

2 RISC CPU

The CPU in ETRAX 100 is a 32-bit RISC CPU with a 16-bit instruction width. The CPU complies with the Axis Code Reduced Instruction Set (CRIS) architecture. It runs at a cycle frequency of 100 MHz, giving a peak performance of 100 MIPS. A summary of the CRIS architecture is given below. The CRIS CPU architecture is described in more detail in "CRIS Programmer's Manual".

2.1 REGISTERS

The processor contains fifteen 32-bit General Registers (R0 - R14), a 32-bit Program Counter (R15 or PC) and sixteen Special Registers (P0 - P15) of which eight are implemented. The remaining eight registers are reserved for future use. The register R14 is used as Stack Pointer (SP). The registers are presented in the figures below:

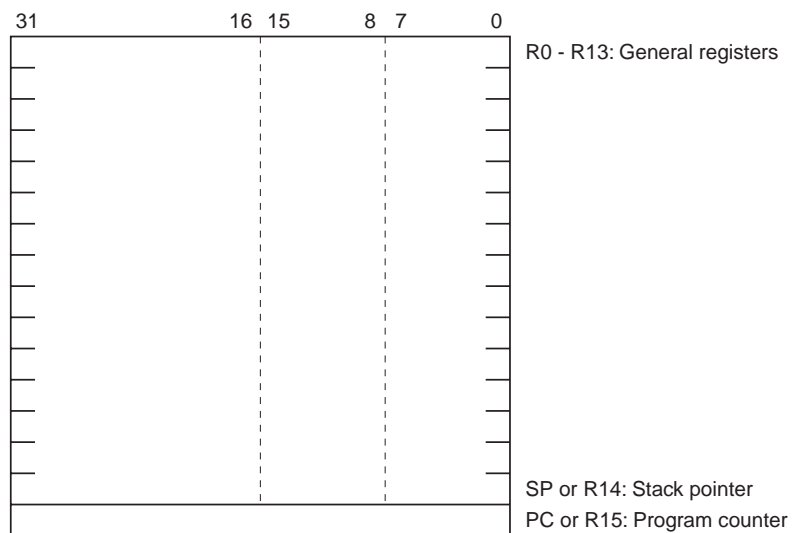


Figure 2-1 General registers

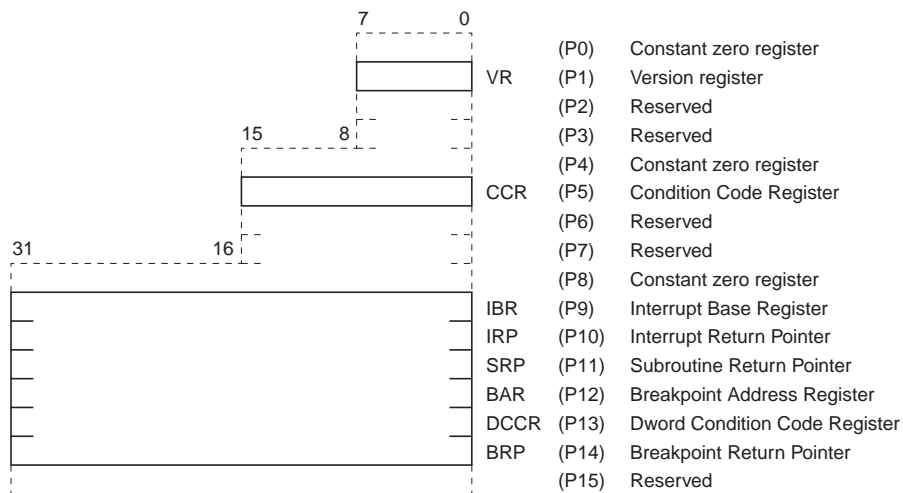


Figure 2-2 Special registers

2.2 FLAGS AND CONDITION CODES

The Condition Code Register (CCR) and the Dword Condition Code Register (DCCR) contain eight different flags. The remaining bits of the CCR and DCCR are always zero. Please note that CCR and DCCR access the same flags.

