챕터1 정리

Ch1: INTRODUCTION NUMBER SYSTEMS AND CONVERSION

1.1 Digital Systems and Switching Circuits

· Digital system ←대응 관계→ Analog system

- Analog system: 현실에서 접하는 모든 물리량을 말함. 연속적임. Ex) 압력, 온도, 속도 등

  공학적 관점, 전자기기적 관점에서 Analog system은 처리하기 힘들다.

  → 따라서 Digital system으로 conversion하여 처리한다.

- Digital system: Binary Digit으로 구성됨; 0 또는 1의 이진 체계. Discrete한 특징(이산적)

  전자공학적으로 구현 및 프로세싱이 편리함. 고밀도로 집적되면 다양한 자료 표현 가능함. 컴퓨터 문제, 데이터 프로세싱, 컨트롤, 커뮤니케이션, 측정 모두 디지털 기반.

  → 신뢰성 및 정확성(Reliable)

· Binary Digit

- Binary: 0과 1, bit 단위.

  스위치의 관점에서는 1은 on, 0은 off 상태. Voltage의 관점에선 1은 high, 0은 low.

· Design

- System Design: 컴퓨터 같이 여러 가지 모듈로 구성되어 있음. 인간으로 치면 사람의 단위

  예) 컴퓨터

- Logic design: 사람으로 치면 피부, 위, 소장, 대장 같은 각종 장기들의 단위

  예) 레지스터, 메모리, 연산처리장치 등

- Circuit Design: 사람으로 치면 각각의 장기를 구성하는 세포의 단위

예) Flip-Flops, 기초 소자 등

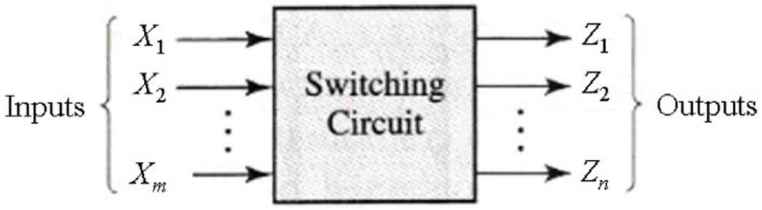
· Switching Circuit

- Switching Circuit: 논리 회로를 Switching Circuit이라고도 한다.

스위치의 on off 기반이기 때문. 크게 두 개의 카테고리로 구분함

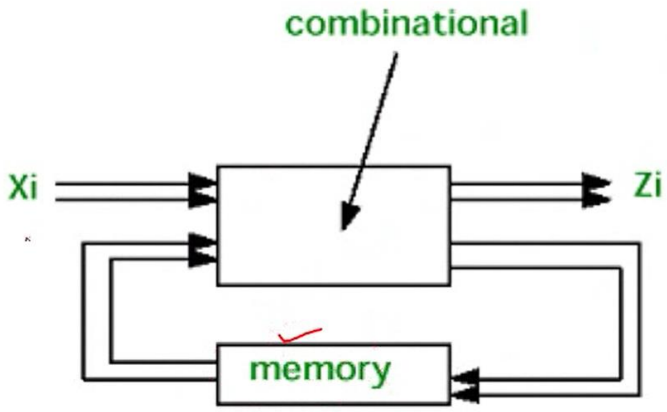
Combinational Circuit과 Sequential Circuit(중간 이후 – 메모리 요소를 포함함)

- Combinational Circuit: 출력이 현재 입력의 조합으로만 표현됨. 과거 입력과는 무관함



- Sequential Circuit: 출력이 현재의 입력뿐 아니라 과거의 입력까지 포함되어 표현됨

메모리 요소를 포함함. 메모리 요소로서는 Flip-Flops이 대표적임



1.2 Number Systems and Conversion

· Number Systems

- Number Systems: 다양한 수 체계가 존재함. 10진법, 12진법, 60진법, 2진법 등

Binary number: Base 2(혹은 radix 2) – 0, 1

Octal numbers: Base 8(radix 8) – 0, 1, 2, 3, 4, 5, 6, 7

Hexadecimal Numbers: Base 16(radix 16) - (0, 1, 2, ... , 9, A, B, C, D, E, F)

