REPRESENTASI DATA

Arsitektur Komputer

Abstraksi Data

- *Raw* data kehidupan manusia
  - Personal data input [lewat 5 indra]
  - Mass media [audio/visual] data input [populer, ilmiah, fiksi, riset, dll.]
- Pengertian Dasar
  - Data ➔ Informasi
  - Komunikasi
  - Kode / notasi / simbol / Icon
- Bentuk Kode
  - Karakter umum: Numerik & Alfabet
  - Karakter khusus

Lanjutan …

- Jenis Representasi
  - Eksternal (human-readable)
    a. notasi dalam bahasa pemrograman
    b. fasilitas untuk: pemrogram, desainer bahasa, pemakai
    c. berbasis notasi aljabar (desimal, sign/magnitude)

Lanjutan …

- Internal (machine readable)
  a. notasi aktual dalam komputer
  b. fasilitas untruk konstruksi HW
  c. meningkatkan efisiensi
  d. HW menjadi handal

Konsep Tipe Data

- Definisi matematik
  - cardinal / unsigned numbers integer: 0,1,2, …]
  - signed numbers integer: -2, -1, 0, 1, 2, …]
  - real number [4.5, -8.47, -0.6* 107]
  - character
  - string
  - boolean [true, false]
  - beragam struktur gabungan 6 tipe tersebut

Lanjutan…

- ADT (abstract data type) [a specified set of items which certain properties & operations]: boolean, rate, time, speed, area, ..
- Variabel, konstanta, atau ADT
- Format Dasar

  - Tipe informasi : (a) instruksi, (b) data : numbers / numerical : (fixed-point & floating point), nonnumerical

  - Length : bit, byte,word, double/long word

  - Storage bit order : most (left) - least (right) significant

Sistem Bilangan

- Biner : basis 2; {0, 1}
- Oktal: basis 8; {0, 1, 2, 3, 4, 5, 6, 7}
- Desimal: basis 10; {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
- Heksadesimal: basis 16;
  {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}

Lanjutan…

- Contoh:
  - Bilangan biner: 11101_2
  - Bilangan oktal: 54731_8
  - Bilangan desimal: 1250910
  - Bilangan heksadesimal: A18E616

Konversi Bilangan

- Satu sistem bilangan bisa dialih bentuk ke sistem bilangan lain.
- Contoh, biner dan oktal ke desimal:

a. $1\ 1\ 1\ 0\ 12$ = d4 d3 d2 d1 d0

$= 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1$ + 1×2^0

$= 16 + 8 + 4 + 0 + 1 = 29_{10}$

b. $668 = 6 \times 8^1 + 6 \times 8^0$

$= 48 + 6 = 54_{10}$

Lanjutan…

- Contoh, desimal ke oktal:

129	
8 ----- 1	
16	Baca dari bawah ke atas,
8 ----- 0	
2	maka: $\mathbf{129_{10} = 201_8}$

- Berapa konversi: 1492_{10} = ……. $_2$?

Lanjutan …

- Algoritma konversi dari desimal

```
place = 0;
number = number to be converted
while (number > 0 )
    {
        digit[place] = number mod B ;
        number = number div B ;
        place = place + 1 ;
    }
```

Unsigned Integer

– *Unsigned integer* hanya untuk bilangan positif, jika negatif perlu format baru.

– Kelebihan biner

- mudah dibuat
- tidak ambigu (noise immunity)
- bisa dicopy dengan sempurna (flawlessly)
- untuk bilangan 8-bit memungkinkan 256 pola 0 & 1, sejumlah: 128 negatif + 128 positif.

Unsigned Integer

- Unsigned variable disimpan lansung dalam format biner

No. of bits	Binary	
	Min	Max
n	0	$2^n - 1$

Unsigned Integer

Type	No. of bytes	No. of bits	Range
unsigned char	1	8	0 … 255
unsigned short	2	16	0 … 65,535
unsigned int	4	32	0 … 4,294,967,295
unsigned long	4	32	0 … 4,294,967,295

