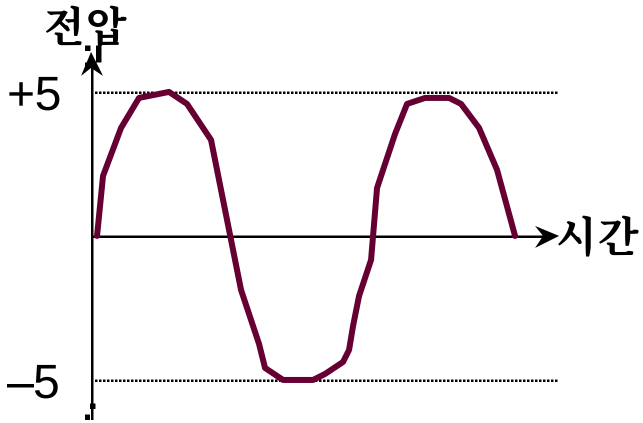
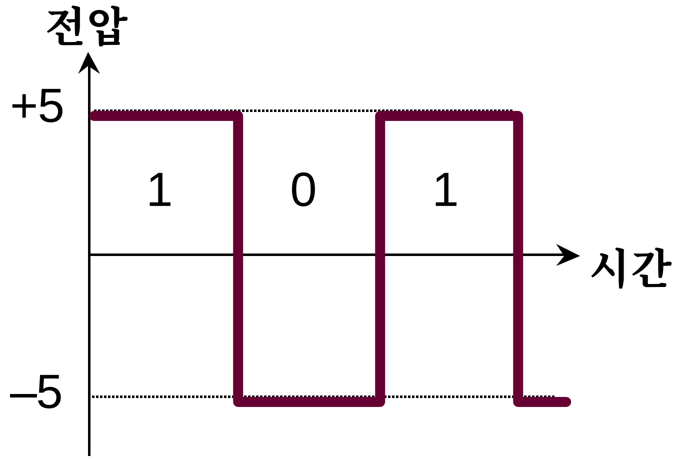
제1장 Binary System

Digital and Analog

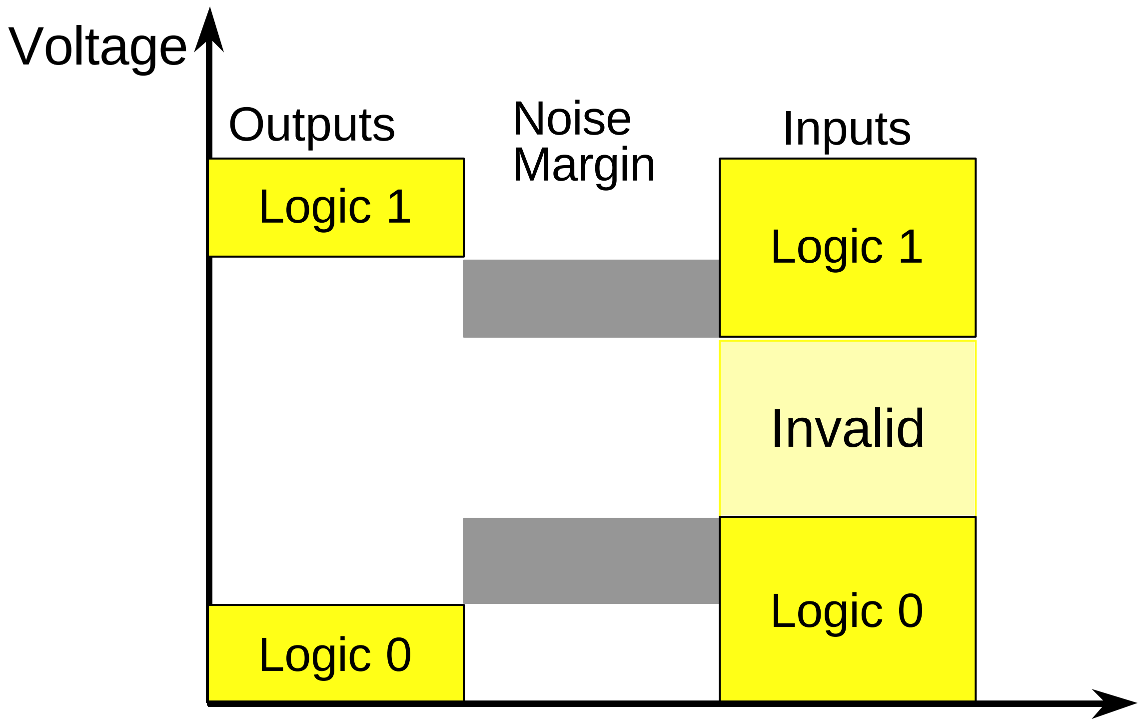
- Digital and analog
  - Analog (연속의)
    - data가 연속적으로 변화
    - 입력측의 작은 error가
    - 출력측에 큰 error를 발생
  - Digital (discrete, 불연속의, 이산의)
    - data가 불연속적인 값을 가짐
    - analog보다 정확하고 신뢰성이 높다





