8	77	-	(r)	-

OPERATION: dst ⇐ (SSP)
SSP ⇐ SSP + 1

The contents of the system stack addressed by the system stack pointer are loaded into the destination location and then the system stack pointer is incremented automatically by one.

FLAGS: No flags affected

EXAMPLE:

Instruction	HEX	Binary
POP (r2)	77 D2	0111 0111 1101 0010

If the system stack pointer contains 2000 (decimal), working register 2 contains 52 (decimal) and system stack location 2000 contains 124 (decimal), after this instruction register 52 will contain 124 and the system stack pointer will contain 2001.

POPU POPU

Pop Byte from User Stack

POPU dst

INSTRUCTION FORMAT:

[OPC] [dst]	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
	2	8	20	-	R	-
	2	8	20	-	r	-
	2	8	21	-	(R)	-
	2	8	21	-	(r)	-

OPERATION: dst ⇐ (USP)
USP ⇐ USP + 1

The contents of the user stack addressed by the user stack pointer are loaded into the destination location and then the user stack pointer is increment automatically by one.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
POPU (r2)	21 D2	0010 0001 1101 0010

If the user stack pointer contains 2000 (decimal), working register 2 contains 52 (decimal) and user stack location 2000 contains 124 (decimal), after this instruction register 52 will contain 124 and the user stack pointer will contain 2001.

POPUW POPUW

Pop Word from User Stack

POPUW dst

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [dst,0]	2	10	B7	-	RR	-
	2	10	B7	-	rr	-

OPERATION: dst ⇐ (USP)
USP ⇐ USP + 2

The contents of the user stack addressed by the user stack pointer are loaded into the destination register pair and the user stack pointer is automatically increment by two.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
POPUW rr2	B7 D2	1011 0111 1101 0010

If the user stack pointer contains 2000 (decimal), user stack location 2000 contains 11 (hex) and user stack location 2001 contains 24 (hex), after this instruction working register 2 will contain 11 (hex), working register 3 will contain 24 (hex) and the user stack pointer will contain 2002.

POPW POPW

Pop Word from System Stack

POPW dst

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [dst,0]	2	10	75	-	RR	-
	2	10	75	-	rr	-

OPERATION: dst ⇐ (SSP)
SSP ⇐ SSP + 2

The contents of the system stack pointer are loaded into the destination register pair and the system stack pointer is automatically incremented by two.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
POPW rr2	75 D2	0111 0101 1101 0010

If the system stack pointer contains 2000 (decimal), system stack location 2000 contains 11 (hex), system stack location 2001 contains 24 (hex), after this instruction working register 2 will contain 11 (hex), working register 3 will contain 24 (hex) and the system stack pointer will contain 2002.

PUSH PUSH

Push Byte on System Stack

PUSH src

INSTRUCTION FORMAT:

[OPC] [src]	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
	2	8	66	-	-	R
	2	8	66	-	-	r
	2	8	F7	-	-	(R)
	2	8	F7	-	-	(r)
[OPC] [XTN] [src]	3	12	8F	F1	-	#N

OPERATION: SSP ⇐ SSP - 1
(SSP) ⇐ src

The system stack pointer is decremented automatically by one and then the operand loaded into the location addressed by the decremented system stack pointer.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
PUSH (R32)	F7 20	1111 0111 0010 0000

If the system stack pointer contains 2000 (decimal), register 32 contains 100 and register 100 contains 60 (decimal), after this instruction system stack pointer location 1999 will contain 60.

PUSHU PUSHU

Push Byte on User Stack

PUSHU src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [src]	2	8	30	-	-	R
	2	8	30	-	-	r
	2	8	31	-	-	(R)
	2	8	31	-	-	(r)
[OPC] [XTN] [src]	3	12	8F	F3	-	#N

OPERATION: USP ⇐ USP - 1
(USP) ⇐ src

The user stack pointer is decremented automatically by one and then the contents of the source operand loaded into the location addressed by the decremented user stack pointer.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
PUSHU #20	8F F3 14	1000 1111 1111 0011 0001 0100

If the user stack pointer contains 2000 (decimal), after this instruction user stack pointer location 1999 will contain 20.

PUSHUW PUSHUW

Push Word on User Stack

PUSHUW src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [src,0]	2	8/10	B6	-	-	RR
	2	8/10	B6	-	-	rr
[OPC] [XTN] [src h] [src l]	4	16	8F	C3	-	#NN

OPERATION: USP ⇐ USP - 2
(USP) ⇐ src

The user stack pointer is automatically decremented by two and then the contents of the source operand is loaded into the user stack.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
PUSHUW RR32	B6 20	1011 0110 0010 0000

If the stack pointer contains 2000 (decimal) and register pair 32 contains 6000 (hex), after this instruction the user stack pointer will contain 1998, user stack location 1999 will contain 00 (hex) and user stack location 1998 will contain 60 (hex).

NOTE: See also PEAUW instruction.

PUSHW PUSHW

Push Word on System Stack

PUSHW src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [src,0]	2	8/10	74	-	-	RR
	2	8/10	74	-	-	rr
[OPC] [XTN] [src h] [src l]	4	16	8F	C1	-	#NN

OPERATION: SSP ⇐ SSP - 2
(SSP) ⇐ src

The system stack pointer is automatically decremented by two and then the contents of the source register pair is loaded into the system stack.

FLAGS: No flag affected.

EXAMPLE:

Instruction	HEX	Binary
PUSHW RR32	74 20	0111 0100 0010 0000

If the system stack pointer contains 2000 (decimal) and register pair 32 contains 6000 (hex), aftert this instruction the system stack pointer will contain 1998, system stack location 1999 will contain 00 (hex) and system stack location 1998 will contain 60 (hex).

NOTE: See also PEAW instruction.

RCF RCF

Reset Carry Flag

RCF

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode	
					dst	src
[OPC]	1	4	11	-	-	-

OPERATION: C ⇐ 0

The carry flag is reset to zero, regardless of its previous content.

FLAGS: C: reset to zero.

No other flags affected.

EXAMPLE:

Instruction	HEX	Binary
RCF	11	0001 0001

Regardless of its prior condition, after this instruction the carry flag will be reset to zero.

RET RET

Return From Subroutine

RET

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode	
					dst	src
[OPC]	1	8/10	46	-	-	-

OPERATION: PC ⇐ (SSP)
SSP ⇐ SSP + 2

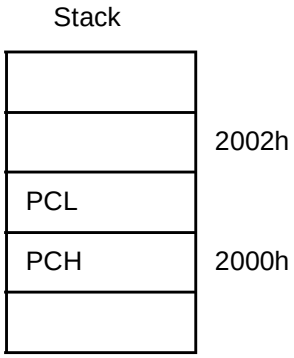
This instruction is normally used to return to the previously executed procedure at the end of procedure entered by a CALL statement. The contents of the location addressed by the system stack pointer are popped into the program counter. The next statement executed is that addressed by the new content of the PC.

FLAGS: No flags affected.

EXAMPLE : If the program counter contains 35B4 (hex), the system stack pointer contains 2000 (hex), external data memory location 2000 (hex) contains 18 (hex), and location 2001 (hex) contains 85 (hex), then the instruction:

RET

leaves the value 2002 (hex) in the system stack pointer and 1885 (hex), the addressed of the next instruction, in the program counter.



RETS RETS

Return From Far Subroutine

RETS

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode	
					dst	src
[OPC]	2	12/10*	F6	01	-	-

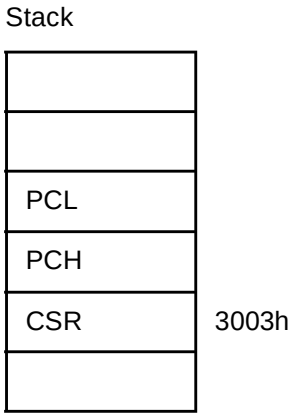
* depends if stack is in memory/register file

OPERATION: PC ⇐ (SSP)
SSP ⇐ SSP+2
CSR ⇐ (SSP)
SSP ⇐ SSP+1

This instruction is normally used to return to the previously executed procedure at the end of procedure entered by a CALLS statement. The contents of the location addressed by the system stack pointer are popped into the CSR and PC registers. The next statement executed is that addressed by the new content of the PC.

FLAGS: No flags affected.

EXAMPLE : If the program counter contains 35B4 (hex), the system stack pointer contains 2000 (hex), external data memory 2000 (hex) contains 18 (hex), and data memory pair 2001 (hex) contains 85A1 (hex), after the instruction CSR contains 18 (hex), PC points to instruction address 85A1 (hex) and the system stack pointer contains 2003 (hex).



RLC RLC

Rotate Left Through Carry

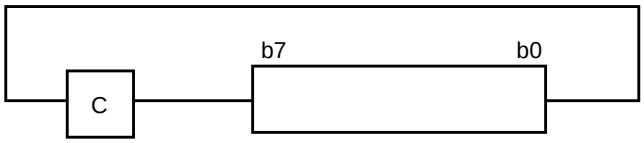
RLC dst

INSTRUCTION FORMAT:

[OPC] [dst]	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src
	2	4	B0	-	R	-
	2	4	B0	-	r	-
	2	6	B1	-	(R)	-
	2	6	B1	-	(r)	-

OPERATION: dst(0) ⇐ C
C ⇐ dst(7)
dst(n+1) ⇐ dst(n) Where n=0-6

The contents of the destination register are shifted one place to the left with bit 7 shifted into the carry flag and the carry flag shifted into bit 0. The destination register can be directly or indirectly addressed.



FLAGS:

C: Set if carry from MSB (bit 7 was 1).

Z: Set if the result is zero, otherwise cleared.

S: Set if the result bit 7 is set, otherwise cleared.

V: Set if result bit 7 is changed, otherwise cleared.

D: Unaffected.

H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
RLC (r2)	B1 D2	1011 0001 1101 0010

If the carry flag is zero, working register 2 contains 155 (decimal) and register 155 contains 11001100B, after this instruction register 155 will contain 10011000B and the carry flag will be set to 1.

RLCW RLCW

Rotate Left Through Carry Word

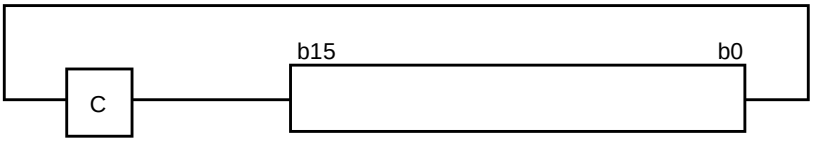
RLCW dst

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode	
					dst	src
[OPC] [dst,0]	2	8	8F	-	RR	-
	2	8	8F	-	rr	-

OPERATION: dst(0) ⇐ C
C ⇐ dst(15)
dst(n+1) ⇐ dst(n) where n=0-14

The contents of the destination register pair are shifted one place to the left with bit 15 shifted into the carry flag and the carry flag shifted into bit 0.



FLAGS:

C: Set if carry from MSB bit 15 was 1.

Z: Undefined.

S: Set if the result bit 15 is set, otherwise cleared.

V: Set if result bit 15 is changed, otherwise cleared.

D: Unaffected.

H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
RLCW rr2	8F D2	1000 1111 1101 0010

If the carry flag is zero, and working register pair 2 contains 11001100/11001100B, after this instruction it will 10011001/10011000B and the carry flag will be set to 1.

ROL ROL

Rotate Left Byte

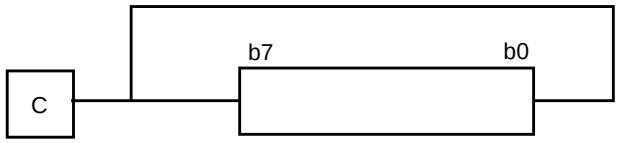
ROL dst

INSTRUCTION FORMAT:

[OPC] [dst]	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src
	2	4	A0	-	R	-
	2	4	A0	-	r	-
	2	6	A1	-	(R)	-
	2	6	A1	-	(r)	-

OPERATION: C ⇐ dst(7)
dst(0) ⇐ dst(7)
dst(n+1) ⇐ dst(n) Where n=0-6

The contents of the destination register are shifted one place to the left with bit 7 shifted into bit 1 and into the carry flag. The destination register can be directly or indirectly addressed.



FLAGS:

C: Set if carry from MSB (bit 7 was 1).

Z: Set if the result is zero, otherwise cleared.

S: Set if the result bit 7 is set, otherwise cleared.

V: Set if result bit 7 is changed, otherwise cleared.

D: Unaffected.

H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
ROL (r2)	A1 D2	1010 0001 1101 0010

If working register 2 contains 146 (decimal) and register 146 contains 11001100B, after this instruction register 146 will contain 10011001B and the carry flag will be set to 1.

ROR ROR

Rotate Right Byte

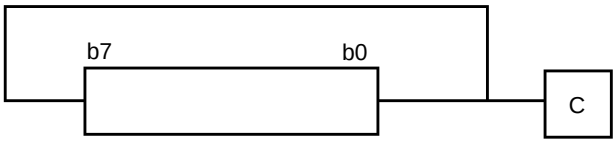
ROR dst

INSTRUCTION FORMAT:

[OPC] [dst]	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src
	2	4	C0	-	R	-
	2	4	C0	-	r	-
	2	6	C1	-	(R)	-
	2	6	C1	-	(r)	-

OPERATION: C ⇐ dst(0)
dst(7) ⇐ dst(0)
dst(n) ⇐ dst(n+1) Where n=0-6

The contents of the destination register are shifted one place to the right with bit 0 shifted into bit 7 and into the carry flag. The destination register can be directly or indirectly addressed.