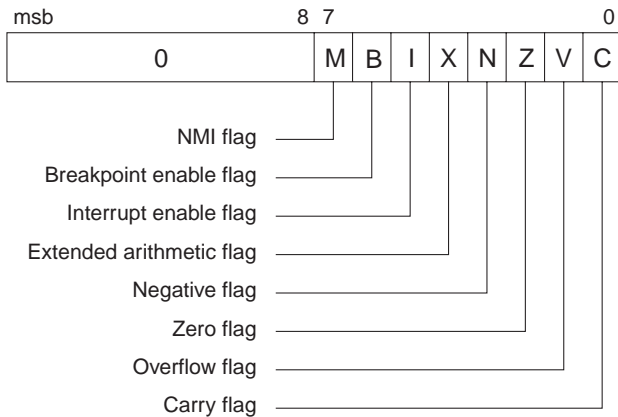


Figure 2-3 Condition Code Register/Dword Condition Code Register

The flags can be tested using one of the 15 condition codes specified below:

Code	Alt	Condition	Encoding	Boolean function
CC	HS	Carry Clear	0000	\bar{C}
CS	LO	Carry Set	0001	C
NE		Not Equal	0010	\bar{Z}
EQ		Equal	0011	Z
VC		Overflow Clear	0100	\bar{V}
VS		Overflow Set	0101	V
PL		Plus	0110	\bar{N}
MI		Minus	0111	N
LS		Low or Same	1000	C+Z
HI		High	1001	$\bar{C}*\bar{Z}$
GE		Greater or Equal	1010	$N*V+\bar{N}*\bar{V}$
LT		Less Than	1011	$N*\bar{V}+\bar{N}*V$
GT		Greater Than	1100	$N*V*\bar{Z}+\bar{N}*V*Z$
LE		Less or Equal	1101	$Z+N*\bar{V}+\bar{N}*V$
A		Always True	1110	1
		(Reserved)	1111	

Table 2-1 Condition codes

2.3 DATA ORGANIZATION IN MEMORY

The data types supported by CRIS are:

Name	Description	Size Modifier
Byte	8-bit integer	.B
Word	16-bit integer	.W
Dword	32-bit integer or address	.D

Table 2-2 Data types supported by CRIS

Each address location contains one byte of data. Data is stored in memory with the least significant byte at the lowest address ("little endian"). The CRIS CPU data bus in ETRAX 100 is 32-bit wide. A conversion from 32-bit to 16-bit is performed by the bus interface in case of an external 16-bit data bus mode.

Data can be aligned to any address. If the data crosses a 32-bit boundary, the CPU will split the data access into two separate accesses. The use of unaligned word and dword data will thus degrade the performance.

The figures below show examples of data organization with a 16-bit bus and a 32-bit bus:

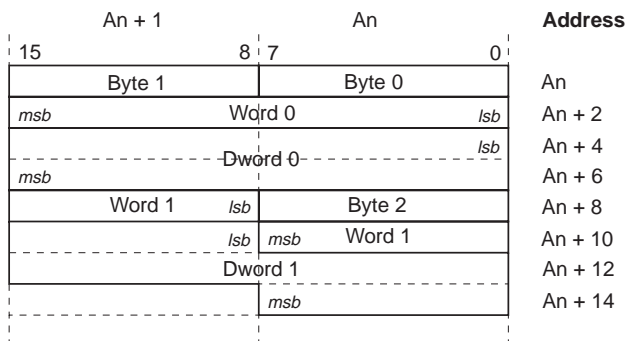


Figure 2-4 Example of data organization with a 16-bit bus

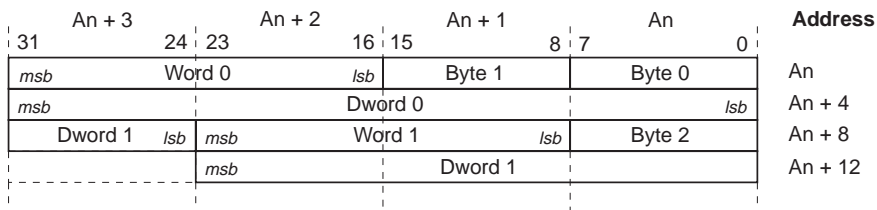


Figure 2-5 Example of data organization with a 32-bit bus

2.4 INSTRUCTION FORMAT

The basic instruction word is 16 bits long. Instructions must be 16-bit aligned.

