The RISC Architecture

NW 5.12.10, rev. 5.10.13

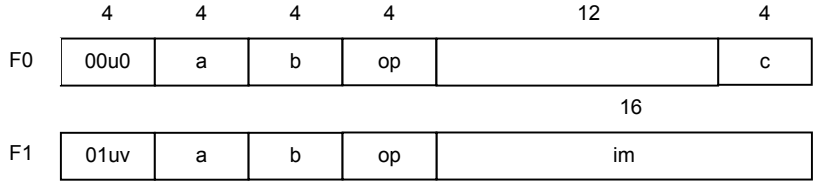
Resources and registers

From the viewpoints of the programmer and the compiler designer the computer consists of an arithmetic unit, a control unit and a store. The arithmetic unit contains 16 registers R0 – R15, with 32 bits each. The control unit consists of the instruction register IR, holding the instruction currently being executed, and the program counter PC, holding the word-address of the instruction to be fetched next. All branch instructions are conditional. The memory consists of 32-bit words, and it is byte-addressed. Furthermore, there are 4 flag registers N, Z, C and V, called the *condition codes*.

There are four types of instructions and instruction formats. *Register instructions* operate on registers only and feed data through a shifter or the arithmetic logic unit ALU. *Memory instructions* fetch and store data in memory. *Branch instructions* affect the program counter.

1. Register instructions (formats F0 and F1)

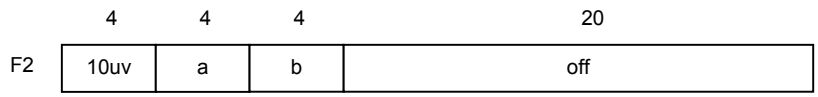
Register instructions assign the result of an operation to the destination register R.a. The first operand is the register R.b. The second operand *n* is either register R.c or is the constant *im*.



0	MOV a, n	R.a := n	
1	LSL a, b, n	R.a := R.b ← n	(shift left by n bits)
2	ASR a, b, n	R.a := R.b → n	(shift right by n bits with sign extension)
3	ROR a, b, n	R.a := R.b rot n	(rotate right by n bits)
4	AND a, b, n	R.a := R.b & n	logical operations
5	ANN a, b, n	R.a := R.b & ~n	
6	IOR a, b, n	R.a := R.b or n	
7	XOR a, b, n	R.a := R.b xor n	
8	ADD a, b, n	R.a := R.b + n	integer arithmetic
9	SUB a, b, n	R.a := R.b – n	
10	MUL a, b, n	R.a := R.a x n	
11	DIV a, b, n	R.a := R.b div n	
12	FAD a, b, c	R.a := R.b + R.c	floating-point arithmetic
13	FSB a, b, c	R.a := R.b – R.c	
14	FML a, b, c	R.a := R.a x R.c	
15	FDV a, b, c	R.a := R.b / R.c	

Immediate values are extended to 32 bits with 16 v-bits to the left. Apart from R.a these instructions also affect the flag registers N (negative) and Z (zero). The ADD and SUB instructions also set the flags C (carry, borrow) and V (overflow).

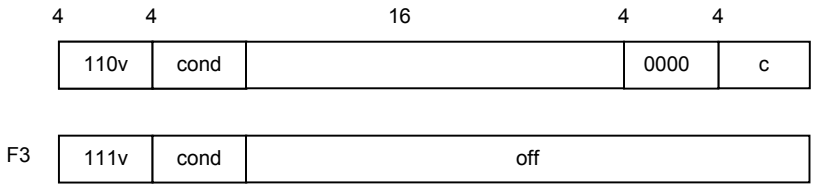
2. Memory instructions (format F2)



LD	a, b, im	R.a := Mem[R.b + off]	u = 0
ST	a, b, im	Mem[R.b + off] := R.a	u = 1

If v = 0, access is for a word (4 bytes). If v = 1, a single byte is accessed.

3. Branch instructions (Format F3)



Bcond dest

If u = 0, the destination address is taken from register R.c. If u = 1, it is PC+1 + offset. If v = 1, the link address PC+1 is deposited in register R15.

code	cond		condition	code	cond		condition
0000	MI	negative (minus)	N	1000	PL	positive (plus)	~N
0001	EQ	equal (zero)	Z	1001	NE	positive (plus)	~Z
0010	CS	carry set	C	1010	CC	carry clear	~C
0011	VS	overflow set	V	1011	VC	overflow clear	~V
0100	LS	less or same	~C|Z	1100	HI	high	~(~C}Z)
0101	LT	less than	N≠V	1101	GE	greater or equal	~(N≠V)
0110	LE	less or equal	(N≠V)|Z	1110	GT	greater than	~((N≠V)|Z)
0111		always	true	1111		never	false

Special features

Modifier bit u = 1 changes the effect of certain instructions as follows:

ADD', SUB'	add, subtract also carry C
MUL'	unsigned multiplication
MOV' form 0, c = 0:	R.a := H
MOV' form 0, c = 1:	R.a := [N, Z, C, V]
MOV' form 1	R.a := [R.c[15:0], 16'b0] (R.c left shifted 16 bits)

The MUL instruction deposits the high 32 bits of the product in the auxiliary register H. The DIV instruction deposits the remainder in H.