

CoolRISC™

Quick reference for the CoolRISC 816

Instruction set

Instruction	Modification
Jump addr(16bits) or ip	~*~*
Jcc addr(16bits) or ip	~*~*
Call addr(16bits) or ip	~*~*
Calls addr(16bits) or ip	~*~*
Ret	~*~*
Rets	~*~*
Reti	~*~*
Push	~*~*
Pop	~*~*
Move reg,#data(8bits)	~*,Z,a
Move reg1,reg2	~*,Z,a
Move reg,eaddr	~*,Z,a
Move eaddr,reg	~*~*
Move addr(8bits),#data(8bits)	~*~*
Cmvd reg1,reg2	~*,Z,a
Cmvs reg,eaddr	~*,Z,a
Shl reg1,reg2	C,V,Z,a
Shl reg	C,V,Z,a
Shl reg,eaddr	C,V,Z,a
Shlc reg1,reg2	C,V,Z,a
Shlc reg	C,V,Z,a
Shlc reg,eaddr	C,V,Z,a
Shr reg1,reg2	C,V,Z,a
Shr reg	C,V,Z,a
Shr reg,eaddr	C,V,Z,a
Shrc reg1,reg2	C,V,Z,a
Shrc reg	C,V,Z,a
Shrc reg,eaddr	C,V,Z,a
Shra reg1,reg2	C,V,Z,a
Shra reg	C,V,Z,a
Shra reg,eaddr	C,V,Z,a
Cpl1 reg1,reg2	~*,Z,a
Cpl1 reg	~*,Z,a
Cpl1 reg,eaddr	~*,Z,a
Cpl2 reg1,reg2	C,V,Z,a
Cpl2 reg	C,V,Z,a
Cpl2 reg,eaddr	C,V,Z,a
Cpl2c reg1,reg2	C,V,Z,a
Cpl2c reg	C,V,Z,a
Cpl2c reg,eaddr	C,V,Z,a

Inc reg1,reg2	C,V,Z,a
Inc reg	C,V,Z,a
Inc reg,eaddr	C,V,Z,a
Incc reg1,reg2	C,V,Z,a
Incc reg	C,V,Z,a
Incc reg,eaddr	C,V,Z,a
Dec reg1,reg2	C,V,Z,a
Dec reg	C,V,Z,a
Dec reg,eaddr	C,V,Z,a
Decc reg1,reg2	C,V,Z,a
Decc reg	C,V,Z,a
Decc reg,eaddr	C,V,Z,a
And reg,#data(8bits)	~*,Z,a
And reg1,reg2,reg3	~*,Z,a
And reg1,reg2	~*,Z,a
And reg,eaddr	~*,Z,a
Or reg,#data(8bits)	~*,Z,a
Or reg1,reg2,reg3	~*,Z,a
Or reg1,reg2	~*,Z,a
Or reg,eaddr	~*,Z,a
Xor reg,#data(8bits)	~*,Z,a
Xor reg1,reg2,reg3	~*,Z,a
Xor reg1,reg2	~*,Z,a
Xor reg,eaddr	~*,Z,a
Add reg,#data(8bits)	C,V,Z,a
Add reg1,reg2,reg3	C,V,Z,a
Add reg1,reg2	C,V,Z,a
Add reg,eaddr	C,V,Z,a
Addc reg,#data(8bits)	C,V,Z,a
Addc reg1,reg2,reg3	C,V,Z,a
Addc reg1,reg2	C,V,Z,a
Addc reg,eaddr	C,V,Z,a
Subd reg,#data(8bits)	C,V,Z,a
Subd reg1,reg2,reg3	C,V,Z,a
Subd reg1,reg2	C,V,Z,a
Subd reg,eaddr	C,V,Z,a
Subdc reg,#data(8bits)	C,V,Z,a
Subdc reg1,reg2,reg3	C,V,Z,a
Subdc reg1,reg2	C,V,Z,a
Subdc reg,eaddr	C,V,Z,a
Subs reg,#data(8bits)	C,V,Z,a
Subs reg1,reg2,reg3	C,V,Z,a
Subs reg1,reg2	C,V,Z,a
Subs reg,eaddr	C,V,Z,a
Subsc reg,#data(8bits)	C,V,Z,a
Subsc reg1,reg2,reg3	C,V,Z,a
Subsc reg1,reg2	C,V,Z,a
Subsc reg,eaddr	C,V,Z,a
Mul reg,#data(8bits)	u,u,u,a
Mul reg1,reg2,reg3	u,u,u,a
Mul reg1,reg2	u,u,u,a
Mul reg,eaddr	u,u,u,a
Mula reg,#data(8bits)	u,u,u,a
Mula reg1,reg2,reg3	u,u,u,a
Mula reg1,reg2	u,u,u,a
Mula reg,eaddr	u,u,u,a
Mshl reg,#shift(3bits)	u,u,u,a
Mshr reg,#shift(3bits)	u,u,u,a *
Mshra reg,#shift(3bits)	u,u,u,a
Cmp reg,#data(8bits)	C,V,Z,a

Cmp reg1,reg2	C,V,Z,a
Cmp reg,eaddr	C,V,Z,a
Cmpa reg,#data(8bits)	C,V,Z,a
Cmpa reg1,reg2	C,V,Z,a
Cmpa reg,eaddr	C,V,Z,a
Tstb reg,#bit(3bit)	Z,a
Setb reg,#bit(3bit)	Z,a
Clr b reg,#bit(3bit)	Z,a
Invb reg,#bit(3bit)	Z,a
Sflag	~*,~*,a
Rflag reg	C,V,Z,a
Rflag eaddr	C,V,Z,a
Freq divn	~*~*
Halt	~*~*
Nop	~*~*
Pmd #s	~*~*

