09 ARM Machine Code: Data Processing Instructions

CPE 221

The University of Alabama in Huntsville

Rahul Bhadani

March 19, 2025

Table of Content

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Data Processing Instructions

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

Bit 25	Bits 24:21	Bit 20
I	cmd	S
I = 1 when Src2 is an immediate	Specific data-processing instruction	S= 1 when an instruction sets the condition flags

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Encoding Immediates in ARM Assembly

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

Encoding Immediates for Data Processing Instructions

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

Bit 11:8	Bits 7:0
rot	imm8

imm8 = 8-bit immediate

rot = 4-bit rotation

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

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

4-bit Rotation, 8-bit Immediate

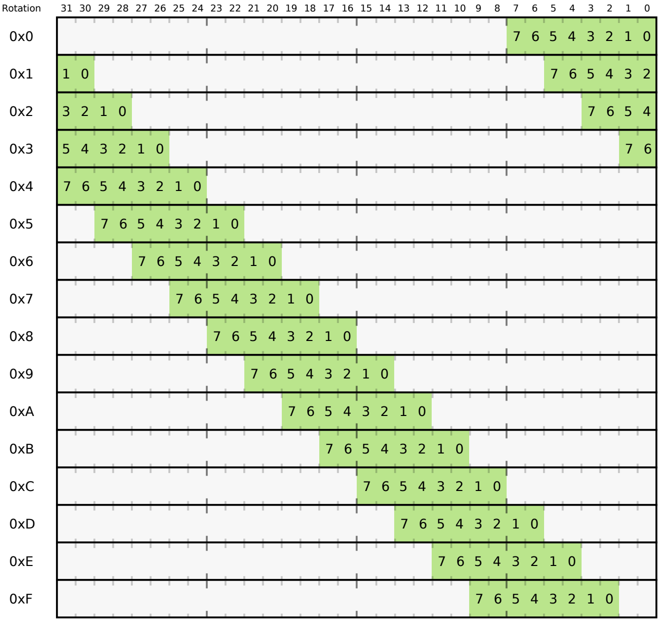
ARM doesn't use the 12-bit immediate value as a 12-bit number. Instead, it's an 8-bit number with a 4-bit rotation, like this:

Bit 11:8	Bits 7:0
rot	imm8

The 4-bit rotation value has 16 possible settings (i.e. 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30), so it's not possible to rotate the 8-bit value to any position in the 32-bit word. The most useful way to use this rotation value is to multiply it by two. It can then represent all even numbers from zero to 30.

4-bit Rotation, 8-bit Immediate

To form the constant for the data processing instruction, the 8-bit immediate value is extended with zeroes to 32 bits, then rotated the specified number of places to the right. For some values of rotation, this can allow splitting the 8-bit value between bytes. See the table in the next slide for all possible rotations.


Rotation
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0x0 7 6 5 4 3 2 1 0
0x1 1 0 7 6 5 4 3 2
0x2 3 2 1 0 7 6 5 4
0x3 5 4 3 2 1 0 7 6
0x4 7 6 5 4 3 2 1 0
0x5 7 6 5 4 3 2 1 0
0x6 7 6 5 4 3 2 1 0
0x7 7 6 5 4 3 2 1 0
0x8 7 6 5 4 3 2 1 0
0x9 7 6 5 4 3 2 1 0
0xA 7 6 5 4 3 2 1 0
0xB 7 6 5 4 3 2 1 0
0xC 7 6 5 4 3 2 1 0
0xD 7 6 5 4 3 2 1 0
0xE 7 6 5 4 3 2 1 0
0xF 7 6 5 4 3 2 1 0

Example 1

Encode `SUB R2, R3, 0xFF0`

- `0xFF0 = 1111 1111 0000`
- It is 12 bits but need to find a way to represent using 8 bits.
- Given, 0XFF0, we first write it as 32-bit: 0x00000FF0 which is 0000 0000 0000 0000 0000 1111 1111 0000
- Since the above number cannot be accomodated in 8 bits (minimum need is 12 bits), we need to rotate it.
- By looking at the table in the previous slide, we see that 0000 0000 0000 0000 0000 1111 1111 0000 is corresponding to 0xE which means we need $14 \times 2 = 28$ rotation.
- Hence rot $=$ 1110, imm8 $=$ 1111 1111

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Note

Not all 32 bits can be represented by the above logic, so not all immediates greater than 8 bits are valid immediates.

