

CompactRISC16 (CR16) Instruction Set Architecture (ISA)

Computer Design Laboratory ECE 3710 Group 2

Fall 2021

The University of Utah

Table 1: Assembly Instructions and Machine Encodings

Mnemonic	Operands	Function	Opcode	Rdest	ImmHi/ Opcode Ext	ImmLo/ Rsrc	Notes
			[15:12]	[11:8]	[7:4]	[3:0]	
ADD	Rdest, Rsrc	$Rdest = Rdest + Rsrc$	0000	Rdest	0000	Rsrc	
ADDI	Rdest, Imm	$Rdest = Rdest + Imm$	0001	Rdest	ImmHi	ImmLo	Sign extended Imm
ADDC	Rdest, Rsrc	$Rdest = Rdest + Rsrc + 1$	0000	Rdest	0001	Rsrc	
ADDCI	Rdest, Imm	$Rdest = Rdest + Imm + 1$	0010	Rdest	ImmHi	ImmLo	Sign extended Imm
MUL	Rdest, Rsrc	$Rdest = Rdest * Rsrc$	0000	Rdest	0010	Rsrc	
MULI	Rdest, Imm	$Rdest = Rdest * Imm$	0011	Rdest	ImmHi	ImmLo	Sign extended Imm
SUB	Rdest, Rsrc	$Rdest = Rdest - Rsrc$	0000	Rdest	0011	Rsrc	
SUBI	Rdest, Imm	$Rdest = Rdest - Imm$	0100	Rdest	ImmHi	ImmLo	Sign extended Imm
CMP	Rdest, Rsrc	$Rdest - Rsrc$	0000	Rdest	0100	Rsrc	
CMPI	Rdest, Imm	$Rdest - Imm$	0101	Rdest	ImmHi	ImmLo	Sign extended Imm
NOT	Rdest, Rsrc	$Rdest = !Rsrc$	0000	Rdest	0101	Rsrc	
NOTI	Rdest, Imm	$Rdest = !Imm$	0110	Rdest	ImmHi	ImmLo	Zero extended Imm
AND	Rdest, Rsrc	$Rdest = Rdest \& Rsrc$	0000	Rdest	0110	Rsrc	
ANDI	Rdest, Imm	$Rdest = Rdest \& Imm$	0111	Rdest	ImmHi	ImmLo	Zero extended Imm
OR	Rdest, Rsrc	$Rdest = Rdest Rsrc$	0000	Rdest	0111	Rsrc	NOP instruction is OR R0, R0
ORI	Rdest, Imm	$Rdest = Rdest Imm$	1000	Rdest	ImmHi	ImmLo	Zero extended Imm
XOR	Rdest, Rsrc	$Rdest = Rdest \wedge Rsrc$	0000	Rdest	1000	Rsrc	
XORI	Rdest, Imm	$Rdest = Rdest \wedge Imm$	1001	Rdest	ImmHi	ImmLo	Zero extended Imm
LSH	Rdest, Ramount	$Rdest = Rdest \ll Ramount$	0000	Rdest	1001	Ramount	$0 \leq Ramount \leq 15$ since registers are only 16-bits
LSHI	Rdest, ImmLo	$Rdest = Rdest \ll Imm$	0000	Rdest	1010	ImmLo	$0 \leq ImmLo \leq 15$
RSH	Rdest, Ramount	$Rdest = Rdest \gg Ramount$	0000	Rdest	1011	Ramount	$0 \leq Ramount \leq 15$
RSHI	Rdest, ImmLo	$Rdest = Rdest \gg Imm$	0000	Rdest	1100	ImmLo	$0 \leq ImmLo \leq 15$
ALSH	Rdest, Ramount	$Rdest = Rdest \lll Ramount$	0000	Rdest	1101	Ramount	$0 \leq Ramount \leq 15$
ALSHI	Rdest, ImmLo	$Rdest = Rdest \lll Imm$	0000	Rdest	1110	ImmLo	$0 \leq ImmLo \leq 15$
ARSH	Rdest, Ramount	$Rdest = Rdest \ggg Ramount$	0000	Rdest	1111	Ramount	$0 \leq Ramount \leq 15$
ARSHI	Rdest, Imm	$Rdest = Rdest \ggg Imm$	1111	Rdest	0000	ImmLo	$0 \leq ImmLo \leq 15$
MOV	Rdest, Rsrc	$Rdest = Rsrc$	1111	Rdest	0001	Rsrc	Copies Rsrc into Rdest
MOVIL	Rdest, Lower Imm	$Rdest[7:0] = Imm$	1010	Rdest	ImmHi	ImmLo	Zero extended Imm, moves immediate value into lower bits of Rdest
MOVIU	Rdest, Upper Imm	$Rdest[15:8] = Imm$	1011	Rdest	ImmHi	ImmLo	Zero padded Imm, moves immediate value into upper bits of Rdest
J[condition]	Rtarget	if [condition]: PC = Rtarget	1111	condition	0010	Rtarget	[condition] bit patterns are in Table 2.
B[condition]	Displacement Imm	if [condition]: PC += Imm	1100	condition	ImmHi	ImmLo	[condition] bit patterns are in Table 2. Immediate is sign extended 2's complement for program counter/address displacement.
CALL	Rtarget	Pushes PC onto stack, PC = Rtarget	1111	xxxx	0011	Rtarget	Used for nested subroutines