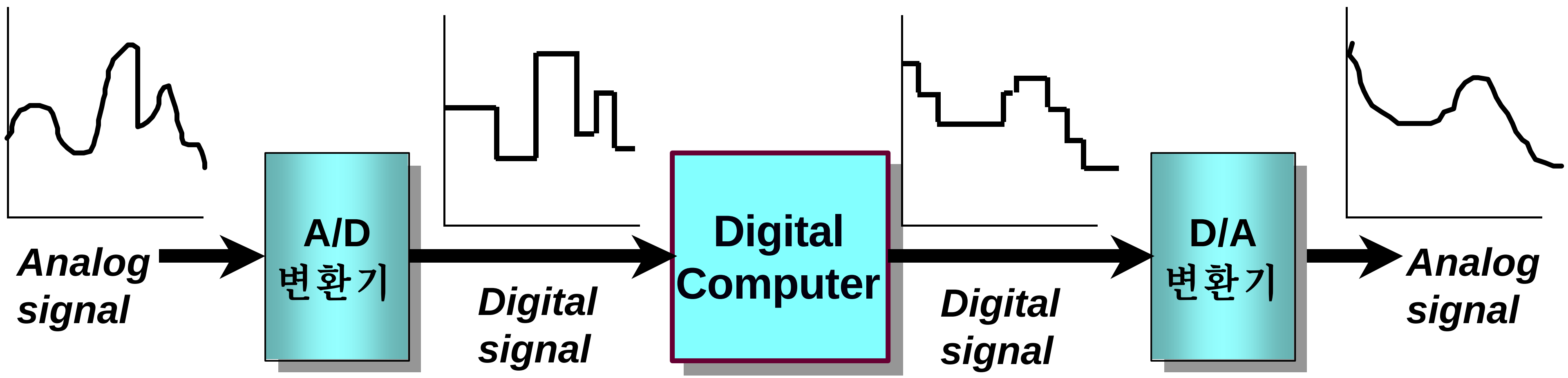
Binary System

- Digital binary system : 두개의 discrete value를 사용
  (cf) 다치논리(multi-valued logic)
  - "1", yes, 5 [V], N극, closed switch, 전류의 흐르는 상태
  - "0", no, 0 [V], S극, opened switch, 전류가 흐르지 않는 상태



Digital Computer

- 컴퓨터의 종류
  - 취급하는 자료에 따라서
    - Analog computer
    - Digital computer
  - 용도에 따라서
    - Special purpose computer
    - General purpose computer : application program
- **Digital computer**
  - Discrete information processing system
  - EDPS(Electronic data processing system)



Number System

수체계	기수 (Radix)	사용되는 숫자
2 진법(Binary)	2	0, 1
8 진법(Octal)	8	0, 1, …, 6, 7
10 진법(Decimal)	10	0, 1, …,, 7, 8, 9
16 진법(Hexadecimal)	16	0, 1, …, 8, 9, A, B, C, D, E, F

- ❑ 수의 표시

$$(a_n a_{n-1} a_{n-2} \ldots a_0 \,.\, a_{-1} a_{-2} \ldots a_{-m})_r$$

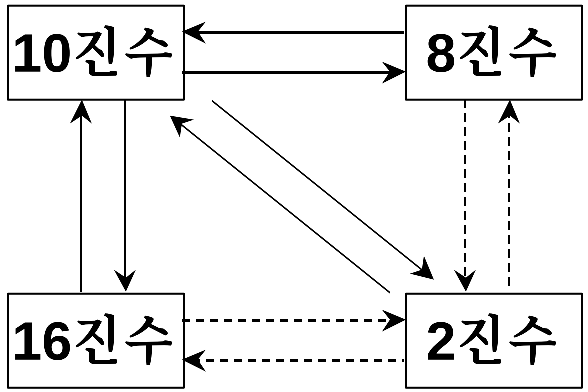
- ❑ 10진수로의 변환

$$a_n r^n + a_{n-1} r^{n-1} + a_{n-2} r^{n-2} + \cdots + a_0 r^0 + a_{-1} r^{-1} + \cdots + a_{-m} r^{-m}$$ ⇨ 십진수

진법변환 (10진수⇨r진수)

