CompactRISC16 (CR16) Instruction Set Architecture (ISA)

Computer Design Laboratory ECE 3710
Fall 2021
The University of Utah

Table 1: Assembly Instructions and Machine Encodings

					ImmHi/	ImmLo/	
			Opcode	Rdest	Opcode Ext	Rsrc	
Mnemonic	**Operands**	**Function**	**[15:12]**	**[11:8]**	**[7:4]**	**[3:0]**	**Notes**
ADD	Rsrc, Rdest	`Rdest = Rdest + Rsrc`	0000	Rdest	0000	Rsrc	
ADDI	Imm, Rdest	`Rdest = Rdest + Imm`	0001	Rdest	ImmHi	ImmLo	Sign extended Imm
ADD	Rsrc, Rdest	`Rdest = Rdest + Rsrc`	0000	Rdest	0000	Rsrc	
ADDI	Imm, Rdest	`Rdest = Rdest + Imm`	0001	Rdest	ImmHi	ImmLo	Sign extended Imm
ADDU	Rsrc, Rdest	`Rdest = Rdest + Rsrc`	0000	Rdest	0001	Rsrc	
ADDUI	Imm, Rdest	`Rdest = Rdest + Imm`	0010	Rdest	ImmHi	ImmLo	Sign extended Imm
ADDC	Rsrc, Rdest	`Rdest = Rdest + Rsrc + 1`	0000	Rdest	0010	Rsrc	
ADDCI	Imm, Rdest	`Rdest = Rdest + Imm + 1`	0011	Rdest	ImmHi	ImmLo	Sign extended Imm
ADDCUI	Imm, Rdest	`Rdest = Rdest + Imm + 1`	0100	Rdest	ImmHi	ImmLo	Sign extended Imm
MUL	Rsrc, Rdest	`Rdest = Rdest * Rsrc`	0000	Rdest	0011	Rsrc	
MULI	Imm, Rdest	`Rdest = Rdest * Imm`	0101	Rdest	ImmHi	ImmLo	Sign extended Imm
SUB	Rsrc, Rdest	`Rdest = Rdest - Rsrc`	0000	Rdest	0100	Rsrc	
SUBI	Imm, Rdest	`Rdest = Rdest - Imm`	0110	Rdest	ImmHi	ImmLo	Sign extended Imm
CMP	Rsrc, Rdest	`Rdest - Rsrc`	0000	Rdest	0101	Rsrc	
CMPI	Imm, Rdest	`Rdest - Imm`	0111	Rdest	ImmHi	ImmLo	Sign extended Imm
AND	Rsrc, Rdest	`Rdest = Rdest & Rsrc`	0000	Rdest	0110	Rsrc	
ANDI	Imm, Rdest	`Rdest = Rdest & Imm`	1000	Rdest	ImmHi	ImmLo	Zero extended Imm
OR	Rsrc, Rdest	`Rdest = Rdest | Rsrc`	0000	Rdest	0111	Rsrc	NOP instruction is OR R0, R0
ORI	Imm, Rdest	`Rdest = Rdest | Imm`	1001	Rdest	ImmHi	ImmLo	Zero extended Imm
XOR	Rsrc, Rdest	`Rdest = Rdest ^ Rsrc`	0000	Rdest	1000	Rsrc	
XORI	Imm, Rdest	`Rdest = Rdest ^ Imm`	1010	Rdest	ImmHi	ImmLo	Zero extended Imm
LSH	Ramount, Rdest	`Rdest = Rdest << Ramount`	0000	Rdest	1001	Ramount	$0 \leq$ Ramount ≤ 15 since registers are only 16-bits
LSHI	Imm, Rdest	`Rdest = Rdest << Imm`	0000	Rdest	1010	ImmLo	$0 \leq$ ImmLo ≤ 15
RSH	Ramount, Rdest	`Rdest = Rdest >> Ramount`	0000	Rdest	1011	Ramount	$0 \leq$ Ramount ≤ 15
RSHI	Imm, Rdest	`Rdest = Rdest >> Imm`	0000	Rdest	1100	ImmLo	$0 \leq$ ImmLo ≤ 15
ALSH	Ramount, Rdest	`Rdest = Rdest <<< Ramount`	0000	Rdest	1101	Ramount	$0 \leq$ Ramount ≤ 15
ALSHI	Imm, Rdest	`Rdest = Rdest <<< Imm`	0000	Rdest	1110	ImmLo	$0 \leq$ ImmLo ≤ 15
ARSH	Ramount, Rdest	`Rdest = Rdest >>> Ramount`	0000	Rdest	1111	Ramount	$0 \leq$ Ramount ≤ 15
ARSHI	Imm, Rdest	`Rdest = Rdest >>> Imm`	1111	Rdest	0000	ImmLo	$0 \leq$ ImmLo ≤ 15
MOV	Rsrc, Rdest	`Rdest = Rsrc`	1111	Rdest	0001	Rsrc	Copies Rsrc into Rdest
MOVIL	Lower Imm, Rdest	`Rdest[7:0] = Imm`	1011	Rdest	ImmHi	ImmLo	Zero extended Imm, moves immediate value into lower bits of Rdest
MOVIU	Upper Imm, Rdest	`Rdest[15:8] = Imm`	1100	Rdest	ImmHi	ImmLo	Zero padded Imm, moves immediate value into upper bits of Rdest
B[condition]	Displacement Imm	`Relative jump by Imm if [condition]`	1101	condition	ImmHi	ImmLo	Immediate is used as a 2's complement program counter/address displacement. [condition] bit patterns are in Table 2.

J[condition]	Rtarget	`Absolute jump to Rtarget if [condition]`	1111	condition	0010	Rtarget	[condition] bit patterns are in Table 2.
JAL	Rlink, Rtarget	`Jump to Rtarget, Rlink = PC + 1`	1111	Rlink	0011	Rtarget	Stores the address of the next instruction in Rlink and jumps to Rtarget, used for subroutines
LPC	Rdest	`Rdest = PC`	1111	Rdest	0100	xxxx	Sets Rdest to the current instruction address/PC
LSF	Rdest	`Rdest[4:0] = status flags`	1111	Rdest	0101	xxxx	Sets 5 least significant bits of Rdest to the current status flags
SSF	Rsrc	`Status flags = Rsrc[4:0]`	1111	xxxx	0110	Rsrc	Sets current status flags to 5 least significant bits of Rsrc
LOAD	Raddr, Rdest	`Rdest = Main memory value at Raddr`	1111	Raddr	0111	Rdest	Used to load data at Raddr into Rdest from main memory
STORE	Rsrc, Raddr	`Main memory value at Raddr = Rsrc`	1111	Raddr	1000	Rsrc	Used to store data at Raddr from Rsrc to main memory
LOADX	Raddr, Rdest	`Rdest = External memory at Raddr`	1111	Raddr	1001	Rdest	Used to load data at Raddr into Rdest from external/peripheral memory/registers
STOREX	Rsrc, Raddr	`External memory value at Raddr = Rsrc`	1111	Raddr	1010	Rsrc	Used to store data at Raddr from Rsrc to external/peripheral memory/registers
NOP		`No Operation`					Alias for: OR R0, R0

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

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

Table 3: Register Conventions

Register Index	Convention
`4'd15`	Stack Pointer with an address starting at `0xFFFF` (2^{16}) and grows downward towards dynamically allocated memory
`4'd14`	Rlink
`4'd13`	Return value of subroutine
`4'd12`	1st subroutine argument
`4'd11`	2nd subroutine argument
`4'd10`	3rd subroutine argument
`4'd9`	Caller-owned
`4'd8`	Caller-owned
`4'd7`	Caller-owned
`4'd6`	Caller-owned
`4'd5`	Callee-owned
`4'd4`	Callee-owned
`4'd3`	Callee-owned
`4'd2`	Callee-owned
`4'd1`	Callee-owned
`4'd0`	Callee-owned