FLAGS:

C: Set if carry from LSB (bit 0 was 1).

Z: Set if the result is zero, otherwise cleared.

S: Set if the result bit 7 is set, otherwise cleared.

V: Set if result bit 7 is changed, otherwise cleared.

D: Unaffected.

H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
ROR R32	C0 20	1100 0000 0010 0000

If the carry flag is set to one and register 32 contains 11001100B, after this instruction register 32 will contain 01100110B and the carry flag will be reset to zero.

RRC RRC

Rotate Right Through Carry Byte

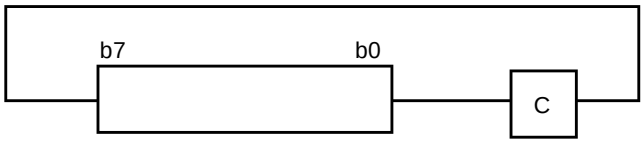
RRC dst

INSTRUCTION FORMAT:

[OPC] [dst]	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src
	2	4	D0	-	R	-
	2	4	D0	-	r	-
	2	6	D1	-	(R)	-
	2	6	D1	-	(r)	-

OPERATION: dst(7) ⇐ C
C ⇐ dst(0)
dst(n) ⇐ dst(n+1) Where n=0-6

The contents of the destination register are shifted one place to the right with bit 0 shifted into the carry flag and the carry flag shifted into bit 7. The destination register can be directly or indirectly addressed.



FLAGS:

C: Set if carry from LSB (bit 0 was 1).

Z: Set if the result is zero, otherwise cleared.

S: Set if the result bit 7 is set, otherwise cleared.

V: Set if result bit 7 is changed, otherwise cleared.

D: Unaffected.

H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
RRC (R32)	D1 20	1101 0001 0010 0000

If the carry flag is zero, register 32 contains 155 and register 155 contains 00110011B, after this instruction register 155 will contain 00011001B and the carry flag will be set to 1.

RRCW RRCW

Rotate Right Through Carry Word

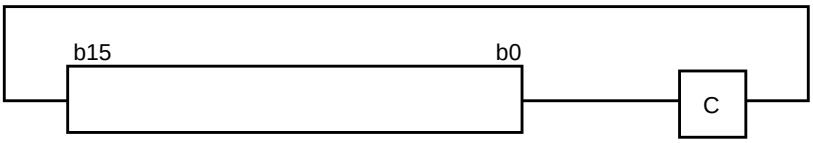
RRCW dst

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode	
					dst	src
`[ OPC ] [ dst,0 ]`	2	8	36	-	RR	-
	2	8	36	-	rr	-

OPERATION: dst(15) ⇐ C
C ⇐ dst(0)
dst(n) ⇐ dst(n+1) where n=0-14

The contents of the destination register pair are shifted one place to the right with bit 0 shifted into the carry flag and the carry flag shifted into bit 15.

FLAGS: C: Set if carry from LSB (bit 0 was 1).



Z: Undefined.

S: Set if the result bit 15 is set, otherwise cleared.

V: Set if result bit 15 is changed, cleared otherwise.

D: Unaffected.

H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
RRCW R32	36 20	0011 0110 0010 0000

If the carry flag is set and register 32 pair contains 11001100/11001100B, after this instruction register 32 will contain 11100110/01100110B and the zero flag will be reset to 0.

SBC SBC

Subtract with carry (byte) Register, Register

SBC dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [dst | src]	2	4	22	-	r	r
	2	6	23	-	r	(r)
[OPC] [src] [dst]	3	6	24	-	R	R
	3	6	24	-	r	R
	3	6	24	-	R	r
[OPC] [src] [XTN | dst]	3	6	E6	-	(r)	R
	3	6	E6	2	(r)	r
[OPC] [XTN | src] [dst]	3	6	E7	2	R	(r)

OPERATION: dst ⇐ dst - src - C

The source byte, along with the carry, is subtracted from the destination byte and the result is stored in the destination byte. The source and destination byte can be addressed either directly or indirectly.

FLAGS:
C: Cleared if carry from MSB of result, set otherwise indicating a borrow.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Always reset to one.
H: Cleared if carry from low-order nibble occurred.

EXAMPLE:

Instruction	HEX	Binary
SBC r8,(r4)	23 84	0010 0011 1000 0100

If the carry flag is reset, working register 8 contains 100 (decimal), working register 4 contains 200 (decimal) and register 200 contains 25 (decimal), after this instruction working register 8 will contain 75.

SBC SBC

Subtract with carry (byte) Register, Memory

SBC dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [XTN |src,1] [dst]	3	8	72	2	R	(rr)	a
	3	8	72	2	r	(rr)	a
	3	12	B4	2	R	(rr)+	b
	3	12	B4	2	r	(rr)+	b
	3	12	C2	2	R	-(rr)	c
	3	12	C2	2	r	-(rr)	c
[OPC] [ofs,1|src,0] [XTN | dst]	3	12	60	2	r	rr(rrx)	a
[OPC] [XTN |src,1] [ofs] [dst]	4	12	7F	2	R	N(rr)	a
	4	12	7F	2	r	N(rr)	a
[OPC] [XTN | dst] [src h] [src l]	4	10	C4	2	r	NN	a
[OPC] [XTN |src,0] [ofs h] [ofs l] [dst]	5	14	7F	2	R	NN(rr)	a
	5	14	7F	2	r	NN(rr)	a

OPERATION a: dst ⇐ dst - src - C

The source byte, along with the carry, is subtracted from the destination byte and the result is stored in the destination byte. The destination byte is held in the destination register. The source byte can be addressed directly, indirectly or by indexing.

OPERATION b: dst ⇐ dst - src - C
rr ⇐ rr + 1

The source byte, along with the carry, is subtracted from the destination byte and the result is stored in the destination byte. The source byte is in the memory location addressed by the source register pair, the destination byte is in the destination register. The contents of the source register pair are incremented after the SBC has been carried out.

OPERATION c: rr ⇐ rr - 1
dst ⇐ dst - src - C

The source byte, along with the carry, is subtracted from the destination byte and the result is stored in the destination byte. The source byte is in the memory location addressed by the source register pair, the destination byte is in the destination register. The contents of the source register pair are decremented before the SBC is carried out.

SBC SBC

Subtract with carry (byte) Register, Memory

SBC dst,src (Cont'd)

FLAGS:
C: Cleared if carry from MSB of result, set otherwise indicating a borrow.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Always reset to one.
H: Cleared if carry from low-order nibble occurred.

EXAMPLE:

Instruction	HEX	Binary
SBC r8,6(rr4)	7F 25 06 D4	0111 1111 0010 0101 0000 0110 1101 1000

If the carry flag is set, working register 8 contains 110 (decimal), working register pair 4 contain 4200 (decimal) and memory address 4204 (decimal) contains 10 (decimal), after this instruction working register 8 contains 99 (decimal).

SBC SBC

Subtract with carry (byte) Memory, Register

SBC dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [XTN |dst,0] [src]	3	12	72	2	(rr)	R	a
	3	12	72	2	(rr)	r	a
	3	14	B4	2	(rr)+	R	b
	3	14	B4	2	(rr)+	r	b
	3	14	C2	2	-(rr)	R	c
	3	14	C2	2	-(rr)	r	c
[OPC] [ofd,1|dst,1] [XTN | src]	3	14	60	2	rr(rrx)	r	a
[OPC] [XTN |dst,1] [ofd] [src]	4	14	26	2	N(rr)	R	a
	4	14	26	2	N(rr)	r	a
[OPC] [XTN | src] [dst h] [dst l]	4	12	C5	2	NN	r	a
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src]	5	16	26	2	NN(rr)	R	a
	5	16	26	2	NN(rr)	r	a

OPERATION a: dst ⇐ dst - src - C

The source byte, along with the carry, is subtracted from destination byte and the result is stored in the destination byte. The source byte is held in the source register. The destination byte can be addressed directly, indirectly or by indexing.

OPERATION b: dst ⇐ dst - src - C
rr ⇐ rr + 1

The source byte, along with the carry, is subtracted from the destination byte and the result is stored in the destination byte. The source byte is in the source register, the destination byte is in the memory location addressed by the destination register pair. The contents of the destination register pair are incremented after the SBC has been carried out.

OPERATION c: rr ⇐ rr - 1
dst ⇐ dst - src - C

The source byte, along with the carry, is subtracted from the destination byte and the result is stored in the destination byte. The source byte is in the source register , the destination byte is in the memory location addressed by the destination register pair. The contents of the destination register pair are decremented before the SBC is carried out.

SBC SBC

Subtract with carry (byte) Memory, Register

SBC dst,src (Cont'd)

FLAGS:
C: Cleared if carry from MSB of result, set otherwise indicating a borrow.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Always reset to one.
H: Cleared if carry from low-order nibble occurred.

EXAMPLE:

Instruction	HEX	Binary
SBC (rr8)+,R255	B4 28 FF	1011 0100 0010 1000 1111 1111

If the carry flag is set, working register pair 8 contains 4028 (decimal) memory location 4028 contains 110 (decimal) and register 255 contains 101 (decimal), after this instruction memory location 4028 will contain 8 and working register pair 8 will contain 4029.

SBC SBC

Subtract with carry (byte) Memory, Memory

SBC dst,src

INSTRUCTION FORMAT:

[OPC] [XTN |src,0] [dst,0]	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
	3	14	73	2	(RR)	(rr)
	3	14	73	2	(rr)*	(rr)

OPERATION: dst ⇐ dst - src - C

The source byte, along with the carry, is subtracted from the destination byte and the result is stored in the destination byte. The source byte is in the memory location addressed by the source register pair, the destination byte is in the memory location addressed by the destination register pair.

FLAGS:
C: Cleared if carry from MSB of result, set otherwise indicating a borrow.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Always reset to one.
H: Cleared if carry from low-order nibble occurred.

EXAMPLE:

Instruction	HEX	Binary
SBC (rr4),(rr8)	73 28 D4	0111 0011 0010 1000 1101 0100

If the carry flag is set, working register pair 4 contains 2800 (decimal), memory location 2800 contains 46 (decimal), working register pair 8 contains 4200 (deciamal) and memory location 4200 contains 45 (decimal), after this instruction memory location 2800 will contain 0.

SBC SBC

Subtract with carry (byte) All, Immediate

SBC dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst] [src]	3	6	25	-	R	#N
	3	6	25	-	r	#N
[OPC] [XTN |dst,0] [src]	3	10	F3	2	(rr)	#N
[OPC] [XTN] [src] [dst h] [dst l]	5	16	2F	21	NN	#N

OPERATION: dst ⇐ dst - src - C

The source byte, along with the carry, is subtracted from the destination byte and the result is stored in the destination byte. The source byte is the immediate value in the operand, the destination byte can be in memory or in the register file.

FLAGS:
C: Cleared if carry from MSB of result, set otherwise indicating a borrow.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Always reset to one.
H: Cleared if carry from low-order nibble occurred.

EXAMPLE:

Instruction	HEX	Binary
SBC (rr8),#32	F3 28 20	1111 0011 0010 1000 0010 0000

If the carry flag is set, working register pair 8 contains 4028 (decimal) and memory location 4028 contains 74 (decimal), after this instruction memory location 4028 will contain 41.

SBCW SBCW

Subtract With Carry (Word) - Register, Register

SBCW dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst,0|src,0]	2	8	2E	-	rr	rr
[OPC] [src,0] [dst,0]	3	8	27	-	RR	RR
	3	8	27	-	rr	RR
	3	8	27	-	RR	rr
[OPC] [src,0] [XTN | dst]	3	10	96	2	(r)	RR
	3	10	96	2	(r)	rr
[OPC] [XTN | src] [dst,0]	3	10	A6	2	RR	(r)
	3	10	A6	2	rr	(r)

OPERATION: dst ⇐ dst - src - C

The source word, along with the carry flag, is subtracted from the destination word and the result is stored in the destination word. The source and destination word can be addressed either directly or indirectly.

FLAGS:
C: Cleared if carry from MSB of result, otherwise set indicating borrow.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Undefined.
H: Undefined.

EXAMPLE:

Instruction	HEX	Binary
SBCW (r8),RR64	96 40 28	1001 0110 0100 0000 0010 1000

If the carry flag is set, register pair 64 contains 1102 (decimal), working register 8 contains 200 (decimal) and register pair 200 contains 2550 (decimal), after this instruction register pair 200 will hold 1447.

