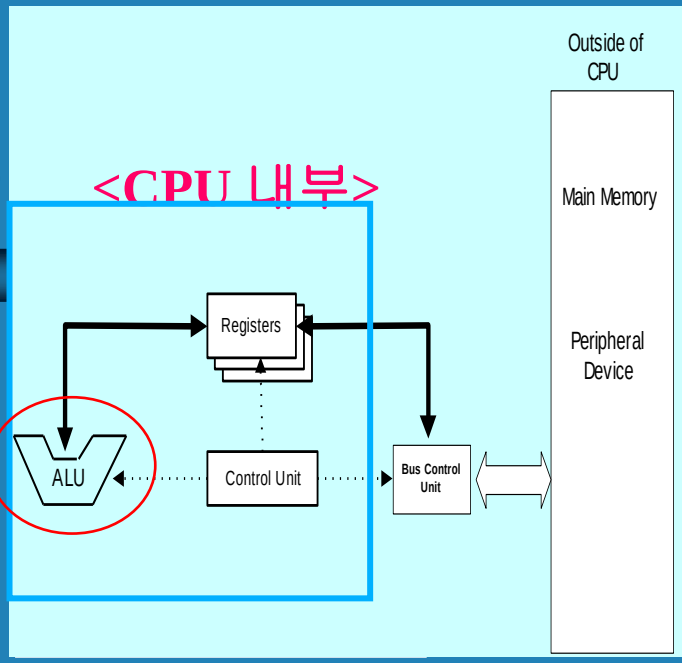
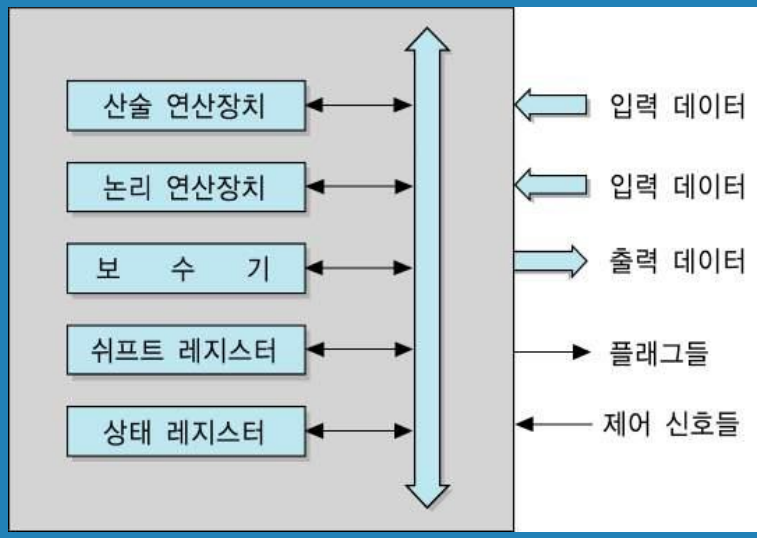
제3장 컴퓨터 산술과 논리 연산 (1-1)

- ALU의 구성요소
- Fixed Point vs. Floating Point
- 정수의 표현(Fixed Point 표현)
  - signed magnitude (부호화 크기 표현)
  - signed 1's complement (부호화 1의 보수)
  - signed 2's complement (부호화 2의 보수)

3.1 ALU의 구성요소

- 산술 연산장치
  - + - × ÷ 수행
- 논리 연산장치
  - AND, OR, XOR, NOT
- Shift register
  - bit들의 왼(오른)쪽 shift
- Complementor
  - 2의 보수(음수화)
- Status register
  - 연산 결과의 상태의 flag 저장





3.2 정수의 표현
Fixed Point vs. Floating Point

- 정수 → Binary Number System
  - 00000000 = 0
  - 00000001 = 1
  - 10000000 = 128

- 실수(Decimal: -13.625) → Binary : -1101.101
  - 부호와 소수점 저장은 어떻게 하는 가?
- Fixed Point Representation vs. Floating Point Representation
  - (고정 소수점 표현 vs. 부동 소수점 표현)

N진법 수 $a_3a_2a_1.b_1b_2b_3$를 10진법 변환

- $(a_3a_2a_1.b_1b_2b_3)_2$

$$a_3 \times n^3 + a_2 \times n^2 + a_1 \times n^1 + a_0 \times n^0 + b_1 \times n^{-1} + b_2 \times n^{-2} + b_3 \times n^{-3}$$

- 예) 이진수 $(1011.01)_2$

$$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} = 11.25$$

- 10진수를 2진수를 변환하는 방법 (**fixed point** 표현할 경우)
  - 먼저 수를 정수부와 소수부로 분리시켜 처리
  - 예) 13.75
    - 정수부 13은 2로 나눈 후 나머지 값들 처리
    - 소수부 0.75에 2를 곱하여 소수부 윗자리를 비교하여 처리

