Custom 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 [15:12]	Rdest [11:8]	ImmHi/ Opcode Ext [7:4]	ImmLo/ Rsrc [3:0]	Notes	Clock Cycles
ADD	Rdest, Rsrc	`Rdest = Rdest + Rsrc`	0000	Rdest	0000	Rsrc		3
ADDI	Rdest, Imm	`Rdest = Rdest + Imm`	0001	Rdest	ImmHi	ImmLo	Sign extended Imm	3
ADDC	Rdest, Rsrc	`Rdest = Rdest + Rsrc + 1`	0000	Rdest	0001	Rsrc		3
ADDCI	Rdest, Imm	`Rdest = Rdest + Imm + 1`	0010	Rdest	ImmHi	ImmLo	Sign extended Imm	3
MUL	Rdest, Rsrc	`Rdest = Rdest * Rsrc`	0000	Rdest	0010	Rsrc		3
MULI	Rdest, Imm	`Rdest = Rdest * Imm`	0011	Rdest	ImmHi	ImmLo	Sign extended Imm	3
SUB	Rdest, Rsrc	`Rdest = Rdest - Rsrc`	0000	Rdest	0011	Rsrc		3
SUBI	Rdest, Imm	`Rdest = Rdest - Imm`	0100	Rdest	ImmHi	ImmLo	Sign extended Imm	3
CMP	Rdest, Rsrc	`Rdest - Rsrc`	0000	Rdest	0100	Rsrc		3
CMPI	Rdest, Imm	`Rdest - Imm`	0101	Rdest	ImmHi	ImmLo	Sign extended Imm	3
NOT	Rdest, Rsrc	`Rdest = !Rsrc`	0000	Rdest	0101	Rsrc		3
NOTI	Rdest, Imm	`Rdest = !Imm`	0110	Rdest	ImmHi	ImmLo	Zero extended Imm	3
AND	Rdest, Rsrc	`Rdest = Rdest & Rsrc`	0000	Rdest	0110	Rsrc		3
ANDI	Rdest, Imm	`Rdest = Rdest & Imm`	0111	Rdest	ImmHi	ImmLo	Zero extended Imm	3
OR	Rdest, Rsrc	`Rdest = Rdest | Rsrc`	0000	Rdest	0111	Rsrc		3
ORI	Rdest, Imm	`Rdest = Rdest | Imm`	1000	Rdest	ImmHi	ImmLo	Zero extended Imm	3
XOR	Rdest, Rsrc	`Rdest = Rdest ^ Rsrc`	0000	Rdest	1000	Rsrc		3
XORI	Rdest, Imm	`Rdest = Rdest ^ Imm`	1001	Rdest	ImmHi	ImmLo	Zero extended Imm	3
LSH	Rdest, Ramount	`Rdest = Rdest << Ramount`	0000	Rdest	1001	Ramount	$0 \leq$ Ramount ≤ 15 since registers are only 16-bits	3
LSHI	Rdest, ImmLo	`Rdest = Rdest << Imm`	0000	Rdest	1010	ImmLo	$0 \leq$ ImmLo ≤ 15	3
RSH	Rdest, Ramount	`Rdest = Rdest >> Ramount`	0000	Rdest	1011	Ramount	$0 \leq$ Ramount ≤ 15	3
RSHI	Rdest, ImmLo	`Rdest = Rdest >> Imm`	0000	Rdest	1100	ImmLo	$0 \leq$ ImmLo ≤ 15	3
ALSH	Rdest, Ramount	`Rdest = Rdest <<< Ramount`	0000	Rdest	1101	Ramount	$0 \leq$ Ramount ≤ 15	3
ALSHI	Rdest, ImmLo	`Rdest = Rdest <<< Imm`	0000	Rdest	1110	ImmLo	$0 \leq$ ImmLo ≤ 15	3
ARSH	Rdest, Ramount	`Rdest = Rdest >>> Ramount`	0000	Rdest	1111	Ramount	$0 \leq$ Ramount ≤ 15	3
ARSHI	Rdest, Imm	`Rdest = Rdest >>> Imm`	1111	Rdest	0000	ImmLo	$0 \leq$ ImmLo ≤ 15	3
MOV	Rdest, Rsrc	`Rdest = Rsrc`	1111	Rdest	0001	Rsrc	Copies Rsrc into Rdest	3
MOVIL	Rdest, Lower Imm	`Rdest[7:0] = Imm`	1010	Rdest	ImmHi	ImmLo	Zero extended Imm, moves immediate value into lower bits of Rdest	3
MOVIU	Rdest, Upper Imm	`Rdest[15:8] = Imm`	1011	Rdest	ImmHi	ImmLo	Zero padded Imm, moves immediate value into upper bits of Rdest	3
J[condition]	Rtarget	`if [condition]:` `PC = Rtarget`	1111	condition	0010	Rtarget	[condition] bit patterns are in Table 2.	T: 5 F: 3
B[condition]	Displacement Imm	`if [condition]:` `PC += Imm + 1`	1100	condition	ImmHi	ImmLo	[condition] bit patterns are in Table 2. Immediate is sign extended 2's complement for program counter/address displacement.	T: 4 F: 3
CALL	Rtarget	`Pushes PC + 1 onto stack,` `PC = Rtarget`	1111	xxxx	0011	Rtarget	Used for nested subroutines	5

