3. 데이터의 표현

목표

- 데이터의 표현 방법 및 처리 방법을 이해함.

데이터 종류

디지털

- 제한 수의 불연속적인 값으로 표현
  - 명령어
  - 데이터
- 이진수, binary number
  - 0, 1의 불연속적인 값 이용
  - Bit, binary digit

디지털 컴퓨터

- 디지털 데이터 처리
- 처리 대상 데이터 종류
  - 수
  - 문자
  - 특수 목적 등…
- **디지털 값**으로 표현되어야 함
  - 이산적인 값 : 0, 1
  - 예: 정수 15 ←→ 이진 조합 00001111

왜 “데이터 표현”을 알아야 하나?

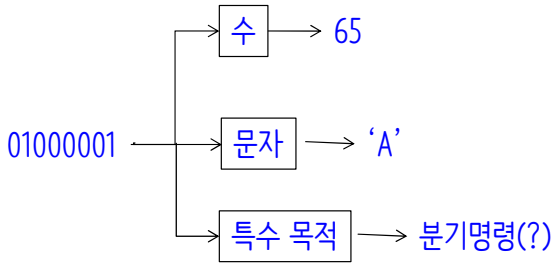
디지털 컴퓨터에서 데이터(수, 문자, 특수 목적 등)를 어떻게 표현하고, 그것을 처리하는 방법 이해

→ 시스템 소프트웨어 및 하드웨어 설계의 기본적인 개념

데이터 표현

- 0과 1의 나열을 해석하는 방법

- 종류
  - 수
  - 문자
  - 특수목적 등…

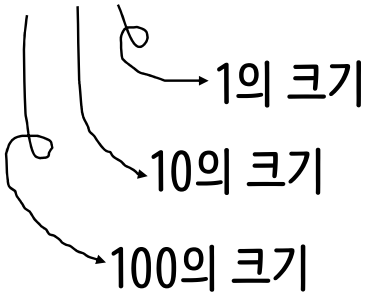


자리 표기법, Positional Notation

수 111은 크기는?

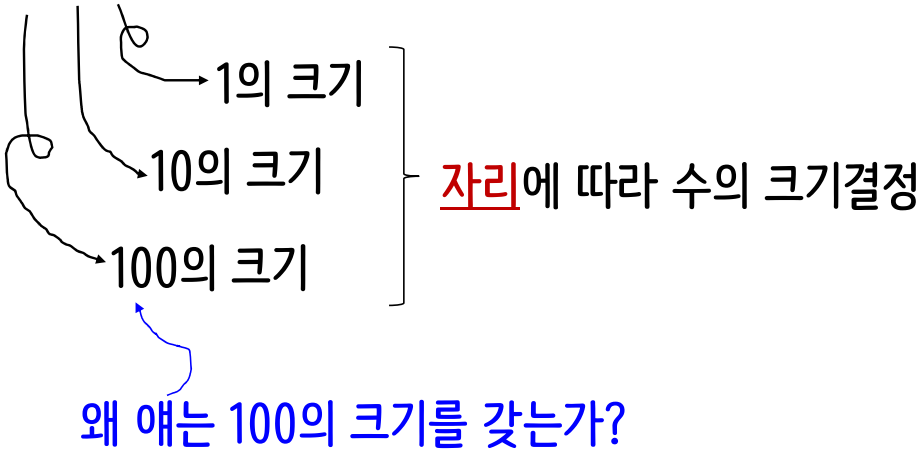
자리 표기법, Positional Notation

수 1 1 1에서 왜 같은 '1'이 아닌가?



자리 표기법, Positional Notation

수 1 1 1에서 왜 같은 '1'이 아닌가?