· Conversion

- Conversion

Decimal: $953.78_{10} = 9\times10^{2}+5\times10^{1}+3\times10^{0}+7\times10^{-1}+8\times10^{-2}$

Binary: $1011.11_{2} = 1\times2^{3}+0\times2^{2}+1\times2^{1}+1\times2^{0}+1\times2^{-1}+1\times2^{-2}$

$= 8+0+2+1+\frac{1}{2}+\frac{1}{4} = 11\frac{3}{4} = 11.75_{10}$

Radix(Base): $N = (a_4a_3a_2a_1a_0.a_{-1}a_{-2}a_{-3})_R$

$= a_4\times R^4 + a_3\times R^3 + a_2\times R^2 + a_1\times R^1 + a_0\times R^0$

$+ a_{-1}\times R^{-1} + a_{-2}\times R^{-2} + a_{-3}\times R^{-3}$

Example: $147.3_8 = 1\times8^2+4\times8^1+7\times8^0+3\times8^{-1} = 64+32+7+\frac{3}{8}$

$= 103.375_{10}$

Hexa-Decimal: $A2F_{16} = 10\times16^2+2\times16^1+15\times16^0 = 2560+32+15 = 2607_{10}$

- 10진수를 임의의 R진법 체계의 표현식으로 바꿀 때

Successive Division Radix Conversion 테크닉: 소수점을 제외한 부분

Step 1) Base R로 계속 나눔

Step 2) 나머지를 계속 모음

$$N = (a_n a_{n-1}\cdots a_2 a_1 a_0)_R = a_nR^n + a_{n-1}R^{n-1} + \cdots + a_2R^2 + a_1R^1 + a_0$$

$$\frac{N}{R} = a_nR^{n-1} + a_{n-1}R^{n-2} + \cdots + a_2R^1 + a_1 = Q_1, \text{ remainder } a_0$$

$$\frac{Q_1}{R} = a_nR^{n-2} + a_{n-1}R^{n-3} + \cdots + a_3R^1 + a_2 = Q_2, \text{ remainder } a_1$$

$$\frac{Q_2}{R} = a_nR^{n-3} + a_{n-1}R^{n-4} + \cdots + a_3 = Q_3, \text{ remainder } a_2$$

Successive Multiplication 테크닉: 소수점 부분

Step 1) Base R로 계속 곱함

Step 2) 정수 부분을 계속 취함

$$F = (.a_{-1}a_{-2}a_{-3}\cdots a_{-m})_R = a_{-1}R^{-1} + a_{-2}R^{-2} + a_{-3}R^{-3} + \cdots + a_{-m}R^{-m}$$

$$FR = a_{-1} + a_{-2}R^{-1} + a_{-3}R^{-2} + \cdots + a_{-m}R^{-m+1} = a_{-1} + F_1$$

$$F_1R = a_{-2} + a_{-3}R^{-1} + \cdots + a_{-m}R^{-m+2} = a_{-2} + F_2$$

$$F_2R = a_{-3} + \cdots + a_{-m}R^{-m+3} = a_{-3} + F_3$$

Example:

$F = .625$	$F_1 = .250$	$F_2 = .500$	
$\times \quad 2$	$\times \quad 2$	$\times \quad 2$	$.625_{10} = .101_2$
1.250	0.500	1.000	
$(a_{-1} = 1)$	$(a_{-2} = 0)$	$(a_{-3} = 1)$	

- Conversion of Binary to Octal, Hexa-decimal

1개의 bit은 0, 1 두가지를 표현함. 2개면 4가지, 3개면 8가지가 표현이 가능함

이 말은 즉, bit 단위를 3개를 고려한다면 8진수로, 4개는 16진수로 표현이 가능함

$$1001101.010111_2 = \underbrace{0100}_{4}\ \underbrace{1101}_{D}\ .\ \underbrace{0101}_{5}\ \underbrace{1100}_{C} = 4D.5C_{16}$$

1.3 Binary Arithmetic

· 사칙연산(Arithmetic)

- Addition

0 + 0 = 0

0 + 1 = 1

1 + 0= 1

1 + 1 = 0, and carry 1 to the next column

$$\begin{array}{rl} & 1111 \leftarrow \textbf{carries} \\ 13_{10} = & 1101 \\ 11_{10} = & \underline{1011} \\ & 11000 = 24_{10} \end{array}$$

