Operasi Aritmatika Biner Dan Representasi Floating Point

Q Fadlan, S.ST, M.Kom
STMIK TAZKIA BOGOR

AGENDA

1. PENDAHULUAN
2. ARITMATIKA BINER (PENJUMLAHAN, PENGURANGAN, PEMBAGIAN, PERKALIAN)
3. KASUS KHUSUS
4. REPRESENTASI FLOATING POINT

PENDAHULUAN

CAPAIAN PEMBELAJARAN

- Memahami konsep dasar operasi bilangan dalam sistem komputer
- Menguasai teknik perhitungan dan manipulasi data binary
- Memahami representasi bilangan real dalam komputer
- Menganalisis keterbatasan dan solusi dalam komputasi numerik

KOMPETENSI YANG DIHARAPKAN

- Mampu melakukan operasi aritmatika dalam sistem biner
- Mampu mengkonversi dan merepresentasikan bilangan floating point
- Dapat mengidentifikasi dan menangani kasus khusus dalam komputasi
- Memahami implementasi di level hardware komputer

MANFAAT DALAM ARSITEKTUR KOMPUTER

- Memahami cara kerja ALU (Arithmetic Logic Unit)
- Mengerti proses komputasi di level hardware
- Memahami desain FPU (Floating Point Unit)
- Mengoptimasi kinerja sistem komputer

TENTANG MATERI

Fondasi penting untuk memahami cara kerja internal komputer dalam memproses dan memanipulasi data numerik

ARITMATIKA BINER

ARITHMETIC LOGIC UNIT (ALU) PADA KOMPUTER

Fungsi Utama ALU

- Unit yang melakukan operasi aritmatika (tambah, kurang, kali, bagi)
- Unit yang melakukan operasi logika (AND, OR, NOT, XOR)
- Komponen inti CPU untuk pemrosesan data numerik

Pemrosesan Data di ALU

- Bekerja dengan data biner (0 dan 1)
- Menggunakan register untuk menyimpan operand
- Menghasilkan flags (carry, overflow, zero, sign)

ARITMATIKA BINER

Pengertian Aritmatika Biner

- Operasi matematika yang menggunakan sistem bilangan biner (0 dan 1)
- Operasi dasar yang dilakukan ALU di level hardware
- Fundamental dari semua perhitungan dalam komputer digital

Operasi Dasar

- Penjumlahan
- Pengurangan
- Perkalian
- Pembagian

Semua perhitungan dalam komputer, sekompleks apapun, pada dasarnya adalah kombinasi dari operasi aritmatika biner sederhana

PENJUMLAHAN BINER / BINARY ADDITION

ATURAN DASAR

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 0 (dengan carry 1)

CONTOH

^{1} (carry)

1 0 1 (5 desimal)

+ 0 1 1 (3 desimal)

---

1 0 0 0 (8 desimal)

PENGURANGAN BINER / BINARY SUBTRACTION

ATURAN DASAR

0 - 0 = 0

1 - 0 = 1

1 - 1 = 0

0 - 1 = 1 (dengan borrow 1)

CONTOH

1 0 1 (5 desimal)

\- 0 1 1 (3 desimal)

---

0 1 0 (2 desimal)

PERKALIAN BINER / BINARY MULTIPLICATION

ATURAN DASAR

0 × 0 = 0

0 × 1 = 0

1 × 0 = 0

1 × 1 = 1

CONTOH

1 0 1 (5 desimal)

× 0 1 1 (3 desimal)

---

1 0 1

1 0 1 0

0 0 0 0 0

---

0 1 1 1 1 (15 desimal)

PEMBAGIAN BINER / BINARY DIVISION

CONTOH

```
                    1 0 1  (5 desimal)
                   ———————————————
11 (3 desimal) | 1111  (15 desimal)
                   11
                  -----
                    01
                    00
                   -----
                      11
                      11
                     ------
                       0
```