When the CPU fetches 32 bits, containing two 16-bit aligned instructions, it saves the upper two bytes in an internal prefetch register. Thus, the CPU will only perform one read for every second instruction when running consecutive code.

The most common instructions follow the same general instruction format:

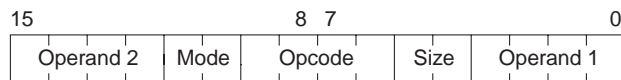


Figure 2-6 General instruction format

The following definitions apply to the instruction descriptions:

Syntax	Definition
m	Size modifier, byte, word or dword
z	Size modifier, byte or word
Rm	General register
Rn	General register
Rp	General register
Rs	Source operand, register addressing mode
[Rs]	Source operand, indirect addressing mode
[Rs+]	Source operand, auto increment addressing mode
s	Source operand, any addressing mode except quick immediate
si	Source operand, any mode except register or quick immediate
se	Source operand, indexed, offset, double indirect or absolute addressing mode
Pn	Special register
Ps	Source operand, special register
i	6-bit signed immediate operand
j	6-bit unsigned immediate operand
c	5-bit immediate shift value
Rd	Destination operand, register addressing mode
[Rd]	Destination operand, indirect addressing mode
[Rd+]	Destination operand, auto increment addressing mode
d	Destination operand, any addressing mode except quick immediate
di	Destination operand, any mode except register or quick immediate
Pd	Destination operand, special register
o	8-bit branch offset, bit 0 is the sign bit
x	8-bit signed immediate value
xx	16-bit signed immediate value
xxxx	32-bit signed immediate value
u	8-bit unsigned immediate value
uu	16-bit unsigned immediate value
uuuu	32-bit unsigned immediate value
cc	Condition code
n	4-bit breakpoint entry number

Table 2-3 Syntax definitions

The following definitions apply in the descriptions of how the flags are affected:

-	Flag not affected
0	Flag cleared
1	Flag set
*	Flag affected according to the result of the operation

Table 2-4 Flags behavior definitions

Instructions that do not have size modifiers operate on 32-bit data, except for instructions operating on 8-bit and 16-bit special registers.

2.4.1 Addressing Modes

The CRIS CPU has four basic addressing modes. These modes are encoded in the mode field of the instruction word. The basic addressing modes are:

- Quick immediate mode
- Register mode
- Indirect mode
- Autoincrement mode (with immediate mode as a special case)

More complex addressing modes can be achieved by combining the basic instruction word with an addressing mode prefix word. The complex addressing modes are:

- Indexed
- Indexed with assign
- Offset
- Offset with assign
- Double indirect
- Absolute

The addressing modes of the CRIS CPU are described in the table below:

Assembler syntax	Addressing mode
i, j	Quick immediate
Rn	Register
Pn	Special register
[Rn]	Indirect
[Rn+]	Post increment
x, u	Byte immediate
xx, uu	Word immediate
xxxx, uuuu	Dword immediate
[Rn+Rm.m]	Indexed
[Rp=Rn+Rm.m]	Indexed with assign
[Rn+[Rm].m]	Indirect offset
[Rn+[Rm+].m]	Autoincrement offset
[Rn+x]	Immediate byte offset
[Rn+xx]	Immediate word offset
[Rn+xxxx]	Immediate dword offset
[Rp=Rn+[Rm].m]	Indirect offset with assign
[Rp=Rn+[Rm+].m]	Autoincrement offset with assign
[Rp=Rn+x]	Immediate byte offset with assign
[Rp=Rn+xx]	Immediate word offset with assign
[Rp=Rn+xxxx]	Immediate dword offset with assign
[[Rn]]	Double indirect
[[Rn+]]	Double indirect with auto increment
[uuuu]	Absolute

Table 2-5 The CRIS CPU addressing modes

2.4.2 Data Transfers

The data transfer instructions of the CRIS CPU, the two predefined assembler macros POP and PUSH, and the word/byte/bit swap instruction set are specified in table 2-6.

