Elec 374 MiniSRC Instruction Set Spec.

Processor State

PC<31..0>:	32-bit Program Counter (PC)
IR<31..0>:	32-bit Instruction Register (IR)
R[0..15]<31..0>:	Sixteen 32-bit registers named R[0] through R[15]
R[15]<31..0>:	Stack Pointer (SP)
R[14]<31..0]:	Return Address Register (RA)
R[10..13]<31..0>:	Four Argument Registers, named A[0] through A[3]
R[8..9]<31..0>:	Two Return Value Registers, named V[0] and V[1]
HI<31..0>:	32-bit HI Register dedicated to keep the high-order word of a Multiplication product, or the Remainder of a Division operation
LO<31..0>:	32-bit LO Register dedicated to keep the low-order word of a Multiplication product, or the Quotient of a Division operation

Memory State

Mem[0..511]<31..0>:	512 words (32 bits per word) of memory
MDR<31..0>:	32-bit memory data register
MAR<31..0>:	32-bit memory address register

I/O State

In.Port<31..0>:	32-bit input port
Out.Port<31..0>:	32-bit output port
Run.Out:	Run/halt indicator
Stop.In:	Stop signal
Reset.In:	Reset signal

Instructions:

The instructions (with their op-code patterns shown in parentheses) perform the following operations:

Notation: x: 0 or 1 - : unused

Load and Store Instructions

1(a): ld, ldi, st

		Assembly language	
Load direct (00000xxxx0000xxxxxxxxxxxxxxxxxxx)	R[Ra] ← M[C (sign-extended)] Direct addressing, Rb = R0	ld	Ra, C
Load indexed (00000xxxxxxxxxxxxxxxxxxxxxxxxxxx)	R[Ra] ← M[R[Rb] + C (sign-extended)] Indexed addressing, Rb ≠ R0 If C = 0 ➔ Register Indirect addressing	ld	Ra, C(Rb)
Load immediate (00001xxxx0000xxxxxxxxxxxxxxxxxxx)	R[Ra] ← C (sign-extended) Immediate addressing, Rb = R0	ldi	Ra, C
(00001xxxxxxxxxxxxxxxxxxxxxxxxxxx)	R[Ra] ← R[Rb] + C (sign-extended) Immediate addressing, Rb ≠ R0 If C = 0 ➔ instruction acts like a simple register transfer If C ≠ 0 and Ra = Rb ➔ Increment/decrement instruction	ldi	Ra, C(Rb)
Store direct (00010xxxx0000xxxxxxxxxxxxxxxxxxx)	M[C (sign-extended)] ← R[Ra] Direct addressing, Rb = R0	st	C, Ra
Store indexed (00010xxxxxxxxxxxxxxxxxxxxxxxxxxx)	M[R[Rb] + C (sign-extended)] ← R[Ra] Indexed addressing, Rb ≠ R0 If C = 0 ➔ Register Indirect addressing	st	C(Rb), Ra

1(b): ldr, str

Load relative (00011xxxx----xxxxxxxxxxxxxxxxxxx)	R[Ra] ← M[PC + C (sign-extended)] Relative addressing	ldr	Ra, C
Store relative (00100xxxx----xxxxxxxxxxxxxxxxxxx)	M[PC + C (sign-extended)] ← R[Ra] Relative addressing	str	C, Ra

Arithmetic and Logical Instructions

2(a): add, sub, and, or, shr, shl, ror, rol

Add (00101xxxxxxxxxxxx---------------)	R[Ra] ← R[Rb] + R[Rc]	add	Ra, Rb, Rc

Sub (00110xxxxxxxxxxxx---------------)	R[Ra] ← R[Rb] - R[Rc]	sub	Ra, Rb, Rc
AND (00111xxxxxxxxxxxx---------------)	R[Ra] ← R[Rb] ∧ R[Rc]	and	Ra, Rb, Rc
OR (01000xxxxxxxxxxxx---------------)	R[Ra] ← R[Rb] ∨ R[Rc]	or	Ra, Rb, Rc
Shift right (01001xxxxxxxxxxxx---------------)	Shift R[Rb] right into R[Ra] by count in R[Rc]	shr	Ra, Rb, Rc
Shift left (01010xxxxxxxxxxxx---------------)	Shift R[Rb] left into R[Ra] by count in R[Rc]	shl	Ra, Rb, Rc
Rotate right (01011xxxxxxxxxxxx---------------)	Rotate R[Rb] right into R[Ra] by count in R[Rc]	ror	Ra, Rb, Rc
Rotate left (01100xxxxxxxxxxxx---------------)	Rotate R[Rb] left into R[Ra] by count in R[Rc]	rol	Ra, Rb, Rc

2(b): addi, andi, ori

Add immediate (01101xxxxxxxxxxxxxxxxxxxxxxxxxxx)	R[Ra] ← R[Rb] + C (sign-extended) Immediate addressing If C = 0 → instruction acts like a simple register transfer If C ≠ 0 and Ra = Rb → Increment/decrement instruction Similar to ldi, however Rb can be any register.	addi	Ra, Rb, C
AND immediate (01110xxxxxxxxxxxxxxxxxxxxxxxxxxx)	R[Ra] ← R[Rb] ∧ C (sign-extended) Immediate addressing	andi	Ra, Rb, C
OR immediate (01111xxxxxxxxxxxxxxxxxxxxxxxxxxx)	R[Ra] ← R[Rb] ∨ C (sign-extended) Immediate addressing	ori	Ra, Rb, C

2(c): mul, div, neg, not

Multiply (10000xxxxxxxx-------------------)	HI, LO ← R[Ra] × R[Rb]	mul	Ra, Rb
Divide (10001xxxxxxxx-------------------)	HI, LO ← R[Ra] ÷ R[Rb]	div	Ra, Rb

Negate (10010xxxxxxxx-------------------)	R[Ra] ← - R[Rb]	neg	Ra, Rb
NOT (10011xxxxxxxx-------------------)	R[Ra] ← $\overline{\text{R[Rb]}}$	not	Ra, Rb

Conditional Branch Instructions

brzr, brnz, brmi, brpl

Branch (10100xxxxxxxx-----------------xx) PC ← R[Rb] if R[Ra] meets the condition

"branch if zero"	C2 = 00	brzr	Ra, Rb
"branch if nonzero"	C2 = 01	brnz	Ra, Rb
"branch if positive"	C2 = 10	brpl	Ra, Rb
"branch if negative"	C2 = 11	brmi	Ra, Rb

Jump Instructions

jr, jal

jr (10101xxxx--------------------------)	PC ← R[Ra] If Ra = R14, it is for procedure return	jr	Ra

jal
(10110xxxx--------------------------)
	R[14] ← PC + 4
	PC ← R[Ra]
		jal	Ra

Input/Output and MFHI/MFLO Instructions

in, out, mfhi, mflo

Input (10111xxxx-----------------------)	R[Ra] ← In.Port	in	Ra
Output (11000xxxx-----------------------)	Out.Port ← R[Ra]	out	Ra
Move from HI (11001xxxx-----------------------)	R[Ra] ← HI	mfhi	Ra
Move from LO (11010xxxx-----------------------)	R[Ra] ← LO	mflo	Ra

Miscellaneous Instructions

nop, halt

No-operation (11011---------------------------)	Do nothing	nop
Halt (11100---------------------------)	Halt the control stepping process	halt