- Subtraction

0 – 0 = 0

0 – 1 = 1, and borrow 1 from the next column

1 – 0 = 1

1 – 1 = 0

$$\begin{array}{rl} 1 \leftarrow & \text{(indicates a borrow from the 3rd column)} \\ 11101 & \\ -10011 & \\ \hline 1010 & \end{array} \qquad \begin{array}{rl} 1111 \leftarrow & \text{borrows} \\ 10000 & \\ - \quad 11 & \\ \hline 1101 & \end{array} \qquad \begin{array}{rl} 111 \leftarrow & \text{borrows} \\ 111001 & \\ - \quad 1011 & \\ \hline 101110 & \end{array}$$

- Multiplication

0 x 0 = 0

0 x 1 = 0

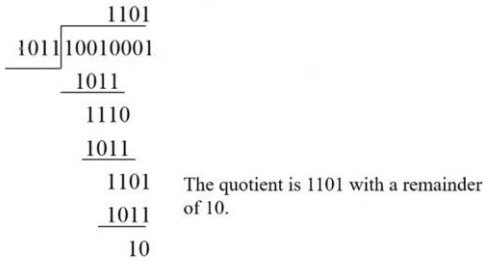
1 x 0 = 0

1 x 1 = 1

$$\begin{array}{rl} 1111 & \text{multiplicand} \\ \underline{1101} & \text{multiplier} \\ 1111 & \text{first partial product} \\ \underline{0000} & \text{second partial product} \\ (01111) & \text{sum of first two partial products} \\ \underline{1111} & \text{third partial product} \\ (1001011) & \text{sum after adding third partial product} \\ \underline{1111} & \text{fourth partial product} \\ 11000011 & \text{final product (sum after adding fourth partial product)} \end{array}$$

- Division

쓰고 빼주기 계속 반복. 10진수 나눗셈과 다를 바 없다.



1.4 Representation of Negative Numbers

· Unsigned & Signed number

- 음수의 필요성

사칙연산에서 덧셈과 뺄셈이 가장 중요한데, if 뺄셈을 덧셈으로 표현이 가능하다면?

→ 모든 사칙연산이 덧셈으로 표현 가능하므로 컴퓨터 내부에서 계산이 편하고 빨라짐

적분 같은 것을 고려했을 때 극한으로 구간을 나눈 다음에 면적의 합을 구함.

이 때 각 구간은 밑변 * 높이인데 이것 마저도 덧셈이 됨.

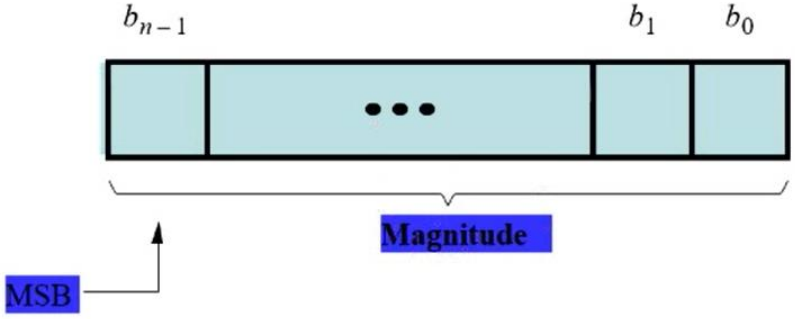
→ 그런 의미에서 음수를 어떻게 표현할 것인가? 잘 표현해야 함.

- Unsigned & Signed number

Unsigned number: 부호 없이 magnitude만 표현

제일 상위 bit인 MSB(Most Significant Bit)과

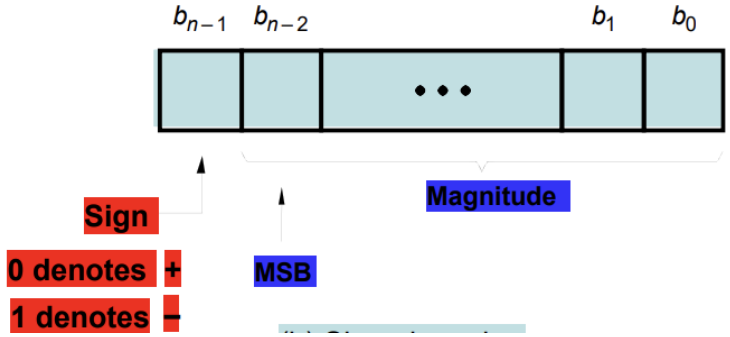
제일 하위 bit인 LSB(Least Significant Bit)



Signed number: 부호까지 고려.