KASUS KHUSUS

KASUS KHUSUS - OVERFLOW

A. OVERFLOW

Overflow adalah kondisi yang terjadi dalam komputer ketika hasil perhitungan melebihi kapasitas penyimpanan yang tersedia. Bayangkan seperti mengisi air ke dalam gelas - jika kamu menuang air terlalu banyak, air akan tumpah keluar. Sama halnya dengan komputer, yang memiliki "tempat" terbatas (misalnya 8-bit) untuk menyimpan angka. Ketika hasil perhitungan lebih besar dari kapasitas ini (contoh: dalam sistem 8-bit, mencoba menghitung 255 + 1), komputer tidak bisa menyimpan hasil yang benar dan "kembali ke nol" - seperti odometer sepeda yang berputar kembali ke 0000 setelah mencapai 9999. Hal ini bisa menyebabkan kesalahan serius dalam program komputer, terutama dalam aplikasi penting seperti perhitungan keuangan atau sistem kontrol.

KASUS KHUSUS - OVERFLOW

Evolusi Sistem Bit Pada Komputer

- 4-bit : Prosesor pertama (Intel 4004) - 1971
- 8-bit : Komputer rumah pertama (Apple I, Commodore) - 1970an
- 16-bit : IBM PC/XT - 1980an
- 32-bit : Era Windows 95/98 - 1990an
- 64-bit : Saat ini umum digunakan - 2000an ke atas

KOMPUTER PUNYA KAPASITAS DALAM MENYIMPAN INFORMASI !!!

KASUS KHUSUS - OVERFLOW

CONTOH PERHITUNGAN OVERFLOW

Misalkan kita punya sistem 4-bit, berarti hanya bisa menyimpan:
0000 (0) sampai 1111 (15)

Mari jumlahkan 14 + 3:

```
    1110 (14)
  + 0011 (3)
  ---------------
   10001 ← Ups! Kita hanya punya 4 bit
    0001 ← Yang tersimpan hanya 4 bit terakhir = 1
```

Seharusnya 14 + 3 = 17
Tapi komputer menghasilkan 1!

KASUS KHUSUS - OVERFLOW

OVERFLOW SEPERTI ODOMETER

Sistem 4-bit mencoba menghitung 15 + 1:

```
   1111 (15)
+  0001 (1)
-------
 0000 ← Kembali ke 0!
```

Seperti odometer mobil: 9999 + 1 = 0000

KASUS KHUSUS - OVERFLOW

Analogi Sederhana:

1. Seperti jam dinding 12 jam
2. Jika sekarang jam 11 malam
3. Ditambah 2 jam
4. Harusnya jam 1 malam
5. Tapi jam hanya bisa menunjukkan 1-12
6. Tidak bisa menunjukkan jam 13

KASUS KHUSUS - OVERFLOW

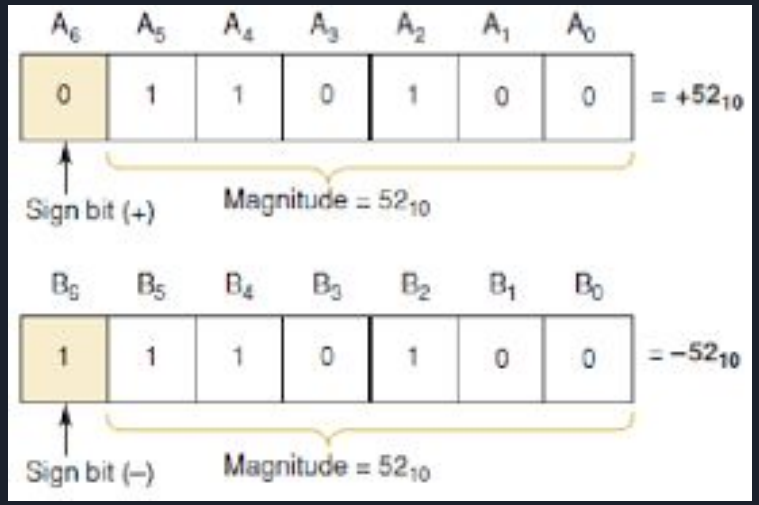
Mengapa overflow penting dipahami:

1. Hasil perhitungan bisa salah total
2. Tidak ada peringatan error
3. Bisa menyebabkan bug serius

Tips: Selalu ingat batasan sistem:

1. 4-bit: 0-15 atau -8 sampai +7
2. 8-bit: 0-255 atau -128 sampai +127
3. Dan seterusnya...

