숭실대학교
Soongsil University

Computer Architecture

과제 #1: ARM Instructions 분석

2021년 1학기
Young Geun Kim (김영근)

이진수

- $a_5a_4a_3a_2a_1a_0.a_{-1}a_{-2}a_{-3}$

  $= a_nr^n+a_{n-1}r^{n-1}+\ldots+a_2r^2+a_1r+a_0+a_{-1}r^{-1}+a_{-2}r^{-2}+\ldots+a_{-m}r^{-m}$

$7392 = 7 \times 10^3 + 3 \times 10^2 + 9 \times 10^1 + 2 \times 10^0$

$(11010)_2 = 1\times2^4+1\times2^3+0\times2^2+1\times2^1+0\times2^0 = (26)_{10}$

2^{10} = 1Kilo

2^{20} = 1Mega

2^{30} = 1Giga

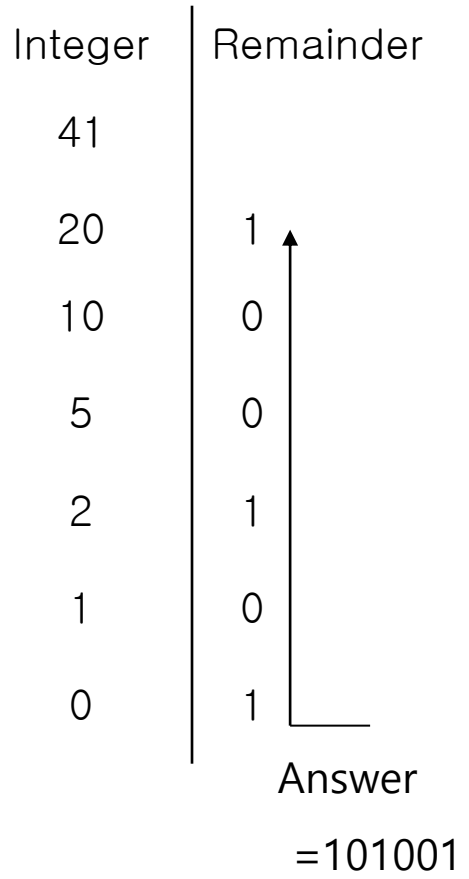
Table 1-1
Powers of Two

n		n		n	
0	1	8	256	16	65,536
1	2	9	512	17	131,072
2	4	10	1,024	18	262,144
3	8	11	2,048	19	524,288
4	16	12	4,096	20	1,048,576
5	32	13	8,192	21	2,097,152
6	64	14	16,384	22	4,194,304
7	128	15	32,768	23	8,388,608

이진수 변환

- Ex 1-1) Convert decimal 41 to binary.

	Integer Quotient		Remainder	Coefficient
41/2 =	20	+	½	$a_0 = 1$
20/2 =	10	+	0	$a_1 = 0$
10/2 =	5	+	0	$a_2 = 0$
5/2 =	2	+	½	$a_3 = 1$
2/2 =	1	+	0	$a_4 = 0$
1/2 =	0	+	½	$a_5 = 1$



answer : $(41)_{10} = (a_5a_4a_3a_2a_1a_0)_2 = (101001)_2$

8진수와 16진수

Table 1-2
Numbers with Different Bases

Decimal (base 10)	Binary (base 2)	Octal (base 8)	Hexadecimal (base 16)
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

보수 (부호있는 연산을 위한 음수의 표현 방식)

- N이라는 숫자의 (r-1)의 보수는 (r^n-1) - N
  - n은 N의 자릿수

- r=10, r-1=9, N의 9의 보수는 $(10^n - 1)$ - N

  Ex) the 9's complements of 546700 is 999999 (= 1000000 – 1) -546700 = 453299

  the 9's complements of 012398 is 999999-012398 = 987601

- 이진수의 경우, r=2, r-1=1

  N의 1의 보수는 $(2^n - 1)$ - N

  Ex) the 1's complements of 1011000 is (10000000 – 1) - 1011000 = 1111111 – 1011000 = 0100111

  the 1's complements of 0101101 is 1010010

보수 (부호있는 연산을 위한 음수의 표현 방식)

- N이라는 숫자의 r의 보수는 r^n - N
  - n은 N의 자릿수

- r^n-N = [(r^n-1)-N] + 1
  - R의 보수는 (r-1)의 보수 + 1임

Ex) the 2's complements of 1011000 is 0100111 + 1 = 0101000

the 2's complements of 0101101 is 1010010 + 1 = 1010011

보수 (부호있는 연산을 위한 음수의 표현 방식)

- Ex1-7) X=1010100, Y=1000011, (a) X-Y, (b) Y-X

X =	1010100
2's complement of Y =	+0111101
Sum =	10010001
Discard end carry 2^7 =	−10000000
Answer: X−Y =	0010001

Y =	1000011
2's complement of X =	+0101100
Sum =	1101111

There is no carry.