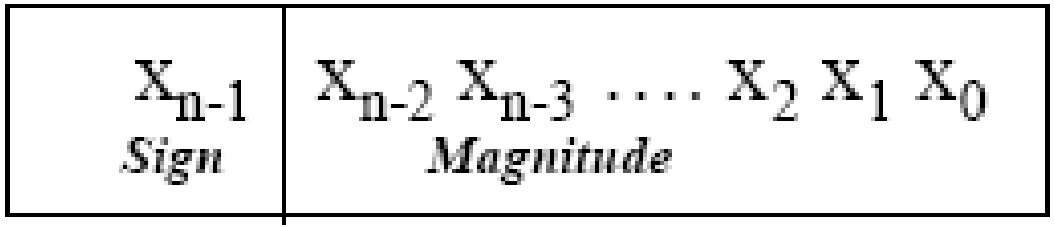
Signed Integer

- Representasi
  - *Signed Integer* direpresentasikan selain oleh nilai bilangan juga dengan adanya tambahan tanda [negatif].

- Tipe
  - Sign and Magnitude (SM)
  - One's Complement (OC)
  - Two's Complement (TC)

Signed Integer

- Notasi Sign Magnitude (SM)
  - Format:



  - Bila digit pada posisi terkiri (sign)
    - **Nol** maka bilangan tersebut **positif**,
    - **Satu** maka bilangan tersebut **negatif**

    [negatif nol =positif nol = nol].

  - Rentang SM (untuk integer **m** dan **I**):

$$-(2^{m-1} - 1) \leq I \leq +(2^{m-1} - 1)$$

Signed Integer

- Sign Magnitude (SM):
  - Penjumlahan, aturan:
    - sign tidak dijumlahkan, hanya magnitude
    - buang *carry out* dari msb magnitude
    - jumlahkan yang sign-nya sama (+ ke + atau - ke -)
    - sign hasil = sign penambah

  - Contoh

```
  0  0101  (5)        1  1010  (–10)
+ 0  0011  (3)      + 1  0011  (–3)
-------------      -------------
  0  1000  (8)        1  1101  (–13)
```

Signed Integer

- Sign Magnitude (SM):
  - Pengurangan, aturan:
    - lakukan jika sign sama, jika sign berbeda, ubah soal ke penjumlahan
    - perbandingkan magnitude, lakukan:

      a - b menjadi a + (-b)

      a + b menjadi a - (-b)
  - Contoh
    - (1)

      1 11000 (-24)

      - 1 00010 (-2)

      ———————

      1 10110 (-22)

(2)

0 00111 (7)

- 0 11000 (24)

———————

ubah

→ 0 11000 (24)

- 0 00111 (7)

———————

1 10001 (-17)

Signed Integer

- One's complement (OC)
  - Definisi
    - (2^n - 1 → memberikan jumlah n digit)
    - komplementasi bilangan biner (ubah 1 ke 0 dan 0 ke 1)
  - Contoh
    - OC dari 00101110 adalah 11010001
  - Rentang nilai

Number of Bits	Number of Values	Min Value	Max Value
16	$2^{16} - 1$	$-(2^{15} - 1)$	$2^{15} - 1$
32	$2^{32} - 1$	$-(2^{31} - 1)$	$2^{31} - 1$
N	$2^{N} - 1$	$-(2^{N-1} - 1)$	$2^{N-1} - 1$

Signed Integer

- Signed integers are generally stored in 2's complement format.
- Most Significant Bit (MSB) is considered the sign bit
  - 1 for negative numbers -- 0 for positive numbers

No. of bits	Binary	
	2's compliment	
	Min	Max
n	-2^{n-1}	$2^{n-1} - 1$

Signed Integers

- To change the sign of any number
  1. Invert all the bits
  2. Add 1

2_{10} == 0010_2

=> 1101

\+ 1

\----

1110_2 => -2_{10}

-2_{10} == 1110_2

=> 0001

\+ 1

\----

0010_2 => 2_{10}

BILANGAN FLOATING-POINT

- Representasi bilangan *floating-point* mempunyai tiga bagian:
  1. Mantissa
  2. Basis
  3. Eksponen

- Contoh :

Bilangan	Mantissa	Basis	Eksponen
3×10^6	3	10	6
110×2^8	110	2	8
6132.784	0.6132784	10	4
34.58	0.3458	10	2

- Mantissa dan eksponen direpresentasikan secara eksplisit dalam komputer. Tetapi basisnya (base) adalah yg digunakan oleh komputer tersebut.

- Umumnya komputer mengikuti basis 2.

- Umumnya sebuah bilangan *f* direpresentasikan sebagai $f = m \times r^e$ di mana m adalah mantissa, *r* adalah basis dari sistem bilangan dan e adalah eksponen (pangkat dari basis yg digunakan).