CALLD	Displacement Imm	`Pushes PC + 1 onto stack, PC += Imm + 1`	1101	ImmHi	ImmMid	ImmLo	Used for nested subroutines. Immediate is sign extended 2's complement for program counter/address displacement.	4
RET		`Pops top of stack into PC`	1111	xxxx	0100	xxxx	Used to return from nested subroutine	6
LPC	Rdest	`Rdest = PC`	1111	Rdest	0101	xxxx	Sets Rdest to PC	3
LSF	Rdest	`Rdest = status flags`	1111	Rdest	0110	xxxx	Sets Rdest to the current status flags (zero extended)	3
SSF	Rsrc	`Status flags = Rsrc[4:0]`	1111	xxxx	0111	Rsrc	Sets the current status flags to `Rsrc[4:0]`	3
PUSH	Rsrc	`Main memory value at rsp = Rsrc, rsp--`	1111	xxxx	1000	Rsrc	Pushes Rsrc onto top of stack	4
POP	Rdest	`rsp++, Rdest = Main memory value at rsp`	1111	Rdest	1001	xxxx	Pops top of stack into Rdest	4
LOAD	Rdest, Raddr	`Rdest = Main memory value at Raddr`	1111	Rdest	1010	Raddr	Used to load data at Raddr into Rdest from main memory	4
STORE	Raddr, Rsrc	`Main memory value at Raddr = Rsrc`	1111	Raddr	1011	Rsrc	Used to store data at Raddr from Rsrc to main memory	4
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	4
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	4
NOP		`No Operation`					Pseudo instruction for: `OR R0, R0`	3

Note that during the execution cycle of an instruction, PC (the "Program Counter") always points to the next instruction (e.g. `PC + 1`). As a result of this, `B[condition]` and `CALLD` will displace the current PC by `Imm + 1`. This can be thought of as executing the "next instruction" after PC displacement.

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

Mnemonic	Bit Pattern	Description	Function	Status Flags
EQ	0000	Equal	`Rsrc == Rdest`	`Z=1`
NE	0001	Not Equal	`Rsrc != Rdest`	`Z=0`
CS	0010	Carry Set	`C == 1`	`C=1`
CC	0011	Carry Clear	`C == 0`	`C=0`
FS	0100	Flag Set	`F == 1`	`F=1`
FC	0101	Flag Clear	`F == 0`	`F=0`
LT	0110	Less Than	`signed: Rdest < Rsrc`	`N=0 and Z=0`
LE	0111	Less than or Equal	`signed: Rdest <= Rsrc`	`N=0`
LO	1000	Lower than	`unsigned: Rdest < Rsrc`	`L=0 and Z=0`
LS	1001	Lower than or Same as	`unsigned: Rdest <= Rsrc`	`L=0`
GT	1010	Greater Than	`signed: Rdest > Rsrc`	`N=1`
GE	1011	Greater than or Equal	`signed: Rdest >= Rsrc`	`N=1 or Z=1`
HI	1100	Higher than	`unsigned: Rdest > Rsrc`	`L=1`
HS	1101	Higher than or Same as	`unsigned: Rdest >= Rsrc`	`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