SBCW SBCW

Subtract With Carry (Word) - Register, Memory

SBCW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [dst,0|src,1]	2	12	2E	-	rr	(rr)	a
[OPC] [XTN |src,0] [dst,0]	3	12	7E	2	RR	(rr)	a
[OPC] [XTN |src,1] [dst,0]	3	14	D5	2	RR	(rr)+	b
	3	14	D5	2	rr	(rr)+	b
	3	14	C3	2	RR	-(rr)	c
	3	14	C3	2	rr	-(rr)	c
[OPC] [ofs,0|src,0] [XTN |dst,0]	3	14	60	2	rr	rr(rrx)	a
[OPC] [XTN |src,1] [ofs] [dst,0]	4	14	86	2	RR	N(rr)	a
	4	14	86	2	rr	N(rr)	a
[OPC] [XTN |dst,0] [src h] [src l]	4	14	E2	2	rr	NN	a
[OPC] [XTN |src,0] [ofs h] [ofs l] [dst,0]	5	16	86	2	RR	NN(rr)	a
	5	16	86	2	rr	NN(rr)	a

OPERATION a: dst ⇐ dst - src - C

The source word, along with the carry flag, is subtracted from the destination word and the result is stored in the destination word. The destination word is held in the destination register. The source word can be addressed directly, indirectly or by indexing.

OPERATION b: dst ⇐ dst - src - C
rr ⇐ rr + 2

The source word, along with the carry flag, is subtracted from the destination word and the result is stored in the destination word. The source word is in the memory location addressed by the source register pair, the destination word is in the destination register. The contents of the source register pair are incremented after the subtraction has been carried out.

OPERATION c: rr ⇐ rr - 2
dst ⇐ dst - src - C

The source word, along with the carry flag, is subtracted from the destination word and the result is stored in the destination word. The source word is in the memory location addressed by the source register pair, the destination word is in the destination register. The contents of the source register pair are decremented before the subtraction is carried out.

SBCW SBCW

Subtract With Carry (Word) - Register, Memory

SBCW dst,src (Cont'd)

FLAGS:
C: Cleared if carry from MSB of result, otherwise set indicating borrow.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Undefined.
H: Undefined.

EXAMPLE:

Instruction	HEX	Binary
SBCW RR64,-(rr4)	C3 25 40	1100 0011 0010 0101 0100 0000

If the carry flag is set, working register pair 8 contains 1184 (decimal), register pair 64 contains 5000 (decimal) and memory pair 1182 contains 1100 (decimal), after this instruction register pair 64 will contain 3899 and register pair 4 will contain 1182.

SBCW SBCW

Subtract With Carry (Word) - Memory, Register

SBCW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode		Oper
					dst	src	
[OPC] [dst,1|src,0]	2	16	2E	-	(rr)	rr	a
[OPC] [XTN |dst,1] [src,0]	3	18	BE	2	(rr)	RR	a
[OPC] [XTN |dst,0] [src,0]	3	18	D5	2	(rr)+	RR	b
	3	18	D5	2	(rr)+	rr	b
	3	18	C3	2	-(rr)	RR	c
	3	18	C3	2	-(rr)	rr	c
[OPC] [ofd,0|dst,1] [XTN |src,0]	3	18	60	2	rr(rrx)	rr	a
[OPC] [XTN |dst,1] [ofd] [src,1]	4	18	86	2	N(rr)	RR	a
	4	18	86	2	N(rr)	rr	a
[OPC] [XTN |src,1] [dst h] [dst l]	4	18	E2	2	NN	rr	a
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src,1]	5	20	86	2	NN(rr)	RR	a
	5	20	86	2	NN(rr)	rr	a

OPERATION a: dst ⇐ dst - src - C

The source word, along with the carry flag, is subtracted from the destination word and the result is stored in the destination word. The source word is held in the source register. The destination word can be addressed directly, indirectly or by indexing.

OPERATION b: dst ⇐ dst - src - C
rr ⇐ rr + 2

The source word, along with the carry flag, is subtracted from the destination word and the result is stored in the destination word. The source word is in the source register, the destination word is in the memory location addressed by the destination register pair. The contents of the destination register pair are incremented after the subtraction has been carried out.

OPERATION c: rr ⇐ rr - 2
dst ⇐ dst - src - C

The source word, along with the carry flag, is subtracted from the destination word and the result is stored in the destination word. The source word is in the source register , the destination word is in the memory location addressed by the destination register pair. The contents of the destination register pair are decremented before the subtraction is carried out.

SBCW SBCW

Subtract With Carry (Word) - Memory, Register

SBCW dst,src (Cont'd)

FLAGS:

C: Cleared if carry from MSB of result, otherwise set indicating borrow.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Undefined.
H: Undefined.

EXAMPLE:

Instruction	HEX	Binary
SBCW (rr4)+,RR64	D5 24 40	1101 0101 0010 0100 0100 0000

If the carry flag is set, register pair 64 contains 1250 (decimal), working register pair 4 contains 1064 (decimal) and memory pair 1064 contains 1750, after this instruction has been carried out memory pair 1064 will contain 499 and working register pair 4 will contain 1066.

SBCW SBCW

Subtract With Carry (Word) - Memory, Memory

SBCW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [dst,1|src,1]	2	20	2E	-	(rr)	(rr)

OPERATION: dst ⇐ dst - src - C

The source word, along with the carry flag, is subtracted from the destination word and the result is stored in the destination word. The source word is in the memory location addressed by the source register pair, the destination word is in the memory location addressed by the destination register pair.

FLAGS:
C: Cleared if carry from MSB of result, otherwise set indicating borrow.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Undefined.
H: Undefined.

EXAMPLE:

Instruction	HEX	Binary
SBCW (rr4),(rr6)	2E 57	0010 1110 0101 0111

If the carry flag is zero, working register pair 6 contains 1002 (decimal), memory pair 1002 contains 2300 (decimal), working register pair 4 contains 1060 (decimal) and memory pair 1060 contains 2700 (decimal), after this instruction memory pair 1060 will contain 400.

SBCW SBCW

Subtract With Carry (Word) - All, Immediate

SBCW dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst,1] [src h] [src l]	4	10	27	-	RR	#NN
	4	10	27	-	rr	#NN
[OPC] [XTN |dst,0] [src h] [src l]	4	18	BE	2	(rr)	#NN
[OPC] [XTN |dst,1] [ofd] [src h] [src l]	5	20	06	2	N(rr)	#NN
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src h] [src l]	6	22	06	2	NN(rr)	#NN
[OPC] [XTN] [src h] [src l] [dst h] [dst l]	6	22	36	21	NN	#NN

OPERATION: dst ⇐ dst - src - C

The source word, along with the carry flag, is subtracted from the destination word and the result is stored in the destination word. The source word is the immediate value in the operand, the destination word can be in memory or in the register file.

FLAGS:
C: Cleared if carry from MSB of result, otherwise set indicating borrow.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Undefined.
H: Undefined.

EXAMPLE:

Instruction	HEX	Binary
SBCW RR64,#4268	27 41 10 AC	0010 0111 0100 0001 0001 0000 1010 1100

If the carry flag is zero, register pair 64 contains 5000 (decimal), after this instruction has been carried out register pair 64 will contain the decimal value 732.

SCF SCF

Set Carry Flag

SCF

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC]	1	4	01	-	-	-

OPERATION: C ⇐ 1

The carry flag is set to 1.

FLAGS: C: Set to one.
No other flags affected.

EXAMPLE:

Instruction	HEX	Binary
SCF	01	0000 0001

Regardless of its prior condition, after this instruction the carry flag will be set to one.

SDM SDM

Set Data Memory

SDM

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC]	1	4	FE	-	-	-

OPERATION: Set Data Memory.

After executing this instruction, accesses to operands are performed as if these operands were located in data memory. This means that:

– wait states defined for data memory are used;

– When using extended addressing mechanism through MMU, DPRx registers are used to extend addresses contained in the instruction or registers defined in the instruction.

See MMU usage for further details.

This instruction sets to one bit 0 of the flag register R231.

FLAGS: No flags affected.

SLA SLA

Shift Left Arithmetic (Byte)

SLA dst

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst | dst]	2	4	42	-	r	-
[OPC] [dst] [dst]	3	6	44	-	R	-
[OPC] [XTN |dst,0] [dst,0]	3	14	73	4	(rr)	-

OPERATION: dst C ⇐ dst(7)
dst(0) ⇐ 0
dst(n+1) ⇐ dst(n) where n=0-6

The content of the destination is shifted one place to the left with the most significant bit shifted into the carry flag and a zero shifted into bit 0. The destination register can be a register or a memory indirectly addressed.

FLAGS:
C: Set if MSB set, otherwise cleared.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Always reset to zero.
H: Set if carry from low-order nibble occurred.

EXAMPLE:

Instruction	HEX	Binary
SLA r6	44 66	0100 0100 0110 0110

If working register 6 contains A4 hex , after this instruction the carry bit will be set and working register 8 will contain 48 hex.

NOTE: This instruction is logically and functionally equivalent to the ADD dst, dst operation and is recognized and translated into the corresponding ADD instruction by the ST9 assembler.

SLAW SLAW

Shift Left Arithmetic Word

SLAW dst

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst | dst]	2	8	4E	-	rr	-
[OPC] [dst,0] [dst,0]	3	8	47		RR	-
[OPC] [dst,1] [dst,1]	2	20	4E	-	(rr)	-

OPERATION: dst C ⇐ dst(15)
dst(0) ⇐ 0
dst(n+1) ⇐ dst(n) where n=0-14

The content of the destination is shifted one place to the left with the most significant bit shifted into the carry flag and a zero shifted into bit 0. The destination register can be a register or a memory indirectly addressed.

FLAGS:
C: Set if MSB set, otherwise cleared.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Undefined
H: Undefined

EXAMPLE:

Instruction	HEX	Binary
SLAW RR4	47 04 04	0100 0111 0000 0100 0000 0100

If working register pair 4 contains A438 hex , after this instruction the carry bit will be set and working register pair 4 will contain 4870 hex.

NOTE: This intruction is logically and functionally equivalent to the ADD dst, dst operation and is recognized and translated into the corresponding ADDW instruction by the ST9 assembler.

SPM SPM

Set Program Memory

SPM

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC]	1	4	EE	-	-	-

OPERATION: Set Program Memory.

After executing this instruction, accesses to operands are performed as if these operands were located in program memory. This means that:

– wait states defined for program memory are used;

– When using extended addressing mechanism through MMU, CSR register is used to extend addresses contained in the instruction or registers defined in the instruction.

The only exceptions are instructions which use explicitly an operand in the stack (PUSH, PUSHW, POP, POPW, PUSHU, PUSHW, POPU, POPUW, PEA, PEAU, CALL, CALLS, RET, RETS and Interrupts).

See MMU usage for further details.

This instruction resets to zero bit 0 of the flag register R231.

FLAGS: No flags affected.

SPP SPP

Set Page Pointer

SPP src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [src ,1,0]	2	4	C7	-	-	#N

OPERATION: Set to N the Page Pointer Register (R234), where $0 \leq N \leq 63$

This instruction selects one of the 64 pages available to be used for the storage of control information relevant to particular peripherals. Each page is composed of 16 registers based on the top group (F) of the register file. After selecting a page any address on the top group (R240-R255) will be referred to the selected page.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
SPP #5	C7 16	1100 0111 0001 0110

This instruction will select page 5 of paged registers. Then operations addressing group F of the register file are related to page 5.

The page pointer register (R234) contains 0x14.

SRA SRA

Shift Right Arithmetic Byte

SRA dst

INSTRUCTION FORMAT:

[OPC] [dst]	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
	2	4	E0	-	R	-
	2	4	E0	-	r	-
	2	6	E1	-	(R)	-
	2	6	E1	-	(r)	-

OPERATION:

dst(7) ⇐ dst(7)
C ⇐ dst(0)
dst(n) ⇐ dst(n+1) Where n=0-6

The contents of the destination register are shifted one place to the right with the bit 0 shifted into the carry flag. Bit 7 (the sign bit) is unchanged but its value is also carried into bit position 6. The destination register can be directly or indirectly addressed.

FLAGS:

C: Set if carry from LSB (bit 0 was 1).
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
SRA (r2)	E1 D2	1110 0001 1101 0010

If the carry flag is one, working register 2 contains 137 (decimal) and register 137 contains 11001100, after this instruction register 137 will contain 11100110 and the carry flag will be zero.

SRAW SRAW

Shift Right Arithmetic Word

SRAW dst

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [dst,0]	2	8	2F	-	RR	-
	2	8	2F	-	rr	-

OPERATION:

dst(15) ⇐ dst(15)
C ⇐ dst(0)
dst(n) ⇐ dst(n+1) where n=0-14

The contents of the destination register pair are shifted one place to the right with bit 0 shifted into the carry flag. Bit 15 (the sign bit) is unchanged but its value is also carried into bit position 14.

FLAGS:

C: Set if carry from LSB (bit 0 was 1).
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is negative, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
SRAW rr2	2F D2	0010 1111 1101 0010

If the carry flag is one, working register pair 2 contains 11001100/11001100B, after this instruction working register pair 2 will contain 11100110/01100110B and the carry flag will be zero.

SRP SRP

Set Register Pointer

SRP src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [src,0,0,0]	2	4	C7	-	-	#N

OPERATION: Set Register Pointer

This instruction selects one pair of the thirty-two groups of 8 registers available in the register file. The pair will always start from the lowest even number equal or lower to the number given in the instruction.

When this instruction is followed by a SRP1 instruction, that is when the mode is changed to the twin working register groups, an 8 register group is selected, equivalent to the SRP0 instruction.

