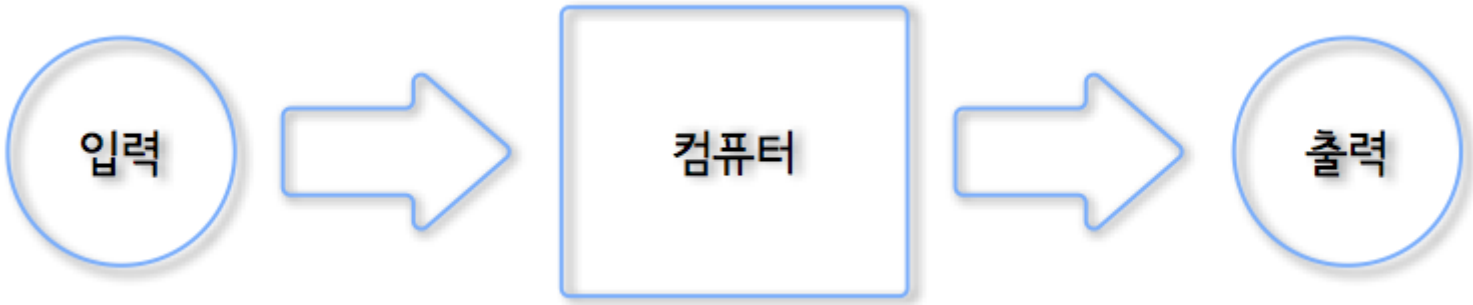
Lecture 02
컴퓨터와 기수법

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

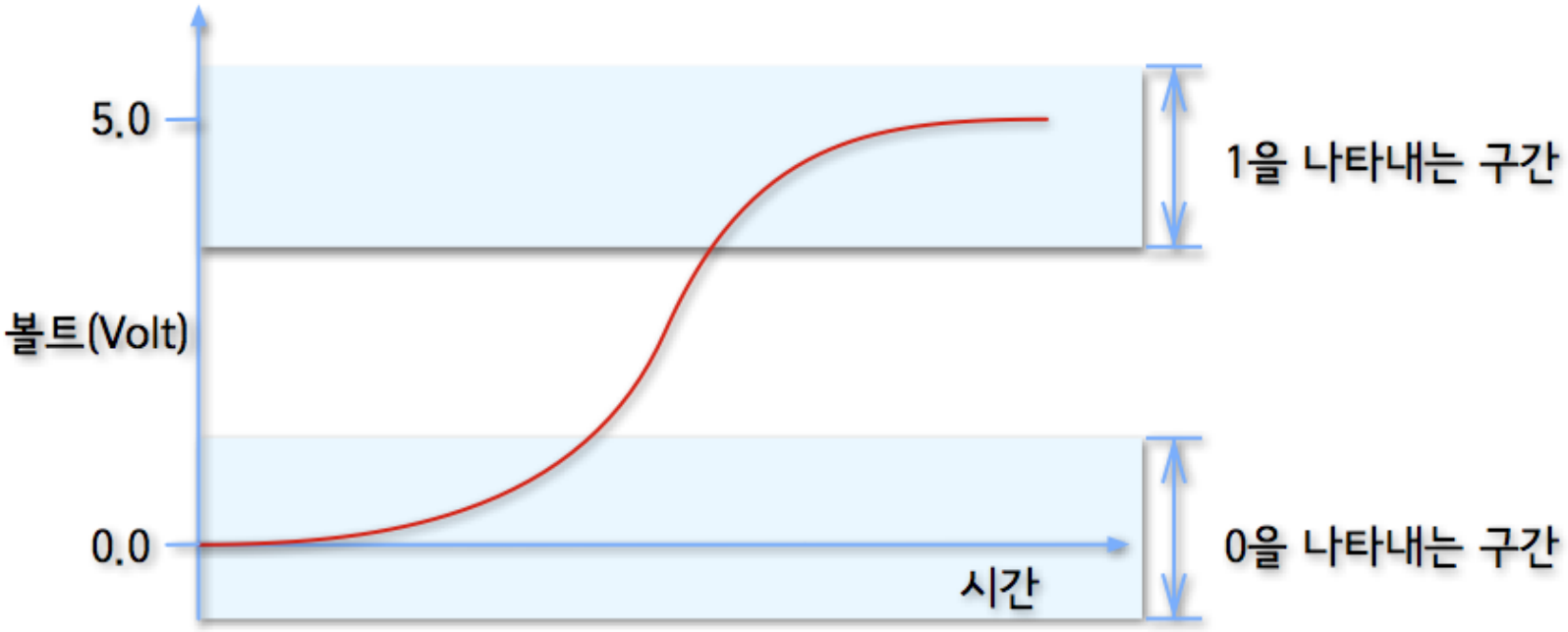


데이터

- 사전적 정의
  - 추론, 논의, 계산의 기반이 되는 측정이나 통계에 의해 얻어진 사실적 정보
- 컴퓨터공학에서 데이터의 정의
  - 컴퓨터로 처리할 수 있는 형태로 구성된 사실적 정보
- 비트(bit)
  - 이진(二陣) 숫자를 의미하는 binary digit의 준말
  - 정보의 단위로 0이나 1을 나타냄
  - 컴퓨터 내부에서 정보는 비트들의 열인 비트 패턴으로 인코딩됨
  - 비트패턴의 의미는 해석에 따라 다름
- 바이트(byte)
  - 정보의 단위로 8 개의 비트로 구성된 연속된 비트의 열

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(竝列) 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(assembler)

r-진법

- 위치기반 기수법(positional number system)
  - 수는 숫자들을 연이어 나열한 것으로 표현하고, 그 표현이 가지는 값은 각 숫자가 가지는 값을 더한 것
  - 각 숫자의 값은 숫자의 위치에 따른 무게 값(weight)에 따라 결정됨
- $x_{p-1}x_{p-2}\cdots x_0 . x_{-1}x_{-2}\cdots x_{-n}$로 표현된 수의 값 x
  - $x = \sum_{i=-n}^{p-1} x_i \cdot r^i$
  - r : 기수(基數)
  - 위치 i의 무게 값 : r^i

흔히 사용하는 기수법의 기수와 숫자

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

- r-진법으로 나타낸 수 N의 기수를 문맥으로 추측이 곤란한 경우는 보통 아래첨자 r을 N의 뒷부분에 붙여서 혼동을 피함
  - 예) 101_2: 이진수 101
  - 예) 101_{10} : 십진수 101

고정소수점 표현

- Fixed-point representation
- 소수점이 어떤 위치에 고정되어 있다고 가정
- 소수점을 컴퓨터 내부에서 표시할 필요가 없음
- r-진법은 왼쪽 p 개의 숫자와 오른쪽 n 개의 숫자 사이에 소수점의 위치를 고정
  - $x_{p-1}x_{p-2}\cdots x_0 \,.\, x_{-1}x_{-2}\cdots x_{-n}$

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 정수 N을 이진 정수로 변환하는 방법

$i \leftarrow 0$
repeat
  $N \leftarrow N$ 을 2로 나눈 몫
  $b_i \leftarrow N$을 2로 나눈 나머지
  $i \leftarrow i + 1$
until $(N = 0)$

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 소수 N을 이진 소수로 변환하는 방법

- 소수점 아래 n 자리까지

```
i ← −1
while (−1×i ≤ n) do
    N ← 2×N
    if (N < 1) then
        b_i ← 0
    else
        b_i ← 1
    end if
    N ← N − 1
    i ← i − 1
end while
```

i	N	b_i
–1	0.735 x 2 = 1.47	1
–2	0.47 x 2 = 0.94	0
–3	0.94 x 2 = 1.88	1
–4	0.88 x 2 = 1.76	1

정수부와 소수부가 조합되어 있는 경우

- 각 부분을 따로 변환한 다음 그 결과를 다시 조합

- 예) 108.735_{10}는 십진정수의 변환과 십진소수의 변환 결과를 조합하여 1101100.1011_2로 변환됨

십진수를 팔진수 또는 십육진수로 변환하기

- 십진수를 이진수로 변환하는 것과 비슷함
  - 정수 : 연속된 나눗셈
  - 소수 : 연속된 곱셈
- 십진수를 이진수로 변환한 후, 이진수를 팔진수와 십육진수로 변환

$$108.735_{10} = 001\ 101\ 100\ .\ 101\ 111\ 000\ 010_2 = 154.5702_8$$

$$108.735_{10} = 0110\ 1100\ .\ 1011\ 1100\ 0010\ 1000_2 = 6C.BC28_{16}$$

	i	N	b_i
정수부	0	108 / 16 = 6	C
	1	6 / 16 = 0	6
	i	N	b_{-i}
소수부	-1	0.735 x 16 = 11.76	B
	-2	0.76 x 16 = 12.16	C
	-3	0.16 x 16 = 2.56	2
	-4	0.56 x 16 = 8.96	8

부호 없는 수(unsigned number)

- 이진수가 0이나 양의 값을 나타낼 때
- n-비트 부호 없는 이진 정수의 값 x의 범위
  - $0 \leq x \leq 2^n - 1$
- $x_{n-1}x_{n-2}\cdots x_0$을 부호 없는 이진수로 해석하고 그 값을 대응시키는 함수
  - $f_{B2U}^n(x_{n-1}, x_{n-2}, \cdots, x_0) : B^n \dashrightarrow N_n$
    - $N_n = \{0, 1, 2, \cdots, 2^n - 1\}$
  - $f_{B2U}^n(x_{n-1}, x_{n-2}, \cdots, x_0) = \sum_{i=0}^{n-1} x_i \cdot 2^i$
    - $f_{B2U}^4(1, 0, 1, 1) = 11_{10}$

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 x의 r의 보수(補數, radix complement, complement) $\bar{x}$는,
  - $\bar{x} = \begin{cases} r^n - x, & x \neq 0 \\ 0, & x = 0 \end{cases}$

- x의 $(r-1)$의 보수(diminished radix complement) x'은,
  - $x' = (r^n - 1) - x$

보수 구하기

- x의 $(r-1)$의 보수는 x의 각 자리에 있는 숫자의 $(r-1)$의 보수를 취하면 됨
  - 각 자리의 숫자를 $(r-1)$에서 빼는 것
  - $(r^n-1)-x = (r^n-1^n)-x = ((r-1)\cdot r^{n-1}+(r-1)\cdot r^{n-2}+\cdots+(r-1)\cdot r^0)-x$
  - 예) 십진수 836의 $(10-1)$의 보수는 $(10^3-1)-836 = 999-836 = 163$

- x가 0 이 아닌 경우, x의 $(r-1)$의 보수에 1을 더하면 x의 r의 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_2이고 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 x의 2의 보수 $\bar{x}$
  - $\bar{x} = \begin{cases} 2^n - x, & x \neq 0 \\ 0, & x = 0 \end{cases}$
- 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_2이 2의 보수 표현에서 -6_{10}을 나타냄
- MSB는 부호 비트

2의 보수 표현의 범위와 값

- 가장 큰 수는 $2^{n-1}-1$ 이고 가장 작은 수는 -2^{n-1}

- $x_{n-1}x_{n-2}\cdots x_0$를 2의 보수 표현으로 해석하여 그 값에 대응시키는 함수
  - $f_{B2C}^{n}(x_{n-1},\ x_{n-2},\cdots,x_0) : B^n \dashrightarrow\ C_n$
    - $C_n = \{-2^{n-1}, -2^{n-1}+1, \cdots, -1, 0, 1, \cdots, 2^{n-1}-1\}$
  - $f_{B2C}^{n}(x_{n-1}, x_{n-2}, \cdots, x_0) = -x_{n-1}\cdot 2^{n-1} + \sum_{i=0}^{n-2} x_i \cdot 2^i$
    - 예) $f_{B2C}^{4}(1,0,1,1) = -5_{10}$

f_{B2U}^n과 f_{B2C}^n의 관계

- $b = (x_{n-1}, x_{n-2}, \cdots, x_0)$
- (1) : $f_{B2U}^n(b) = \sum_{i=0}^{n-1} x_i \cdot 2^i$
- (2) : $f_{B2C}^n(b) = -x_{n-1} \cdot 2^{n-1} + \sum_{i=0}^{n-2} x_i \cdot 2^i$
- (1)-(2)

$$f_{B2U}^n(b) - f_{B2C}^n(b) = \sum_{i=0}^{n-1} x_i \cdot 2^i + x_{n-1} \cdot 2^{n-1} - \sum_{i=0}^{n-2} x_i \cdot 2^i$$

$$\boxed{f_{B2U}^n(b) = f_{B2C}^n(b) + x_{n-1} \cdot 2^n}$$

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 m과 $n(\neq 0)$이 주어졌을 때 $m = qn + r$과 $0 \leq r < |n|$을 만족하는 유일한 정수 q와 r이 항상 존재
    - q는 m을 n으로 나눈 몫이고 r은 나머지

우리가 잘 알고 있는 나눗셈

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

mod

- 두 정수 x와 $m(\neq 0)$이 주어질 때 x/m의 몫 q 는 $\lfloor x/m \rfloor$으로 나타낼 수 있고, 그 나머지 r 을 나타내는 연산이 mod
  - $q = \left\lfloor \frac{x}{m} \right\rfloor,\ m \neq 0$
  - $r = x \bmod m = x - mq,\ m \neq 0$
  - m은 modulus라고 불림
- 앞으로 이용할 성질
  - $(a + b) \bmod n = \big((a \bmod n) + (b \bmod n)\big) \bmod n$

합동관계(合同關係, congruence relation)

- $x \bmod m = y \bmod m$
  - x와 y의 관계를 $x \equiv y \pmod{m}$이라 표기
  - x와 y는 모듈로 m에 대하여 합동
  - x와 y가 m으로 나누었을 때 유클리드 나눗셈의 나머지가 같음

모듈로-m 연산(modulo-m operation)

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