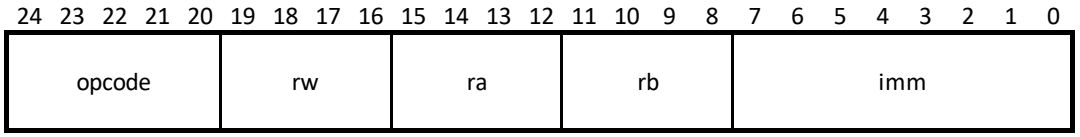
Instruction Format



Instruction Fields

opcode = ins<24:20>	Encoded operation field for this instruction
rw = ins<19:16>	encoded register file write address
ra = ins<15:12>	encoded register file read address for port A
rb = ins<11:8>	encoded register file read address for port B
imm = ins<7:0>	Immediate or literal value for data specified by this instruction

Note: The 01 and 11 rows of the table all have immediate fields that replace the rb operand value

Opcode Map

instn<22:20>

instn<24:23>	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
00	OR	AND		NOT	CMP		ADD	SUB
01	ORI	ANDI		NOTI	CMPI		ADDI	SUBI
10	BRZ	BRO	MF					
11	LD	ST						HALT

Data Path Structures

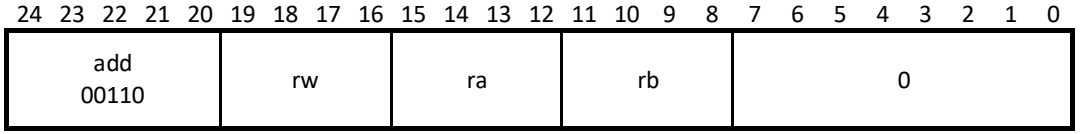
RF	Register File: 18 total entries, each of which is 8 bits wide Registers 0 through 15 are real registers addressable in the instruction word. Register 16 is Lo and register 17 is Hi, accessible via the mflo and mfhi instructions ***RF[0] is not a real register: reads return 0, writes are ignored***
RF[i]	Register file 8-bit value at location i

Control Signals

C1_OpAdd	Result provided by the Add unit (add[i], sub[i])
C1_CiB0	Carry in to b0 of the adder (sub[i])
C1_OpCmp	Result provided by the CMP unit (cmp[i])
C1_OpAND	Result provided by the AND unit (and[i])

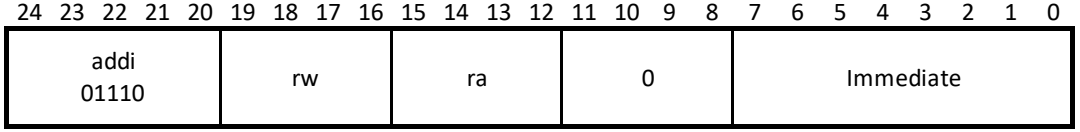
C1_OpOR	Result provided by the OR unit (or[i], not[i])
C1_OpBInv	Invert the B input to the processing units (not[i], sub[i])
C1_LoadPC	Load PC from the Immediate field of the instruction
I0_InstnImm	Place the instruction immediate field on the B output of the register file (ori, andi, noti, cmpi, addi, subi)
C0_BDecLO	Read the LO register onto the B output of the register file
C0_BDecHI	Read the HI register onto the B output of the register file

ADD - RF[rw] <- RF[ra] + RF[rb]
Add register ra to register rb and store the result in register rw
Assembler Format: add rw, ra, rb



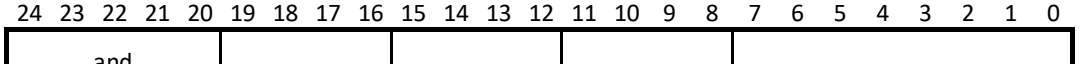
C1_opAdd <- 1

ADDI - RF[rw] <- RF[ra] + Immediate
Add register ra to the immediate and store the result in register rw
Assembler Format: addi rw, ra, immediate



C1_opAdd <- 1
I0_InstnImm <- 1

AND - RF[rw] <- RF[ra] & RF[rb] (Bitwise AND)
Bit-wise AND register ra with register rb and store the result in register rw
Assembler Format: and rw, ra, rb



and 00001	rw	ra	rb	0

C1_OpAND <- 1

ANDI - RF[rw] <- RF[ra] & Immediate (Bitwise AND)
Bit-wise AND register ra with the immediate and store the result in register rw
Assembler Format: andi rw, ra, immediate

24 23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4 3 2 1 0
andi 01001	rw	ra	0	Immediate

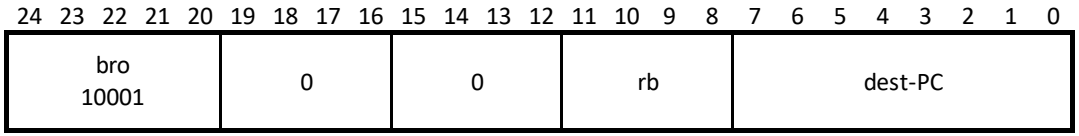
C1_OpAND <- 1
I0_InstnImm <- 1

BRZ - If RF[rb]<0> == 0, PC <- dest-PC
Branch to dest-PC if bit 0 of register rb is 1; execute the next instruction in sequence before taking the branch
Assembler Format: brz rb, dest-PC
NOTE: The BRZ instruction has a delay slot. That is, the next instruction in sequences is executed independent of whether the branch is taken or not