After having selected the window pair every absolutely addressed register that refers to group D (R208-R223) will be referenced to the working window pair.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
SRP #3	C7 18	1100 0111 0001 1000

```
SRP #3
LD   r3, #10
LD   R21, #20
```

The first instruction will select the second pair of register (R16-R31) as working register window. The second instruction therefore will load the value 10 (decimal) in working register 3 which is R19. The register R21 in the third instruction is equivalent to r5. After this instruction register R21 will contain 20 (decimal).

SRP0 SRP0

Set Register Pointer 0

SRP0 src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [src 1,0,0]	2	4	C7	-	-	#N

OPERATION: Set Register Pointer 0

This instruction activates the twin register mode and therefore register pointer 0 will refer to one of the thirty-two available groups in the register file.

In particular, after having selected the appropriate window every register between R208 and R215 will be equivalent to working registers r0-r7 and therefore will refer to the window pointed to by RP0.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
SRP0 #3	C7 1C	1100 0111 0001 1100

```
SRP0#3
LD   r3, #10
LD   r5, #20
```

This instruction will select the window R24-R31. The second instruction will therefore load in the third register of the selected window the immediate data, that is register R27 will contain 10 (decimal). The third instruction will load in the sixth working register, that is R29, the value 20 (decimal).

SRP1 SRP1

Set Register Pointer 1

SRP1 src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [src 1,0,1]	2	4	C7	-	-	#N

OPERATION: Set Register Pointer 1

This instruction activates the twin register mode and therefore register pointer 1 will refer to one of the thirty-two available groups of 8 registers in the register file.

In particular after having selected the appropiate window every register between R216 and R223 will be equivalent to working registers r8-r15 and therefore will refer to the window pointed to by RP1.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
SRP1 #2	C715	1100 0111 0001 0101

```
SRP#3
SRP1#2
LD   r3, #10
LD   r10, #20
```

The first instruction will select the window pair R16-R31. With the second instruction the mode will be changed to the twin register groups and register RP0 will point to R24-R31 while register RP1 will point to R16-R23. The first load instruction will therefore refer to register pointer zero since the value of the short register is between 0-7 and will place the value 10 (decimal) into R19. The second load refers to register pointer one since the value of the short register is between 8-15 and will place 20 (decimal) into R26.

SUB SUB

Subtract (byte) Register, Register

SUB dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [dst | src]	2	4	52	-	r	r
	2	6	53	-	r	(r)
[OPC] [src] [dst]	3	6	54	-	R	R
	3	6	54	-	r	R
	3	6	54	-	R	r
[OPC] [src] [XTN | dst]	3	6	E6	5	(r)	R
	3	6	E6	5	(r)	r
[OPC] [XTN | src] [dst]	3	6	E7	5	R	(r)

OPERATION: dst ⇐ dst - src

The source byte is subtracted from the destination byte and the result is stored in the destination byte. The source and destination byte can be addressed either directly or indirectly.

FLAGS:
C: Cleared if carry from MSB of result, set otherwise indicating a borrow.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Always reset to one.
H: Cleared if carry from low-order nibble occurred.

EXAMPLE:

Instruction	HEX	Binary
SUB (r8),R255	E6 FF 58	1110 0110 1111 1111 0101 1000

If working register 8 contains 28 (decimal), register 28 contains 43 (decimal) and register 255 contains 21 (decimal), after this instruction register 28 will contain 22.

SUB SUB

Subtract (byte) Register, Memory

SUB dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode		Oper
					dst	src	
[OPC] [XTN |src,1] [dst]	3	8	72	5	R	(rr)	a
	3	8	72	5	r	(rr)	a
	3	12	B4	5	R	(rr)+	b
	3	12	B4	5	r	(rr)+	b
	3	12	C2	5	R	-(rr)	c
	3	12	C2	5	r	-(rr)	c
[OPC] [ofs,1|src,0] [XTN | dst]	3	14	60	5	r	rr(rrx)	a
[OPC] [XTN |src,1] [ofs] [dst]	4	14	7F	5	R	N(rr)	a
	4	14	7F	5	r	N(rr)	a
[OPC] [XTN | dst] [src h] [src l]	4	14	C4	5	r	NN	a
[OPC] [XTN |src,0] [ofs h] [ofs l] [dst]	5	16	7F	5	R	NN(rr)	a
	5	16	7F	5	r	NN(rr)	a

OPERATION a: dst ⇐ dst - src

The source byte is subtracted from the destination byte and the result is stored in the destination byte. The source byte can be addressed directly, indirectly or by indexing.

OPERATION b: dst ⇐ dst - src
rr ⇐ rr + 1

The source byte is subtracted from the destination byte and the result is stored in the destination byte. The source byte is in the memory location addressed by the source register pair, the destination byte is in the destination register. The contents of the source register pair are incremented after the SUB has been carried out.

OPERATION c: rr ⇐ rr - 1
dst ⇐ dst - src

The source byte is subtracted from the destination byte and the result is stored in the destination byte. The source byte is in the memory location addressed by the source register pair, the destination byte is in the destination register. The contents of the source register pair are decremented before the SUB is carried out.

SUB SUB

Subtract (byte) Register, Memory

SUB dst,src (Cont'd)

FLAGS:
C: Cleared if carry from MSB of result, set otherwise indicating a borrow.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Always reset to one.
H: Cleared if carry from low-order nibble occurred.

EXAMPLE:

Instruction	HEX	Binary
SUB r8,(rr4)	72 55 D8	0111 0010 0101 0101 1101 1000

If working register 8 contains 213 (decimal), working register pair 4 contain 4200 (decimal) and memory location 4200 contains 25 (decimal), after this instruction register 8 will contain 188.

SUB SUB

Subtract (byte) Memory, Register

SUB dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [XTN |dst,0] [src]	3	12	72	5	(rr)	R	a
	3	12	72	5	(rr)	r	a
	3	14	B4	5	(rr)+	R	b
	3	14	B4	5	(rr)+	r	b
	3	14	C2	5	-(rr)	R	c
	3	14	C2	5	-(rr)	r	c
[OPC] [ofd,1|dst,1] [XTN | src]	3	14	60	5	rr(rrx)	r	a
[OPC] [XTN |dst,1] [ofd] [src]	4	14	26	5	N(rr)	R	a
	4	14	26	5	N(rr)	r	a
[OPC] [XTN | src] [dst h] [dst l]	4	12	C5	5	NN	r	a
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src]	5	16	26	5	NN(rr)	R	a
	5	16	26	5	NN(rr)	r	a

OPERATION a: dst ⇐ dst - src

The source byte is subtracted from destination byte and the result is stored in the destination byte. The source byte is held in the source register. The destination byte can be addressed directly, indirectly or by indexing.

OPERATION b: dst ⇐ dst - src
rr ⇐ rr + 1

The source byte is subtracted from the destination byte and the result is stored in the destination byte. The source byte is in the source register, the destination byte is in the memory location addressed by the destination register pair. The contents of the destination register pair are incremented after the SUB has been carried out.

OPERATION c: rr ⇐ rr - 1
dst ⇐ dst - src

The source byte is subtracted from the destination byte and the result is stored in the destination byte. The source byte is in the source register , the destination byte is in the memory location addressed by the destination register pair. The contents of the destination register pair are decremented before the SUB is carried out.

SUB SUB

Subtract (byte) Memory, Register

SUB dst,src (Cont'd)

FLAGS:
C: Cleared if carry from MSB of result, set otherwise indicating a borrow.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Always reset to one.
H: Cleared if carry from low-order nibble occurred.

EXAMPLE:

Instruction	HEX	Binary
SUB (rr8),R255	72 58 FF	0111 0010 0101 1000 1111 1111

If working register pair 8 contains 4028 (decimal) memory location 4028 contains 144 (decimal) and register 255 contains 22 (decimal), after this instruction memory location 4028 will contain 122.

SUB SUB

Subtract (byte) Memory, Memory

SUB dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [XTN |src,0] [dst,0]	3	14	73	5	(RR)	(rr)
	3	14	73	5	(rr)*	(rr)

OPERATION: dst ⇐ dst - src

The source byte is subtracted from the destination byte and the result is stored in the destination byte. The source byte is in the memory location addressed by the source register pair, the destination byte is in the memory location addressed by the destination register pair.

FLAGS:
C: Cleared if carry from MSB of result, set otherwise indicating a borrow.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Always reset to one.
H: Cleared if carry from low-order nibble occurred.

EXAMPLE:

Instruction	HEX	Binary
SUB (rr4),(rr8)	73 58 D4	0111 0011 0101 1000 1101 0100

If working register pair 4 contains 2800 (decimal), memory location 2800 contains 46 (decimal), working register pair 8 contains 4200 (deciamal) and memory location 4200 contains 45 (decimal), after this instruction memory location 2800 will contain 1.

SUB SUB

Subtract (byte) All, Immediate

SUB dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst] [src]	3	6	55	-	R	#N
	3	6	55	-	r	#N
[OPC] [XTN |dst,0] [src]	3	10	F3	5	(rr)	#N
[OPC] [XTN] [src] [dst h] [dst l]	5	16	2F	51	NN	#N

OPERATION: dst ⇐ dst - src

The source byte is subtracted from the destination byte and the result is stored in the destination byte. The source byte is the immediate value in the operand, the destination byte can be in memory or in the register file.

FLAGS:
C: Cleared if carry from MSB of result, set otherwise indicating a borrow.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Always reset to one.
H: Cleared if carry from low-order nibble occurred.

EXAMPLE:

Instruction	HEX	Binary
SUB (rr8),#32	F3 58 20	1111 0011 0101 1000 0010 0000

If working register pair 8 contains 4028 (decimal) and memory location 4028 contains 74 (decimal), after this instruction memory location 4028 will contain 42.

SUBW SUBW

Subtract (Word) - Register, Register

SUBW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [dst,0|src,0]	2	8	5E	-	rr	rr
[OPC] [src,0] [dst,0]	3	8	57	-	RR	RR
	3	8	57	-	rr	RR
	3	8	57	-	RR	rr
[OPC] [src,0] [XTN | dst]	3	10	96	5	(r)	RR
	3	10	96	5	(r)	rr
[OPC] [XTN | src] [dst,0]	3	10	A6	5	RR	(r)
	3	10	A6	5	rr	(r)

OPERATION: dst ⇐ dst - src

The source word, along with the carry flag, is subtracted from the destination word and the result is stored in the destination word. The source and destination words can be addressed either directly or indirectly.

FLAGS:
C: Cleared if carry from MSB of result, otherwise set indicating borrow.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Undefined.
H: Undefined.

EXAMPLE:

Instruction	HEX	Binary
SUBW (r8),RR64	96 40 58	1001 0110 0100 0000 0101 1000

If register pair 64 contains 1102 (decimal), working register 8 contains 200 (decimal) and register pair 200 contains 2550 (decimal), after this instruction register pair 200 will hold 1448.

SUBW SUBW

Subtract (Word) - Register, Memory

SUBW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [dst,0|src,1]	2	12	5E	-	rr	(rr)	a
[OPC] [XTN |src,0] [dst,0]	3	12	7E	5	RR	(rr)	a
[OPC] [XTN |src,1] [dst,0]	3	14	D5	5	RR	(rr)+	b
	3	14	D5	5	rr	(rr)+	b
	3	14	C3	5	RR	-(rr)	c
	3	14	C3	5	rr	-(rr)	c
[OPC] [ofs,0|src,0] [XTN |dst,0]	3	14	60	5	rr	rr(rrx)	a
[OPC] [XTN |src,1] [ofs] [dst,0]	4	14	86	5	RR	N(rr)	a
	4	14	86	5	rr	N(rr)	a
[OPC] [XTN |dst,0] [src h] [src l]	4	14	E2	5	rr	NN	a
[OPC] [XTN |src,0] [ofs h] [ofs l] [dst,0]	5	16	86	5	RR	NN(rr)	a
	5	16	86	5	rr	NN(rr)	a

OPERATION a: dst ⇐ dst - src

The source word, along with the carry flag, is subtracted from the destination word and the result is stored in the destination word. The destination word is held in the destination register. The source word can be addressed directly, indirectly or by indexing.

OPERATION b: dst ⇐ dst - src
rr ⇐ rr + 2

The source word, along with the carry flag, is subtracted from the destination word and the result is stored in the destination word. The source word is in the memory location addressed by the source register pair, the destination word is in the destination register. The contents of the source register pair are incremented after the subtraction has been carried out.

OPERATION c: rr ⇐ rr - 2
dst ⇐ dst - src

The contents of the source register pair are decremented before the subtraction is carried out. The source word, along with the carry flag, is subtracted from the destination word and the result is stored in the destination word. The source word is in the memory location addressed by the source register pair, the destination word is in the destination register.

SUBW SUBW

Subtract (Word) - Register, Memory

SUBW dst,src (Cont'd)

FLAGS:
C: Cleared if carry from MSB of result, otherwise set indicating borrow.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Undefined.
H: Undefined.

EXAMPLE:

Instruction	HEX	Binary
SUBW RR64,-(rr4)	C3 55 40	1100 0011 0101 0101 0100 0000

If working register pair 8 contains 1184 (decimal), register pair 64 contains 5000 (decimal) and memory location 1182 contains 1100 (decimal), after this instruction register pair 64 will contain 3900 and register pair 4 will contain 1182.