Instruction	Flag operation							Description	
	M	B	I	X	N	Z	V		C
CLEAR.m d	-	-	-	0	-	-	-	-	Clear destination operand
MOVE.m s, Rd	-	-	-	0	*	*	0	0	Move from source to general register
MOVE.m Rs, di	-	-	-	0	-	-	-	-	Move from general register to memory
MOVE s, Pd	-	-	-	0	-	-	-	-	Move from source to special register
MOVE Ps, d	-	-	-	0	-	-	-	-	Move from special register to destination
MOVEM Rs, di	-	-	-	0	-	-	-	-	Move multiple registers to memory
MOVEM si, Rd	-	-	-	0	-	-	-	-	Move from memory to multiple registers
MOVEQ i, Rd	-	-	-	0	*	*	0	0	Move 6-bit signed immediate
MOVS.z s, Rd	-	-	-	0	*	*	0	0	Move with sign extend
MOVU.z s, Rd	-	-	-	0	0	*	0	0	Move with zero extend
POP Rd	-	-	-	0	*	*	0	0	Pop register from stack
POP Pd	-	-	-	0	-	-	-	-	Pop special register from stack
PUSH Rs	-	-	-	0	-	-	-	-	Push register onto stack
PUSH Ps	-	-	-	0	-	-	-	-	Push special register onto stack
SWAP<option> Rd	-	-	-	0	*	*	0	0	Swap operand bits, see 2-7

Table 2-6 The CRIS CPU data transfer instructions

Option	Description
N	Invert each bit
W	Swap the words of the operand
B	Swap the two bytes within each word of the operand
R	Reverse the bit order within each byte of the operand

Table 2-7 Options for the word/byte/bit swap instruction

2.4.3 Arithmetic Instructions

The arithmetic instructions of the CRIS CPU are described in the table below:

Instruction	Flag operation								Description
	M	B	I	X	N	Z	V	C	
ABS Rs, Rd	-	-	-	0	*	*	0	0	Absolute value
ADD.m s, Rd	-	-	-	0	*	*	*	*	Add source to destination register
ADDI Rs.m, Rd	-	-	-	0	-	-	-	-	Add scaled index to base
ADDQ j, Rs	-	-	-	0	*	*	*	*	Add 6-bit unsigned immediate
ADDS.z s, Rd	-	-	-	0	*	*	*	*	Add sign extended source to register
ADDU.z s, Rd	-	-	-	0	*	*	*	*	Add zero extended source to register
BOUND.m s, Rd	-	-	-	0	*	*	0	0	Adjust table index (unsigned min)
CMP.m s, Rd	-	-	-	0	*	*	*	*	Compare source to register
CMPQ i, Rd	-	-	-	0	*	*	*	*	Compare with 6-bit signed immediate
CMPS.z si, Rd	-	-	-	0	*	*	*	*	Compare with sign extended source
CMPU.z si, Rd	-	-	-	0	*	*	*	*	Compare with zero extended source
DSTEP Rs, Rd	-	-	-	0	*	*	0	0	Divide step
MSTEP Rs, Rd	-	-	-	0	*	*	0	0	Multiply step
NEG.m Rs, Rd	-	-	-	0	*	*	*	*	Negate (2's complement)
SUB.m s, Rd	-	-	-	0	*	*	*	*	Subtract source from register
SUBQ j, Rd	-	-	-	0	*	*	*	*	Subtract 6-bit unsigned immediate
SUBS.z s, Rd	-	-	-	0	*	*	*	*	Subtract with sign extended source
SUBU.z s, Rd	-	-	-	0	*	*	*	*	Subtract with zero extended source
TEST.m s	-	-	-	0	*	*	0	0	Compare operand with 0

Table 2-8 Arithmetic instructions

2.4.4 Logical Instructions

The logical instructions of the CRIS CPU are described in the table below:

Instruction	Flag operation								Description
	M	B	I	X	N	Z	V	C	
AND.m s, Rd	-	-	-	0	*	*	0	0	Bitwise logical AND
ANDQ i, Rd	-	-	-	0	*	*	0	0	AND with 6-bit signed immediate
NOT Rd	-	-	-	0	*	*	0	0	Logical NOT (1's complement)
OR.m s, Rd	-	-	-	0	*	*	0	0	Bitwise logical OR
ORQ i, Rd	-	-	-	0	*	*	0	0	OR with 6-bit signed immediate
XOR Rs, Rd	-	-	-	0	*	*	0	0	Bitwise Exclusive OR

Table 2-9 Logical instructions

2.4.5 Shift Instructions

The shift instructions of the CRIS CPU are shown in the table below. When the shift count is contained in a register, the 6 least significant bits of the register are used as an unsigned shift count.

