

Lecture 02

Number Systems

이재진

서울대학교 컴퓨터공학부

<http://aces.snu.ac.kr>

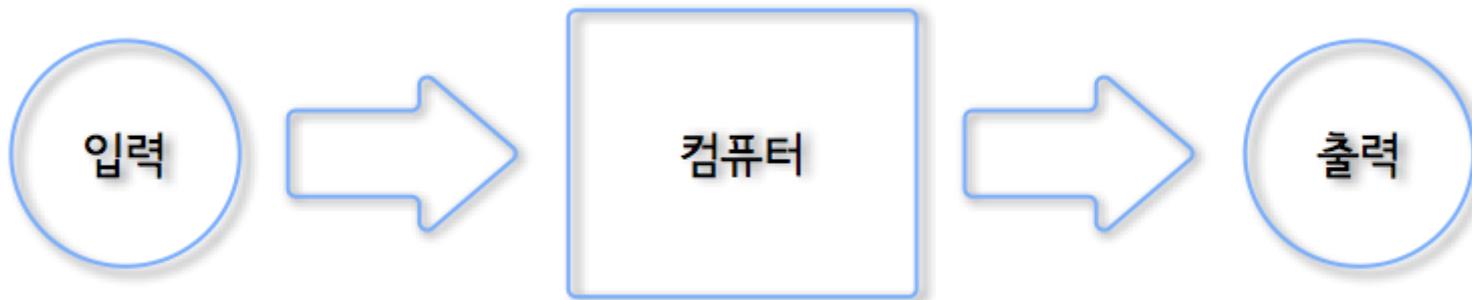


THUNDER Research Group
Seoul National University
서울대학교 천둥 연구실



컴퓨터

- 입력으로 데이터를 받아들여 저장하고, 저장한 데이터를 내장된 프로그램(program)의 지시에 따라 처리하여 그 결과를 원하는 형식으로 출력하는 전자기기
- 블랙박스로 추상화
 - 내부의 구조 및 동작방식에 관계없이 입력과 출력, 입력과 출력의 관계, 그 기기가 제공하는 기능만을 관심의 대상으로 삼음



입력, 출력, 데이터

- 입력(input)
 - 컴퓨터에 사람이거나 다른 컴퓨터, 또는 그 환경이 제공하는 정보
- 출력(output)
 - 컴퓨터가 생산한 결과물로서 텍스트, 오디오, 그래프, 그림, 이미지 등의 형식으로 표현됨
- 데이터
 - 사전적 정의
 - 추론, 논의, 계산의 기반이 되는 측정이나 통계에 의해 얻어진 사실적 정보
 - 컴퓨터공학에서 데이터의 정의
 - 컴퓨터로 처리할 수 있는 형태로 구성된 사실적 정보
 - 두 개의 상태(0과 1)로 표현되는 전기 신호로 컴퓨터에 저장됨
 - 디지털 정보



