10 ARM Machine Code: Branching and Memory Instructions

CPE 221

The University of Alabama in Huntsville

Rahul Bhadani

April 1, 2025

Table of Content

Memory Instruction

0 0 0 0 1 1 1 0 1 0 1 0 0 0 0 1 1 0 1 1

1 0 1 0 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1

1 0 1 0 0 0 0 0 1 0 0 1 0 0 1 0 0 1 1 1

Memory Instructions

31:28	27:26	25	24	23	22	21	20	19:16	15:12	11:0
cond	op	$\bar{I}$	P	U	B	W	L	Rn	Rd	Src2
4 bits	2 bits	1 bit	1 bit	1 bit	1 bit	1 bit	1 bit	4 bits	4 bits	12 bits

If $\bar{I} = 1$ (no immediate) but with register shifting:

Bits 11:7	Bits 6:5	Bit 4	Bits 3:0
shamt5	sh	1/0	Rm
5-bit shift amount	specific shift instructions	1 in the text-book, 0 in the cpulator	Register used for shifting

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Memory Instructions

31:28	27:26	25	24	23	22	21	20	19:16	15:12	11:0
cond	op	$\bar{I}$	P	U	B	W	L	Rn	Rd	Src2
4 bits	2 bits	1 bit	1 bit	1 bit	1 bit	1 bit	1 bit	4 bits	4 bits	12 bits

If $\bar{I} = 1$ (no immediate) no shifting, just a register:

Bits 11:7	Bits 6:5	Bit 4	Bits 3:0
shamt5	sh	0	Rm
00000	00	0 (default)	Register

Memory Instructions

31:28	27:26	25	24	23	22	21	20	19:16	15:12	11:0
cond	op	$\bar{I}$	P	U	B	W	L	Rn	Rd	Src2
4 bits	2 bits	1 bit	1 bit	1 bit	1 bit	1 bit	1 bit	4 bits	4 bits	12 bits

If $\bar{I}$ =0 (immediate):

Bits 11:0
imm12
12 bit immediate

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Memory Instructions

31:28	27:26	25	24	23	22	21	20	19:16	15:12	11:0
cond	op	$\bar{I}$	P	U	B	W	L	Rn	Rd	Src2
4 bits	2 bits	1 bit	1 bit	1 bit	1 bit	1 bit	1 bit	4 bits	4 bits	12 bits

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

Memory Instructions

31:28	27:26	25	24	23	22	21	20	19:16	15:12	11:0
cond	op	$\bar{I}$	P	U	B	W	L	Rn	Rd	Src2
4 bits	2 bits	1 bit	1 bit	1 bit	1 bit	1 bit	1 bit	4 bits	4 bits	12 bits

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

Memory Instructions

31:28	27:26	25	24	23	22	21	20	19:16	15:12	11:0
cond	op	$\bar{I}$	P	U	B	W	L	Rn	Rd	Src2
4 bits	2 bits	1 bit	1 bit	1 bit	1 bit	1 bit	1 bit	4 bits	4 bits	12 bits

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

Example 1

Encode `STR R11, [R5], #-26`

This instruction has indexing type: post-indexing: P = 0, W = 0.
Immediate offset is subtracted from the base. So I = 0, U = 0.

31:28	27:26	25	24	23	22	21	20	19:16	15:12	11:0
cond	op	$\bar{I}$	P	U	B	W	L	Rn	Rd	Src2
1110	01	0	0	0	0	0	0	0101	1011	0000 0001 1010

0x E405B017

Example 2

Encode `LDR R2, [R3, #16]!`

Pre-indexed Word Load:

- Load word into R2, L = 1, B = 0 (no Byte)
- Base register R3
- Immediate offset of 16, which is positive, U = 1
- Pre-indexed (update base register) P = 1 W = 1, P

31:28	27:26	25	24	23	22	21	20	19:16	15:12	11:0
cond	op	$\bar{I}$	P	U	B	W	L	Rn	Rd	Src2
1110	01	0	1	1	0	1	1	0011	0010	0000 0001 0000

0x E5B32010

Example 3

Encode `STRB R7, [R4], R6`

Post-indexed Byte Store:

- Store byte from R7, L = 0, Bye, hence B = 1
- Base register R4
- Offset from R6 register, no Immediate, sets U = 1
- Post-indexed, P = 0, W = 0

31:28	27:26	25	24	23	22	21	20	19:16	15:12	11:0
cond	op	$\bar{I}$	P	U	B	W	L	Rn	Rd	Src2
1110	01	1	0	1	1	0	0	0100	0111	0000 0000 0110