❑ 10진수 ⇨ r진수

- ◆ 정수부분 : r로나눈 나머지의 역순
- ◆ 소수부분 : 소수점이하를 r로 곱한 결과의 정수



[예] $(28.46)_{10} = (\quad ? \quad)_2$

(i) 정수부분

```
2) 28
2) 14  ······  0
2)  7  ······  0
2)  3  ······  1
    1  ······  1
```

\ $(28)_{10} = (11100)_2$

(ii) 소수부분

0.46 * 2 = 0.92 ⇨ 0
0.92 * 2 = 1.84 ⇨ 1
0.84 * 2 = 1.68 ⇨ 1
0.68 * 2 = 1.36 ⇨ 1
0.36 * 2 = 0.72 ⇨ 0
⋮

\ $(0.46)_{10} = (0.01110.....)_2$

(i), (ii)에 의해서 $(28.46)_{10} = (11100.01110........)_2$

진법변환 (r진수⇨10진수)

[예] $(259.61)_{10}$ = (? $)_7$

(i) 정수부분

7) 259
7) 37 ······ 0
5 ······ 2

\ $(259)_{10} = (520)_7$

(ii) 소수부분

0.61 * 7 = 4.27 ⇨ 4
0.27 * 7 = 1.89 ⇨ 1
0.89 * 7 = 6.23 ⇨ 6
0.23 * 7 = 1.61 ⇨ 1
⋮

\ $(0.61)_{10}$ = (0.4161..... $)_7$

(i), (ii)에 의해서 $(259.61)_{10}$ = (520.4161...... $)_7$

❑ r진수 ⇨ 10진수

$$a_n r^n + a_{n-1} r^{n-1} + a_{n-2} r^{n-2} + \cdots + a_0 r^0 + a_{-1} r^{-1} + \cdots + a_{-m} r^{-m}$$ ⇨ 십진수

[예] $(231.463)_8 = 2*8^2 + 3*8^1 + 1*8^0 + 4*8^{-1} + 6*8^{-2} + 3*8^{-3}$

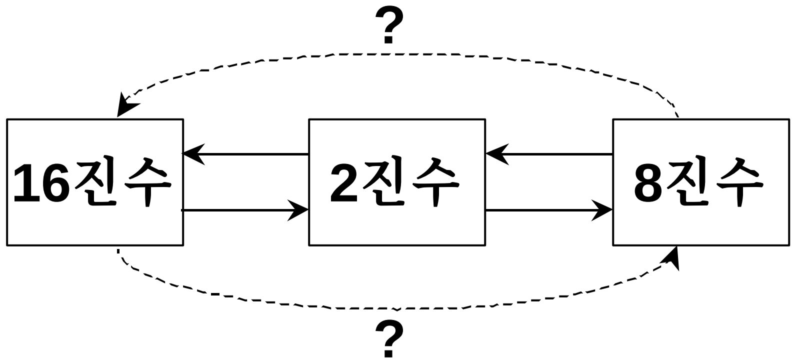
$(10111.01101)_2 = 1*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 1*2^0 + 0*2^{-1} + 1*2^{-2} + 1*2^{-3} + 0*2^{-4} + 1*2^{-5}$

$= 2^4 + 2^2 + 2^1 + 2^0 + 2^{-2} + 2^{-3} + 2^{-5}$

진법변환 (2의 지수승끼리의 변환)

❑ 소숫점을 기준으로 해서

- ◆ 2^n진수 한자리 ⇔ 2진수 n자리
- ◆ 8진수 한자리 ⇔ 2진수 3자리
- ◆ 16진수 한자리 ⇔ 2진수 4자리



[예] $(231.462)_8 = (010\ 011\ 001\ .\ 100\ 110\ 010)_2 = (\ 10\ 011\ 001\ .\ 100\ 110\ 01\ )_2$

$= (\ 10011001.10011001\ )_2$

$(10011001.10011001)_2 = (10\ 011\ 001\ .\ 100\ 110\ 01)_2 = (010\ 011\ 001\ .\ 100\ 110\ 010)_2$

$= (231.462)_8$

$(4B2F.9A2)_{16} = (0100\ 1011\ 0010\ 1111.\ 1001\ 1010\ 0010)_2$

$= (\ 100\ 1011\ 0010\ 1111.\ 1001\ 1010\ 001\ )_2$

$= (\ 1001011\ 00101111.\ 10011010001\ )_2$

$(1001011\ 00101111.\ 10011010001)_2 = (\ 100\ 1011\ 0010\ 1111.\ 1001\ 1010\ 001\ )_2$