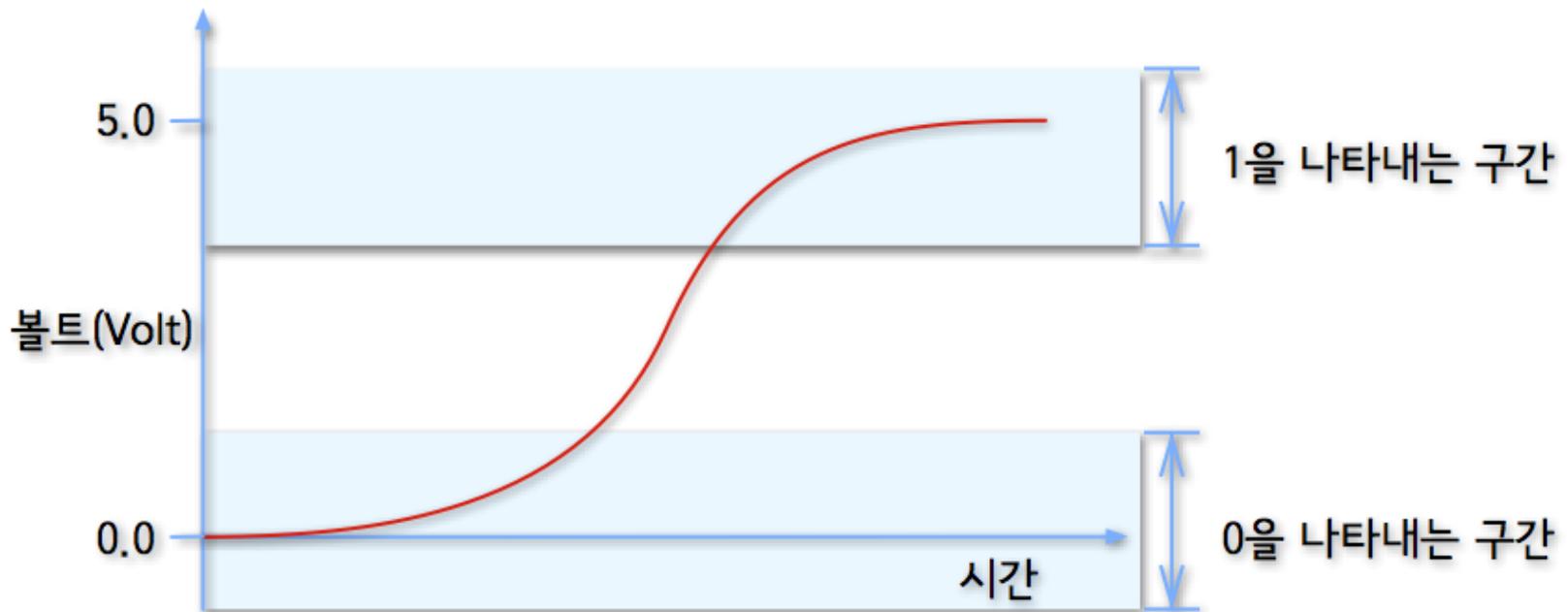
비트와 바이트

- 비트(bit)는 컴퓨터공학에서 가장 근본적인 정보의 표현 단위
 - Binary digit 의 준말
 - 두 개의 값, 0 과 1
 - 컴퓨터 내부에서 정보는 비트들의 패턴으로 표현됨
- 비트 패턴(pattern)
 - 비트로 구성된 열(sequence) 또는 스트링(string)
 - 비트 패턴의 의미는 해석에 따라 달라짐
 - 같은 비트 패턴이라도 수치, 기호, 이미지, 오디오 등으로 다르게 해석될 수 있음
- 바이트(byte) 정보 표현의 다른 단위
 - 8 개의 비트를 묶어 한 개의 바이트라고 함
 - 역사적으로 한 개의 알파벳 문자를 나타내는 비트 열을 바이트라고 함



0과 1을 나타내는 전기 신호

- 데이터는 전기신호의 형태로 저장되고 처리됨
 - 명확히 구분되는 두 가지의 서로 다른 상태 : 0과 1



하드웨어와 소프트웨어

- 하드웨어(hardware)
 - 컴퓨터 시스템의 물리적 구성요소
- 소프트웨어(software)
 - 여러 개의 프로그램으로 구성된 집합
- 프로그램(program)
 - 주어진 입력을 가지고 원하는 출력을 얻기 위해 무엇을 해야 하는지 컴퓨터에게 지시하여 컴퓨터를 동작시키는 역할을 함



순차 컴퓨터 시스템

- 순차(順次) 컴퓨터 시스템(sequential computer system)은 우리가 흔히 알고 있는 컴퓨터
- 병렬(並列) 컴퓨터 시스템(parallel computer system)과 비교하여 이야기할 때 사용하는 용어
- 추상화된 요소인 하드웨어, 운영체제, 응용 프로그램



응용 소프트웨어(application software)

- 줄여서 응용(application)으로 부르기도 함
- 사용자가 특별한 작업을 수행할 때 도움을 주는 응용 프로그램의 집합
 - Spreadsheet



시스템 소프트웨어

- 컴퓨터 하드웨어를 운용하고 응용 소프트웨어를 실행하기 위한 플랫폼(platform)
 - 운영체제, 커맨드라인 인터프리터(command-line interpreter), 윈도우 시스템(window system), 컴파일러(compiler) 및 디버거(debugger)



유틸리티 소프트웨어(utility software)

- 시스템 소프트웨어의 일종
- 컴퓨터 하드웨어와 소프트웨어를 관리하고 튜닝(tuning)할 때 이용
 - 바이러스 스캐너, 데이터 압축 유틸리티, 디스크 파티션 유틸리티, 알카이브 유틸리티(archive utility), 시스템 모니터, 텍스트 에디터, 어셈블러(assembly)



기수법(記數法)