SUBW SUBW

Subtract (Word) - Memory, Register

SUBW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [dst,1|src,0]	2	16	5E	-	(rr)	rr	a
[OPC] [XTN |dst,1] [src,0]	3	18	BE	5	(rr)	RR	a
[OPC] [XTN |dst,0] [src,0]	3	18	D5	5	(rr)+	RR	b
	3	18	D5	5	(rr)+	rr	b
	3	18	C3	5	-(rr)	RR	c
	3	18	C3	5	-(rr)	rr	c
[OPC] [ofd,0|dst,1] [XTN |src,0]	3	18	60	5	rr(rrx)	rr	a
[OPC] [XTN |dst,1] [ofd] [src,1]	4	18	86	5	N(rr)	RR	a
	4	18	86	5	N(rr)	rr	a
[OPC] [XTN |src,1] [dst h] [dst l]	4	18	E2	5	NN	rr	a
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src,1]	5	20	86	5	NN(rr)	RR	a
	5	20	86	5	NN(rr)	rr	a

OPERATION a: dst ⇐ dst - src

The source word is subtracted from the destination word and the result is stored in the destination word. The source word is held in the source register. The destination word can be addressed directly, indirectly or by indexing.

OPERATION b: dst ⇐ dst - src
rr ⇐ rr + 2

The source word is subtracted from the destination word and the result is stored in the destination word. The source word is in the source register, the destination word is in the memory location addressed by the destination register pair. The contents of the destination register pair are incremented after the subtraction has been carried out.

OPERATION c: rr ⇐ rr - 2
dst ⇐ dst - src

The contents of the source register pair are decremented before the subtraction is carried out. The source word, along with the carry flag, is subtracted from the destination word and the result is stored in the destination word. The source word is in the source register , the destination word is in the memory location addressed by the destination register pair.

SUBW SUBW

Subtract (Word) - Memory, Register

SUBW dst,src (Cont'd)

FLAGS:
C: Cleared if carry from MSB of result, otherwise set indicating a borrow.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Undefined.
H: Undefined.

EXAMPLE:

Instruction	HEX	Binary
SUBW (rr4)+,RR64	D5 54 40	1101 0101 0101 0100 0100 0000

If register location 64 contains 1250 (decimal), working register pair 4 contains 1064 (decimal) and memory pair 1064 contains 11750, after this instruction has been carried out memory pair 1064 will contain 500 and workig register pair 4 will contain 1066.

SUBW SUBW

Subtract (Word) - Memory, Memory

SUBW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [dst,1|src,1]	2	20	5E	-	(rr)	(rr)

OPERATION: dst ⇐ dst - src

The source word is subtracted from the destination word and the result is stored in the destination word. The source word is in the memory location addressed by the source register pair, the destination word is in the memory location addressed by the destination register pair.

FLAGS:
C: Cleared if carry from MSB of result, otherwise set indicating borrow.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Undefined.
H: Undefined.

EXAMPLE:

Instruction	HEX	Binary
SUBW (rr4),(rr6)	5E 57	0101 1110 0101 0111

If working register pair 6 contains 1002 (decimal), memory pair 1002 contains 2300 (decimal), working register pair 4 contains 1060 (decimal) and memory pair 1060 contains 2700 (decimal), after this instruction memory pair 1060 will contain 400.

SUBW SUBW

Subtract (Word) - All, Immediate

SUBW dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst,1] [src h] [src l]	4	10	57	-	RR	#NN
	4	10	57	-	rr	#NN
[OPC] [XTN |dst,0] [src h] [src l]	4	18	BE	5	(rr)	#NN
[OPC] [XTN |dst,1] [ofd] [src h] [src l]	5	20	06	5	N(rr)	#NN
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src h] [src l]	6	22	06	5	NN(rr)	#NN
[OPC] [XTN] [src h] [src l] [dst h] [dst l]	6	22	36	51	NN	#NN

OPERATION: dst ⇐ dst - src

The source word is subtracted from the destination word and the result is stored in the destination word. The source word is the immediate value in the operand, the destination word can be in memory or in the register file.

FLAGS:
C: Cleared if carry from MSB of result, otherwise set indicating borrow.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result is less than zero, otherwise cleared.
V: Set if arithmetic overflow occurred, cleared otherwise.
D: Undefined.
H: Undefined.

EXAMPLE:

Instruction	HEX	Binary
SUBW RR64,#4268	57 41 10 AC	0101 0111 0100 0001 0001 0000 1010 1100

If register pair 64 contains 5000 (decimal), after this instruction has been carried out register pair 64 will contain the decimal value 732.

SWAP SWAP

Swap Nibbles

SWAP dst

INSTRUCTION FORMAT:

[OPC] [dst]	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
	2	8	F0	-	R	-
	2	8	F0	-	r	-
	2	8	F1	-	(R)	-
	2	8	F1	-	(r)	-

OPERATION: dst(0-3) ⇐- dst(4-7)

The upper and lower nibbles of the destination register are swapped. The destination register can be directly or indirectly addressed.

FLAGS:
C: Undefined.
Z: Set if the result is zero, otherwise cleared.
S: Set if the result bit 7 is set, otherwise cleared.
V: Undefined.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
SWAP R32	F0 20	1111 0000 0010 0000

If register 32 contains 1110 0111B, after this instruction the contents become 01111110B.

TCM TCM

Test complement under mask (byte) Register, Register

TCM dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst | src]	2	4	82	-	r	r
	2	6	83	-	r	(r)
[OPC] [src] [dst]	3	6	84	-	R	R
	3	6	84	-	r	R
	3	6	84	-	R	r
[OPC] [src] [XTN | dst]	3	6	E6	8	(r)	R
	3	6	E6	8	(r)	r
[OPC] [XTN | src] [dst]	3	6	E7	8	R	(r)

OPERATION: NOT dst AND src

Selected bits in the destination byte are tested for a logical one value. The bits to be tested are selected by setting to one the corresponding bits in the source byte (mask). TCM instruction complements the destination byte, which is then ANDed with the source byte. The zero flag can then be checked to determine the result. The destination byte remains unaltered by this instruction. The source byte is held in the source register and the destination byte in the destination register.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 7 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
TCM r8,R64	84 40 D8	1000 0100 0100 0000 1101 1000

If working register 8 contains 11001100 and register 64 contains 1000 0100, after this instruction the zero flag will be reset to zero.

TCM TCM

Test complement under mask (byte) Register, Memory

TCM dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [XTN |src,1] [dst]	3	8	72	8	R	(rr)	a
	3	8	72	8	r	(rr)	a
	3	12	B4	8	R	(rr)+	b
	3	12	B4	8	r	(rr)+	b
	3	12	C2	8	R	-(rr)	c
	3	12	C2	8	r	-(rr)	c
[OPC] [ofs,1|src,0] [XTN | dst]	3	12	60	8	r	rr(rrx)	a
[OPC] [XTN |src,1] [ofs] [dst]	4	12	7F	8	R	N(rr)	a
	4	12	7F	8	r	N(rr)	a
[OPC] [XTN | dst] [src h] [src l]	4	10	C4	8	r	NN	a
[OPC] [XTN |src,0] [ofs h] [ofs l] [dst]	5	14	7F	8	R	NN(rr)	a
	5	14	7F	8	r	NN(rr)	a

OPERATION a: NOT dst AND src

Selected bits in the destination byte are tested for a logical one value. The bits to be tested are selected by setting to one the corresponding bits in the source byte (mask). TCM instruction complements the destination byte, which is then ANDed with the source byte. The zero flag can then be checked to determine the result. The destination byte remains unaltered by this instruction. The source byte is held in the source memory location and the destination byte in the destination register. The destination register is addressed directly, the memory location is addressed either directly, indirectly or by indexing.

OPERATION b: NOT dst AND src
rr ⇐ rr + 1

As operation 'a' (indirect memory addressing only), but the contents of the destination register pair are incremented after the TCM has been carried out.

OPERATION c: rr ⇐ rr - 1 NOT dst AND src

As operation 'a' (indirect memory addressing only), but the contents of the destination register pair are decremented before the TCM is carried out.

TCM TCM

Test complement under mask (byte) Register, Memory

TCM dst,src (Cont'd)

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 7 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
TCM r8,4028	C4 88 0F BC	1100 0100 1000 1000 0000 1111 1011 1100

If working register 8 contains 11001100 and memory location 4028 contains 10000101, after this instruction the zero flag will be reset to zero.

TCM TCM

Test complement under mask (byte) Memory, Register

TCM dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [XTN |dst,0] [src]	3	10	72	8	(rr)	R	a
	3	10	72	8	(rr)	r	a
	3	12	B4	8	(rr)+	R	b
	3	12	B4	8	(rr)+	r	b
	3	12	C2	8	-(rr)	R	c
	3	12	C2	8	-(rr)	r	c
[OPC] [ofd,1|dst,1] [XTN | src]	3	12	60	8	rr(rrx)	r	a
[OPC] [XTN |dst,1] [ofd] [src]	4	12	26	8	N(rr)	R	a
	4	12	26	8	N(rr)	r	a
[OPC] [XTN | src] [dst h] [dst l]	4	10	C5	8	NN	r	a
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src]	5	14	26	8	NN(rr)	R	a
	5	14	26	8	NN(rr)	r	a

OPERATION a: NOT dst AND src

Selected bits in the destination byte are tested for a logical one value. The bits to be tested are selected by setting to one the corresponding bits in the source byte (mask). TCM instruction complements the destination byte, which is then ANDed with the source byte. The zero flag can then be checked to determine the results. The destination byte remains unaltered by this instruction. The source byte is held in the source memory location and the destination byte in the destination register. The source register is addressed directly, the memory location is addressed either directly, indirectly or by indexing.

OPERATION b: NOT dst AND src
rr ⇐ rr + 1

As operation 'a' (indirect memory addressing only), but the contents of the source register pair are incremented after the TCM has been carried out.

OPERATION c: rr ⇐ rr - 1
NOT dst AND src

As operation 'a' (indirect memory addressing only), but the contents of the source register pair are decremented before the TCM is carried out.

TCM TCM

Test complement under mask (byte) Memory, Register

TCM dst,src (Cont'd)

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 7 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
TCM 4028,r8	C5 88 0F BC	1100 0101 1000 1000 0000 1111 1011 1100

If memory location 4028 contains 11001100 and working register 8 contains 10000101, after this instruction the zero flag will be reset to zero.

TCM TCM

Test complement under mask (byte) Memory, Memory

TCM dst,src

INSTRUCTION FORMAT:

[OPC] [XTN |src,0] [dst,0]	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
	3	12	73	8	(RR)	(rr)
	3	12	73	8	(rr)*	(rr)

OPERATION: NOT dst AND src

Selected bits in the destination byte are tested for a logical one value. The bits to be tested are selected by setting to one the corresponding bits in the source byte (mask). TCM instruction complements the destination byte, which is then ANDed with the source byte. The zero flag can then be checked to determine the results. The destination byte remains unaltered by this instruction. The source byte is in the memory location addressed by the source register pair, the destination byte is in the memory location addressed by the destination register pair.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 7 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
TCM (rr4),(rr8)	73 88 D4	0111 0011 1000 1000 1101 0100

If working register pair 4 contains 2800 (decimal), memory location 2800 contains 11001100, working register pair 8 contain 4200 (decimal) and memory location 4200 contains 11001100, after this instruction the zero flag will be set to one.

TCM TCM

Test complement under mask (byte) All, Immediate

TCM dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst] [src]	3	6	85	-	R	#N
	3	6	85	-	r	#N
[OPC] [XTN |dst,0] [src]	3	8	F3	8	(rr)	#N
[OPC] [XTN] [src] [dst h] [dst l]	5	14	2F	81	NN	#N

OPERATION: NOT dst AND src

Selected bits in the destination byte are tested for a logical one value. The bits to be tested are selected by setting to one the corresponding bits in the source byte (mask). TCM instruction complements the destination byte, which is then ANDed with the source byte. The zero flag can then be checked to determine the results. The destination byte remains unaltered by this instruction. The source byte is the immediate value in the operand, the destination byte can be in memory or in the register file.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 7 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
TCM (rr8),#32	F3 88 20	1111 0011 1000 1000 0010 0000

If working register pair 8 contains 4028 (decimal) and memory location 4028 contains 11101100, after this instruction the zero flag will be set to one.

TCMW TCMW

Test Complement Under Mask (Word) - Register, Register

TCMW dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst,0|src,0]	2	8	8E	-	rr	rr
[OPC] [src,0] [dst,0]	3	8	87	-	RR	RR
	3	8	87	-	rr	RR
	3	8	87	-	RR	rr
[OPC] [src,0] [XTN | dst]	3	10	96	8	(r)	RR
	3	10	96	8	(r)	rr
[OPC] [XTN | src] [dst,0]	3	10	A6	8	RR	(r)
	3	10	A6	8	rr	(r)

OPERATION: NOT dst AND src

Selected bits in the destination word are tested for a logical one value. The bits to be tested are selected by setting to one the corresponding bit in the source word (mask). The TCMW instruction complements the destination word, which is then ANDed with source word. The zero flag can then be checked to determine the result. The destination word remains unaltered by this instruction. The source and the destination word can be addressed either directly or indirectly.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 15 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
TCMW (r8) RR64	96 40 88	1001 0110 0100 0000 1000 1000

If register pair 64 contains 11001100/11001100B, working register 8 contains 200 (decimal) and register pair 200 contains 01001000/01001000B, after this instruction the zero flag will be reset to zero.

