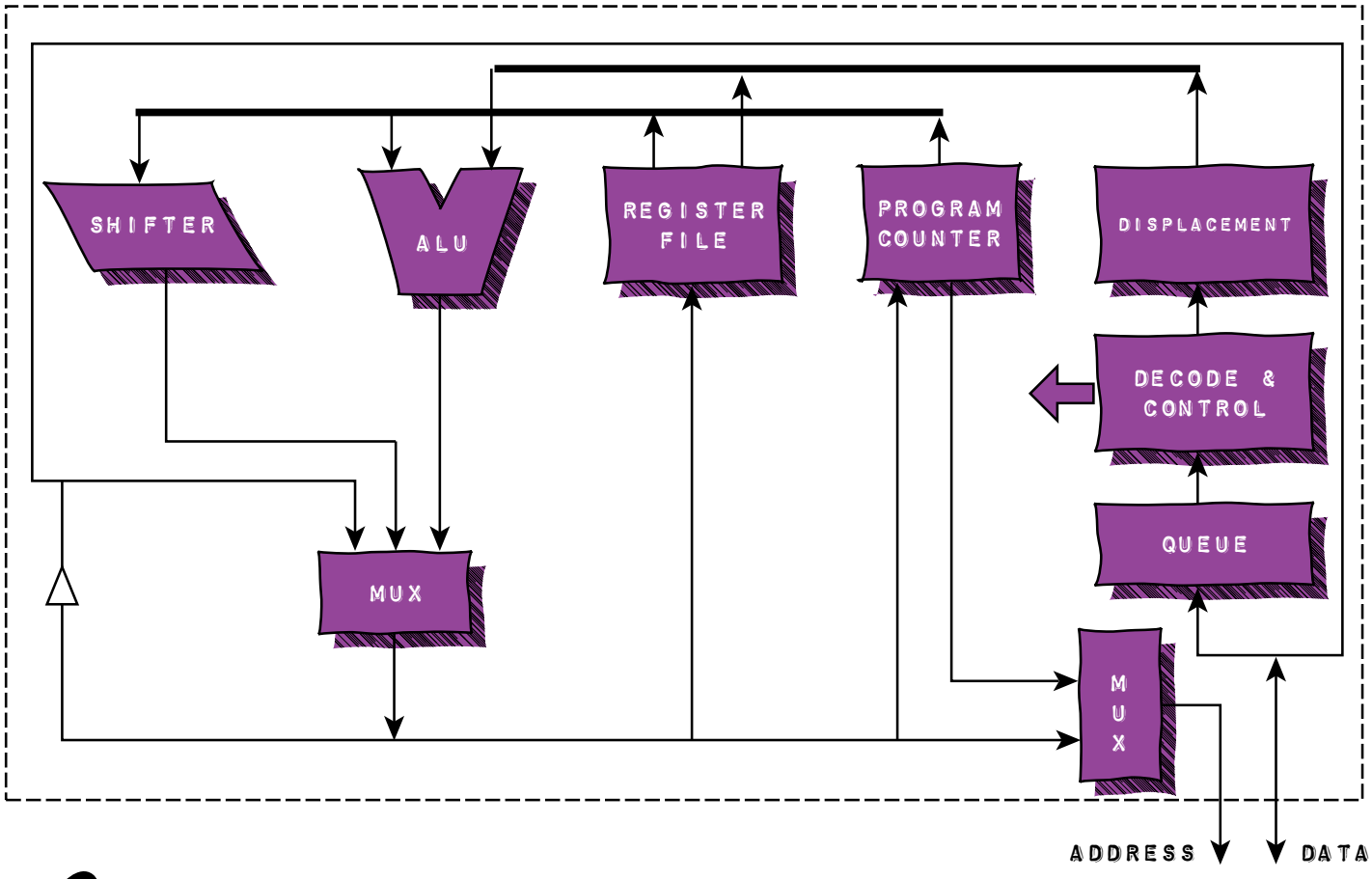
NATIONAL SEMICONDUCTOR
COMPACTRISC™ 32-BIT ARCHITECTURE

INTRODUCTION

CompactRISC processor cores are designed specifically for embedded applications. The CompactRISC architecture is available in a wide range of implementations, supported by a common set of software development tools from multiple vendors. The CompactRISC architecture is scalable over a range of 8-, 16-, 32- and 64-bit processors, with common: variable-length instruction set, registers, addressing modes, interrupt and trap handling, debug support, and non-aligned data accesses, and efficient HLL execution.