- Number Systems
- 수를 시각적으로 표현하는 방법
 - 정해놓은 기본적인 심볼(숫자, digit)들의 조합을 사용하여 나타냄
- 정수, 소수 혹은 실수를 표현할 수 있음
- 실수 = 정수 부분 + 소수 부분
 - 소수점(radix point, '.')으로 정수부와 소수부를 구분
 - 소수부는 0 이상 1 미만의 실수
 - 예) 3.14159265
 - 정수부 : 3
 - 소수부 : 14159265



위치기반 기수법 (Positional Number System)

- 수는 숫자들을 나열한 스트링(string)을 통하여 표현
 - 각 숫자가 가지는 값은 숫자의 위치에 따른 무게값에 따라 결정됨
- 십진법 (Decimal Number System)
 - 사용하는 숫자 : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
 - 십진수 $d_{p-1}d_{p-2} \cdots d_1 d_0 . d_{-1} d_{-2} \cdots d_{-n}$ 의 값은 아래와 같이 주어짐

$$\sum_{i=-n}^{p-1} d_i \cdot 10^i$$

- 기수(base, radix) : 10
- 가장 왼쪽의 숫자(d_{p-1}) : Most Significant Digit (MSD)
- 가장 오른쪽의 숫자(d_{-n}) : Least Significant Digit (LSD)
- 예) $754.82 = 7 \times 100 + 5 \times 10 + 4 \times 1 + 8 \times 0.1 + 2 \times 0.01$



기수가 r 인 기수법

- 앞에서 다룬 10진법에서 기수를 10이 아닌 다른 수 r 로 대치하면 r -진법 체계가 됨
 - r 개의 서로 다른 숫자들을 사용
 - i 번째 위치한 숫자의 무게값은 r^i
 - r 진수 $x_{p-1}x_{p-2} \cdots x_1 x_0 . x_{-1} x_{-2} \cdots x_{-n}$ 는 아래와 같은 값을 가짐

$$\sum_{i=-n}^{p-1} x_i \cdot r^i$$

- $r \leq 10$ 일 때, 10진수에서 쓰는 숫자들 중 처음 r 개의 숫자들을 사용
- $r > 10$ 일 때, 10진수에서 쓰는 숫자들과 처음 $r - 10$ 개의 알파벳을 사용
 - 대소문자 구별 없이 사용 가능



고정소수점 표현과 부동소수점 표현

- 고정소수점 표현(fixed-point representation)
 - 소수점이 어떤 위치에 고정되어 있다고 가정
 - 정수일 경우 소수점은 LSD의 오른쪽에 비명시적으로 존재 (3., 145.)
 - 소수일 경우 소수점은 MSD의 왼쪽에 비명시적으로 존재 (.176, .997)
 - 소수점을 컴퓨터 내부에서 표시할 필요가 없음
 - r -진법은 왼쪽 p 개의 숫자와 오른쪽 n 개의 숫자 사이에 소수점의 위치를 고정
 - $x_{p-1}x_{p-2} \cdots x_0 \cdot x_{-1}x_{-2} \cdots x_{-n}$
- 부동소수점(浮動小數點, floating-point numbers) 표현
 - 소수점의 위치가 유효숫자의 개수에 따라 상대적으로 어디든 위치할 수 있음



흔히 쓰는 기수법

이름	기수	숫자
이진법	2	0, 1
팔진법	8	0, 1, 2, 3, 4, 5, 6, 7
십진법	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
십육진법	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

- 혼동을 피하기 위하여 나타낸 수의 오른쪽 끝에 붙은 첨자를 이용하여 사용한 기수법을 표현
 - 예) 101
 - 2진수 : 101_2
 - 8진수 : 101_8
 - 10진수 : 101_{10}
 - 16진수 : 101_{16}



이진법

- Boole 논리, 컴퓨터와 밀접한 연관이 있음
- 컴퓨터는 내부적으로 수치 데이터를 이진수로 나타냄
- 사용하는 숫자(binary digits, bits) : 0, 1
- 이진수 $b_{p-1}b_{p-2} \cdots b_1b_0 \cdot b_{-1} b_{-2} \cdots b_{-n}$ 의 값은 아래와 같이 주어짐

$$\sum_{i=-n}^{p-1} b_i \cdot 2^i$$

- 기수 : 2
- 가장 왼쪽의 숫자(b_{p-1}) : Most Significant Bit (MSB)
- 가장 오른쪽의 숫자(b_{-n}) : Least Significant Bt (LSB)
- 예)
 - $10011 = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 19$
 - $101.001 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} = 5.125$