$= (0100\ 1011\ 0010\ 1111.\ 1001\ 1010\ 0010)_2$

$= (4B2F.9A2)_{16}$

Complements

- 음수의 표현 ⇨ 뺄셈용이
- (r-1)의 보수와 r의 보수

N:양의정수
n: 정수부분의 자릿수
m:소수부분의 자릿수

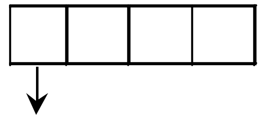
(r-1)의 보수	r 의 보수
$r^n - r^{-m} - N$	$r^n - N$ $(N \neq 0)$ 0 (N=0)
$(21.354)_{10}$의 9 의보수 $= 10^2 - 10^{-3} - 21.354$ $= 99.999 - 21.354$ $= 78.645$	$(21.354)_{10}$의 10 의보수 $= 10^2 - 21.354$ $= 100 - 21.354$ $= 78.646$
$(1101.001)_2$의 1 의보수 $= 2^4 - 2^{-3} - 1101.001$ $= 10000 - 0.001 - 1101.001$ $= 1111.111 - 1101.001$ $= 0010.110$	$(1101.001)_2$의 2 의보수 $= 2^4 - 1101.001$ $= 10000 - 1101.001$ $= 1111.111 - 1101.001$ $= 0010.111$
(r-1)-(각자리수)	[(r-1)의 보수]+ r^{-m}

- 보수를 취한후 다시 보수를 취하면 원래의 수로 복귀

정수의 표현방법

	부호와 절대치	1의 보수를 이용	2의 보수를 이용
+8	표현불가	표현불가	표현불가
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
-0	1000	1111	0000
-1	1001	1110	1111
-2	1010	1101	1110
-3	1011	1100	1101
-4	1100	1011	1100
-5	1101	1010	1011
-6	1110	1001	1010
-7	1111	1000	1001
-8	표현불가	표현불가	1000

[예] 4bit로 정수를 표현



부호bit

정수 표현 방법의 비교

	부호와 절대치	1의 보수를 이용	2의 보수를 이용
수의 표현범위	-7 ~ +7 $-(2^{n-1}-1) \sim +(2^{n-1}-1)$	-7 ~ +7 $-(2^{n-1}-1) \sim +(2^{n-1}-1)$	-8 ~ +7 $-2^{n-1} \sim +(2^{n-1}-1)$
특징	(2^n-1)개의 수 표현 Two different zero 뺄셈기가 따로 필요	(2^n-1)개의 수 표현 Two different zero 뺄셈시 최대 2회연산	2^n개의 수 표현 Only one zero 뺄셈시 1회연산

보수를 이용한 뺄셈(L-M)

- **2's complement 이용**
  - **o 2's complement of M = 2^{n+1}-M**
  - **i) L>M**
    - o L-M = L + (**2^{n+1} - M)**
    - **= 2^{n+1}**+ (L-**M) : carry 발생**
    - o Truncate carry then L-M
  - ii) L<M
    - o L-M= L+ (**2^{n+1} - M)**
    - = **2^{n+1} - (M-L)**
    - o Take 2's complement then M-L

- **1's complement 이용**
  - **o 1's complement of M = 2^{n+1}-M-1**
  - **i) L>M**
    - o L-M = L + (**2^{n+1} - M-1)**
    - **= 2^{n+1}**+ (L-**M)-1 : carry 발생**
    - o Truncate carry and add 1 then L-M
  - ii) L<M
    - o L-M= L+ (**2^{n+1} - M-1)**
    - = **2^{n+1} - (M-L)-1**
    - o Take 1's complement then M-L

보수를 이용한 뺄셈

❑ 보수를 이용한 뺄셈과정

◆ M - N ⇨ M + (N의 보수)

	(r-1)의 보수	r 의 보수	계산결과
carry 발생	최하위 bit 에 1 을 더함	carry 무시	양수
No carry	(r-1)의 보수후 (-)	r 의 보수후 (-)	음수

< 9의 보수 이용 >

```
  2397          1785
-) 1785       -) 2397

  2397          1785
+) 8214       +) 7602
 ①0611        ⓧ9387
+)   ①           ↓ 보수
  0612        ⊖0612
```