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Encoding Shift Instructions

Encoding Shift Instructions

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

Bit 25	Bits 24:21	Bit 20
I	cmd	S
I = 0 for LSL, LSR, ROR, ASR I = 1 for MOV if immediate, otherwise 0	1101	S= 1 when an instruction sets the condition flags

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Encoding Shift Instructions: Register Value Shift by Constant

Register Value Shift by Constant or Immediate (Shift Amount)

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

LSL/LSR/ASR/ROR Rd, Rm, shamt5

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

sh	Instructions
00	LSL/MOV
01	LSR
10	ASR
11	ROR

shamt5 = a constant by which the register Rm is shifted

shamt5 = 00000 for MOV

THE UNIVERSITY OF
ALABAMA IN HUNTSVILLE

Example 2

Encode `LSL R3, R2, #23`

31:28	27:26	25:20	19:16	15:12	11:0
1110	00	0 1101 0	0000	0011	Src2
4 bits	2 bits	6 bits	4 bits	4 bits	12 bits

Src2:

Bits 11:7	Bits 6:5	Bit 4	Bits 3:0
10111	00	0	0010

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Example 3

Encode `MOV R3, R2`

31:28	27:26	25:20	19:16	15:12	11:0
1110	00	0 1101 0	0000	0011	Src2
4 bits	2 bits	6 bits	4 bits	4 bits	12 bits

Src2:

Bits 11:7	Bits 6:5	Bit 4	Bits 3:0
00000	00	0	0010

1110 0001 1010 0000 0011 0000 0000 0010

0x E 1 A 0 3 0 0 2

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Example 4

Encode `MOV R3, #0xFF0`

31:28	27:26	25:20	19:16	15:12	11:0
1110	00	1 1101 0	0000	0011	Src2
4 bits	2 bits	6 bits	4 bits	4 bits	12 bits

Src2: follows the same convention as in Example 1.

1110 0011 1010 0000 0011 1110 1111 1111

0x E 3 A 0 3 E F F

Encoding Shift Instructions: Register Value Shift by Register Value

Register Value Shift by Constant or Immediate (Shift Amount)

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

LSL/LSR/ASR/ROR Rd, Rm, Rs

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

sh	Instructions
00	LSL
01	LSR
10	ASR
11	ROR

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Example 5

Encode `ASR R3, R5, R7`

31:28	27:26	25:20	19:16	15:12	11:0
1110	00	0 1101 0	0000	0011	Src2
4 bits	2 bits	6 bits	4 bits	4 bits	12 bits

Src2:

Bits 11:8	Bit 7	Bits 6:5	Bit 4	Bits 3:0
0111	0	10	1	0101

1110 0001 1010 0000 0011 0111 0101 0101

0x E 1 A 0 3 7 5 5

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Example 6

Encode `ADD R7, R2, R3, LSL #2`

Rn = R2, Rd = R7, Rm = R3

31:28	27:26	25:20	19:16	15:12	11:0
1110	00	0 0100 0	0010	0111	Src2
4 bits	2 bits	6 bits	4 bits	4 bits	12 bits

Src2:

Bits 11:7	Bits 6:5	Bit 4	Bits 3:0
00010	00	0	0011

1110 0000 1000 0010 0111 0001 0000 0011

0x E 0 8 2 7 1 0 3

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Multiply Instructions

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

Multiply Instructions

Multiply instructions use the encoding in below. The 3-bit *cmd* field specifies the type of multiply, as given in in the Table.

Multiply

31:28	27:26	25:24	23:21	20	19:16	15:12	11:8	7:4	3:0
cond	op 00	00	cmd	S	Rd	Ra	Rm	1001	Rn
4 bits	2 bits	6 bits			4 bits	4 bits	4 bits	4 bits	4 bits

Multiply instruction encoding

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Types of Multiply Instructions

cmd	Name	Description	Operation
000	MUL Rd, Rn, Rm	Multiply	Rd $\leftarrow$ Rn $\times$ Rm (low 32 bits)
001	MLA Rd, Rn, Rm, Ra	Multiply Accumulate	Rd $\leftarrow$ (Rn $\times$ Rm)+Ra (low 32 bits)
100	UMULL Rd, Rn, Rm, Ra	Unsigned Multiply Long	{Rd, Ra} $\leftarrow$ Rn $\times$ Rm (all 64 bits, Rm/Rn unsigned)
101	UMLAL Rd, Rn, Rm, Ra	Unsigned Multiply Accumulate Long	{Rd, Ra} $\leftarrow$ (Rn $\times$ Rm)+{Rd, Ra} (all 64 bits, Rm/Rn unsigned)
110	SMULL Rd, Rn, Rm, Ra	Signed Multiply Long	{Rd, Ra} $\leftarrow$ Rn $\times$ Rm (all 64 bits, Rm/Rn signed)
111	SMLAL Rd, Rn, Rm, Ra	Signed Multiply Accumulate Long	{Rd, Ra} $\leftarrow$ (Rn $\times$ Rm)+{Rd, Ra} (all 64 bits, Rm/Rn signed)

The End