24 23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4 3 2 1 0
brz 10000	0	0	rb	dest-PC

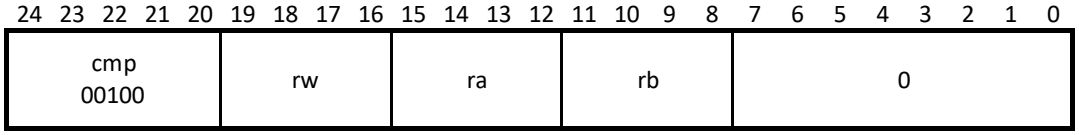
C1_LoadPC <- 1 if RF[rb]<0> == 0, else 0

BRO - If RF[rb]<0> == 1, PC <- dest-PC
Branch to dest-PC if bit 0 of register rb is 1; execute the next instruction in sequence before taking the branch
Assembler Format: bro rb, dest-PC
NOTE: The BRO instruction has a delay slot. That is, the next instruction in sequences is executed independent of whether the branch is taken or not



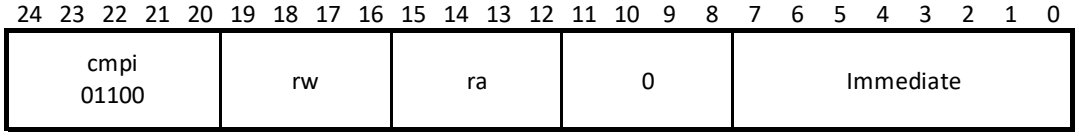
C1_LoadPC <- 1 if RF[rb]<0> == 1, else 0

CMP - RF[rw] <-Compare(RF[ra],RF[rb])
Compare register ra with register rb. If they are the same, store a 1 in register rw; else store a 0 in register rw
Assembler Format: cmp rw, ra, rb



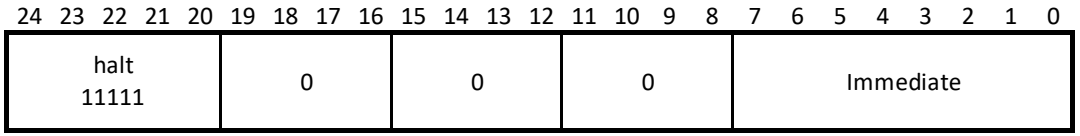
C1_OpCMP <- 1

CMPI - RF[rw] <- Compare(RF[ra], Immediate)
Compare register ra with the immediate. If they are the same, store a 1 in register rw; else store a 0 in register rw
Assembler Format: cmpi rw, ra, immediate

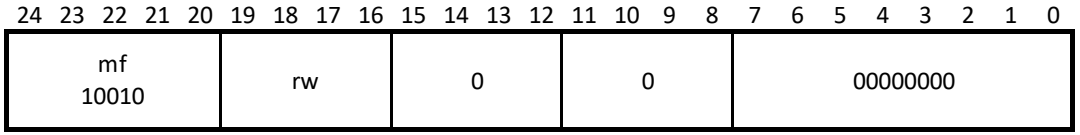


C1_OpCMP <- 1
I0_InstnImm <- 1

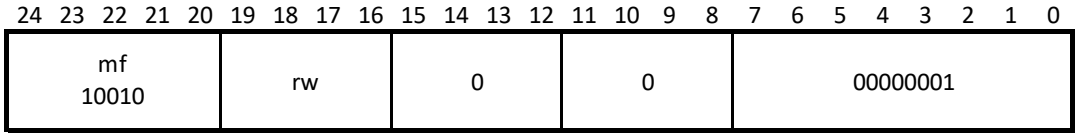
HALT - Halt the CPU
Halt the CPU with the immediate value as the exit code. An exit code of 0 denotes a successful completion. Any other value denotes a failure
Assembler Format: halt immediate



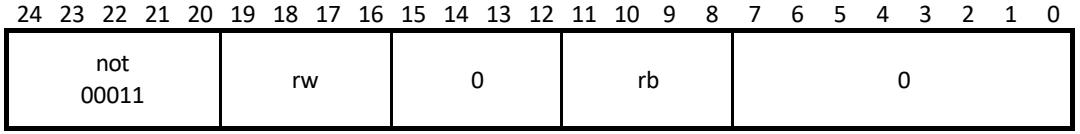
MFLO - RF[rw] <- RF[16]
Store the value of register 16 in register rw
Assembler Format: mflo rw



MFHI - RF[rw] <- RF[17]
Store the value of register 17 in register rw
Assembler Format: mfhi rw