CALLD	Displacement Imm	Pushes PC onto stack, PC += Imm	1101	ImmHi	ImmMid	ImmLo	Used for nested subroutines. Immediate is sign extended 2's complement for program counter/address displacement.
RET		Pops top of stack into PC, PC++	1111	xxxx	0100	xxxx	Used to return from nested subroutine
LPC	Rdest	Rdest = PC + 1	1111	Rdest	0101	xxxx	Sets Rdest to the current instruction address/PC + 1
LSF	Rdest	Rdest = status flags	1111	Rdest	0110	xxxx	Sets Rdest to the current status flags (zero extended)
SSF	Rsrc	Status flags = Rsrc[4:0]	1111	xxxx	0111	Rsrc	Sets the current status flags to Rsrc[4:0]
PUSH	Rsrc	rsp--, Main memory value at rsp = Rsrc	1111	xxxx	1000	Rsrc	Pushes Rsrc onto top of stack
POP	Rdest	Rdest = Main memory value at rsp, rsp++	1111	Rdest	1001	xxxx	Pops top of stack into Rdest
LOAD	Rdest, Raddr	Rdest = Main memory value at Raddr	1111	Rdest	1010	Raddr	Used to load data at Raddr into Rdest from main memory
STORE	Raddr, Rsrc	Main memory value at Raddr = Rsrc	1111	Raddr	1011	Rsrc	Used to store data at Raddr from Rsrc to main memory
LOADX	Rdest, Raddr	Rdest = External memory at Raddr	1111	Rdest	1100	Raddr	Used to load data at Raddr into Rdest from external/peripheral memory/registers
STOREX	Raddr, Rsrc	External memory value at Raddr = Rsrc	1111	Raddr	1101	Rsrc	Used to store data at Raddr from Rsrc to external/peripheral memory/registers
NOP		No Operation					Pseudo instruction for: OR R0, R0

Table 2: Bit Patterns of Conditions for B[condition] and J[condition]

Mnemonic	Bit Pattern	Description	Function	Status Flags
EQ	0000	Equal	<code>Rsrc == Rdest</code>	Z=1
NE	0001	Not Equal	<code>Rsrc != Rdest</code>	Z=0
CS	0010	Carry Set	<code>C == 1</code>	C=1
CC	0011	Carry Clear	<code>C == 0</code>	C=0
FS	0100	Flag Set	<code>F == 1</code>	F=1
FC	0101	Flag Clear	<code>F == 0</code>	F=0
LT	0110	Less Than	<code>signed: Rdest < Rsrc</code>	N=0 and Z=0
LE	0111	Less than or Equal	<code>signed: Rdest <= Rsrc</code>	N=0
LO	1000	Lower than	<code>unsigned: Rdest < Rsrc</code>	L=0 and Z=0
LS	1001	Lower than or Same as	<code>unsigned: Rdest <= Rsrc</code>	L=0
GT	1010	Greater Than	<code>signed: Rdest > Rsrc</code>	N=1
GE	1011	Greater than or Equal	<code>signed: Rdest >= Rsrc</code>	N=1 or Z=1
HI	1100	Higher than	<code>unsigned: Rdest > Rsrc</code>	L=1
HS	1101	Higher than or Same as	<code>unsigned: Rdest >= Rsrc</code>	L=1 or Z=1
UC	1110	Unconditional		N/A
	1111	Never Jump		N/A

Table 3: Register Naming and Conventions

Register Index	Register Name	Meaning
4'd15	rsp	Stack pointer with an address starting at 0xFFFF (2 ¹⁶) and grows downward towards dynamically allocated memory
4'd14	r14	4th subroutine argument
4'd13	r13	3rd subroutine argument
4'd12	r12	2nd subroutine argument
4'd11	r11	1st subroutine argument
4'd10	r10	Return value of subroutine
4'd9	r9	Caller-owned
4'd8	r8	Caller-owned
4'd7	r7	Caller-owned
4'd6	r6	Caller-owned
4'd5	r5	Callee-owned
4'd4	r4	Callee-owned
4'd3	r3	Callee-owned
4'd2	r2	Callee-owned
4'd1	r1	Callee-owned
4'd0	r0	Callee-owned