0x E6C47006

Example 4

Encode `LDR R11, [R7, R8, LSL #2]`

Word Load with Register Shift Offset:

- Load word into R11, L = 1, B = 0
- Base register R7
- Offset from R8, logically shifted left by 2, U = 1, Offset, so P = 1, W = 0
- Register-based offset with shift, sh = 00 for LSL

31:28	27:26	25	24	23	22	21	20	19:16	15:12	11:0
cond	op	$\bar{I}$	P	U	B	W	L	Rn	Rd	Src2
1110	01	1	1	1	0	0	1	0111	1011	00010 00 0 1000

0x E797B108

Branching Instruction

0 0 1 1 1 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1

0 1 0 1 1 0 0 1 1 0 0 0 0 0 1 1 0 1 1 1

1 1 1 1 0 1 0 0 1 0 1 0 1 1 1 1 1 0 1 1

Encding Branching Instrunctions: B Label

31:28	27:26	25	24	23:0
cond	op	1	L	imm24
cond	10	1	L = 0	imm24 is in 2's complement, used to specify an instruction address relative to PC + 8.

Encding Branching Instrunctions: BL Label

Branch with Link

31:28	27:26	25	24	23:0
cond	op	1	L	imm24
cond	10	1	L = 1	imm24 is in 2's complement, used to specify an instruction address relative to PC + 8.

Example 5

Encode

```
0x80A0 BLT THERE
```

When we have the rest of the propgram as:

```
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
```

Example 5

Step 1: We calculate BTA. 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.

Step 2: 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.

31:28	27:26	25	24	23:0
cond	op	1	L	imm24
1011	10	1	0	0000 0000 0000 0000 0000 0011

Example 6

If the instruction at address 0x8400 is `0xEA000005`, what is the branch target address?
Solution:

- This is an unconditional branch (B) instruction
- The immediate field is 5
- Branch target = PC+8 + (imm24 $\times$ 4)
- PC+8 for the instruction at 0x8400 is 0x8408
- 5 $\times$ 4 = 20 (0x14)
- 0x8408 + 0x14 = 0x841C

Conditions Encoding

cond	Mnemonic	Name	CondEx
0000	EQ	Equal	Z
0001	NE	Not equal	$\overline{Z}$
0010	CS / HS	Carry set / unsigned higher or same	C
0011	CC / LO	Carry clear / unsigned lower	$\overline{C}$
0100	MI	Minus / negative	N
0101	PL	Plus / positive or zero	$\overline{N}$
0110	VS	Overflow / overflow set	V
0111	VC	No overflow / overflow clear	$\overline{V}$
1000	HI	Unsigned higher	$\overline{Z}C$
1001	LS	Unsigned lower or same	Z OR $\overline{C}$
1010	GE	Signed greater than or equal	$\overline{N \oplus V}$
1011	LT	Signed less than	N$\oplus$V
1100	GT	Signed greater than	$\overline{Z}$ $(\overline{N \oplus V})$
1101	LE	Signed less than or equal	Z OR (N$\oplus$V)
1110	AL (or none)	Always / unconditional	Ignored

Encoding PUSH and POP Instruction

0 0 0 0 0 0 1 0 0 1 0 1 1 1 1 1 1 0 0 1

1 1 0 1 1 1 0 1 0 1 0 0 0 1 1 0 0 0 1 0

1 1 1 0 0 0 0 1 1 0 0 0 1 1 0 1 1 0 0 1

PUSH Instruction

```
PUSH {R4, R5, LR}
```

is equivalent to

```
STMFD SP!, {R4, R5, LR}
```

31:28	27:26	25	24	23	22	21	20	19:16	15:0
cond	op	I	P	U	S	W	L	Rn	register list
1110	10	0	1	0	0	1	0	1101	0100000000110000

P = 1 Pre-indexing mode, U = 0 for decrement, W = 1 write back, updates SP after storing, L = 0 for STM (Store operation)
Rn is the base register which is SP. Register list uses bit masking. Hence, encoding is 0x E92D4030.

POP Instruction

`POP {R4, R5, LR}`

is equivalent to

`LDMFD SP!, {R4, R5, LR}`

31:28	27:26	25	24	23	22	21	20	19:16	15:0
cond	op	I	P	U	S	W	L	Rn	register list
1110	10	0	0	1	0	1	1	1101	0100000000110000

P = 0 Post-indexing mode, U = 1 for increment, W = 1 write back, updates SP after loading, L = 1 for LDM (Load operation)
Rn is the base register which is SP. Register list uses bit masking. Hence, encoding is 0x E8BD4030.

The End