The answer is Y-X = -(2's complement of 1101111)=-0010001

보수 (부호있는 연산을 위한 음수의 표현 방식)

- Ex) X=1010100, Y=1010100, X-Y

X =	1010100
2's complement of Y =	+0101100
Sum =	10000000
Discard end carry 2^7 =	−10000000
Answer: X−Y =	0000000

The answer is X-Y = 0

이진수에서의 Shift 연산

$(11010)_2 = 1\times2^4+1\times2^3+0\times2^2+1\times2^1+0\times2^0 = (26)_{10}$

Shift Left 2

$(1101000)_2 = 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$

$= (104)_{10} = (26)_{10} \times 4$

$(110100)_2 = 1\times2^5+1\times2^4+0\times2^3+1\times2^2+0\times2^1+0\times2^0 = (52)_{10}$

Shift Right 2

$(1101)_2 = 1\times2^3+1\times2^2+0\times2^1+1\times2^0 = (13)_{10} = (52)_{10} / 4$

inst_data.mif

Byte단위 Address

```
0
4
8
12...20
24
28
36
40
.
.
.
```

Word단위 Index

```
000  :   EA000006;
001  :   EAFFFFFE;
002  :   EA0000A7;
[003..005]  :   EAFFFFFE;
006  :   EA0000A4;
007  :   EAFFFFFE;
008  :   E59F2EC8;
009  :   E3A00040;
00A  :   E5820010;
00B  :   E5820014;
00C  :   E5820018;
00D  :   E582001C;
00E  :   E5820020;
00F  :   E5820024;
010  :   E3A0003F;
011  :   E5820028;
012  :   E3A00008;
013  :   E582002C;
014  :   E59F3E9C;
015  :   E59F1E9C;
016  :   E5831000;
017  :   E59F9E98;
018  :   E3A08000;
019  :   E5898000;
01A  :   E5898004;
01B  :   E5898008;
01C  :   E589800C;
01D  :   E5898010;
01E  :   E5898014;
01F  :   E5898018;
020  :   E59FDE78;
021  :   E5931200;
022  :   E3510001;
023  :   0A000000;
024  :   EAFFFFFB;
```

EA000006

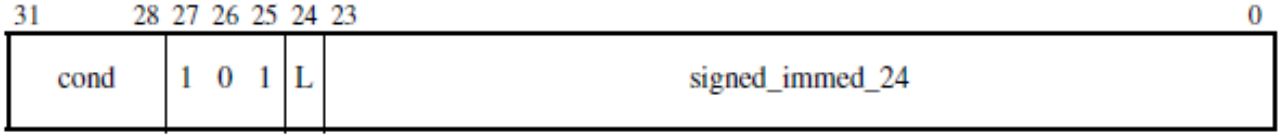
Instruction

Example

◆ EA000006

- ➢ Instruction을 Binary로 변환
  - 1110 **101**0 0000 0000 0000 0000 0000 0110 $_{(2)}$
- ➢ 어떤 Instruction인지 Reference File을 통해 확인
  - B **#6 (= 0000 0000 0000 0000 0000 0110)**;
- ➢ Instruction이 어떤 동작을 하는지 분석
  - 1. Sign-extending the 24-bit signed (two's complement) immediate to 30 bits
    - 0000 0000 0000 0000 0000 0110 -> **00 0000** 0000 0000 0000 0000 0000 0110
  - 2. Shifting the result left two bits to form a 32-bit value
    - 0000 0000 0000 0000 0000 0000 0001 1000 = 6_{10} * 4 = 24_{10}
  - Adding this to the contents of the PC, which contains the address of the branch instruction plus 8 bytes.
    - 현재 명령어의 주소 (0 * 4) + 8 에 위에서 계산한 24를 더해 32를 만듦
    - 32를 4로 나누어 Word 단위의 명령어 순서 중 8번째로 Branch함

A4.1.5 B, BL



Example

◆ EAFFFFFE

- Instruction을 Binary로 변환
  - 1110 **101**0 1111 1111 1111 1111 1111 1110 $_{(2)}$
- 어떤 Instruction인지 Reference File을 통해 확인
  - B **#-2 (= 1111 1111 1111 1111 1111 1110)**;
    (= 0000 0000 0000 0000 0000 0010의 2의 보수)
- Instruction이 어떤 동작을 하는지 분석
  - 1. Sign-extending the 24-bit signed (two's complement) immediate to 30 bits
    - 1111 1111 1111 1111 1111 1110 -> **11 1111** 1111 1111 1111 1111 1111 1110
  - 2. Shifting the result left two bits to form a 32-bit value
    - 1111 1111 1111 1111 1111 1111 11111 1000 = -2_{10} * 4 = -8_{10}
  - Adding this to the contents of the PC, which contains the address of the branch instruction plus 8 bytes.
    - 현재 명령어의 주소 (1 * 4) + 8 에 위에서 계산한 -8을 더해 4를 만듦
    - 4를 4로 나누어 Word 단위의 명령어 순서 중 1번째로 Branch함
      (= 동일한 명령어로 Branch하므로, 같은 명령어가 무한 반복 됨)