진법

- Base 또는 Radix
  - 수 표현을 위해 사용하는 부호의 수
  - r-진법 : 0~(r-1) 이용
- Positional number

진법

$$A_n = A_{n-1} \cdots A_1 A_0 . A_{-1} \cdots A_{-m}$$

$$V(A_n) = A_{n-1} \cdot R^{n-1} + \cdots + A_1 \cdot R^1 + A_0 \cdot R^0 + A_{-1} \cdot R^{-1} + \cdots A_{-m} \cdot R^{-m}$$
$$= \sum_{i=-m}^{n-1} A_i \cdot R^i$$

$$101101.1 = 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1}$$

10 진법

- 기수(Base)가 10인 수 체계
  - 10개의 숫자로 수 표현 : 0 ~ 9

- 10진수의 자리표기법 사용

$$d_{n-1} * 10^{n-1} + d_{n-2} * 10^{n-2} + \ldots + d_1 * 10^1 + d_0 * 10^0$$

수 1 1 1= $1 \times 10^2 + 1 \times 10^1 + 1 \times 10^0$

16 진법

- 기수(Base)가 16인 수 체계
  - 16개의 숫자로 수 표현 : 0 ~ 9, A, B, C, D, E, F

- 16진수의 자리표기법 사용

$$d_{n-1} * 16^{n-1} + d_{n-2} * 16^{n-2} + \ldots + d_1 * 16^1 + d_0 * 16^0$$

수 1 1 1= $1 \times 16^2 + 1 \times 16^1 + 1 \times 16^0$

2 진법

- 기수(Base)가 2인 수 체계
  - 2개의 숫자로 수 표현 : 0, 1 → 비트(binary digit)

- 2진수의 자리표기법 사용

$$d_{n-1} * 2^{n-1} + d_{n-2} * 2^{n-2} + \ldots + d_1 * 2^1 + d_0 * 2^0$$

수 1 1 1= $1x2^2 + 1x2^1 + 1x2^0$

R 진수 표시

수 1 1 1의 크기는?

기수 R을 알아야 크기를 알 수 있음

R 진수 표시

수 111_{10}의 또는 $(111)^{10}$ 크기는?

수 111_{16}의 크기는?

수 111_{2}의 크기는?

10진수를 r-진수로

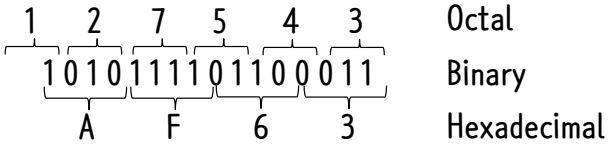
- 방법
  - 소수점 이상은 r로 나눈 나머지를 나온 순서와 반대로 이용
  - 소수점 이하는 r로 곱한 몫을 나온 순서대로 이용
- 예 : 10진수(41.6875)→2진수(101001.1011)

41	
20	1
10	0
5	0
2	1
1	0
0	1

```
0.6875
   X 2
-------
1.3750
   X 2
-------
0.7500
   X 2
-------
1.5000
   X 2
-------
1.0000
```

2진수 → 8진수, 16진수

- 소수점 기준 3비트 또는 4비트 묶음.
- 예 :



BCD, Binary Coded Decimal

- **십진수** 표현 방법
  - 십진수 1자리에 4 비트

Decimal Number	Binary-coded Decimal Number
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
12	0001 0010

글자(문자)의 순서

'B' 보다 앞에 있는 글자는?

'sky'와 'skate' 중 사전상 순서로 앞에 있는 문자열은?

글자(문자)의 순서

'B' 보다 앞에 있는 글자는?

'sky'와 'skate' 중 사전상 순서로 앞에 있는 문자열은?

'앞에 있다'는 것 = 순서가 있다는 ?

→ 값으로 표현되어야 함

문자의 표현

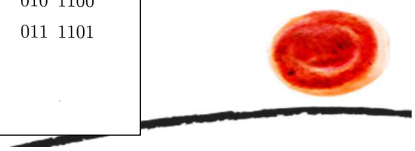
- 표현해야 하는 문자의 수는 유한
- 각 문자들에 이진 비트 열(bit string) 할당
  - 문자 코드(code)
- **문자의 집합**
  - 문자들과 각 문자를 위한 코드 리스트

ASCII 문자 집합

- ASCII 는 정보 교환 목적의 미국 표준 코드
  - American Standard Code for Information Interchange
  - 각 문자에 7비트씩 사용하여 128개의 문자 표현
- 확장형 ASCII
  - 8비트 코드
  - 표현할 수 있는 문자의 수는?

