ESE 501 Digital System Specification and Modeling
Department of Electrical and Computer Engineering
Stony Brook University

ISA for MINI RISC Processor

Mnemonic	Operands	Op Code [15:12]	Rdest [11:8]	ImmHi/ Op Code Ext [7:4]	ImmLo/ Rsrc [3:0]	Notes (* is Baseline)
ADD	Rsrc, Rdest	0000	Rdest	0101	Rsrc	*
ADDI	Imm, Rdest	0101	Rdest	ImmHi	ImmLo	*sign extended Imm
ADDU	Rsrc, Rdest	0000	Rdest	0110	Rsrc	
ADDUI	Imm, Rdest	0110	Rdest	ImmHi	ImmLo	Sign extended Imm
ADDC	Rsrc, Rdest	0000	Rdest	0111	Rsrc	
ADDCI	Imm, Rdest	0111	Rdest	ImmHi	ImmLo	Sign extended Imm
MUL	Rsrc, Rdest	0000	Rdest	1110	Rsrc	
MULI	Imm, Rdest	1110	Rdest	ImmHi	ImmLo	Sign extended Imm
SUB	Rsrc, Rdest	0000	Rdest	1001	Rsrc	*
SUBI	Imm, Rdest	1001	Rdest	ImmHi	ImmLo	*sign extended Imm
SUBC	Rsrc, Rdest	0000	Rdest	1010	Rsrc	
SUBCI	Imm, Rdest	1010	Rdest	ImmHi	ImmLo	Sign extended Imm
CMP	Rsrc, Rdest	0000	Rdest	1011	Rsrc	*
CMPI	Imm, Rdest	1011	Rdest	ImmHi	ImmLo	*sign extended Imm
AND	Rsrc, Rdest	0000	Rdest	0001	Rsrc	*
ANDI	Imm, Rdest	0001	Rdest	ImmHi	ImmLo	*sign extended Imm
OR	Rsrc. Rdest	0000	Rdest	0010	Rsrc	*NOP=OR R0, R0
ORI	Imm, Rdest	0010	Rdest	ImmHi	ImmLo	*zero extended Imm
XOR	Rsrc, Rdest	0000	Rdest	0011	Rsrc	*
XORI	Imm, Rdest	0011	Rdest	ImmHi	ImmLo	*zero extended Imm
MOV	Rsrc, Rdest	0000	Rdest	1101	Rsrc	*
MOVI	Imm, Rdest	1101	Rdest	ImmHi	ImmLo	*zero extended Imm
LSH	Ramount, Rdest	1000	Rdest	0100	Ramount	*-15 to 15 (2s cmpl)
LSHI	Imm, Rdest	1000	Rdest	000s	ImmLo	*s= sign (0=left,2s cmpl)
ASH	Ramount, Rdest	1000	Rdest	0110	Ramount	*-15 to 15 (2s cmpl)
ASHI	Imm, Rdest	1000	Rdest	001s	ImmLo	*s= sign (0=left,2s cmpl)
LUI	Imm, Rdest	1111	Rdest	ImmHi	ImmLo	*load and 8bit left shift
LOAD	Rdest, Raddr	0100	Rdest	0000	Raddr	*
STOR	Rsrc, Raddr	0100	Rsrc	0100	Raddr	*
Bcond	Disp	1100	Cond	DispHi	DispLo	*2s cmpl displacement
Jcond	Rtarget	0100	Cond	1100	Rtarget	*
JAL	Rlink, Rtarget	0100	Rlink	1000	Rtarget	*
NOP		0000	0000	0000	0000	*

COND Values for Jcond, Bcond, Scond

Mnemonic	Bit Pattern	Description	PSR Values
EQ	0000	Equal	*Z=1
NE	0001	Not equal	*Z=0
GE	1101	Greater than or equal	*N=1 or Z=1
CS	0010	Carry set	*C=1
CC	0011	Carry clear	*C=0
GT	0110	Greater than	*N=1
LE	0111	Less than or equal	*N=0
FS	1000	Flag set	F=1
FC	1001	Flag clear	F=0
LT	1100	Less than	*N=0 and Z=0
US	1110	Unconditional	N/A
	1111	Never jump	N/A

Notes on the Instruction Set

All ALU instructions (except CMP, CMPI – see below) write the result back to the destination register. Instructions ending with I are immediate and use the eight least significant bits of the instruction as data, the others are direct, (i.e. instruction" op Rsrc/Imm, Rdest" performs

Rdest <-Rdest op Imm (sign extended) or Rdest <- Rdest op Rsrc respectively)

Successive memory addresses can refer to 16 bit words instead of bytes. Of the baseline subset of instructions, the only ones which can change the program status register (PSR) are the arithmetic instructions ADD, ADDI, SUB, SUBI, CMP, CMPI. CMP and CMPI perform the same operations as SUB, SUBI but affect different PSR flags (see below) and do not write back the result. Only flags FCNZ are needed for the baseline implementation.

ADD, ADDI, SUB, SUBI set the C flag if a carry/borrow from the most significant bit position occurs when the operands are treated as unsigned numbers, and set the F flag if an overflow occurs when the operands are treated as two's complement numbers. (Note: the processor does not know which interpretation you are using, so must set both flags appropriately for each operation.) CMP, CMPI perform a subtraction without write back to Rdest and set the Z flag if the result is zero, set the N flag if Rsrc/Imm > Rdest when the operands are treated as two's complement numbers (N can be computed as the exclusive-or of Rsrc/Imm and Rdest.) All other baseline instructions leave the flag unchanged.

Jcond, Bcond are absolute and relative jumps respectively based on the condition codes specified in the condition code (cond) table. JAL (jump and link) stores the address of the next instruction in Rlink, and jumps to Rtarget. Its main use is for subroutine calls. Return with a JUC Rlink (where Rlink is the same register used to store the link).

LSH is a logical left shift by the number of bits specified in Rsrc/Imm treated as a signed two's complement number (which must be in the range -15 to +15). A negative left shift is effectively a right shift.

LOAD and STOR instruction load from, and store to the data memory location whose address is in register Raddr. The NOP instruction is really OR R0, R0 and does not need to be implemented separately. Unconditional jumps (JUMP) and branches (BR) are equivalent to JUC and BUC respectively, so do not need separate implementation either. Compiler may have these alternative instruction ops for convenience, however.

LUI (load upper immediate) loads the 8-bit immediate data into the upper (most significant) bits of the destination register.

MOV copies the source register or immediate into the destination register.

ASH does an arithmetic left shift interpreting both operands as signed two's complement.

5-Stage Pipeline

IF, RD. EXE, MEM, WB