팔진법

- Boolean 논리, 컴퓨터와 밀접한 연관이 있음
- 사용하는 숫자(octal digits) : 0, 1, 2, 3, 4, 5, 6, 7
- 팔진수 (octal number) $x_{p-1}x_{p-2} \cdots x_1 x_0 . x_{-1} x_{-2} \cdots x_{-n}$ 는 아래와 같은 값을 가짐

$$\sum_{i=-n}^{p-1} x_i \cdot 8^i$$

- 기수 : 8
- 가장 왼쪽의 숫자(x_{p-1}) : Most Significant Digit (MSD)
- 가장 오른쪽의 숫자(x_{-n}) : Least Significant Digit (LSD)
- 예)
 - $731 = 7 \cdot 8^2 + 3 \cdot 8^1 + 1 \cdot 8^0 = 473$
 - $731.462 = 7 \cdot 8^2 + 3 \cdot 8^1 + 1 \cdot 8^0 + 4 \cdot 8^{-1} + 6 \cdot 8^{-2} + 2 \cdot 8^{-3} = 473.59765625$



십육진법

- 사용하는 숫자(hexadecimal digits) : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
- 십육진수(hexadecimal number) $x_{p-1}x_{p-2} \cdots x_1 x_0 \cdot x_{-1} x_{-2} \cdots x_{-n}$ 는 아래와 같은 값을 가짐

$$\sum_{i=-n}^{p-1} x_i \cdot 16^i$$

- 기수 : 16
- 가장 왼쪽의 숫자(x_{p-1}) : Most Significant Digit (MSD)
- 가장 오른쪽의 숫자(x_{-n}) : Least Significant Digit (LSD)
- C 언어 등에서 십육진법으로 표현한 수 앞에 0x를 붙여 십육진수임을 나타내기도 함
- 예)
 - $0x3AE = 3 \cdot 16^2 + 10 \cdot 16^1 + 14 \cdot 16^0 = 798$
 - $3AE.4C = 3 \cdot 16^2 + 10 \cdot 16^1 + 14 \cdot 16^0 + 4 \cdot 16^{-1} + 12 \cdot 16^{-2} = 798.296875$



4-bit 이진수, 팔진수, 십진수, 십육진수

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



이진수에서 팔진수로

$$100011001110_2 = 100\ 011\ 001\ 110_2 = 4316_8$$

$$10.1011001011_2 = 010 . 101\ 100\ 101\ 100_2 = 2.5454_8$$



이진수에서 십육진수로

$$100011001110_2 = 1000 \ 1100 \ 1110_2 = 8CE_{16}$$

$$10.1011001011_2 = 0010 \ . \ 1011 \ 0010 \ 1100_2 = 2.B2C_{16}$$



십진 정수를 이진수로 변환하는 방법

- 십진 정수 N 을 이진수로 변환하여 이진 정수 $b_{p-1}b_{p-2} \cdots b_1b_0$ 를 얻는 알고리즘
 1. i 를 0으로 설정
 2. N 을 2로 나누어 몫과 나머지를 얻음
 3. b_i 를 나머지 값으로, N 을 몫의 값으로 설정
 4. N 이 0 이 아니면 i 를 1 증가 시킨 후 2 단계부터 다시 반복
- 예) 십진수 108을 이진수로 변환

i	N	b_i
0	$108/2 = 54$	0 (LSB)
1	$54/2 = 27$	0
2	$27/2 = 13$	1
3	$13/2 = 6$	1
4	$6/2 = 3$	0
5	$3/2 = 1$	1
6	$1/2 = 0$	1 (MSB)

십진 소수를 이진수로 변환하는 방법

- 십진 소수 N 을 이진수로 변환하여 이진 소수 $b_{-1}b_{-2}\cdots b_{-n}$ 를 얻는 알고리즘
 1. i 를 1 로 설정
 2. N 에 2를 곱하여 그 결과 값을 N 의 값으로 설정
 3. N 이 1보다 작으면 b_{-i} 를 0 으로 설정하고, 아니라면 b_{-i} 를 1 로 설정한 후 N 을 1 감소시킴
 4. i 가 전체 자릿수 n 보다 작으면 i 를 1 증가시킨 후 2 단계부터 다시 반복
- 예) 십진수 0.735를 네자리 이진소수 0.1011로 변환

i	N	b_{-i}
1	$0.735 \times 2 = 1.47$	1
2	$0.47 \times 2 = 0.94$	0
3	$0.94 \times 2 = 1.88$	1
4	$0.88 \times 2 = 1.76$	1