ASCII 문자 집합 (일부)

Character	Binary code	Character	Binary code
A	100 0001	0	011 0000
B	100 0010	1	011 0001
C	100 0011	2	011 0010
D	100 0100	3	011 0011
E	100 0101	4	011 0100
F	100 0110	5	011 0101
G	100 0111	6	011 0110
H	100 1000	7	011 0111
I	100 1001	8	011 1000
J	100 1010	9	011 1001
K	100 1011		
L	100 1100		
M	100 1101	space	010 0000
N	100 1110	.	010 1110
O	100 1111	(	010 1000
P	101 0000	+	010 1011
Q	101 0001	$	010 0100
R	101 0010	*	010 1010
S	101 0011	)	010 1001
T	101 0100	-	010 1101
U	101 0101	/	010 1111
V	101 0110	,	010 1100
W	101 0111	=	011 1101
X	101 1000		
Y	101 1001		
Z	101 1010		



ASCII 코드의 한계

- 8 bit 코드 → 256 개의 글자 코드 표현
- 그럼 '한글'은?

보수, COMPLEMENT

보수, Complement

- 뺄셈 연산과 논리 연산에 사용

- r-진법 보수(n자리수의 수 N에 대해)
  - r의 보수: $r^n - N$
  - r-1의 보수: $(r^n - 1) - N$

(R-1)의 보수 예:

546700_{10}의 9의 보수 : $(10^6-1)-546700 = 453299$

0001111_2의 2의 보수 : $(2^7-1)-0001111 = 1110000$

1의 보수를 구할 때 0은 1로, 1은 0으로 바꾸는 것과 동일.

R의 보수

2389_{10}의 10의 보수 : $10^4-2389 = 7611$

101100_2의 2의 보수 : $2^6-101100 = 010100$

"2의 보수를 구할 때 1의 보수를 구한 후 +1 하라". 왜?

$[\ (2^n-1) - N\ ] + 1 = 2^n - N$

1의 보수

10의 보수를 구할 때도 적용가능.

TIP!! 2의 보수 구하기

- 공식적인 방법 :

  $2^n - d$ 단, 수 d는 n 자리

1110의 경우:
$2^4 - 1110$
= 0010

- Tip 1:
  - 단계 1 : 0→1, 1→0으로 바꿈
  - 단계 2 : 단계 1의 결과에 +1

1110의 경우:
단계1 : 1110 → 0001
단계2 : 0001+1 = 0010

1110의 경우: 0010

- Tip 2: ^^
  - 비트들의 오른쪽에서 왼쪽으로 볼 때(scan), 처음 1이 올 때까지는 그대로, 그 이후부터는 0→1, 1→0으로 바꿈.

부호없는 수의 뺄셈

- r 진수의 부호없는 두 n자리 수 뺄셈: M-N
  - 1. M에 N의 r의 보수 더함 : $M+(r^n-N)=M-N+r^n$
  - 2.M>=N이면, r^n이 캐리 발생시킴
    - 이 캐리를 무시함.
    - (M-N)의 답을 구함.

    → WHY? n 자리만 저장 또는 처리!!
  - 3.M<N이면, r^n-(N-M)
    - 결과에 r의 보수 구한 후
    - 음수 부호 붙임

    → (N-M)의 R의 보수

10진수 뺄셈의 예

- 72532-13250

자리올림수(캐리, carry)는 무시

M = 72532
N 13250의 10의 보수 = + 86750
합 = 1 59282
답 = 59282

- 13250-72532

M = 13250
N 72532의 10의 보수 = + 27468
합 = 40718
답 = - 59282

2진수 뺄셈의 예

- 1010100-1000011

M = 1010100
N의 2의 보수 = + 0111101
합 = 1 0010001
답 = 0010001

- 1000011-1010100

M = 1000011
N의 2의 보수 = + 0101100
합 = 1101111
답 = - 0010001

2진수 뺄셈의 예

- 1010100-1000011

```c
#include <stdio.h>

int main()
{
  int M = 0x54;
  int N = 0x43;

  printf("%x \n", M-N );

  M = 0x43;
  N = 0x54;

  printf("%x \n",(M-N) );

  return 0;
}
```

- 10000

M = 1010100
N의 2의 보수 = + 0111101
합 = 10010001
답 = 0010001
1 1

M = 1000011
N의 2의 보수 = + 0101100
합 = 1101111
답 = - 0010001

결과(16진수) :

```
11
ffffffef
```

고정 소수점 표현

고정 소수점, Fixed-Point

- 소수점의 위치가 고정

- 수의 가장 왼쪽에 소수점
  - 소수
  - .456
- 수의 가장 오른쪽에 소수점
  - 정수
  - 456.