TCMW TCMW

Test Complement Under Mask (Word) - Register, Memory

TCMW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [dst,0|src,1]	2	12	8E	-	rr	(rr)	a
[OPC] [XTN |src,0] [dst,0]	3	14	7E	8	RR	(rr)	a
[OPC] [XTN |src,1] [dst,0]	3	14	D5	8	RR	(rr)+	b
	3	14	D5	8	rr	(rr)+	b
	3	14	C3	8	RR	-(rr)	c
	3	14	C3	8	rr	-(rr)	c
[OPC] [ofs,0|src,0] [XTN |dst,0]	3	14	60	8	rr	rr(rrx)	a
[OPC] [XTN |src,1] [ofs] [dst,0]	4	14	86	8	RR	N(rr)	a
	4	14	86	8	rr	N(rr)	a
[OPC] [XTN |dst,0] [src h] [src l]	4	14	E2	8	rr	NN	a
[OPC] [XTN |src,0] [ofs h] [ofs l] [dst,0]	5	16	86	8	RR	NN(rr)	a
	5	16	86	8	rr	NN(rr)	a

OPERATION a: NOT dst AND src

Selected bits in the destination word are tested for a logical one value. The bits to be tested are selected by setting to one the corresponding bit in the source word (mask). The TCMW instruction complements the destination word, which is then ANDed with source word. The zero flag can then be checked to determine the result. The destination word remains unaltered by this instruction.

The source word is held in the source memory location and the destination word in the destination register pair. The destination register pair is addressed directly, the memory location is addressed either directly, indirectly or by indexing.

OPERATION b: NOT dst AND src
rr ⇐ rr + 2

As operation 'a' (indirect memory addressing only), but the contents of the destination register pair are incremented after the TCMW has been carried out.

OPERATION c: rr ⇐ rr - 2
NOT dst AND src

As operation 'a' (indirect memory addressing only), but the contents of the destination register pair are decremented before the TCMW is carried out.

TCMW TCMW

Test Complement Under Mask (Word) - Register, Memory

TCMW dst,src (Cont'd)

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 15 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
TCMW RR64,-(rr4)	C3 85 40	1100 0011 0011 0101 0100 0000

If working register pair 4 contains 1184 (decimal), register pair 64 contains 11001100/11001100B and memory location 1182 contains 11001100/11001100B, after this instruction the zero flag will be set and register pair 4 will contain 1182.

TCMW TCMW

Test Complement Under Mask (Word) - Memory, Register

TCMW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [dst,1|src,0]	2	14	8E	-	(rr)	rr	a
[OPC] [XTN |dst,1] [src,0]	3	14	BE	8	(rr)	RR	a
[OPC] [XTN |dst,0] [src,0]	3	14	D5	8	(rr)+	RR	b
	3	14	D5	8	(rr)+	rr	b
	3	14	C3	8	-(rr)	RR	c
	3	14	C3	8	-(rr)	rr	c
[OPC] [ofd,0|dst,1] [XTN |src,0]	3	14	60	8	rr(rrx)	rr	a
[OPC] [XTN |dst,1] [ofd] [src,1]	4	14	86	8	N(rr)	RR	a
	4	14	86	8	N(rr)	rr	a
[OPC] [XTN |src,1] [dst h] [dst l]	4	14	E2	8	NN	rr	a
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src,1]	5	16	86	8	NN(rr)	RR	a
	5	16	86	8	NN(rr)	rr	a

OPERATION a: NOT dst AND src

Selected bits in the destination word are tested for a logical one value. The bits to be tested are selected by setting to one the corresponding bit in the source word (mask). The TCMW instruction complements the destination word, which is then ANDed with the source word. The zero flag can then be checked to determine the result. The destination word remains unaltered by this instruction.

The source word is held in the source register pair and the destination word in the destination memory location. The source register pair is addressed directly, the memory location is addressed either directly, indirectly or by indexing.

OPERATION b: NOT dst AND src
rr ⇐ rr + 2

As operation 'a' (indirect memory addressing only), but the contents of the source register pair are incremented after the TCMW has been carried out.

OPERATION c: rr ⇐ rr - 2
NOT dst AND src

As operation 'a' (indirect memory addressing only), but the contents of the source register pair are decremented before the TCMW is carried out.

TCMW TCMW

Test Complement Under Mask (Word) - Memory, Register

TCMW dst,src (Cont'd)

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 15 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
TCMW (rr8),RR64	BE 89 40	1011 1110 1000 1001 0100 0000

If register pair 64 contains 11001100/11001100B, working register pair 8 contains 2000h and memory location 2000 contains 11001100/11001100B, after this instruction the zero flag will be set.

TCMW TCMW

Test Complement Under Mask (Word) - Memory, Memory

TCMW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [dst,1|src,1]	2	16	8E	-	(rr)	(rr)

OPERATION: NOT dst AND src

Selected bits in the destination word are tested for a logical one value. The bits to be tested are selected by setting to one the corresponding bit in the source word (mask). The TCMW instruction complements the destination word, which is then ANDed with source word. The zero flag can then be checked to determine the result. The destination word remains unaltered by this instruction.

The source word is held in the source memory location and the destination word in the destination register pair. The source word is in the memory location addressed by the contents of the source register pair. the destination word is in the memory location addressed by the contents of the destination register pair.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 15 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
TCMW (rr4),(rr6)	8E 57	1000 1110 0101 0111

If working register pair 6 contains 1002 (decimal), memory location pair 1002 contains 11001100/11001100B, working register pair 4 contains 1060 (decimal) and memory location 1060 contains 11001100/11001100B, after this instruction the zero flag wil be set.

TCMW TCMW

Test Complement Under Mask (Word) - All, Immediate

TCMW dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst,1] [src h] [src l]	4	10	87	-	RR	#NN
	4	10	87	-	rr	#NN
[OPC] [XTN |dst,0] [src h] [src l]	4	14	BE	8	(rr)	#NN
[OPC] [XTN |dst,1] [ofd] [src h] [src l]	5	16	06	8	N(rr)	#NN
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src h] [src l]	6	18	06	8	NN(rr)	#NN
[OPC] [XTN] [src h] [src l] [dst h] [dst l]	6	20	36	81	NN	#NN

OPERATION: NOT dst AND src

Selected bits in the destination word are tested for a logical one value. The bits to be tested are selected by setting to one the corresponding bit in the source word (mask). The TCMW instruction complements the destination word, which is then ANDed with source word. The zero flag can then be checked to determine the result. The destination word remains unaltered by this instruction.

The source word is the immediate value held in the operand. The destination word can be in memory or register file.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 15 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
TCMW RR64, #CCCCh	87 41 CC CC	0011 0111 0100 0001 1100 1100 1100 1100

If register pair 64 contains 01001000/01001000B, after this instruction has been carried out the zero flag will be reset.

TM TM

Test under mask (byte) Register, Register

TM dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst | src]	2	4	A2	-	r	r
	2	6	A3	-	r	(r)
[OPC] [src] [dst]	3	6	A4	-	R	R
	3	6	A4	-	r	R
	3	6	A4	-	R	r
[OPC] [src] [XTN | dst]	3	6	E6	A	(r)	R
	3	6	E6	A	(r)	r
[OPC] [XTN | src] [dst]	3	6	E7	A	R	(r)

OPERATION: dst AND src

Selected bits in the destination byte are tested for a logical zero value. The bits to be tested are selected by setting to one the corresponding bits in the source byte (mask) which is then ANDed with the destination byte. The zero flag can then be checked to determine the results. The destination byte remains unaltered by this instruction. The source byte is held in the source register and the destination byte in the destination register.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 7 is set, otherwise cleared.
V: Always reset
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
TM r8,R64	A4 40 D8	1010 0100 0100 0000 1101 1000

If working register 8 contains 01001100 and register 64 contains 00110011, after this instruction the zero flag will be set to one.

TM TM

Test under mask (byte) Register, Memory

TM dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [XTN |src,1] [dst]	3	8	72	A	R	(rr)	a
	3	8	72	A	r	(rr)	a
	3	12	B4	A	R	(rr)+	b
	3	12	B4	A	r	(rr)+	b
	3	12	C2	A	R	-(rr)	c
	3	12	C2	A	r	-(rr)	c
[OPC] [ofs,1|src,0] [XTN | dst]	3	12	60	A	r	rr(rrx)	a
[OPC] [XTN |src,1] [ofs] [dst]	4	12	7F	A	R	N(rr)	a
	4	12	7F	A	r	N(rr)	a
[OPC] [XTN | dst] [src h] [src l]	4	10	C4	A	r	NN	a
[OPC] [XTN |src,0] [ofs h] [ofs l] [dst]	5	14	7F	A	R	NN(rr)	a
	5	14	7F	A	r	NN(rr)	a

OPERATION a: dst AND src

Selected bits in the destination byte are tested for a logical zero value. The bits to be tested are selected by setting to one the corresponding bits in the source byte (mask) which is then ANDed with the destination byte. The zero flag can then be checked to determine the results. The destination byte remains unaltered by this instruction. The source byte is held in the source memory location and the destination byte in the destination register. The destination register is addressed directly, the memory location is addressed either directly, indirectly or by indexing.

OPERATION b: dst AND src
rr ⇐ rr + 1

As operation 'a' (indirect memory addressing only), but the contents of the destination register pair are incremented after the TM has been carried out.

OPERATION c: rr ⇐ rr - 1
dst AND src

As operation 'a' (indirect memory addressing only), but before the TM is carried out the contents of the destination register pair are decremented.

TM TM

Test under mask (byte) Register, Memory

TM dst,src (Cont'd)

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 7 is set, otherwise cleared.
V: Always reset
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
TM r8, 0FBCh	C4 A8 0F BC	1100 0100 1010 1000 0000 1111 1011 1100

If working register 8 contains 11001100 and memory location 4028 contains 10000101, after this instruction the zero flag will be set.

TM TM

Test under mask (byte) Memory, Register

TM dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [XTN |dst,0] [src]	3	10	72	A	(rr)	R	a
	3	10	72	A	(rr)	r	a
	3	12	B4	A	(rr)+	R	b
	3	12	B4	A	(rr)+	r	b
	3	12	C2	A	-(rr)	R	c
	3	12	C2	A	-(rr)	r	c
[OPC] [ofd,1|dst,1] [XTN | src]	3	12	60	A	rr(rrx)	r	a
[OPC] [XTN |dst,1] [ofd] [src]	4	12	26	A	N(rr)	R	a
	4	12	26	A	N(rr)	r	a
[OPC] [XTN | src] [dst h] [dst l]	4	10	C5	A	NN	r	a
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src]	5	14	26	A	NN(rr)	R	a
	5	14	26	A	NN(rr)	r	a

OPERATION a: dst AND src

Selected bits in the destination byte are tested for a logical zero value. The bits to be tested are selected by setting to one the corresponding bits in the source byte (mask) which is then ANDed with the destination byte. The zero flag can then be checked to determine the results. The destination byte remains unaltered by this instruction. The source byte is held in the source register and the destination byte in the destination memory location. The source register is addressed directly, the memory location is addressed either directly, indirectly or by indexing.

OPERATION b: dst AND src
rr ⇐ rr + 1

As operation 'a' (indirect memory addressing only), but the contents of the source register pair are incremented after the TM has been carried out.

OPERATION c: rr ⇐ rr - 1
dst AND src

As operation 'a' (indirect memory addressing only), but before the TM is carried out the contents of the source register pair are decremented.

TM TM

Test under mask (byte) Memory, Register

TM dst,src (Cont'd)

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 7 is set, otherwise cleared.
V: Always reset
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
TM 0FBCh	C5 A8 0F BC	1100 0101 1010 1000 0000 1111 1011 1100

If working register 8 contains 11001100 and memory location 4028 contains 10000101, after this instruction the zero flag will be reset to zero.

TM TM

Test under mask (byte) Memory, Memory

TM dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [XTN |src,0] [dst,0]	3	12	73	A	(RR)	(rr)
	3	12	73	A	(rr)*	(rr)

OPERATION: dst AND src

Selected bits in the destination byte are tested for a logical zero value. The bits to be tested are selected by setting to one the corresponding bits in the source byte (mask) which is then ANDed with the destination byte. The zero flag can then be checked to determine the results. The destination byte remains unaltered by this instruction. The source byte is in the memory location addressed by the source register pair, the destination byte is in the memory location addressed by the destination register pair.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 7 is set, otherwise cleared.
V: Always reset
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
TM (rr4),(rr8)	73 A8 D4	0111 0011 1010 1000 1101 0100

If working register pair 4 contains 2800 (decimal), memory location 2800 contains 11001100, working register pair 8 contains 4200 (decimal) and memory location 4200contains 00110011, after this instruction the zero flag will be set to one.

