The SRC Instruction Set

Register Structure: 32 32-bit General Purpose Registers.
Memory: 2^{32} memory bytes, byte addressable. Word Size: 32 bits.
Instruction size: 32 bits.

Category	Opcode	Operands	Meaning
Load and Store	`ld`	`ra, c2`	Load from absolute address; rb is register 0.
	`ld`	`ra, c2(rb)`	Load from displacement address.
	`ldr`	`ra, c1`	Load from relative address.
	`st`	`ra, c2`	Store into absolute address; rb is register 0.
	`st`	`ra, c2(rb)`	Store into displacement address.
	`str`	`ra, c1`	Store into relative address.
	`la`	`ra, c2`	Load value of absolute address into ra; rb is register 0.
	`la`	`ra, c2(rb)`	Load value of displacement address into ra.
	`lar`	`ra, c1`	Load value of relative address into ra.
Arithmetic	`add`	`ra, rb, rc`	Add rb to rc, and put result in ra.
	`addi`	`ra, rb, c2`	Add rb to immediate constant, and put result in ra.
	`sub`	`ra, rb, rc`	Subtract rc from rb, and put result in ra.
	`neg`	`ra, rc`	Place 2's complement negative of rc into ra.
	`or`	`ra, rb, rc`	OR rb and rc, and put result in ra.
	`ori`	`ra, rb, c2`	OR rb and immediate constant, and put result in ra.
	`and`	`ra, rb, rc`	AND rb and rc, and put result in ra.
	`andi`	`ra, rb, c2`	AND rb and immediate constant, and put result in ra.
	`not`	`ra, rc`	Place logical NOT of rc into ra.
Shift	`shr`	`ra, rb, c3`	Shift rb right into ra by constant shift count c3 ≤ 31.
	`shr`	`ra, rb, rc`	Shift rb right into ra by count in rc; c3 is 0.
	`shra`	`ra, rb, c3`	Shift rb right with sign-extend into ra by constant c3.
	`shra`	`ra, rb, rc`	Shift rb right with sign-extend into ra by count in rc.
	`shl`	`ra, rb, c3`	Shift rb left into ra by constant c3.
	`shl`	`ra, rb, rc`	Shift rb left into ra by count in rc; c3 is 0.
	`shc`	`ra, rb, c3`	Shift rb left circularly into ra by constant c3.
	`shc`	`ra, rb, rc`	Shift rb left circularly into ra by count in rc; c3 is 0.
Branch	`br`	`rb, rc, c3`	Branch to target in rb if rcl satisfies condition c3.
	`brl`	`ra,rb,rc,c3`	Branch to rb if rc satisfies c3, and save PC in ra.

Branch (cont'd.)	`br`	`rb`	Branch unconditionally to rb.
	`brl`	`ra, rb`	Branch unconditionally to rb, and save PC in ra.
	`brlnv`	`ra`	Do not branch, but save PC in ra.
	`brnv`		Branch never.
	`brzr`	`rb, rc`	Branch to rb if rc is zero.
	`brlzr`	`ra, rb, rc`	Branch to rb if rc is zero, and save PC in ra.
	`brnz`	`rb, rc`	Branch to rb if rc is nonzero.
	`brlnz`	`ra, rb, rc`	Branch to rb if rc is nonzero, and save PC in ra.
	`brpl`	`rb, rc`	Branch to rb if rc is positive or zero (sign is plus).
	`brlpl`	`ra, rb, rc`	Branch to rb if rc is positive, and save PC in ra.
	`brmi`	`rb, rc`	Branch to rb if rc is negative (sign is minus).
	`brlmi`	`ra, rb, rc`	Branch to rb if rc is negative, and save PC in ra.
Miscellaneous	`nop`		No operation. Used to insert pipeline bubble.
	`stop`		Set Run to zero, halting the machine.
	`een`		Exception enable. Set overall exception enable bit.
	`edi`		Exception disable. Clear overall exception enable.
	`rfi`		Return from interrupt. PC ←IPC; enable exceptions.
	`svi`	`ra, rb`	Save II and IPC in ra and rb, respectively.
	`ri`	`ra, rb`	Restore II and IPC from ra and rb, respectively.

SRC Assembly Language Conventions

A line of SRC assembly code has the form:

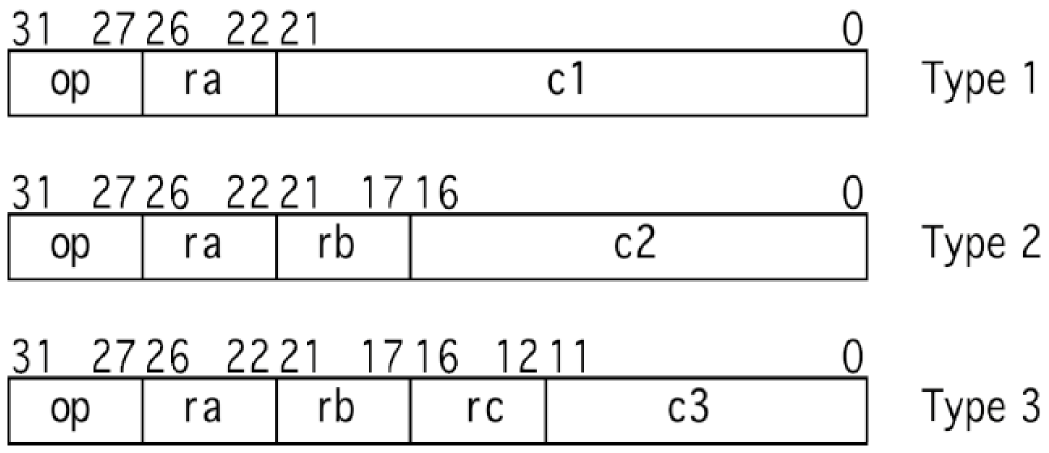
Label: op-code operands ;comments.