정수 표현

- 부호 절대값(signed-magnitude)
- 부호화된 1의 보수(signed 1’s complement)
- 부호화된 2의 보수(signed 1’s complement)

- 양수 표현은 동일

정수 표현

- +14 : 00001110

- -14
  - 부호와 절대값 : 10001110
  - 1의 보수 : 11110001
  - 2의 보수 : 11110010

부호 비트 : 0=양수, 1=음수

부호 비트 역할 : 0=양수, 1=음수

정수 덧셈,뺄셈

- 2의 보수 표현 사용
  - 연산이 간단
- 덧셈
  - 두 수를 덧셈
  - 캐리는 무시
- 뺄셈
  - 감수(subtrahend)의 2의 보수 취한 후 덧셈
  - A-B = A + (-B)

오버플로, overflow

- 주어진 비트로 표현할 수 없는 값이 결과값으로 생성
  - 정상적으로 저장할 수 없음.

- 부호없는 수의 덧셈에서 오버플로
  - 연산 결과 캐리의 값이 1.

- 부호가 있는 수의 덧셈에서 오버플로
  - 부호 비트로 올라오는 캐리와 부호 비트에서 생성되는 캐리의 값이 다름.

오버플로 판단

0 0	0 1	1 1	1 0
0XXXX	0XXXX	1XXXX	1XXXX
0XXXX	0XXXX	1XXXX	1XXXX
0XXXX	1XXXX	1XXXX	0XXXX
오버플로 X	오버플로 O	오버플로 X	오버플로 O

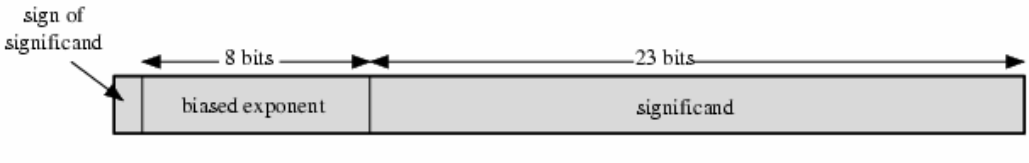
오버플로의 예

Carry:	0 1	Carry:	1 0
+ 70	01000110	- 70	10111010
+ 80	01010000	- 80	10110000
------	--------	------	--------
+ 150	10010110	- 150	01101010

부동 소수점 표현
FLOATING POINT

부동소수점

- 실수 표현
- 구성 : $(-1)^s \, m \, X \, r^e$
  - 부호, s
  - 가수(mantissa), m
  - 지수(exponent), e



IEEE 754 표준

지수

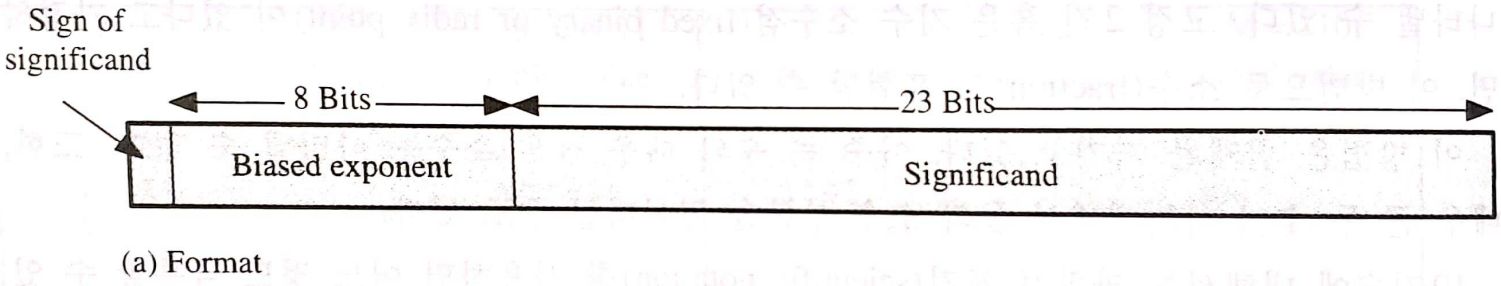
- 소수점 위치 이동

- 바이어스된 표현, biased representation
  - 지수 부분의 값 - 바이어스값 = 지수 값
  - 음수 : 소수점을 왼쪽으로 이동
  - 양수 : 소수점을 오른쪽으로 이동
  - IEEE 754 표준에서 32 비트인 경우 바이어스값 127

가수

- 정규화된 표현, normalization
  - 가수의 MSB가 0이 아닌 값이 되도록
  - ±1.bbb…x 2±E 형태
    - 가수의 MSB는 항상 1
    - 저장할 필요 없음
    - 23비트 이용하여 24비트 표현하는 효과

부동 소수점 표현 예



(a) Format

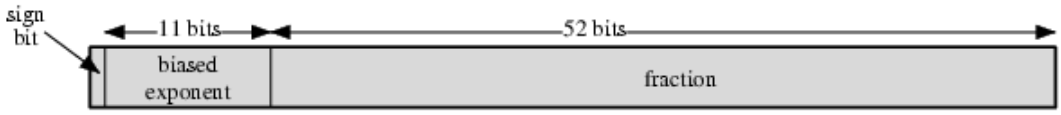
0 10000000 10010010000000000000000 = + 3.140623

1 10000000 10010010000000000000000 = − 3.140623

0 10000001 10010010000000000000000 = + 6.2815

부동소수점 표준 - IEEE 754

(a) Single format



(b) Double format

기타 이진 코드 및 에러 검출 코드

Gray 코드

- 한 숫자에서 다음 숫자로 올라갈 때 한 비트만 변함

표3-5 4비트 Gray code

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

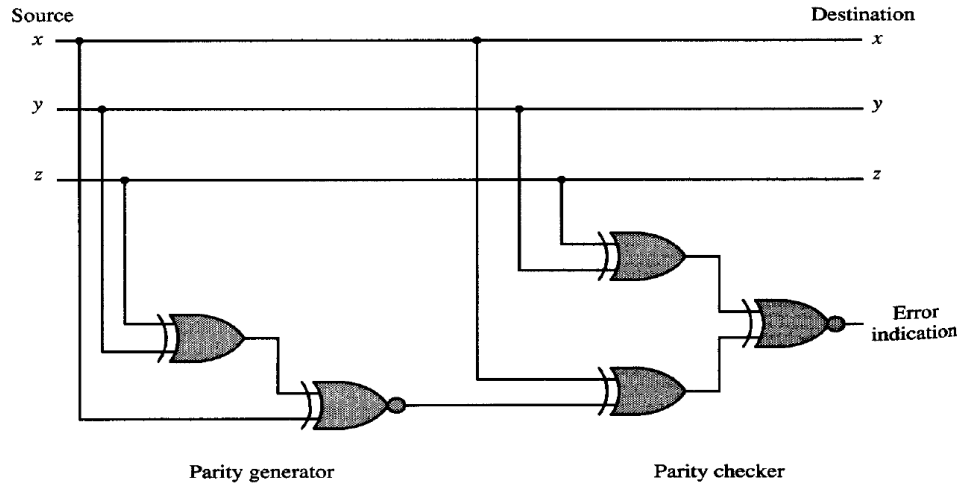
10진수를 위한 코드

표3-6 10진수를 위한 4개의 다른 2진 코드

Decimal digit	BCD 8421	2421	Excess-3	Excess-3 gray
0	0000	0000	0011	0010
1	0001	0001	0100	0110
2	0010	0010	0101	0111
3	0011	0011	0110	0101
4	0100	0111	0111	0100
5	0101	1011	1000	1100
6	0110	1100	1001	1101
7	0111	1101	1010	1111
8	1000	1110	1011	1110
9	1001	1111	1100	1010

패리티 검사

- 전송 중 발생할 수 있는 오류 검출 방법
- 전송 데이터에 1의 수가 홀수/짝수가 되도록 패리티 비트를 추가
- ODD 패리티의 예 :



Q&A