TM TM

Test under mask (byte) All, Immediate

TM dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [dst] [src]	3	6	A5	-	R	#N
	3	6	A5	-	r	#N
[OPC] [XTN |dst,0] [src]	3	8	F3	A	(rr)	#N
[OPC] [XTN] [src] [dst h] [dst l]	5	14	2F	A1	NN	#N

OPERATION: dst AND src

Selected bits in the destination byte are tested for a logical zero value. The bits to be tested are selected by setting to one the corresponding bits in the source byte (mask) which is then ANDed with the destination byte. The zero flag can then be checked to determine the results. The destination byte remains unaltered by this instruction. The source byte is the immediate value in the operand, the destination byte can be in memory or in the register file.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 7 is set, otherwise cleared.
V: Always reset
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
TM (rr8),#20h	F3 A8 20	1111 0011 1010 1000 0010 0000

If working register pair 8 contains 4028 (decimal) and memory location 4028 contains 11101100, after this instruction the zero flag will be reset to zero.

TMW TMW

Test Under Mask (Word) - Register, Register

TMW dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst,0|src,0]	2	8	AE	-	rr	rr
[OPC] [src,0] [dst,0]	3	8	A7	-	RR	RR
	3	8	A7	-	rr	RR
	3	8	A7	-	RR	rr
[OPC] [src,0] [XTN | dst]	3	10	96	A	(r)	RR
	3	10	96	A	(r)	rr
[OPC] [XTN | src] [dst,0]	3	10	A6	A	RR	(r)
	3	10	A6	A	rr	(r)

OPERATION: dst AND src

Selected bits in the destination word are tested for a logical zero value. The bits to be tested are selected by setting to one the corresponding bits in the source word (mask) which is then ANDed with the destination word. The zero flag can then be checked to determine the result. The destination word remains unaltered by this instruction. The source and the destination word can be addressed either directly or indirectly.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 15 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
TMW (r8),RR64	96 40 A8	1001 0110 0100 0000 1010 1000

If register pair 64 contains 11001100/11001100B, working register 8 contains 200 (decimal) and register pair 200 contains 00110011/00110011B, after this instruction the zero flag will be reset to zero.

TMW TMW

Test Under Mask (Word) - Register, Memory

TMW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [dst,0|src,1]	2	12	AE	-	rr	(rr)	a
[OPC] [XTN |src,0] [dst,0]	3	14	7E	A	RR	(rr)	a
[OPC] [XTN |src,1] [dst,0]	3	14	D5	A	RR	(rr)+	b
	3	14	D5	A	rr	(rr)+	b
	3	14	C3	A	RR	-(rr)	c
	3	14	C3	A	rr	-(rr)	c
[OPC] [ofs,0|src,0] [XTN |dst,0]	3	14	60	A	rr	rr(rrx)	a
[OPC] [XTN |src,1] [ofs] [dst,0]	4	14	86	A	RR	N(rr)	a
	4	14	86	A	rr	N(rr)	a
[OPC] [XTN |dst,0] [src h] [src l]	4	14	E2	A	rr	NN	a
[OPC] [XTN |src,0] [ofs h] [ofs l] [dst,0]	5	16	86	A	RR	NN(rr)	a
	5	16	86	A	rr	NN(rr)	a

OPERATION a: dst AND src

Selected bits in the destination word are tested for a logical zero value. The bits to be tested are selected by setting to one the corresponding bits in the source word (mask) which is then ANDed with the destination word. The zero flag can then be checked to determine the result. The destination word remains unaltered by this instruction. The source word is held in the source memory location and the destination word in the destination register pair. The destination register pair is addressed directly, the memory location is addressed either directly, indirectly or by indexing.

OPERATION b: dst AND src
rr ⇐ rr + 2

As operation 'a' (indirect memory addressing only), but the contents of the destination register pair are incremented after the TMW has been carried out.

OPERATION c: rr ⇐ rr - 2
dst AND src

As operation 'a' (indirect memory addressing only), but the contents of the destination register pair are decremented before the TMW is carried out.

TMW TMW

Test Under Mask (Word) - Register, Memory

TMW dst,src (Cont'd)

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 15 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
TMW RR64,-(rr4)	C3 A5 40	1100 0011 1010 0101 0100 0000

If working register pair 4 contains 1184 (decimal), register pair 64 contains 11001100/11001100B and memory location 1182 contains 11001100/11001100B, after this instruction the zero flag will be set and register pair 4 will contain 1182.

TMW TMW

Test Under Mask (Word) - Memory, Register

TMW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [dst,1|src,0]	2	14	AE	-	(rr)	rr	a
[OPC] [XTN |dst,1] [src,0]	3	14	BE	A	(rr)	RR	a
[OPC] [XTN |dst,0] [src,0]	3	14	D5	A	(rr)+	RR	b
	3	14	D5	A	(rr)+	rr	b
	3	14	C3	A	-(rr)	RR	c
	3	14	C3	A	-(rr)	rr	c
[OPC] [ofd,0|dst,1] [XTN |src,0]	3	14	60	A	rr(rrx)	rr	a
[OPC] [XTN |dst,1] [ofd] [src,1]	4	14	86	A	N(rr)	RR	a
	4	14	86	A	N(rr)	rr	a
[OPC] [XTN |src,1] [dst h] [dst l]	4	16	E2	A	NN	rr	a
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src,1]	5	16	86	A	NN(rr)	RR	a
	5	16	86	A	NN(rr)	rr	a

OPERATION a: dst AND src

Selected bits in the destination word are tested for a logical zero value. The bits to be tested are selected by setting to one the corresponding bits in the source word (mask) which is then ANDed with the destination word. The zero flag can then be checked to determine the result. The destination word remains unaltered by this instruction.

The source word is held in the source register pair and the destination word in the destination memory location. The source register pair is addressed directly, the memory location is addressed either directly, indirectly or by indexing.

OPERATION b: dst AND src
rr ⇐ rr + 2

As operation 'a' (indirect memory addressing only), but the contents of the source register pair are incremented after the TMW has been carried out.

OPERATION c: rr ⇐ rr - 2
dst AND src

As operation 'a' (indirect memory addressing only), but the contents of the source register pair are decremented before the TMW is carried out.

TMW TMW

Test Under Mask (Word) - Memory, Register

TMW dst,src (Cont'd)

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 15 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
TMW (rr8),RR64	BE A9 40	1011 1110 1010 1001 0100 0000

If register pair 64 contains 11001100/11001100B, working register pair 8 contains 2000 (decimal) and memory location 2000 contains 11001100/11001100B, after this instruction the zero flag will be set.

TMW TMW

Test Under Mask (Word) - Memory, Memory

TMW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [dst,1|src,1]	2	16	AE	-	(rr)	(rr)

OPERATION: dst AND src

Selected bits in the destination word are tested for a logical zero value. The bits to be tested are selected by setting to one the corresponding bits in the source word (mask) which is then ANDed with the destination word. The zero flag can then be checked to determine the result. The destination word remains unaltered by this instruction.

The source word is held in the source memory location and the destination word in the destination register pair. The source word is in the memory location addressed by the contents of the source register pair. the destination word is in the memory location addressed by the contents of the destination register pair.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 15 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
TMW (rr4),(rr6)	AE 57	1010 1110 0101 0111

If working register pair 6 contains 1002 (decimal), memory location pair 1002 contains 11001100/11001100B, working register pair 4 contains 1060 (decimal) and memory location 1060 contains 11001100/11001100B, after this instruction the zero flag wil be set.

TMW TMW

Test Under Mask (Word) - All, Immediate

TMW dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst,1] [src h] [src l]	4	10	A7	-	RR	#NN
	4	10	A7	-	rr	#NN
[OPC] [XTN |dst,0] [src h] [src l]	4	14	BE	A	(rr)	#NN
[OPC] [XTN |dst,1] [ofd] [src h] [src l]	5	16	06	A	N(rr)	#NN
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src h] [src l]	6	18	06	A	NN(rr)	#NN
[OPC] [XTN] [src h] [src l] [dst h] [dst l]	6	20	36	A1	NN	#NN

OPERATION: dst AND src

Selected bits in the destination word are tested for a logical zero value. The bits to be tested are selected by setting to one the corresponding bits in the source word (mask) which is then ANDed with the destination word. The zero flag can then be checked to determine the result. The destination word remains unaltered by this instruction.

The source word is the immediate value held in the operand. the destination word can be in memory or register file.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 15 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
TMW RR64,#CCCCh	A7 41 CC CC	0011 0111 0100 0001 1100 1100 1100 1100

If register pair 64 contains 01001000/01001000B, after this instruction has been carried out the zero flag will be reset.

UNLINK UNLINK

Unlink code

UNLINK dst

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [dst]	2	10/6	75	-	rr	

OPERATION:

Stack in memory(10 cycles)	Stack in the register file (6 cycles)
SSP = RR	SSP(low) = RR(low)
RR = (SSP)	RR(low) = SSP
SSP = SSP + 2	SSP(low) = SSP(low) + 1
	SSP(high) = undefined

In C functions, the compiler needs to push variables in the user/system stacks and to keep the return address location of the function inside the stack.

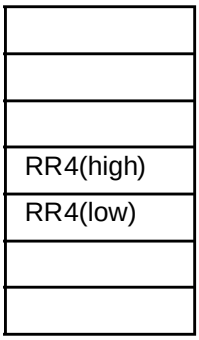
Therefore, a frame pointer is used, and 2 pieces of code named prologue and epilogue need to be added at the beginning and at the end of the function.

The ”Unlink” instruction is used to get the shortest size in the epilogue of a C function.

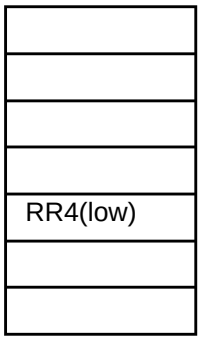
FLAGS: No flags affected.

EXAMPLES: UNLINK RR4

Stack in memory (16 cycles)



Stack in the register file (12 cycles)



After the instruction, RR4 register will have the value taken in the stack as indicated in the above scheme.

UNLINKU UNLINKU

Unlink code

UNLINKU

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC]	3	10/6	87	-	RR	#N

OPERATION:

Stack in memory(10 cycles)	Stack in the register file (6 cycles)
USP = RR	USP(low) = RR(low)
RR = (USP)	RR(low) = USP
SP = USP + 2	USP(low) = USP(low) + 1
	USP(high) = undefined

In C functions, the compiler needs to push variables in the user/system stacks and to keep the return address location of the function inside the stack.

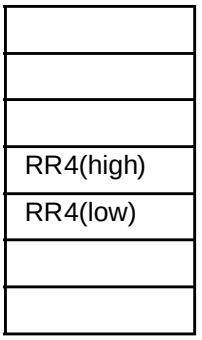
Therefore, a frame pointer is used, and 2 pieces of code named prologue and epilogue need to be added at the beginning and at the end of the function.

The ”Unlinku” instruction is used to get the shortest size in the epilogue of a C function.

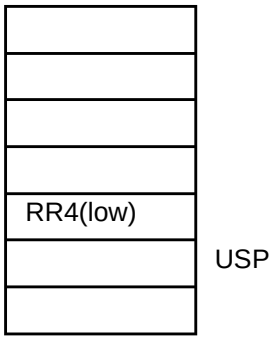
FLAGS: No flags affected.

EXAMPLES: UNLINKU RR4

Stack in memory (16 cycles)



Stack in the register file (12 cycles)



After the instruction, RR4 register will have the value taken in the stack as indicated in the above scheme.

WFI WFI

Wait For Interrupt

WFI

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
[OPC] [XTN]	2	4 + ...	EF	01	-	-

OPERATION : This instruction suspends program operation until an interrupt is acknowledged, although DMA requests are still serviced.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
WFI	EF 01	1110 1111 0000 0001

The program is suspended until an interrupt occurs.

XCH XCH

Exchange Registers

XCH dst,src

INSTRUCTION FORMAT:

[OPC] [src] [dst]	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
	3	6	16	-	R	R
	3	6	16	-	R	r
	3	6	16	-	r	R
	3	6	16	-	r	r

OPERATION: dst ⇐ src

The contents of the destination register are loaded into the source register and the contents of the source register loaded into the destination register.

FLAGS: No flags affected.

EXAMPLE:

Instruction	HEX	Binary
XCH r2,r4	16 D4 D2	0001 0110 1101 0100 1101 0010

If working register 2 contains 26 (decimal) and working register 4 contains 100 (decimal), after this instruction register 2 will contain 100 and register 4 will contain 26.

XOR XOR

Exclusive OR (byte) Register, Register

XOR dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst | src]	2	4	62	-	r	r
	2	6	63	-	r	(r)
[OPC] [src] [dst]	3	6	64	-	R	R
	3	6	64	-	r	R
	3	6	64	-	R	r
[OPC] [src] [XTN | dst]	3	6	E6	6	(r)	R
	3	6	E6	6	(r)	r
[OPC] [XTN | src] [dst]	3	6	E7	6	R	(r)

OPERATION: dst ⇐ dst XOR src

The contents of the source are XORed with the destination byte and the results stored in the destination byte. The contents of the source are not affected.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 7 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
XOR r8,R64	64 40 D8	0110 0100 0100 0000 1101 1000

If working register 8 contains 11001100 and register 64 contains 10000101, after this instruction working register 8 will contain 01001001.