< 10의 보수 이용 >

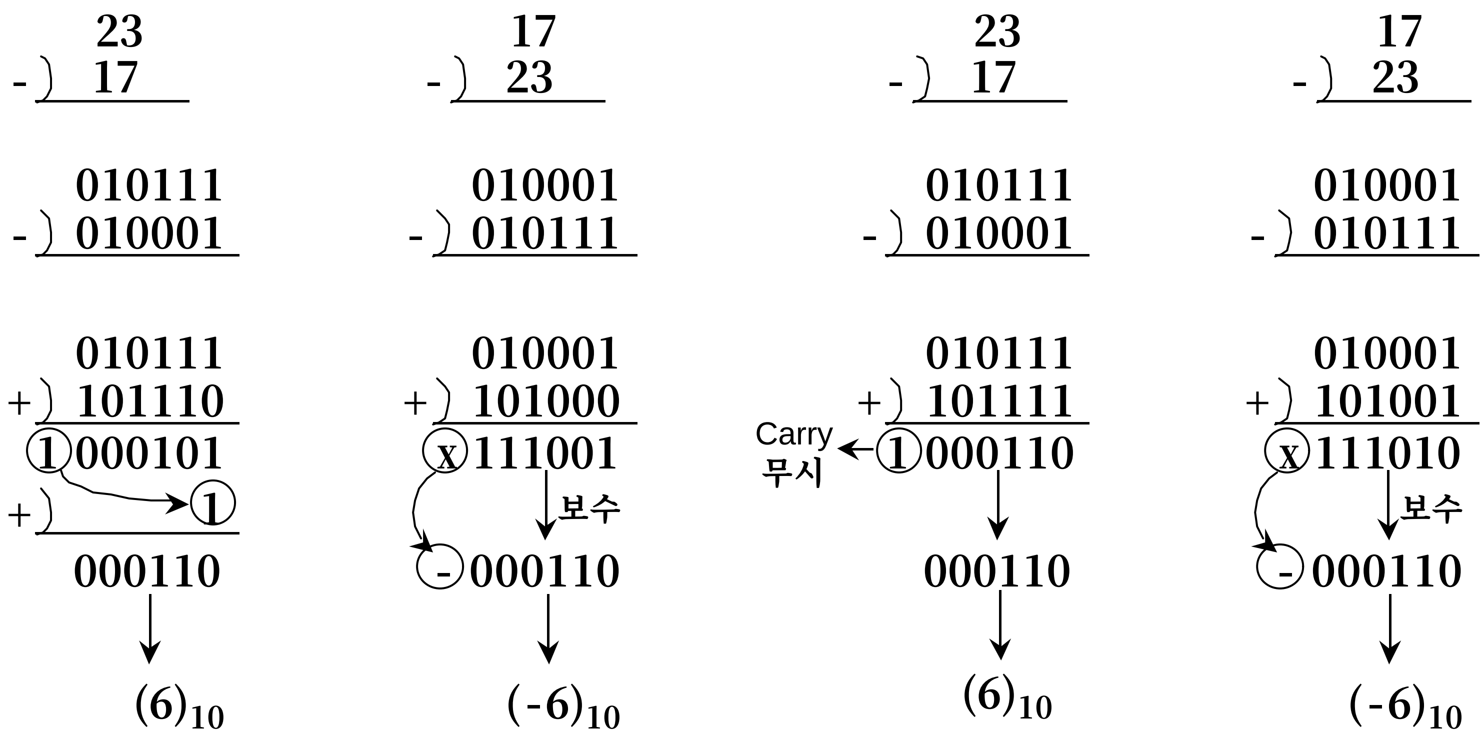
```
       2397          1785
     -) 1785       -) 2397

       2397          1785
     +) 8215       +) 7603
Carry ←①0612        ⓧ9388
무시      ↓           ↓ 보수
       0612        ⊖0612
```

보수를 이용한 뺄셈

[예] 정수를 표현하는데 6bit를 사용한다고 하면

$(23)_{10} = (010111)_2$, $(17)_{10} = (010001)_2$

< 1의 보수 이용 >   < 2의 보수 이용 >



Binary Code

- 코드의 특성에 따른 분류
  - Weighted code(가중치코드)
  - Self-complementary code(자기보수코드)
  - Error detection code(오류검출코드)
    (cf) Error correction code(오류정정코드)
- **8421코드 or BCD코드(Binary Coded Decimal:이진화십진수)**
  - 10진수 한자리를 4bit로 표현
  - Weighted code
  - $(1979)_{10}$에 대한 binary code = $(1110111011)_2$
    BCD code = $(0001\ 1001\ 0111\ 1001)_{BCD}$
- **Excess-3코드(3초과코드,3증코드)**
  - 10진수에 3을 더한 수를 4bit로 표현
  - Non-weighted code, self-complementary code
  - $(1979)_{10}$에 대한 3초과code = $(0100\ 1100\ 1010\ 1100)_{3초과}$