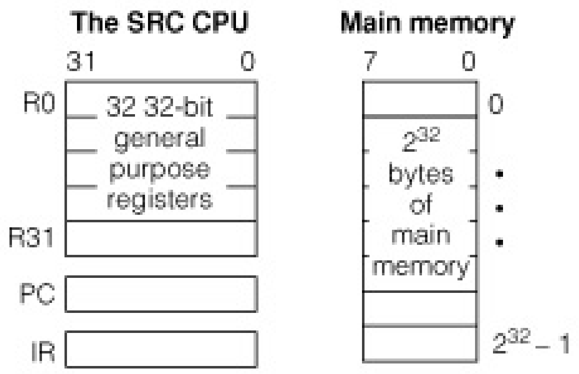
The label and its associated colon may be missing. The op-code field contains either a machine instruction mnemonic of a few lower case letters or an assembler pseudo operation that begins with a period. The pseudo operations used are summarized below and in Table B.1.

.org	Value	Load the program starting at address Value.
.equ	Value	Define the Label symbol to be the constant Value.
.dc	Value [,Value]	Allocate memory words and set to the 32-bit Values.
.dcb	Value [,Value]	Allocate bytes and load them with the 8-bit Values.
.dch	Value [,Value]	Allocate halfwords and load with the 16-bit Values.
.db	Count	Allocate storage for Count bytes.
.dh	Count	Allocate storage for Count 16-bit halfwords.
.dw	Count	Allocate storage for Count 32-bit words.

Values are assumed to be decimal unless terminated by B (binary) or H (hexadecimal).

SRC INSTRUCTION TYPES

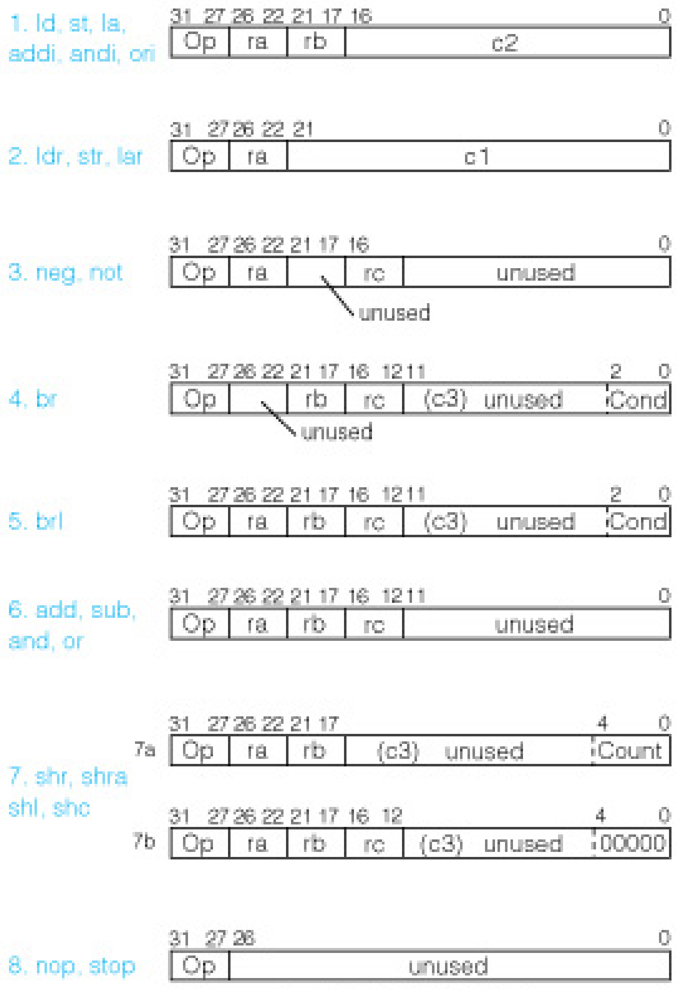



The SRC CPU
31
0
R0
32 32-bit general purpose registers
R31
PC
IR
Main memory
7
0
0
2^{32} bytes of main memory
$2^{32} - 1$

R[7] means contents of register 7

M[32] means contents of memory location 32

Instruction formats



Example

ld r3, A (R[3] = M[A])
ld r3, 4(r5) (R[3] = M[R[5] + 4])
addi r2, r4, 1 (R[2] = R[4] +1)

ldr r5, 8 (R[5] = M[PC + 8])
lar r6, 45 (R[6] = PC + 45)

neg r7, r9 (R[7] = – R[9])

brzr r4, r0
(branch to R[4] if R[0] == 0)

brlnz r6, r4, r0
(R[6] = PC; branch to R[4] if R[0] ≠ 0)

add r0, r2, r4 (R[0] = R[2] + R[4])

shr r0, r1, 4
(R[0] = R[1] shifted right by 4 bits

shl r2, r4, r6
(R[2] = R[4] shifted left by count in R[6])

stop