- Format umum bilangan *floating-point*:

S	Eksponen	Mantissa

- Semula penggunaan format berbeda antar pabrik komputer untuk merepresentasikan bilangan floating-point. Tetapi saat ini telah digunakan format standar ANSI/IEEE secara luas (format IEEE 754). IBM mempunyai standar khusus
- Ada dua format standar IEEE 754 yg dikeluarkan yaitu untuk presisi tunggal (*single precision*) dan format standar untuk bilangan presisi ganda (*double precision*)



Format floating-point presisi tunggal (32 bit)



Format floating-point presisi ganda (64 bit)

NORMALISASI FLOATING-POINT

- Bilangan floating-point dapat direpresentasikan dgn banyak cara seperti yg ditunjukkan untuk bilangan desimal 110 x 28:
  110×2^8, 11×2^9, 1100×2^7,
  1.1×2^{10}, 0.11×2^{11}, $.011 \times 2^{12}$ .... dst.

- Suatu bilangan floating-point berada dalam bentuk ternormalisasi jika *Most Significant Digit* dari mantissa bukan-nol (non-zero).

- Untuk mengubah menjadi bilangan yg ternormalisasi, mantissa harus digeser ke kanan atau ke kiri dgn tepat, menaikkan atau menurunkan eksponen. Jika semua bilangan floating-point direpresentasikan dalam komputer dgn bentuk ternormalisasi, maka posisi bit satu dapat disimpan dengan mengabaikan MSB (selalu 1). Ini disebut *hidden 1 principle*.

Konversi format saintifik ke format standar IEEE

1. Pertama ubah menjadi bilangan biner.

2. Normalisasikan bilangan tsb sehingga terdapat satu digit nonzero di ujung terkiri, lakukan adjust exponent sesuai kebutuhan.

3. Simpan digit biner mantissa ke sisi kanan

4. Tambahkan 127 ke bagian exponent dan ubah hasil penjumlahan tersebut ke biner untuk nilai exponent yg akan disimpan. Untuk double precision, tambahkan 1023 ke exponent.

5. Sign bit = 1 untuk bilangan negatip dan sign bit = 0 untuk bilangan positip.

Single precision



representasi 8-bit
excess -127 exponent

23-bit
mantissa fraction

Bit tanda:
0 = +
1 = –

$$\text{Representasi nilai} = \pm 1.M \times 2^{E' - 127}$$

biased exponent

Double precision



representasi 11-bit
excess -1023 exponent

52-bit
mantissa fraction

Bit tanda:
0 = +
1 = –

$$\text{Representasi nilai} = \pm 1.M \times 2^{E' - 1023}$$

Contoh: Tuliskan +0.0010110... x 2^9 dalam format *single precision* standar IEEE 754

Solusi:

Setelah dinormalisasi : +1.0110... x 2^6
E’ = 6 + 127 = 133 = 10000101

0	10000101	0110...

Contoh: Tuliskan $(-0.75)_{10}$ dalam format *single precision* standar IEEE

Solusi:

- Sign bit, S = 1 (negatip)

0.75 x 2 = 1.5 → 1
0.5 x 2 = 1.0 → 1
0.0 x 2 = 0.0 → 0

$0.11000000 = 0.11000000 \times 2^{0}$

dinormalisasi: 1.10×2^{-1}

M = 10000000000000000000000

E = – 1

E’ = E + 127 = – 1 + 127 = 126 = 01111110

Sehingga representasi single precision $(-0.75)_{10}$:

1	01111110	10000000000000000000000

Latihan:

Typical 32-bit floating-point format

$+1.638125 \times 2^{20} =$

0	10010011	10100011010111 0....

$-1.638125 \times 2^{20} =$

1	10010011	10100011010111011001100

$+1.638125 \times 2^{-20} =$

0	01101011	10100011010111011001100

$-1.638125 \times 2^{-20} =$

1	01101011	10100011010111011001100

Latihan:

A. Tuliskan dalam format *single precision* standar IEEE 754:

1. $(+ 5.375 \times 10^{-3})_{10}$
2. $(+ 200.0)_{10}$
3. $(- 13.275)_{10}$
4. $(- 0.05375 \times 10^{2})_{10}$

B. Tuliskan dalam format notasi saintifik desimal

0	10000101	01100000000000000000000