KASUS KHUSUS - BILANGAN NEGATIF (KOMPLEMEN 2)

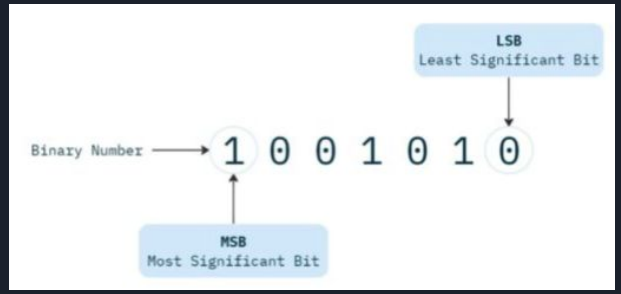


Bilangan negatif dalam komputer tidak direpresentasikan dengan simbol minus (-) seperti yang kita kenal. Untuk menghemat ruang dan menyederhanakan operasi aritmatika, komputer menggunakan sistem bilangan komplemen dua. Dalam sistem ini, bilangan negatif diubah menjadi bentuk biner dengan cara membalik semua bit (0 menjadi 1, 1 menjadi 0) lalu ditambah 1. Bit paling kiri pada representasi komplemen dua tidak selalu menunjukkan tanda secara langsung. Metode ini memungkinkan komputer melakukan operasi aritmatika pada bilangan positif dan negatif secara efisien tanpa perlu sirkuit khusus untuk menangani tanda.

KASUS KHUSUS - BILANGAN NEGATIF (KOMPLEMEN 2)

KONSEP DASAR KOMPLEMEN 2

1. Digunakan untuk merepresentasikan bilangan negatif dalam komputer
2. Bit paling kiri (Most Significant Bit/MSB) menjadi bit tanda
    - 0 = bilangan positif
    - 1 = bilangan negatif
3. Memungkinkan operasi aritmatika langsung tanpa logika khusus



KASUS KHUSUS - BILANGAN NEGATIF (KOMPLEMEN 2)

ATURAN DASAR (menggunakan sistem 4-bit untuk kemudahan)

Positif: Bit paling kiri = 0

0000 = 0 0001 = 1 0010 = 2 0111 = 7

Negatif: Bit paling kiri = 1

1111 = -1 1110 = -2 1101 = -3 1000 = -8

KASUS KHUSUS - BILANGAN NEGATIF (KOMPLEMEN 2)

CONTOH MEMBUAT -5 DALAM 8-BIT

Langkah membuat -5 dalam 8-bit:

1. Tulis 5 dalam biner:

   00000101

2. Inverse semua bit (Komplemen 1):

   11111010

3. Tambahkan 1:

   11111011 (Ini adalah -5 dalam komplemen 2)

KASUS KHUSUS - BILANGAN NEGATIF (KOMPLEMEN 2)

CARA MENGUJI SUDAH BENAR ATAU BELUM:

Gunakan Prinsip Nilai yang sama jika ditambahkan nilai positif dan negatif maka sama dengan 0.
Seperti 5 + (-5) = 0

Apakah 11111011 benar -5?

```
  0000 0101 (+5)
+ 1111 1011 (-5)
--------------------
  0000 0000 (0)
```