Error Detection

❑ **Error detection code(오류검출코드)**

◆ **오류를 정정할수는 없지만, 오류의 발생유무를 검출 가능한 코드**

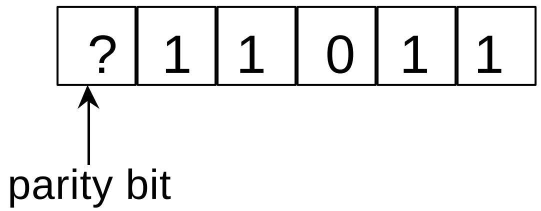
◆ 5043210 code

❑ **Parity bit를 이용한 오류검출 : 1의 갯수로 오류의 유무 판단**

◆ Odd parity : code**에 포함된 모든** 1**의 갯수가 홀수**

◆ Even parity : code**에 포함된 모든** 1**의 갯수가 짝수**

(**예**)



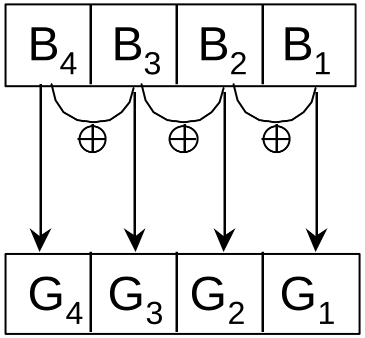
10진수에 대한 2진 코드

10 진수	8421	3 초과	84-2-1	2421	5043210
0	0000	0011	0000	0000	0100001
1	0001	0100	0111	0001	0100010
2	0010	0101	0110	0010	0100100
3	0011	0110	0101	0011	0101000
4	0100	0111	0100	0100	0110000
5	0101	1000	1011	1011	1000001
6	0110	1001	1010	1100	1000010
7	0111	1010	1001	1101	1000100
8	1000	1011	1000	1110	1001000
9	1001	1100	1111	1111	1010000

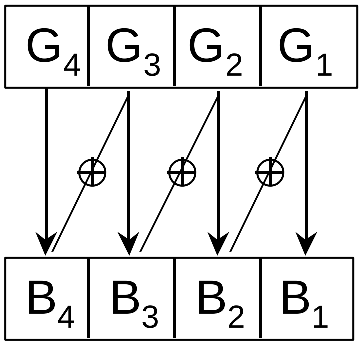
- weighted code : BCD(8421), 84-2-1, 2421, 5043210
- self-complement code : excess-3, 84-2-1, 2421
- error detection : 5043210

Gray Code (Reflected Code)

- 인접한 코드는 1bit만 다름
- Analog signal ⇨ digital signal
- Binary code ⇨ Gray code



- Gray code ⇨ Binary code



10 진수	Binary	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

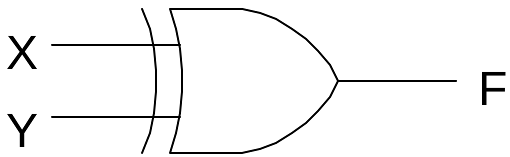
XOR gate

- 배타적논리합 (exclusive-OR)
  - 不等비교기
  - Truth table

X Y	F
0 0	0
0 1	1
1 0	1
1 1	0

$F = X \oplus Y$
$= X'Y + XY'$

  - Gate symbol



(cf) $X \oplus Y = \overline{X \odot Y}$

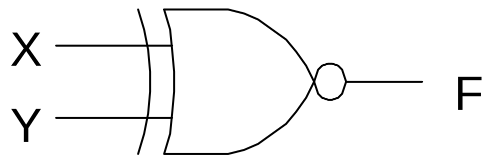
XNOR gate

- Equivalence gate
  - 等비교기
  - Truth table

X Y	F
0 0	1
0 1	0
1 0	0
1 1	1

$F = X \odot Y$
$= XY + X'Y'$

  - Gate symbol



Alphanumeric Code

- **ASCII (American Standard Code for Information Interchange)**
  - 7 bit로 128개의 문자표현

    94 graphic character : 26*2 영문자, 10 숫자,

    32특수문자(%,*,&,$,#,@,...)

    34 non-printing character : 34 제어문자(ESC, BS, NUL,......)
  - 정보교환을 위한 통신코드
- **EBCDIC (Extended Binary Coded Decimal Interchange Code)**
  - 8 bit로 문자표현
  - 문자데이타의 내부표현용
- **Hollerith code**
  - Punch card system에서 사용

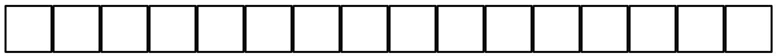
Binary Storage and Register

❑ Binary storage

- 두가지의 안정된 상태를 가지며, 1 bit의 정보를 저장
- Flip-flop, magnetic core, punch card ....