U = undefined *MSHR reg,#1 doesn't shift by 1

Addressing mode

eaddr	Parameters	Data Memory (DM) access	Index update
	addr:8	DM(addr)	
	(ix)	DM(ix)	
	(ix, offset:8)	DM(ix+offset)	
	(ix, r3)	DM(ix+r3)	
	(ix)+	DM(ix)	ix ← ix+1
	(ix, offset:7)+	DM(ix)	ix ← ix+offset
	-(ix)	DM(ix-1)	ix ← ix-1
	-(ix, offset:7)	DM(ix-offset)	ix ← ix-offset

Jump condition

cc	11 Conditions	Test
	CS	C = 1
	CC	C = 0
	ZS	Z = 1
	ZC	Z = 0
	VS	V = 1
	VC	V = 0
	EV	(EV0 OR EV1) = 1
	After CMP d, s	
	EQ	d = s
	NE	d ≠ s
	GT	d > s
	GE	d ≥ s
	LT	d < s
	LE	d ≤ s



Various

ip	Program Memory Index
ix	4 Data Memory (DM) Indexes i0, i1, i2, i3
divn	nodiv, div2, div4, div8, div16

Interrupt

Interrupt	ALL address	Priority
IN0	3	Highest
IN1	1	Medium
IN2	2	Lowest

Registers

reg	16 Registers	Function
reg1	r0	
reg2	r1	
reg3	r2	
	r3	DM offset
	i0l	i0[7:0]
	i0h	i0[15:8]
	i1l	i1[7:0]
	i1h	i1[15:8]
	i2l	i2[7:0]
	i2h	i2[15:8]
	i3l	i3[7:0]
	i3h	i3[15:8]
	ipl	ip[7:0]
	iph	ip[15:8]
	stat	status
	a	accu

I	I	G	I	I	I	E	E
E	E	I	N	N	N	V	V
2	1	E	2	1	0	1	0

7 0
Stat register

Directives

.cpu <i>n</i>	Identifies the CoolRISC processor family for which the content of the file was written. “n” value is 816 for the CR816 processor and 88 for the CR88.
.altregsyn <i>n</i>	Defines the use of alternative register naming syntax. If “n” is set to 1, a % character must precede register names. If “n” is set to 0, no preceding % character is required.
.abort	This directive stops the assembly immediately.
.align <i>abs-expr, abs-expr</i>	Pad the location counter to a particular storage boundary. The first expression is the alignment required. The second expression gives the value to be stored in the padding bytes.
.app-file <i>string</i>	tell GNU-as that we are about to start a new logical file. <i>string</i> is the new file name.
.ascii “<i>string</i>”	It assembles each string into consecutive addresses.
.asciz “<i>string</i>”	Just like .ascii but each string is followed by a zero.
.byte <i>expressions</i>	Each expression is assembled into the next byte. To declare a variable: .byte 12
.comm <i>symbol, length</i>	Declares a named common area in the BSS section.
.data <i>subsection</i>	Tells GNU-as to assemble the following statements onto the end of the data subsection numbered <i>subsection</i> .
.double <i>flonums</i>	It assembles floating point numbers.
.eject	Force a page break at this point.
.else	Is part of the GNU-as support for conditional assembly.
.endif	Is part of the GNU-as support for conditional assembly.
.equ <i>symbol, expression</i>	This directive sets the value of <i>symbol</i> to <i>expression</i> .

.file <i>string</i>	tell GNU-as that we are about to start a new logical file. <i>string</i> is the new file name.
.fill <i>repeat, size, value</i>	This emits <i>repeat</i> copies of <i>size</i> bytes with <i>value</i> .
.float <i>flonums</i>	It has the same effect as .single
.global <i>symbol</i>	Makes the symbol visible to LD
.hword <i>expression</i>	This directive is a synonym for both .short and .word
.if <i>absolute, expression</i>	
.ifdef <i>symbol</i>	
.ifndef <i>symbol</i>	
.include “<i>file</i>”	To include supporting files.
.int <i>expression</i>	
.lcomm <i>symbol, length</i>	Reserve <i>length</i> bytes for a local common denoted by <i>symbol</i> .
.macro <i>macname macargs</i>	Begin the definition of a macro called <i>macname</i> .
.endm	Mark the end of a macro
.exitm	Exit early from the current macro
.org <i>new-lc</i>	Advance the location counter of the current section.
.rep <i>count</i>	
.set <i>symbol, expression</i>	Set the value of <i>symbol</i> to <i>expression</i>
.short <i>expressions</i>	Is the same as “.word”
.skip <i>size, fill</i>	This directive emits <i>size</i> bytes, each of value <i>fill</i> .
.space <i>size, fill</i>	The same as “.skip”.
.string “<i>str</i>”	Copy the character in <i>str</i> to the object file.
.text <i>subsection</i>	Tells GNU-as to assemble the following statements onto the end of the text subsection.
.title “<i>heading</i>”	Use <i>heading</i> as the title when generating assembly listings.
.word <i>expression</i>	Declare a variable with two bytes.

XEMICS SA
Maladière 71
CH-2007 Neuchâtel,
Switzerland
Telephone : +41 32 720 5170
Telefax : +41 32 720 5770
E-mail : info@xemics.ch
Web : www.xemics.ch