Instruction	Flag operation							Description	
	M	B	I	X	N	Z	V		C
ASR.m Rs, Rd	-	-	-	0	*	*	0	0	Right shift Rd with sign fill
ASRQ c, Rd	-	-	-	0	*	*	0	0	Right shift Rd with sign fill
LSL.m Rs, Rd	-	-	-	0	*	*	0	0	Left shift Rd with zero fill
LSLQ c, Rd	-	-	-	0	*	*	0	0	Left shift Rd with zero fill
LSR.m Rs, Rd	-	-	-	0	*	*	0	0	Right shift Rd with zero fill
LSRQ c, Rd	-	-	-	0	*	*	0	0	Right shift Rd with zero fill

Table 2-10 Shift instructions

2.4.6 Bit Test Instructions

The bit test instructions of the CRIS CPU are shown in the table below.

Instruction	Flag operation							Description	
	M	B	I	X	N	Z	V		C
BTST Rs, Rd	-	-	-	0	*	*	0	0	Test bit Rs in register Rd
BTSTQ c, Rd	-	-	-	0	*	*	0	0	Test bit c in register Rd
LZ Rs, Rd	-	-	-	0	0	*	0	0	Number of leading zeroes

Table 2-11 Bit test instructions

The BTST and BTSTQ instructions set the Z flag if the selected bit and all bits to the right of it are zero, and the N flag according to the selected bit in the destination register.

2.4.7 Condition Code Manipulation Instructions

The condition code manipulation instructions of the CRIS CPU are shown in the table below. The predefined assembler macros EI, DI and AX are also shown.

Instruction	Flag operation							Description	
	M	B	I	X	N	Z	V		C
AX	-	-	-	1	-	-	-	-	Arithmetic extend (SETF X)
CLEARF <list>	-	*	*	0	*	*	*	*	Clear flags in list
DI	-	-	0	0	-	-	-	-	Disable interrupts (CLEARF I)
EI	-	-	1	0	-	-	-	-	Enable interrupts (SETF I)
Scc Rd	-	-	-	0	-	-	-	-	Set register according to cc
SETF <list>	*	*	*	*	*	*	*	*	Set flags in list

Table 2-12 Condition code manipulation instructions

2.4.8 Jump and Branch Instructions

The jump and branch instructions of the CRIS CPU and the predefined assembler macros RET, RETB and RETI are shown in the table below:

Instruction	Flag operation							Description	
	M	B	I	X	N	Z	V		C
Bcc o	-	-	-	0	-	-	-	-	Conditional relative branch, see note 1
Bcc xx	-	-	-	0	-	-	-	-	Branch with 16-bit offset, see note 1
BREAK n	-	-	-	0	-	-	-	-	Breakpoint
JBRC s	-	-	-	0	-	-	-	-	Jump to breakpoint routine, see note 2
JIR s	-	-	-	0	-	-	-	-	Jump to interrupt routine
JIRC s	-	-	-	0	-	-	-	-	Jump to interrupt routine, see note 2
JSR s	-	-	-	0	-	-	-	-	Jump to subroutine
JSRC s	-	-	-	0	-	-	-	-	Jump to subroutine, see note 2
JUMP s	-	-	-	0	-	-	-	-	Jump
RET	-	-	-	0	-	-	-	-	Return from subroutine, see note 1
RETB	-	-	-	0	-	-	-	-	Return from breakpoint routine, see note 1
RETI	-	-	-	0	-	-	-	-	Return from interrupt routine, see note 1

Table 2-13 Jump and Branch Instructions

Note 1: The Bcc, RET, RETB and RETI instructions have delayed effect with one delay slot.

Note 2: The JBRC, JIRC and JSRC instructions will add four bytes to the return address stored to either SRP, IRP or BRP. This leaves four bytes unused between the JSRC/JIRC/JBRC instruction and the return point. This can be used to enhance C++ exception support.

2.4.9 No Operation Instruction

Finally, the CRIS CPU also has a no operation instruction, NOP.