XOR XOR

Exclusive OR (byte) Register, Memory

XOR dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [XTN |src,1] [dst]	3	8	72	6	R	(rr)	a
	3	8	72	6	r	(rr)	a
	3	12	B4	6	R	(rr)+	b
	3	12	B4	6	r	(rr)+	b
	3	12	C2	6	R	-(rr)	c
	3	12	C2	6	r	-(rr)	c
[OPC] [ofs,1|src,0] [XTN | dst]	3	12	60	6	r	rr(rrx)	a
[OPC] [XTN |src,1] [ofs] [dst]	4	12	7F	6	R	N(rr)	a
	4	12	7F	6	r	N(rr)	a
[OPC] [XTN | dst] [src h] [src l]	4	10	C4	6	r	NN	a
[OPC] [XTN |src,0] [ofs h] [ofs l] [dst]	5	14	7F	6	R	NN(rr)	a
	5	14	7F	6	r	NN(rr)	a

OPERATION a: dst ⇐ dst XOR src

The source byte is XORed with the destination byte and the result stored in the destination byte. The destination register is addressed directly, the memory location is addressed either directly, indirectly or by indexing.

OPERATION b: dst ⇐ dst XOR src
rr ⇐ rr + 1

The contents of the memory location addressed by the source register pair are XORed with the contents of the directly addressed destination register the result stored in the destination byte. The contents of the source register pair are incremented after the XOR has been carried out.

OPERATION c: rr ⇐ rr - 1
dst ⇐ dst XOR src

The contents of the source register pair are decremented and then the contents of the memory location addressed by the source register pair are XORed with the contents of the directly addressed destination register. The result is stored in the destination byte.

XOR XOR

Exclusive OR (byte) Register, Memory

XOR dst,src (Cont'd)

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 7 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
XOR r8,(rr4)	72 65 D8	0111 0010 0110 0101 1101 1000

If working register 8 contains 11001100, working register pair 4 contains 4200 (decimal) and memory location 4200 contains 10000101, after this instruction register 8 will contain 01001001.

XOR XOR

Exclusive OR (byte) Memory, Register

XOR dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [XTN |dst,0] [src]	3	12	72	6	(rr)	R	a
	3	12	72	6	(rr)	r	a
	3	14	B4	6	(rr)+	R	b
	3	14	B4	6	(rr)+	r	b
	3	14	C2	6	-(rr)	R	c
	3	14	C2	6	-(rr)	r	c
[OPC] [ofs,d|dst,1] [XTN | src]	3	14	60	6	rr(rrx)	r	a
[OPC] [XTN |dst,1] [ofd] [src]	4	14	26	6	N(rr)	R	a
	4	14	26	6	N(rr)	r	a
[OPC] [XTN | src] [dst h] [dst l]	4	12	C5	6	NN	r	a
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src]	5	16	26	6	NN(rr)	R	a
	5	16	26	6	NN(rr)	r	a

OPERATION a: dst ⇐ dst XOR src

The source byte is XORed with the destination byte and the result stored in the destination byte. The source registers are addressed directly, the memory location are addressed either directly, indirectly or by indexing.

OPERATION b: dst ⇐ dst XOR src
rr ⇐ rr + 1

The contents of the memory location addressed by the destination register pair (destination byte) are XORed with the contents of the directly addressed source register the result stored in the destination byte. The contents of the destination register pair are incremented after the XOR has been carried out.

OPERATION c: rr ⇐ rr - 1
dst ⇐ dst XOR src

The contents of the destination register pair are decremented and then the contents of the memory location addressed by the destination register pair (destination byte) are XORed with the contents of the directly addressed source register. The result is stored in the destination byte.

XOR XOR

Exclusive OR (byte) Memory, Register

XOR dst,src (Cont'd)

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 7 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
XOR 4028,r8	C5 18 0F BC	1100 0101 0001 1000 0000 1111 1011 1100

If memory location 4028 contains 11001100 and working register 8 contains 10000101, after this instruction memory location 4028 will contain 01001001.

XOR XOR

Exclusive OR (byte) Memory, Memory

XOR dst,src

INSTRUCTION FORMAT:

[OPC] [XTN |src,0] [dst,0]	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
	3	14	73	6	(RR)	(rr)
	3	14	73	6	(rr)*	(rr)

OPERATION : dst ⇐ dst XOR src

The contents of the memory addressed by the source register pair are XORed with the content of the memory location addressed by the destination register pair.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 7 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
XOR (rr4),(rr8)	73 68 D4	0111 0011 0110 1000 1101 0100

If working register pair 4 contains 2800 (decimal), memory location 2800 contains 11001100, working register pair 8 contains 4200 (decimal) and memory location 4200 contains 1100011, after this instruction memory location 2800 will contain 00001111.

XOR XOR

Exclusive OR (byte) All, Immediate

XOR dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst] [src]	3	6	65	-	R	#N
	3	6	65	-	r	#N
[OPC] [XTN |dst,0] [src]	3	10	F3	6	(rr)	#N
[OPC] [XTN] [src] [dst h] [dst l]	5	16	2F	61	NN	#N

OPERATION: dst ⇐ dst XOR src

The value #N is XORed with the content of the destination register or memory location (destination byte) and stored in the destination byte.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 7 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
XOR (rr8),#32	F3 68 20	1111 0011 0110 1000 0010 0000

If working register pair 8 contains 4028 (decimal) and memory location 4028 contains 11001100, after this instruction memory location 4028 will contain 11101100.

XORW XORW

Exclusive OR (Word) - Register, Register

XORW dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst,0|src,0]	2	8	6E	-	rr	rr
[OPC] [src,0] [dst,0]	3	8	67	-	RR	RR
	3	8	67	-	rr	RR
	3	8	67	-	RR	rr
[OPC] [src,0] [XTN | dst]	3	10	96	6	(r)	RR
	3	10	96	6	(r)	rr
[OPC] [XTN | src] [dst,0]	3	10	A6	6	RR	(r)
	3	10	A6	6	rr	(r)

OPERATION: dst ⇐ dst XOR src

The source word is XORed with the destination word and the result is stored in the destination word. The source and destination word can be addressed either directly or indirectly.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 15 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
XORW (r8),RR64	96 40 68	1001 0110 0100 0000 0110 1000

If register pair 64 contains 11001100/11001100B, working register 8 contains 200 (decimal) and register pair 200 contains 10101010/10101010B, after this instruction register pair 200 will hold 01100110/01100110B.

XORW XORW

Exclusive OR (Word) - Register, Memory

XORW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [dst,0|src,1]	2	12	6E	-	rr	(rr)	a
[OPC] [XTN |src,0] [dst,0]	3	12	7E	6	RR	(rr)	a
[OPC] [XTN |src,1] [dst,0]	3	14	D5	6	RR	(rr)+	b
	3	14	D5	6	rr	(rr)+	b
	3	14	C3	6	RR	-(rr)	c
	3	14	C3	6	rr	-(rr)	c
[OPC] [ofs,0|src,0] [XTN |dst,0]	3	14	60	6	rr	rr(rrx)	a
[OPC] [XTN |src,1] [ofs] [dst,0]	4	14	86	6	RR	N(rr)	a
	4	14	86	6	rr	N(rr)	a
[OPC] [XTN |dst,0] [src h] [src l]	4	14	E2	6	rr	NN	a
[OPC] [XTN |src,0] [ofs h] [ofs l] [dst,0]	5	16	86	6	RR	NN(rr)	a
	5	16	86	6	rr	NN(rr)	a

OPERATION a: dst ⇐ dst XOR src

The source word is XORed with the destination word and the result is stored in the destination word. The destination word is held in the destination register. The source word can be addressed directly, indirectly or by indexing.

OPERATION b: dst ⇐ dst XOR src
rr ⇐ rr + 2

The source word is XORed with the destination word and the result is stored in the destination word. The source word is in the memory location addressed by the source register pair, the destination word is in the destination register. The contents of the source register pair are incremented after the XOR has been carried out.

OPERATION c: rr ⇐ rr - 2
dst ⇐ dst XOR src

The contents of the source register pair are decremented before the XOR is carried out. The source word is XORed with the destination word and the result is stored in the destination word. The source word is in the memory location addressed by the source register pair, the destination word is in the destination register.

XORW XORW

Exclusive OR (Word) - Register, Memory

XORW dst,src (Cont'd)

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 15 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
XORW RR64,-(rr4)	C3 65 40	1100 0011 0110 0101 0100 0000

If working register pair 4 contains 1184 (decimal), register pair 64 contains 10101010/10101010B and memory location 1182 contains 11001100/11001100B, after this instruction register pair 64 will contain 01100110/01100110B and register pair 4 will contain 1182.

XORW XORW

Exclusive OR (Word) - Memory, Register

XORW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Addr Mode dst	Addr Mode src	Oper
[OPC] [dst,1|src,0]	2	16	6E	-	(rr)	rr	a
[OPC] [XTN |dst,1] [src,0]	3	18	BE	6	(rr)	RR	a
[OPC] [XTN |dst,0] [src,0]	3	18	D5	6	(rr)+	RR	b
	3	18	D5	6	(rr)+	rr	b
	3	18	C3	6	-(rr)	RR	c
	3	18	C3	6	-(rr)	rr	c
[OPC] [ofd,0|dst,1] [XTN |src,0]	3	18	60	6	rr(rrx)	rr	a
[OPC] [XTN |dst,1] [ofd] [src,1]	4	18	86	6	N(rr)	RR	a
	4	18	86	6	N(rr)	rr	a
[OPC] [XTN |src,1] [dst h] [dst l]	4	18	E2	6	NN	rr	a
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src,1]	5	20	86	6	NN(rr)	RR	a
	5	20	86	6	NN(rr)	rr	a

OPERATION a: dst ⇐ dst XOR src

The source word is XORed with the destination word and the result is stored in the destination word. The source word is held in the source register. The destination word can be addressed directly, indirectly or by indexing.

OPERATION b: dst ⇐ dst XOR src
rr ⇐ rr + 2

The source word is XORed with the destination word and the result is stored in the destination word. The source word is in the source register, the destination word is in the memory location addressed by the destination register pair. The contents of the destination register pair are incremented after the XOR has been carried out.

OPERATION c: rr ⇐ rr - 2
dst ⇐ dst XOR src

The contents of the destination register pair are decremented before the XOR is carried out. The source word is XORed with the destination word and the result is stored in the destination word. The source word is in the source register , the destination word is in the memory location addressed by the destination register pair.

XORW XORW

Exclusive OR (Word) - Memory, Register

XORW dst,src (Cont'd)

FLAGS:

C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 15 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
XORW (rr4)+,RR64	D5 64 40	1101 0101 0110 0100 0100 0000

If register pair 64 contains 11001100/11001100B, working register pair 4 contains 1064 (decimal) and memory location 1064 contains 10101010/10101010B, after this instruction is carried out memory location 1064 will contain 01100110/01100110B and working register pair 4 will contain 1066.

XORW XORW

Exclusive OR (Word) - Memory, Memory

XORW dst,src

INSTRUCTION FORMAT:	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode	
					dst	src
`[ OPC ] [dst,1|src,1]`	2	20	6E	-	(rr)	(rr)

OPERATION: dst ⇐ dst XOR src

The source word is XORed with the destination word and the result is stored in the destination word. The source word is in the memory location addressed by the source register pair, the destination word is in the memory location addressed by the destination register pair.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 15 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
XORW (rr4),(rr6)	6E 57	0110 1110 0101 0111

If working register pair 6 contains 1002 (decimal), memory location 1002 contains 11001100/11001100B, working register pair 4 contains 1060 (decimal) and memory location 1060 contains 10101010/10101010B, after this instruction memory location 1060 will contains 01100110/01100110B.

XORW XORW

Exclusive OR (Word) - All, Immediate

XORW dst,src

INSTRUCTION FORMAT:

	No. Bytes	No. Cycl	OPC (HEX)	OPC XTN	Address Mode dst	Address Mode src
[OPC] [dst,1] [src h] [src l]	4	10	67	-	RR	#NN
	4	10	67	-	rr	#NN
[OPC] [XTN |dst,0] [src h] [src l]	4	18	BE	6	(rr)	#NN
[OPC] [XTN |dst,1] [ofd] [src h] [src l]	5	20	06	6	N(rr)	#NN
[OPC] [XTN |dst,0] [ofd h] [ofd l] [src h] [src l]	6	22	06	6	NN(rr)	#NN
[OPC] [XTN] [src h] [src l] [dst h] [dst l]	6	22	36	61	NN	#NN

OPERATION: dst ⇐ dst XOR src

The source word is XORed with the destination word and the result is stored in the destination word. The source word is the immediate value in the operand, the destination word can be in memory or in the register file.

FLAGS:
C: Unaffected.
Z: Set if the result is zero, otherwise cleared.
S: Set if result bit 15 is set, otherwise cleared.
V: Always reset to zero.
D: Unaffected.
H: Unaffected.

EXAMPLE:

Instruction	HEX	Binary
XORW RR64,#52428	67 41 CC CC	0110 0111 0100 0001 1100 1100 1100 1100

If register pair 64 contains 10101010/10101010B, after this instruction has benn carried out register pair 64 will contains 01100110/01100110B.

Notes: