



CPE 221: Computer Organization

08 ARM Machine Language
rahul.bhadani@uah.edu

Rahul Bhadani

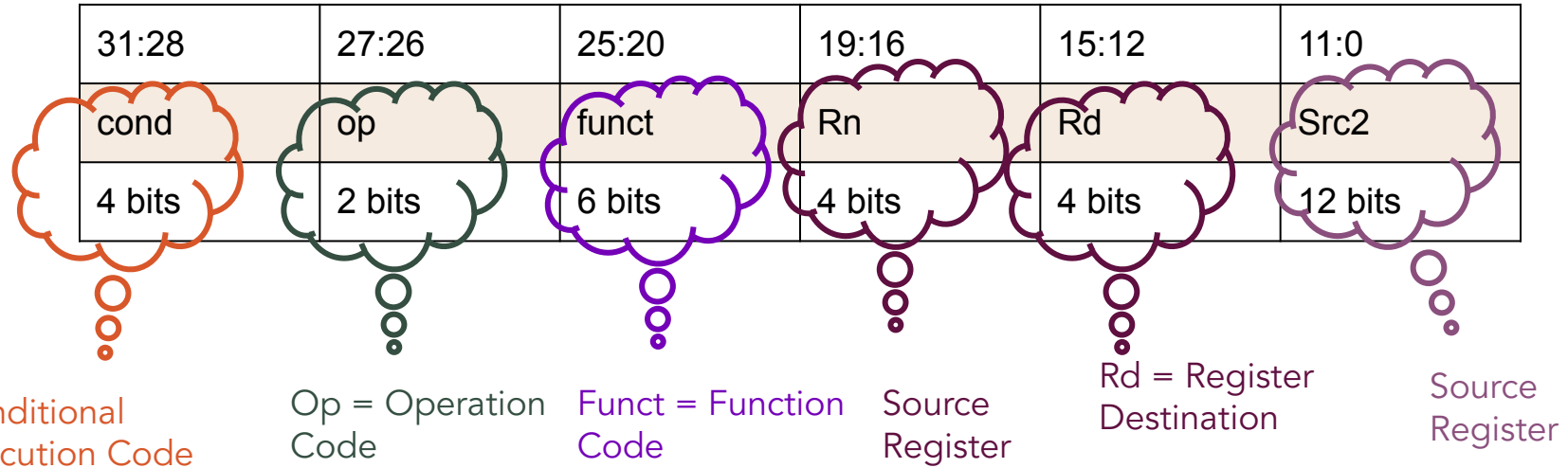


Machine Language

Good design demands good compromise

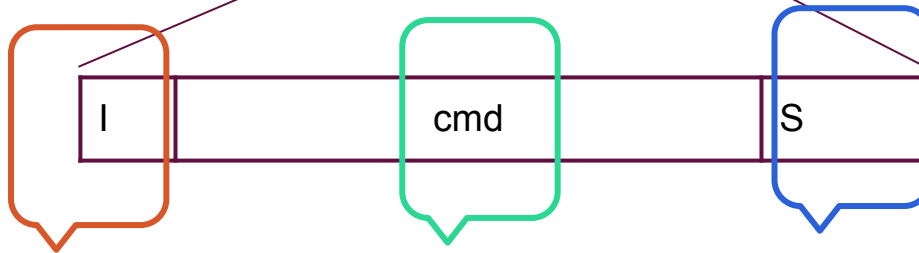
Data-processing Instructions

32-bit instruction



Data-processing Instructions: Functional Code

31:28	27:26	25:20	19:16	15:12	11:0
cond	op	funct	Rn	Rd	Src2
4 bits	2 bits	6 bits	4 bits	4 bits	12 bits



$I = 1$ when Src2 is an immediate

Specific data-processing instruction

$S = 1$ when an instruction sets the condition flags

Data-processing Instructions: Second Source

imm8 = 8-bit immediate
rot = 4-bit rotation

imm8 is rotated right by $2 \times \text{rot}$ to create a 32-bit constant

cond	op	funct	Rn	Rd	Src2
------	----	-------	----	----	------

shamt5 = a constant by which
the register Rm is shifted

Immediate, I = 1

11:8	7:0
rot	imm8

Rs = a register by which the
register Rm is shifted

Register, I = 0

11:7	6:5	4	3:0
shamt5	sh	0	Rm

sh = encodes the type of shift to
perform (LSL, LSR, ASR, ROR)

Register-shifted Register, I = 0

11:8	7	6:5	4	3:0
Rs	0	sh	1	Rm

Example Assembly Code for ADD and SUB

Hex is used for compact representation of the Assembly Code.

Assembly Code

Field Values

Machine Code

ADD R5, R6, R7
(0xE0865007)
SUB R8, R9, R10
(0xE049800A)

31:28	27:26	25	24:21	20	19:16	15:12	11:7	6:5	4	3:0
1110 ₂	00 ₂	0	0100 ₂	0	6	5	0	0	0	7
1110 ₂	00 ₂	0	0010 ₂	0	9	8	0	0	0	10
cond	op	I	cmd	S	Rn	Rd	shamt5	sh		Rm

31:28	27:26	25	24:21	20	19:16	15:12	11:7	6:5	4	3:0
1110	00	0	0100	0	0110	0101	00000	00	0	0111
1110	00	0	0010	0	1001	1000	00000	00	0	1010
cond	op	I	cmd	S	Rn	Rd	shamt5	sh		Rm

Assembly Code

Field Values

Machine Code

ADD R0, R1, #42
(0xE281002A)
SUB R2, R3, #0xFF0
(0xE2432EFF)

31:28	27:26	25	24:21	20	19:16	15:12	11:8	7:0
1110 ₂	00 ₂	1	0100 ₂	0	1	0	0	42
1110 ₂	00 ₂	1	0010 ₂	0	3	2	14	255
cond	op	I	cmd	S	Rn	Rd	rot	imm8

31:28	27:26	25	24:21	20	19:16	15:12	11:8	7:0
1110	00	1	0100	0	0001	0000	0000	00101010
1110	00	1	0010	0	0011	0010	1110	11111111
cond	op	I	cmd	S	Rn	Rd	rot	imm8

Example Assembly Code for Shift Instruction

Assembly Code

Field Values

Machine Code

LSL R0, R9, #7
(0xE1A00389)

ROR R3, R5, #21
(0xE1A03AE5)

31:28	27:26	25	24:21	20	19:16	15:12	11:7	6:5	4	3:0
1110 ₂	00 ₂	0	1101 ₂	0	0	0	7	00 ₂	0	9
1110 ₂	00 ₂	0	1101 ₂	0	0	3	21	11 ₂	0	5
cond	op	I	cmd	S	Rn	Rd	shamt5	sh		Rm

31:28	27:26	25	24:21	20	19:16	15:12	11:7	6:5	4	3:0
1110	00	0	1101	0	0000	0000	00111	00	0	1001
1110	00	0	1101	0	0000	0011	10101	11	0	0101
cond	op	I	cmd	S	Rn	Rd	shamt5	sh		Rm

Assembly Code

Field Values

Machine Code

LSR R4, R8, R6
(0xE1A04638)

ASR R5, R1, R12
(0xE1A05C51)

31:28	27:26	25	24:21	20	19:16	15:12	11:8	7	6:5	4	3:0
1110 ₂	00 ₂	0	1101 ₂	0	0	4	6	0	01 ₂	1	8
1110 ₂	00 ₂	0	1101 ₂	0	0	5	12	0	10 ₂	1	1
cond	op	I	cmd	S	Rn	Rd	Rs		sh		Rm

31:28	27:26	25	24:21	20	19:16	15:12	11:8	7	6:5	4	3:0
1110	00	0	1101	0	0000	0100	0110	0	01	1	1000
1110	00	0	1101	0	0000	0101	1100	0	10	1	0001
cond	op	I	cmd	S	Rn	Rd	Rs		sh		Rm