Instruction	Flag operation							Description	
	M	B	I	X	N	Z	V		C
NOP	-	-	-	0	-	-	-	-	No operation

Table 2-14 No Operation Instruction

2.5 INTERRUPTS

The CRIS CPU uses vectorized interrupts that are generated either externally to, or internally by, the ETRAX 100. The interrupt acknowledge sequence is as follows:

1. Perform an INTA cycle, where the 8-bit vector number is read from the bus.
2. Store the contents of PC to the Breakpoint Return Pointer (BRP) for NMI or to the Interrupt Return Pointer (IRP) for other interrupts. Note that the return address is not automatically pushed on the stack.
3. Read the interrupt vector from the address [IBR + <vector number> * 4].
4. Start the execution at the address pointed to by the interrupt vector.

The Interrupt Base Register (IBR) have bit 31-16 implemented. The remaining bits are always zero.

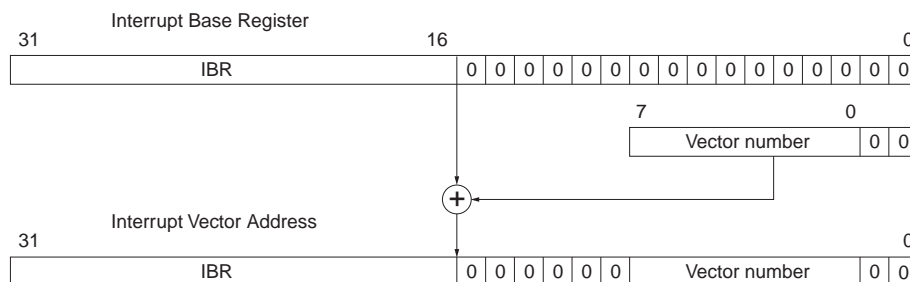


Figure 2-7 Interrupt Vector address calculation

2.5.1 NMI

The Non Maskable Interrupt (NMI) is handled in the same way as the normal interrupt, with two differences:

1. The return address is stored in the Breakpoint Return Pointer (BRP) instead of the IRP.
2. The NMI is enabled/disabled by the M flag instead of the I flag. The M flag can be set with the SETF M instruction. Move to CCR has no effect. Once set, the M flag can only be cleared by an NMI acknowledge cycle or system reset.

2.6 SOFTWARE BREAKPOINT MECHANISM

The CRIS CPU has a breakpoint instruction (BREAK n). This instruction saves the current value of PC in the Breakpoint Return Pointer (BRP) register and performs a jump to address (IBR + 8 * n).

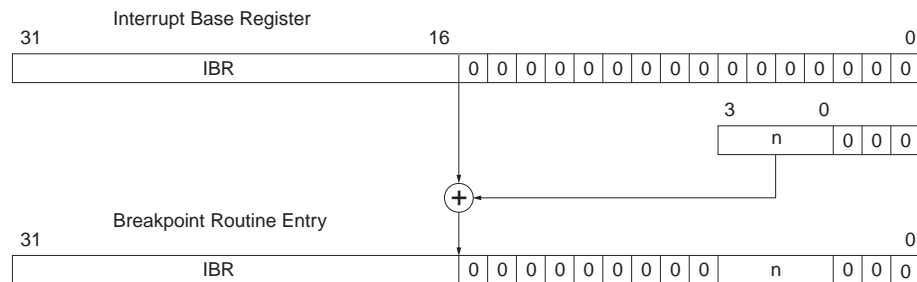


Figure 2-8 Breakpoint Routine Entry address calculation

2.7 HARDWARE BREAKPOINT MECHANISM

The CPU contains a hardware breakpoint mechanism. The hardware breakpoint address is loaded in the BAR register (Breakpoint Address Register), and the hardware breakpoint mechanism is enabled by setting the Breakpoint enable flag B (see Figure 2-3 on page 12).

For each CPU read or write cycle, the address is compared with the contents of the BAR register. In order to detect a read or write in the dword (and not just a single byte) of the address location, bit 1 and 0 are ignored in the comparison. Bit 31 is also ignored in the comparison since that bit handles the cache (address bit 31 set will bypass the cache and directly access the main memory).

An address hit is handled in the same way as an NMI with interrupt vector number 0x20, except that a breakpoint hit is not affected by the M flag.

The hardware breakpoint mechanism is disabled after reset.