REPRESENTASI FLOATING POINT

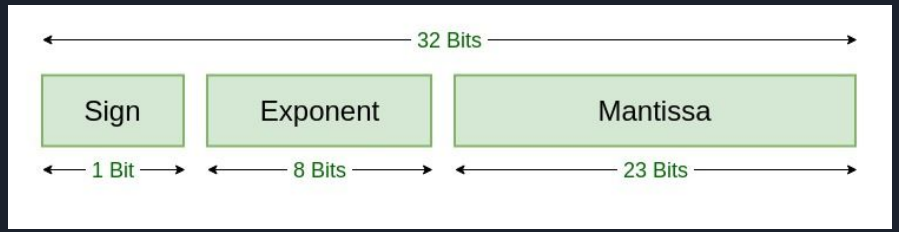
REPRESENTASI FLOATING POINT

Bayangkan komputer sebagai kalkulator super canggih. Selain bisa menghitung bilangan bulat (1, 2, 3, dan seterusnya), komputer juga bisa menghitung bilangan pecahan (1.5, 3.14, dan sebagainya). Cara komputer merepresentasikan bilangan pecahan (bilangan dengan koma) ini disebut representasi floating-point

Kenapa disebut floating-point?

1. Floating: Artinya "mengambang". Titik desimal pada bilangan floating-point bisa berada di posisi mana saja, tidak selalu tetap seperti pada bilangan desimal biasa.
2. Point: Merujuk pada titik desimal itu sendiri.

REPRESENTASI FLOATING POINT



Komponen Floating Point Format Standar IEEE 754

1. Sign (tanda): Menunjukkan positif atau negatif (1 bit)
2. Mantissa: Bagian pecahan yang signifikan
3. Exponent: Pangkat dari basis (biasanya 2)

REPRESENTASI FLOATING POINT

CONTOH KASUS ANGKA 123,45

1. Bagian bulat adalah 123 dan bagian desimal adalah 0,45
2. Konversi bagian bulat dan desimal ke dalam biner

```
123 ÷ 2 = 61  sisa 1
61  ÷ 2 = 30  sisa 1
30  ÷ 2 = 15  sisa 0
15  ÷ 2 = 7   sisa 1
7   ÷ 2 = 3   sisa 1
3   ÷ 2 = 1   sisa 1
1   ÷ 2 = 0   sisa 1
```

123 = 1111011 (biner)

```
0,45 × 2 = 0,90  → 0
0,90 × 2 = 1,80  → 1
0,80 × 2 = 1,60  → 1
0,60 × 2 = 1,20  → 1
0,20 × 2 = 0,40  → 0
...
```

0,45 = 0,0111... (biner)

REPRESENTASI FLOATING POINT

3 . Nilai biner dari 123,45 adalah 1111011,0111...

4. Normalisasi

- Geser koma agar ada 1 angka di depan koma
- Hitung berapa kali menggeser

```
1111011,0111... = 1,1110110111... × 2⁶
                  ↑               ↑
                  bentuk normal   geser 6 kali ke kanan
```

REPRESENTASI FLOATING POINT

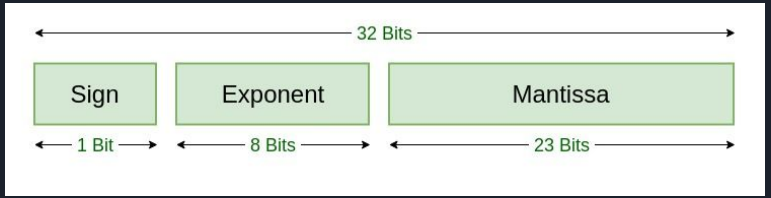
5. Identifikasi Komponen IEEE 754 (Single Precision)

Bit Tanda (Sign Bit)

Jumlah: 1 bit
Fungsi: Menentukan bilangan positif atau negatif
Aturan:

0 = bilangan positif
1 = bilangan negatif

Contoh:
+10,5 menggunakan bit tanda 0
-10,5 menggunakan bit tanda 1



REPRESENTASI FLOATING POINT

Eksponen (Exponent)

- Jumlah: 8 bit (Single Precision)
- Fungsi: Menentukan posisi titik desimal atau pangkat dari 2
- Menggunakan format bias 127
    - Nilai sebenarnya = nilai eksponen - 127