십진수를 이진수로 변환하기

- 정수부와 소수부를 구분하여 이진수로 변환
- 변환 이후, 정수부 변환 결과와 소수부 변환 결과를 소수점과 조합하여 표기
- 예) 108.735_{10} 는 십진정수의 변환과 십진소수의 변환 결과를 조합하여 1101100.1011_2 로 변환됨



십진수를 팔진수로 변환하기

- 십진수를 이진수로 변환하는 과정과 거의 같음
- 예) 십진수 108.735를 팔진수로 변환

i	N	b_i
0	$108/8 = 13$	4 (<i>LSB</i>)
1	$13/8 = 1$	5
2	$1/8 = 0$	1 (<i>MSB</i>)

i	N	b_{-i}
1	$0.735 \times 8 = 5.88$	5
2	$0.88 \times 8 = 7.04$	7
3	$0.04 \times 8 = 0.32$	0
4	$0.32 \times 8 = 2.56$	2



십진수를 십육진수로 변환하기

- 십진수를 이진수로 변환하는 과정과 거의 같음
- 예) 십진수 108.735를 십육진수로 변환

i	N	b_i
0	$108/16 = 6$	12(<i>C</i>) (<i>LSB</i>)
1	$6/16 = 0$	6 (<i>MSB</i>)

i	N	b_{-i}
1	$0.735 \times 16 = 11.76$	11(<i>B</i>)
2	$0.76 \times 16 = 12.16$	12(<i>C</i>)
3	$0.16 \times 16 = 2.56$	2
4	$0.56 \times 16 = 8.96$	8



십진수를 변환하는 다른 방법

- 주어진 십진수를 이진수로 변환
- 이진수로 변환된 수를 다시 팔진수, 혹은 십육진수로 변환
- 예)
 - $108.731_{10} = 001\ 101\ 100.101\ 111\ 000\ 010_2 = 154.5702_8$
 - $108.731_{10} = 0110\ 1100.1011\ 1100\ 0010\ 1000_2 = 6C.BC28_8$



부호가 있거나 없는 이진수

- Unsigned/signed binary numbers
- n 비트로 2^n 개의 서로 다른 이진수를 표현 가능
- 부호가 없는(unsigned) 이진수
 - 0을 포함한 2^n 개의 양수
- 부호가 있는(signed) 이진수
 - 음수와 0, 양수를 모두 나타낼 수 있음
 - MSB를 부호(sign) 비트로 사용하고, 나머지 비트들은 수의 크기를 나타냄
- 컴퓨터 하드웨어를 최대한 간단히 구현하기 위하여 2의 보수 표현을 많이 사용함
 - 거의 모든 컴퓨터시스템에서 2의 보수 표현을 사용



부호 없는 수(unsigned number)

- 이진수가 0이나 양의 값을 나타낼 때
- n -비트 부호 없는 이진 정수의 값 x 의 범위
 - $0 \leq x \leq 2^n - 1$



부호 붙은 수(signed number)

- n 개의 비트를 이용하여 0과 양수뿐만 아니라 음수도 인코딩할 수 있음
- 부호 붙은 수를 n 개의 비트로 인코딩하는 방법
 - 부호 붙은 크기 표현(signed magnitude representation)
 - 0과 양수의 경우 MSB를 0으로, 음수의 경우 MSB를 1로 설정하고 나머지 비트가 그 수의 크기를 나타냄
 - 1의 보수 표현
 - 2의 보수 표현
 - 하드웨어의 구현이 간단함
 - 대부분의 컴퓨터가 사용 중



보수

- r -진법으로 표현된 n 개의 자리를 가진 수 x 의 r 의 보수(補數, radix complement, complement) \bar{x} 는,

- $$\bar{x} = \begin{cases} r^n - x, & x \neq 0 \\ 0, & x = 0 \end{cases}$$

- x 의 $(r - 1)$ 의 보수(diminished radix complement) x' 은,

- $$x' = (r^n - 1) - x$$



보수 구하기

- x 의 $(r - 1)$ 의 보수는 x 의 각 자리에 있는 숫자의 $(r - 1)$ 의 보수를 취하면 됨
 - 각 자리의 숫자를 $(r - 1)$ 에서 빼는 것
 - $(r^n - 1) - x = (r^n - 1^n) - x = ((r - 1) \cdot r^{n-1} + (r - 1) \cdot r^{n-2} + \dots + (r - 1) \cdot r^0) - x$
 - 예) 십진수 836의 $(10 - 1)$ 의 보수는 $(10^3 - 1) - 836 = 999 - 836 = 163$
- x 가 0 이 아닌 경우, x 의 $(r - 1)$ 의 보수에 1을 더하면 x 의 r 의 보수를 얻을
 - 예) 세자리 십진수 836의 $(10 - 1)$ 의 보수는 163 이고 10의 보수는 $10^3 - 836 = 1000 - 836 = 164$