제일 상위 bit이 양수/음수를 결정. Sign bit → 0이 양수, 1이 음수

MSB는 그 다음 bit



· 곱셈을 활용해서 뺄셈을 수행하기 위한 특수한 음수 표현 체계

- 도입의 이유

기본적으로 두 보수 체계를 도입한 이유는 뺄셈을 덧셈을 활용해서 수행하기 위함임

+N	Positive integers (all systems)	-N	Negative integers		
			Sign and magnitude	2's complement N^*	1's complement $\overline{N}$
+0	0000	-0	1000	-	1111
+1	0001	-1	1001	1111	1110
+2	0010	-2	1010	1110	1101
+3	0011	-3	1011	1101	1100
+4	0100	-4	1100	1100	1011
+5	0101	-5	1101	1011	1010
+6	0110	-6	1110	1010	1001
+7	0111	-7	1111	1001	1000
		-8	-	1000	-

- 2's complement representation for Negative Numbers(2의 보수 표현법)

$$N^* = 2^n - N$$

예) 음수 -5 → $2^4 - 5$ → $10000_2 - 0101_2 = 1011_2$

- 1's complement representation for Negative Numbers(1의 보수 표현법)

$$\overline{N} = (2^n - 1) - N$$

예)
$$\begin{aligned} 2^n - 1 &= 111111 \\ N &= 010101 \\ \hline \overline{N} &= 101010 \end{aligned}$$

- 두 식 사이의 관계

$$N^* = 2^n - N = (2^n - 1 - N) + 1 = \overline{N} + 1$$

→ 2's complement: 1's complement + '1'

$N = 2^n - N^*$ and $N = (2^n - 1) - \overline{N}$ ← to obtain the magnitude of the negative integer by 2's or 1's complements

→ 2의 보수나 1의 보수로 표현된 식의 magnitude만을 알고 싶다면...

· 실제로 덧셈을 활용한 뺄셈이 되는가?

- 2의 보수 체계

Case 1

+3 0011
+4 0100
+7 0111 (correct answer) → 옳게 됨

Case 2

+5 0101
+6 0110
+11 1011 ← wrong answer because of **overflow** (+11 requires 5 bits including sign)

→ 잘못됨

양수 + 양수 해서 음수가 나오는 overflow 현상이 발생. 충분한 자릿수를 쓰지 않았기 때문임. Bit의 크기를 충분히 크게 하면 해결됨.

Case 3

+5 0101
−6 1010
−1 1111 (correct answer) → 옳게 됨

Case 4

−5 1011
+6 0110
+1 (1)0001 ← correct answer when the carry from the sign bit is ignored (this is *not* an overflow)

→ Sign bit를 무시되면 옳음

Carry가 발생하지만 Sign bit를 무시하면 옳은 결과. Overflow가 아님

Case 5

−3 1101
−4 1100
−7 (1)1001 ← correct answer when the last carry is ignored (this is *not* an overflow)

→ carry 무시하면 옳음

Overflow가 아님

Case 6

−5 1011
−6 1010
(1)0101 ← wrong answer because of overflow (-11 requires 5 bits including sign) → 잘못됨

Overflow 발생. 충분한 자릿수를 쓰지 않았기 때문에 발생. 5비트 이상을 사용하면 해결

- 1의 보수 체계

Case 3

+5	0101	
−6	1001	
−1	1110	(correct answer) → 옳게 됨

Case 4

		1010	
−5		0110	
+6	(1)	0000	
		↳ 1	(end-around carry)
		0001	(correct answer, *no* overflow) → 옳게 됨

1의 보수 체계에서 발생한 carry를 첫째 자리로 옮겨서 더해주는 것을 end-around carry 라고 함

Case 5

		1100	
−3		1011	
−4	(1)	0111	
		↳ 1	(end-around carry)
		1000	(correct answer, *no* overflow) → 옳게 됨