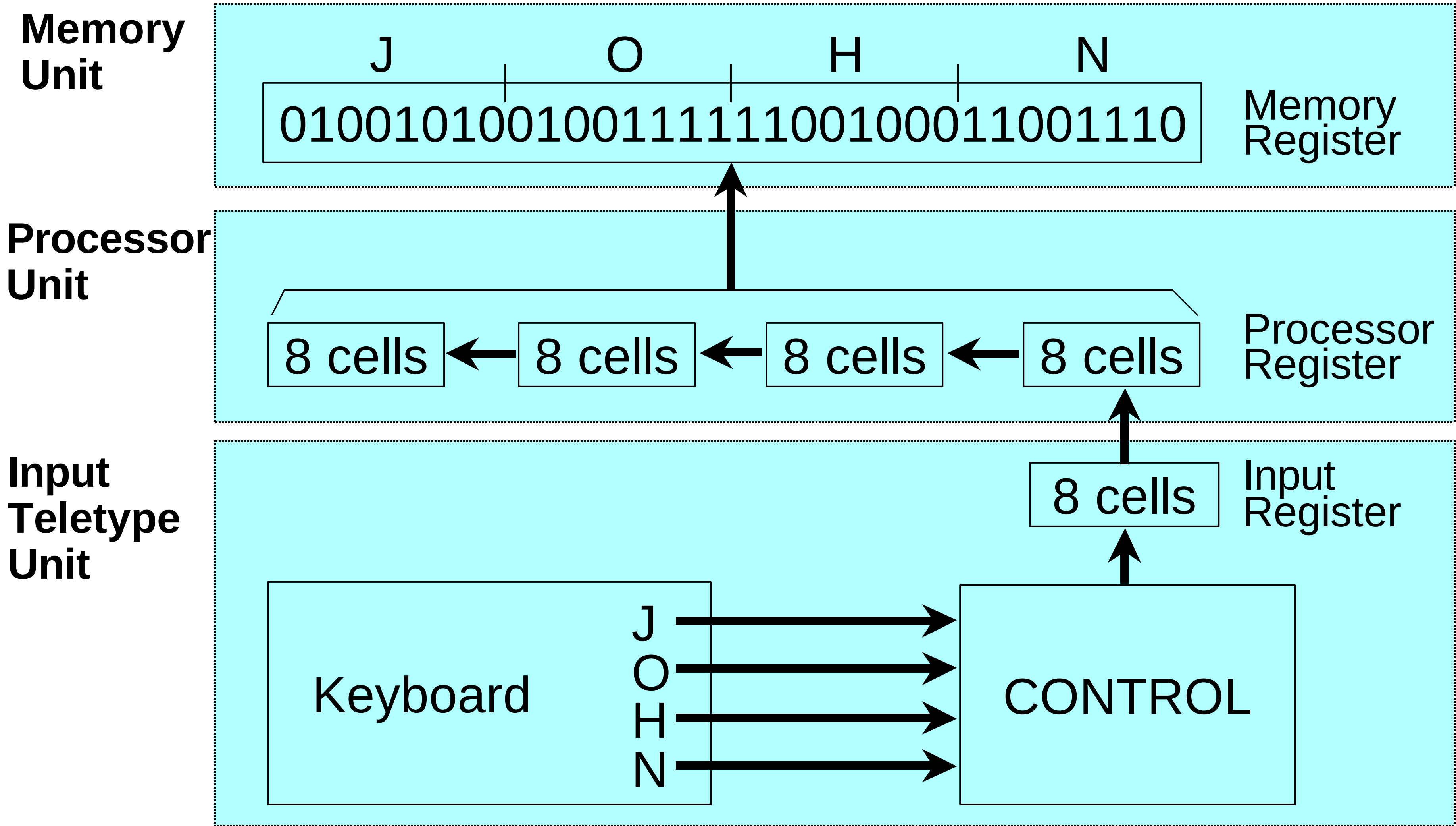
❑ Register

- Binary storage의 집합으로 n bit의 정보를 저장
- 16bit register

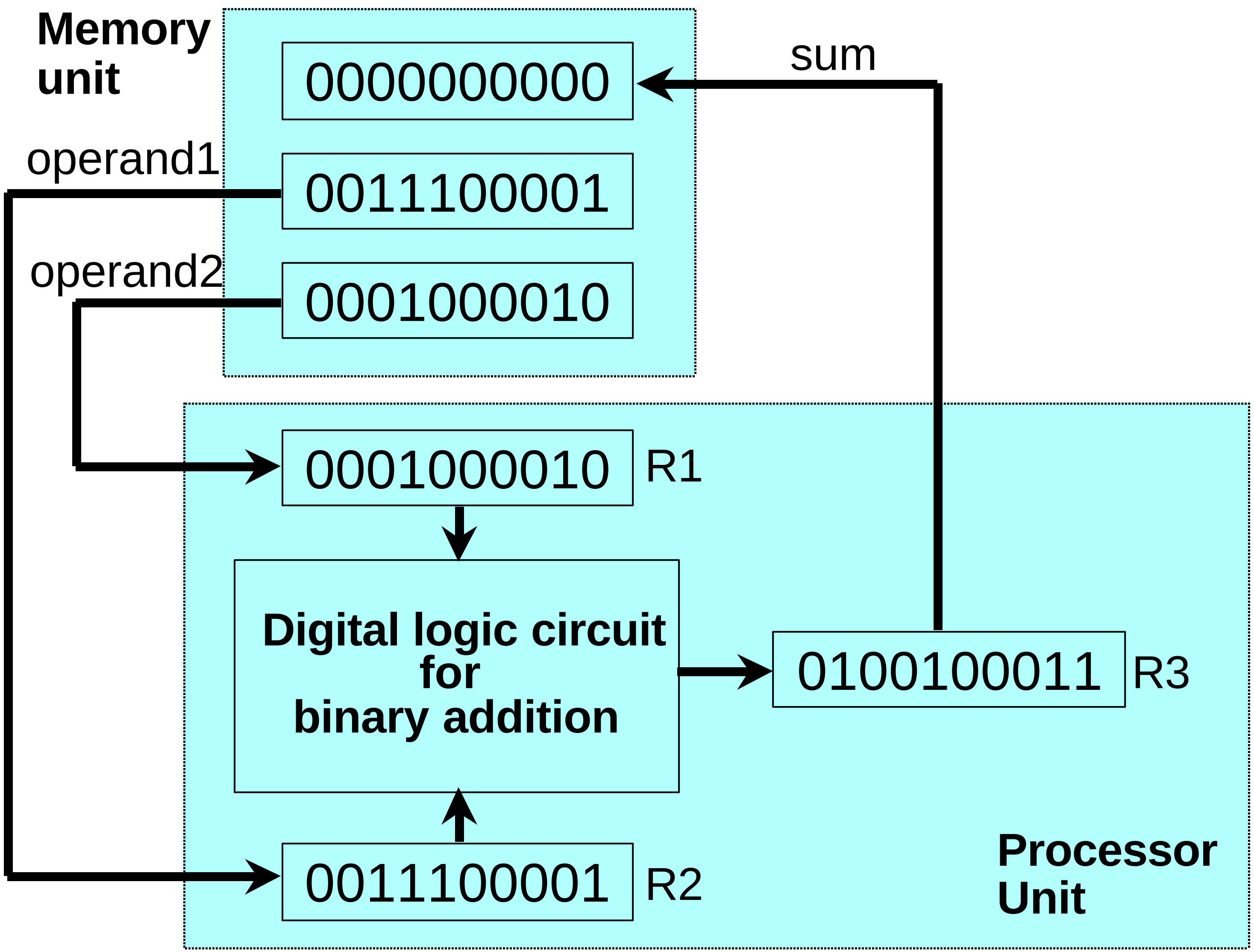


→ 2^{16}개의 정보를 표현 (0 부터 2^{16}-1 까지를 표현)

Register에 의한 정보전송



2진 정보처리의 예


Memory unit
sum
0000000000
operand1
0011100001
operand2
0001000010
0001000010
R1
Digital logic circuit for binary addition
0100100011
R3
0011100001
R2
Processor Unit

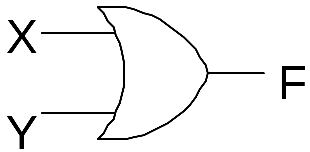
OR gate

- 논리합 (logical-OR)
  - 하나라도 1 이면 1
  - Truth table

X Y	F
0 0	0
0 1	1
1 0	1
1 1	1

$F = X + Y$

  - Gate symbol



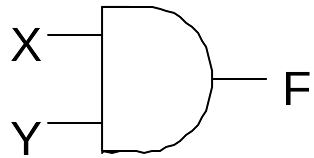
AND gate

- 논리곱 (logical-AND)
  - 모두 1 이어야만 1
  - Truth table

X Y	F
0 0	0
0 1	0
1 0	0
1 1	1

$F = X \cdot Y = XY$

  - Gate symbol



NOT gate　　Gate에 대한 입출력신호

- 논리부정(inverter)
  - 반대의 논리를 취함
  - Truth table

X	F
0	1
1	0

$F = \overline{X}$

  - Gate symbol