Memory Instructions

11:7	6:5	4	3:0
shamt5	sh	1	Rm

31:28	27:26	25	24	23	22	21	20	19:16	15:12	11:0
cond	op	I	P	U	B	W	L	Rn	Rd	Src2

Function
Operation
I = Immediate
P = Pre-index
U = Add
W = Writeback
L = Load
B = Byte

Note: Book
is using I bar
(\bar{I}) instead I

P	W	Index Mode
0	0	Post-index
0	1	Not supported
1	0	Offset
1	1	Pre-index

L	B	Instruction
0	0	STR
0	1	STRB
1	0	LDR
1	1	LDRB

STR, a Memory Instruction in Machine Code

STR R11, [R5], #-26

Post-indexing: $P = 0$, $W = 0$.

Immediate offset is subtracted from the base. So $I = 0$, $U = 0$.

Assembly Code

Field Values

Machine Code

	31:28	27:26	25:20	19:16	15:12	11:0	31:28	27:26	25:20	19:16	15:12	11:0
STR R11, [R5], #-26	1110 ₂	01 ₂	0000000 ₂	5	11	26	1110	01	000000	0101	1011	0000 0001 1010
	cond	op	IPUBWL	Rn	Rd	imm12	E	4	0	5	B	0 1 A

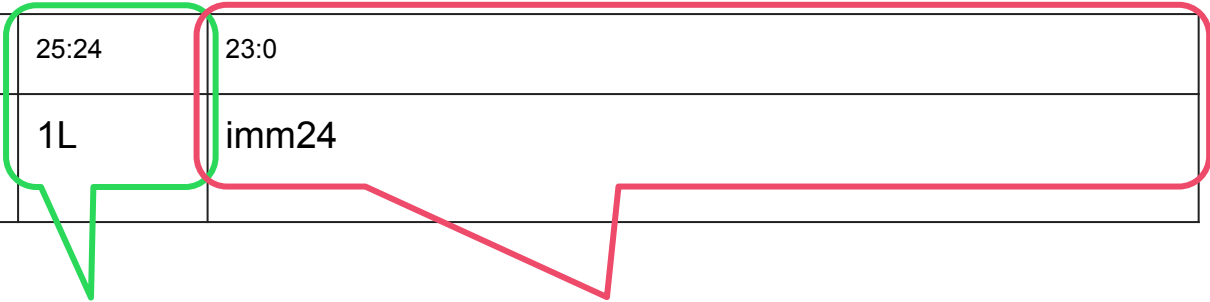
Branch Instruction

24-bit signed immediate.

Branch-instruct:

4-bit condition flag, 2-bit op code, 2-bit function code

31:28	27:26	25:24	23:0
cond	op = 10	1L	imm24

The diagram shows a table representing the Branch Instruction format. The first row shows bit ranges: 31:28, 27:26, 25:24, and 23:0. The second row shows the corresponding fields: cond, op = 10, 1L, and imm24. A green bracket highlights the 25:24 bit range (1L) with a callout explaining its function. A red bracket highlights the 23:0 bit range (imm24) with a callout explaining its function.

L = 1 for BL
L = 0 for B

In 2's complement, used to specify an instruction address relative to PC + 8.

Example of Machine Code for Branch Instruction

ARM Assembly Code

```
0x80A0 BLT THERE
0x80A4 ADD R0, R1, R2
0x80A8 SUB R0, R0, R9
0x80AC ADD SP, SP, #8
0x80B0 MOV PC, LR
0x80B4 THERE SUB R0, R0, #1
0x80B8 ADD R3, R3, #0x5
```

BTA = Branch Target Address, the instruction address to be executed when the branch is taken. BLT has BTA of 0x80B4, the instruction address of THERE.

24-bit immediate field gives the number of instructions between the BTA and PC+8 (two instructions past the branch)

imm24 here is 3, as BTA is three instructions past PC + 8 (0x80A8).

The processor calculates BTA from the instruction by sign-extending the 24-bit immediate, shifting it left 2 (to convert word to bytes) and adding it to PC-8. (Why 8?, Due to pipelining, the PC points to two instructions ahead of the current instruction, we will see this later)

If the immediate is 101010101010101010101010 (24 bits), sign-extending it would result in 111111111010101010101010 (32 bits) if it's negative.