10진수 23.625를 2진수로 변환

- 정수부 23과 소수부 0.625를 분리
- 정수부 23 은 계속 2로 나누어 나머지를 구함.
  - 10111
- 소수부 0.625는 계속 2를 곱하여 소수점 윗자리를 구함.
  - 0.101
- 따라서 10진수 23.625는 2진수 10111.101

정수의 표현 (Fixed Point Representation)

- 정수는 양의 정수와 음의 정수가 있음
  - 모두 표현할 수 있는 2진 표현 필요
  - n bit로 정수 표현
    - 최상위 비트는 부호 비트,
    - 그 이하 나머지 n-1개 비트는 수의 정보
- 정수 표현 방법
  - **signed magnitude** (부호화 크기 표현)
  - **signed 1's complement** (부호화 1의 보수)
  - **signed 2's complement** (부호화 2의 보수)

제3장 컴퓨터 산술과 논리 연산 (1-2)

- 정수의 표현
- 수 표현 방식에 따른 정수의 표현 범위
- 비트의 확장
  - 작은 타입에서 큰 타입의 변수로 수를 넣을때
  - 수 표현 방식에 따른 비트의 확장 방법

정수의 표현 (Fixed Point Representation)

- 정수는 양의 정수와 음의 정수가 있음
  - 모두 표현할 수 있는 2진 표현 필요
  - n bit로 정수 표현
    - 최상위 비트는 부호 비트,
    - 그 이하 나머지 n-1개 비트는 수의 정보
- 정수 표현 방법
  - **signed magnitude** (부호화 크기 표현)
  - **signed 1's complement** (부호화 1의 보수)
  - **signed 2's complement** (부호화 2의 보수)

예제) -7을 부호를 포함하여 5비트로 표현하라.

- +7은 5비트 크기로 0111이므로
  - signed magnitude : 1 0111
  - signed 1's complement : 1 1000
  - signed 2's complement : 1 1001
- 위와 같이 8비트로 정수를 표시할때 3방식에서 각각 표현 가능한 수의 범위 표시
  - signed magnitude : $-(2^7-1) \sim +(2^7-1)$
  - signed 1's complement : $-(2^7-1) \sim +(2^7-1)$
  - signed 2's complement : $-(2^7) \sim +(2^7-1)$

8-비트 보수로 표현된 정수들

- 8-비트 2진수로 표현할 수 있는 10진수의 범위
  - 1의 보수 : $-(2^7 - 1) \sim +(2^7 - 1)$
  - 2의 보수 : $-2^7 \sim +(2^7 - 1)$

10진수	1의 보수	2의 보수
127	01111111	01111111
126	01111110	01111110
⋮	⋮	⋮
1	00000001	00000001
+0	00000000	00000000
−0	11111111	−
−1	11111110	11111111
−2	11111101	11111110
⋮	⋮	⋮
−126	10000001	10000010
−127	10000000	10000001
−128	−	10000000

3.2.2 보수 (Complement) 표현

- 음의 정수를 나타내기 위해 보수의 표현이 사용
- 2진수 체제에서는 1의 보수(1's complement)와 2의 보수(2's complement)가 있음.
  - 1의 보수 :
    - 각 자리의 수가 1이면 0으로, 0이면 1로 변환

    -9 :
  - 2의 보수 :
    - 1의 보수에서 1을 더함

    -9 :

(Signed) 2's complement 로 표현된 **10101110**을 10진수 변환 ?

- 이것을 다시 2의 보수로 표현(양수)
  - 01010010
- 10진수로 변환 시킴
  - 82
- 음 (-) 부호를 붙인다.
  - -82

2's complement 로 표현 덧셈 연산

- 두 수를 더하고, 만약 carry 발생하면 버림

(a) (+3) + (+4) = +7

(b) (−3) + (+3) = 0

(c) (−6) + (+2) = −4

(d) (−4) + (−1) = −5

3.2.3 비트 확장 (Bit Extension)

- 데이터의 길이(비트 수)를 늘리는 방법
  - 목적: 데이터를 더 많은 비트의 레지스터에 저장하거나 더 긴 데이터와의 연산 수행

[예] 8-비트 데이터를 16-비트 데이터로 확장

- **부호화-크기 표현의 경우** : 부호 비트를 맨좌측 위치로 이동시키고, 그 외의 비트들은 0으로 채운다

```
+21 =              00010101   (부호화-크기,  8 비트)
+21 =  0000000000010101   (부호화-크기, 16 비트)
-21 =              10010101   (부호화-크기,  8 비트)
-21 =  1000000000010101   (부호화-크기, 16 비트)
```

3.2.3 비트 확장 (Bit Extension)

- 부호 비트 확장(sign-bit extension)
  - 2의 보수 표현의 경우 :
  - 확장되는 상위 비트들을 부호 비트와 같은 값으로 세트

```
+21 =              00010101   (2의 보수,  8 비트)
+21 =  0000000000010101   (2의 보수, 16 비트)
-21 =              11101011   (2의 보수,  8 비트)
-21 =  1111111111101011   (2의 보수, 16 비트)
```