Case 6

		1010	
−5		1001	
−6	(1)	0011	
−11		↳ 1	(end-around carry)
		0100	(wrong answer because of overflow) → 잘못됨, overflow 발생

충분한 자릿수를 사용하지 않았기 때문임. 5bit 이상을 이용하면 문제 해결

$$Case\,4: \quad -A+B \ (\text{where } B>A)$$
$$\overline{A}+B=(2^n-1-A)+B=2^n+(B-A)-1$$

$$Case\,5: \quad -A-B \ (A+B<2^{n-1})$$
$$\overline{A}+\overline{B}=(2^n-1-A)+(2^n-1-B)=2^n+[2^n-1-(A+B)]-1$$

→ end-around carry가 수학적으로 맞음을 증명

- 두 보수 체계의 차이점

2의 보수는 발생되는 carry를 무시하면 해결되고 1의 보수는 end-around carry를 사용하면 해결된다. 즉 덧셈으로 뺄셈이 가능해진다

1.5 Binary Codes

· Binary Codes

- Binary code: 문자들의 정보를 주고받기 위한 규약.

  Bit 하나로는 2개, 두개로는 4개, ..., n개로는 2^n가지가 표현 가능함.

  n가지를 표현하려면 $\log_2 n$을 올림 한 정수의 개수만큼 필요함

· Weighted Binary codes

- 8421 BCD(Binary Coded Decimal): 10진수를 2진수로 표현

- 6-3-1-1 Code

$$N = w_3a_3 + w_2a_2 + w_1a_1 + w_0a_0$$

→ 경우의 수 존재

Weighted Binary codes

- 8421 BCD
  - each decimal uses BCD conversion
  - each decimal digit to be transmitted is encoded separately

6-3-1-1 Code:

$$N = w_3a_3 + w_2a_2 + w_1a_1 + w_0a_0$$
$$N = 6\cdot1 + 3\cdot0 + 1\cdot1 + 1\cdot1 = 8$$

dec	8421 NBCD	6311
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0100
4	0100	0101
5	0101	0111
6	0110	1000
7	0111	1001
8	1000	1011
9	1001	1100

two possibilities 0110

30/34

· Error Detection Codes

- Gray Code: 인접한 숫자 사이에 오로지 one bit change만 일어나게 함

  예) 8421에서는 3과 4를 볼 때 3개가 바뀐다, 이렇게 되지 않게끔

dec	gray	even parity	
		message	p
0	0000	0000	0
1	0001	0001	1
2	0011	0010	1
3	0010	0011	0
4	0110	0100	1
5	0111	0101	0
6	0101	0110	0
7	0100	0111	1
8	1100	1000	1
9	1101	1001	0
10	1111	1010	0
11	1110	1011	1
12	1010	1100	0
13	1011	1101	1

etc

- Parity Bit: 정보를 전송할 때 묶음 단위("packet")로 보내는데, 메시지와 parity 코드를 보냄.

  이 때 메시지까지 포함하여 1의 개수가 짝수가 되도록 항상 조절함.

  이 다음에 수신자 측에서 전송 상에서 오류가 있었는지를 확인할 수 있음.

- Ex-3 Code: self-complementing code; 8421 code + 3(0011)

  자릿수마다 1과 0의 자리가 다름. (0과 9 비교, 2과 8 비교, ...)

Decimal Digit	8-4-2-1 Code (BCD)	6-3-1-1 Code	Excess-3 Code	2-out-of-5 Code	Gray Code
0	0000	0000	0011	00011	0000
1	0001	0001	0100	00101	0001
2	0010	0011	0101	00110	0011
3	0011	0100	0110	01001	0010
4	0100	0101	0111	01010	0110
5	0101	0111	1000	01100	1110
6	0110	1000	1001	10001	1010
7	0111	1001	1010	10010	1011
8	1000	1011	1011	10100	1001
9	1001	1100	1100	11000	1000

→ 937.25의 코딩

· Alphanumeric Code

- Alphanumeric Code: 문자를 규정하기 위함

- ASCII Code: American Standard Code for International Interchange (7 - bits (128))

- EBCDIC : Extended Binary Coded Decimal Interchange Code (8 - bits (256))

1010011	1110100	1100001	1110010	1110100
S	t	a	r	t