1의 보수 표현(1's complement representation)

- $r = 2$ 일 경우에 $(r - 1)$ 의 보수를 사용하여 음수를 표현
- 예) 6_{10} 의 1의 보수는 $(2^4 - 1) - 6 = 15 - 6 = 9_{10}$
 - 9_{10} 는 1001_2 이고 1의 보수 표현에서 -6_{10} 을 나타냄
- n -비트 1의 보수 표현으로 나타낼 수 있는 가장 큰 정수는 $2^{n-1} - 1$
이고 가장 작은 정수는 $-2^{n-1} + 1$
- MSB는 부호 비트(sign bit)
 - 1 이면 음수, 0이면 양수
- 0을 표현하는 방법이 두 가지인 단점
 - 0000 (+0)
 - 1111 (-0)



2의 보수 표현(2's complement representation)

- 현재 대부분의 컴퓨터가 사용 중
- n -비트 이진수로 표현되는 정수 x 의 2의 보수 \bar{x}
 - $\bar{x} = \begin{cases} 2^n - x, & x \neq 0 \\ 0, & x = 0 \end{cases}$
 - 가장 큰 수는 $2^{n-1} - 1$ 이고 가장 작은 수는 -2^{n-1}
- 먼저 1의 보수를 구하고 1을 더하면 됨
 - 가장 오른쪽에 나오는 1과 뒤따라 나오는 0을 제외한 모든 비트들을 뒤집는(flip) 것과 같음
 - 예) 6_{10} 의 2의 보수는 $10_{10}(1010_2)$ 이고, 1010_2 이 2의 보수 표현에서 -6_{10} 을 나타냄
- MSB는 부호 비트



모듈라 연산

- Modular arithmetic 또는 clock arithmetic
- 모듈로 연산(modulo operation)을 이용
 - 유클리드 나눗셈에서 나머지를 구하는 연산
 - mod로 표기
- 유클리드 알고리즘(Euclidean algorithm)
 - 유클리드 나눗셈(Euclidean division)이라고도 불림
 - 두 정수 m 과 $n(\neq 0)$ 이 주어졌을 때 $m = qn + r$ 과 $0 \leq r < |n|$ 을 만족하는 유일한 정수 q 와 r 이 항상 존재
 - q 는 m 을 n 으로 나눈 몫이고 r 은 나머지



우리가 잘 알고 있는 나눗셈

- 음수의 경우 유클리드 나눗셈과 다름
- 우리가 알고 있는 나눗셈에서 -7 을 3 으로 나누면 몫이 -2 이고 나머지가 -1
- 유클리드 나눗셈은 -7 을 3 으로 나누면 몫이 -3 이고 나머지가 1



mod

- 두 정수 x 와 $m(\neq 0)$ 이 주어질 때 x/m 의 몫 q 는 $\lfloor x/m \rfloor$ 으로 나타낼 수 있고, 그 나머지 r 을 나타내는 연산이 mod
 - $q = \left\lfloor \frac{x}{m} \right\rfloor, m \neq 0$
 - $r = x \bmod m = x - mq, m \neq 0$
 - m 은 modulus라고 불림



합동관계(合同關係, congruence relation)

- $x \bmod m = y \bmod m$
 - x 와 y 의 관계를 $x \equiv y \pmod{m}$ 이라 표기
 - x 와 y 는 모듈로 m 에 대하여 합동
 - x 와 y 가 m 으로 나누었을 때 유클리드 나눗셈의 나머지가 같음



모듈로- m 연산(modulo- m operation)

- 모듈러스가 m 인 합동관계를 이용한 연산
- m 개의 수 $0, 1, 2, \dots, m - 1$ 을 연산에 이용
- 예) 모듈로-8 덧셈
 - $1 + 4 \equiv 5 \pmod{8}$
 - $8 + 5 \equiv 5 \pmod{8}$
 - $7 + 3 \equiv 2 \pmod{8}$
 - $0 + 8 \equiv 0 \pmod{8}$