Assembly Code

Field Values

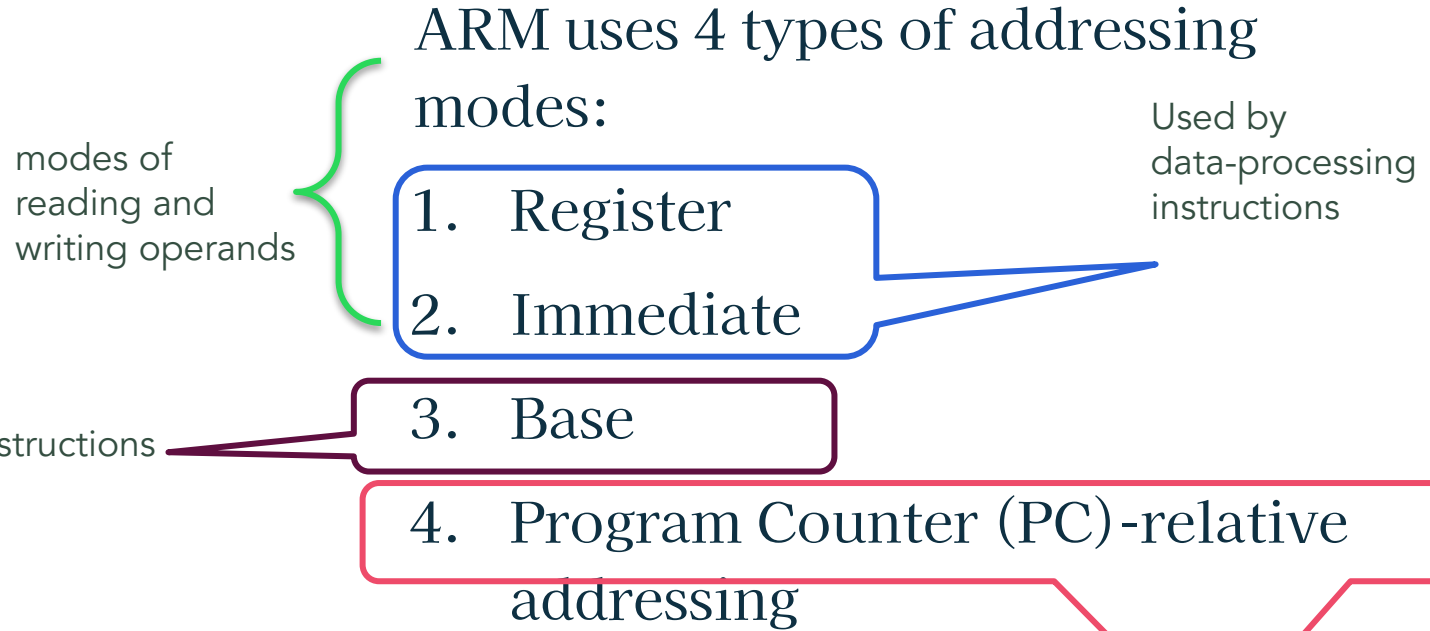
Machine Code

BLT THERE
(0xBA000003)

31:28	27:26	25:24	23:0
1011 ₂	10 ₂	10 ₂	3
cond opfunc		imm24	

31:28	27:26	25:24	23:0
1011	10	10	0000 0000 0000 0000 0000 0011
cond opfunc		imm24	

Addressing Mode



Quick tips about deciphering machine code

1. If the op code is 00, it is a data-processing instruction
2. If the op code is 01, it is a memory instruction
3. If the op code is 10, it is a branch instruction

Stored Program Concept

Instructions are stored in the memory - hence the *Stored Program Concept*.
Stored Program offers general purpose computing, can execute multitude of applications.

Instructions in a stored programs are retrieved or fetched from memory and executed by the processor.

Processor fetches the instructions from memory sequentially, fetched instructions are decoded and executed by the digital hardware.

The address of the current instruction is kept in a 32-bit register known as the program counter.

A read to the PC returns the address of the current instruction plus 8.