FEATURES

- Less than 2.5mm^2 @ 0.5µ
- 4Gbytes of linear address space
- 0.5mA per MHZ @ 3 Volts, 0.5µ
- Static 0 to 30 Megahertz



National Semiconductor®

INTRODUCTION

The CR32 has 24 internal registers grouped by function as follows:

- 16 general purpose registers
- The following processor registers:
  3 dedicated address registers
  1 processor status register
  1 configuration register
  3 debug registers

GENERAL PURPOSE REGISTER

Registers R0-R13 are used for general purposes, such as holding variables, addresses or index values. The SP general purpose register is usually used as a pointer to the program run-time stack. The RA general purpose register is usually used as a return address from sub-routines. If a general purpose register is specified by an operation that is 8 or 16 bits long, then only the low part of the register is used; the high part is not referenced or modified.

DEDICATED ADDRESS REGISTERS

PC

Program Counter - The value in the PC register, points to the first byte of the instruction currently being executed.

ISP

Interrupt Stack Pointer - The ISP register points to the lowest address of the last item stored on the interrupt stack. This stack is used when interrupt and trap service routines are invoked.

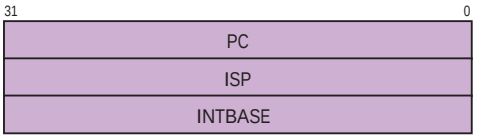
INTBASE

Interrupt Base Register - The INTBASE register holds the address of the dispatch table for interrupts and traps.

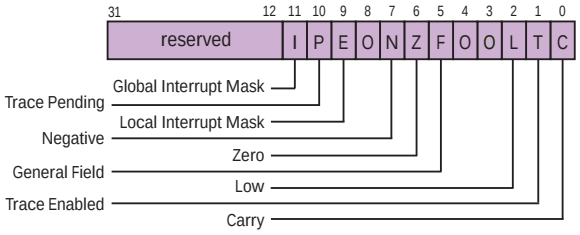
PROCESS STATUS REGISTERS

The Processor Status Register (PSR) holds status information and selects operating modes for the CR32. It is 32-bits wide. The figure to the right shows the format of the PSR.