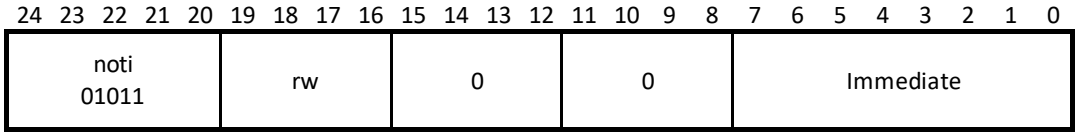


NOT - RF[rw] <- ~RF[rb] (Bitwise Invert)
Bit-wise invert register rb and store the result in register rw
Assembler Format: not rw, rb



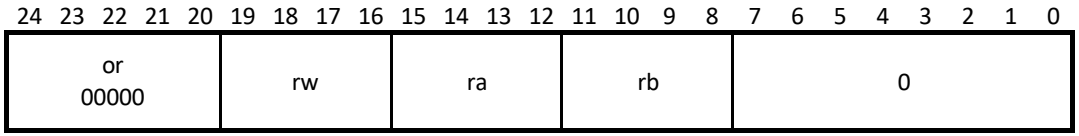
C1_opOr <- 1
C1_OpBInv <- 1

NOTI - RF[rw] <- ~Immediate (Bitwise Invert)
Bit-wise invert immediate and store the result in register rw
Assembler Format: noti rw, immediate



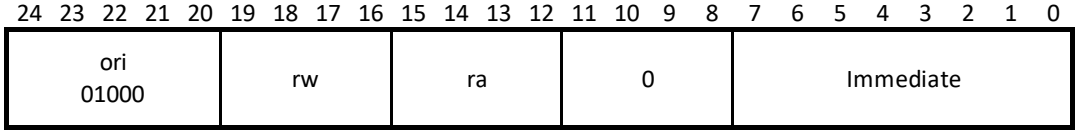
C1_opOr <- 1
C1_OpBInv <- 1
I0_InstnImm <- 1

OR - RF[rw] <- RF[ra] | RF[rb] (Bitwise OR)
Bit-wise OR register ra with register rb and store the result in register rw
Assembler Format: or rw, ra, rb



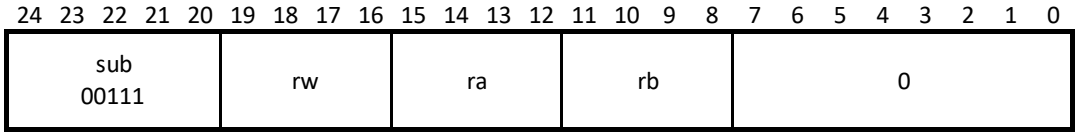
C1_opOr <- 1

ORI - RF[rw] <- RF[ra] | Immediate (Bitwise OR)
Bit-wise OR register ra with the immediate and store the result in register rw
Assembler Format: ori rw, ra, immediate



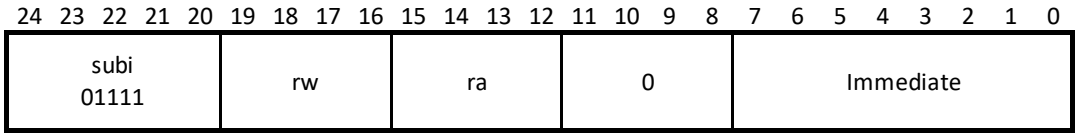
C1_opOr <- 1
I0_InstnImm <- 1

SUB - RF[rw] <- RF[ra] - RF[rb]
Subtract register rb from register ra and store the result in register rw
Assembler Format: sub rw, ra, rb



C1_opAdd <- 1
C1_CiB0 <- 1
C1_OpBInv <- 1

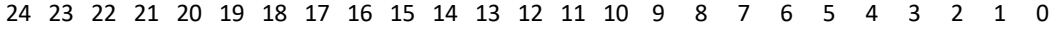
SUBI- RF[rw] <- RF[ra] - Immediate
Subtract immediate from register rb and store the result in register rw
Assembler Format: subi rw, ra, immediate



C1_opAdd <- 1
C1_CiB0 <- 1
C1_OpBInv <- 1
I0_InstnImm <- 1

Pseudo Instructions

LI - RF[rw] <- Immediate
Load immediate value into register rw
Assembler Format: li rw, immediate



ori 01000	rw	0	0	Immediate

C1_opOr <- 1
I0_InstnImm <- 1

NOP - Do nothing
Assembler Format: nop

24 23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4 3 2 1 0
or 00000	0	0	0	0

C1_opOr <- 1

BR - PC <- dest-PC
Branch unconditionally to dest-PC; execute the next instruction in sequence before taking the branch
Assembler Format: br dest-PC
NOTE: The BRO instruction has a delay slot. That is, the next instruction in sequences is executed independent of whether the branch is taken or not

24 23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4 3 2 1 0
brz 10000	0	0	0	dest-PC

C1_LoadPC <- 1 if RF[rb]<0> == 0, else 0