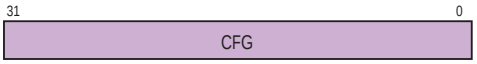
Dedicated Address Register



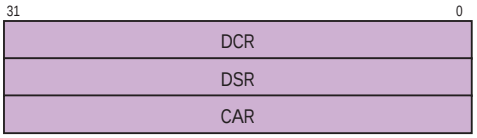
Process Status Register



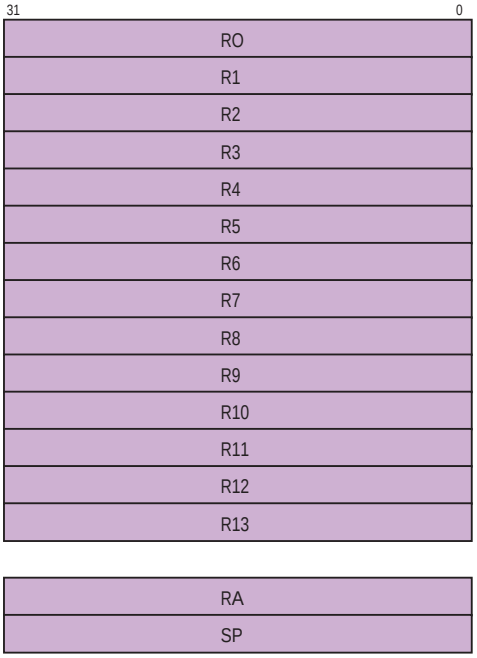
Configuration Register



Debug Registers



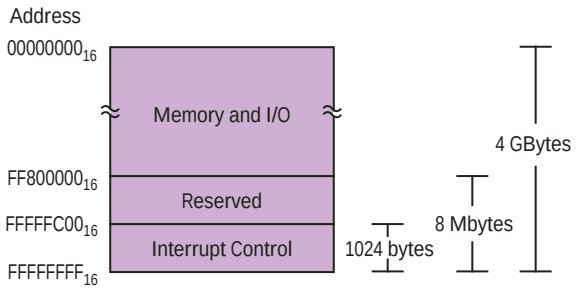
General Purpose Registers



MEMORY ORGANIZATION

The CR32 implements full 32-bit addresses. This allows the CPU to access up to 4 Gbytes of memory. The memory is a uniform linear address space. Memory locations are numbered sequentially starting at 0 and ending at 2^{32}-1. The number specifying a memory location is called an address.

The CR32 can access the memory by the load and store instructions. It can access non-aligned data: bytes, word, and double-words can be referenced on any boundary.



STACKS

The CR32 provides support for two kinds of stacks: the interrupt stack and the program stack.

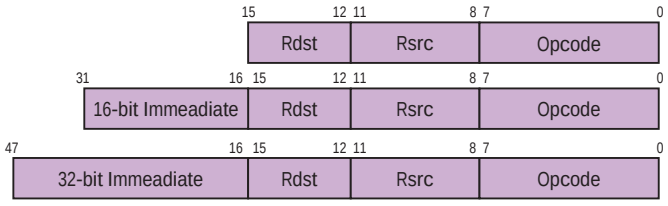
Interrupt Stack

The interrupt stack is used by the processor to save and restore the program state during the handling of an exception condition. The interrupt stack is accessed via the ISP processor register.

Program Stack

The program stack is normally used by programs at run time, to save and restore register values upon procedure entry and exit. It is also used to store local and temporary variables. The program stack is accessed via the SP general purpose register.

INSTRUCTION FORMAT



Instruction Commands

The following instructions are included in the CR32.

Mnemonic	Operands	Description
MOVES		
MOVi	Rsrc/imm, Rdest	Move
MOVXi	Rsrc, Rdest	Move with sign extension
MOVZi	Rsrc, Rdest	Move with zero extension
INTEGER ARITHMETIC		
ADD[U]i	Rsrc/imm, Rdest	Add
ADDCi	Rsrc/imm, Rdest	Add with carry
MULi	Rsrc/imm, Rdest	Multiply
SUBi	Rsrc/imm, Rdest	Subtract (Rdest := Rdest – Rsrc)
SUBCi	Rsrc/imm, Rdest	Subtract w/carry (Rdest – Rsrc – PSR.C)
INTEGER COMPARISON		
CMPi	Rsrc/imm, Rdest	Compare (Rdest – Rsrc)
LOGICAL AND BOOLEAN		
ANDi	Rsrc/imm, Rdest	Logical AND
BICi	Rsrc/imm, Rdest	Clear selected bits
ORi	Rsrc/imm, Rdest	Logical OR
Scond	Rdest	Save condition code as boolean
XORi	Rsrc/imm, Rdest	Logical exclusive OR
SHIFTS		
ASHUi	Rsrc/imm, Rdest	Arithmetic left/right shift
LSHi	Rsrc/imm, Rdest	Logical left/right shift
BITS		
TBIT	Roffset/imm, Rsrc	Test bit
PROCESSOR REGISTER MANIPULATION		
LPR	Rsrc, Rproc	Load processor register
SPR	Rproc, Rdest	Store processor register
JUMPS AND LINKAGE		
Bcond	disp	Conditional branch
BAL	Rlink, disp	Branch and link
BR	disp	Branch
EXCP	vector	Trap (vector)
Jcond	Rtarget	Conditional Jump
JAL	Rlink, Rtarget	Jump and link
JUMP	Rtarget	Jump
RETX		Return from exception
LOAD AND STORE		
LOADi	disp(Rbase), Rdest	
	abs, Rdest	
STORi	Rsrc, disp(Rbase)	
	Rsrc, abs	
MISCELLANEOUS		
CINV		Cache Invalidation
DI		Disable maskable interrupts
EI		Enable maskable interrupts
NOP		No operation
WAIT		Wait for interrupt