Contoh:

- Jika nilai eksponen 130
- Nilai sebenarnya = 130 - 127 = 3
- Berarti 2^3

Mantissa/Fraction

- Jumlah: 23 bit (Single Precision)
- Fungsi: Menyimpan nilai signifikan dari bilangan
- Selalu dimulai dengan 1 (hidden bit) yang tidak disimpan

Contoh:

- Bilangan 1,0101
- Hidden bit adalah 1
- Yang disimpan hanya 0101

REPRESENTASI FLOATING POINT

Jadi Komponen IEEE 754 (Single Precision) untuk 1,1110110111... x 2^6 adalah

Sign Bit (1 bit)

- Karena angka positif → 0

Exponent (8 bit)

- Bias = 127 (untuk single precision)
- Nilai eksponent = 6
- Nilai yang disimpan = 6 + 127 = 133
- 133 dalam biner = 10000101

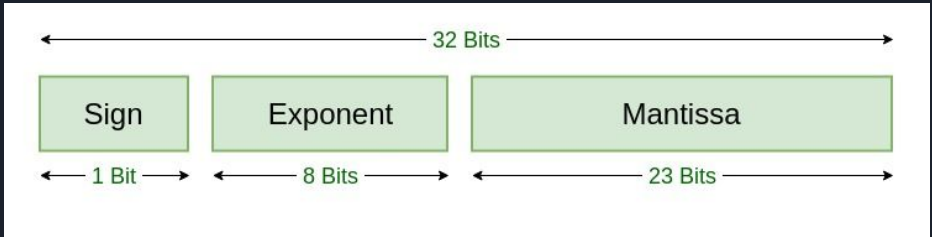
Mantissa (23 bit)

- Ambil angka setelah koma: 1110110111...
- Tambahkan 0 hingga 23 bit
- Menjadi: 11101101100000000000000

REPRESENTASI FLOATING POINT

6. Menyusun Format IEEE 754

0	10000101	11101101100000000000000
↑	↑	↑
Sign	Exponent	Mantissa
(1)	(8)	(23)



REPRESENTASI FLOATING POINT

Cara Membacanya Kembali:

- Sign bit = 0 → angka positif
- Exponent = 10000101 (133) - 127 = 6
- Mantissa = 1,11101101100000000000000
- Nilai = 1,11101101100000000000000 × 2^6 = 123,45

REPRESENTASI FLOATING POINT

JENIS-JENIS FORMAT IEEE 754

Single Precision (32-bit):

- 1 bit tanda (sign)
- 8 bit eksponen
- 23 bit mantissa/fraction
- Total: 32 bit

Double Precision (64-bit):

- 1 bit tanda
- 11 bit eksponen
- 52 bit mantissa/fraction
- Total: 64 bit

Half Precision (16-bit):

- 1 bit tanda
- 5 bit eksponen
- 10 bit mantissa
- Total: 16 bit

REPRESENTASI FLOATING POINT

Peran Fundamental

- Floating point adalah cara komputer menyimpan dan menghitung bilangan pecahan
- Tanpa floating point, komputer hanya bisa bekerja dengan bilangan bulat
- Ini adalah konsep dasar yang memungkinkan perhitungan kompleks di komputer

Standar IEEE 754

- Standar internasional yang memastikan konsistensi perhitungan
- Memungkinkan pertukaran data antar sistem komputer berbeda
- Ada dua format utama: Single (32 bit) dan Double (64 bit) Precision

REPRESENTASI FLOATING POINT

CONTOH KASUS PENGGUNAAN FLOATING POINT

Probabilitas dalam AI

- Prediksi AI sering dalam bentuk probabilitas
- Contoh output classifier:
  - Kucing: 0.85 (85%)
  - Anjing: 0.12 (12%)
  - Burung: 0.03 (3%)
- Membutuhkan presisi floating point untuk akurasi

Floating point merupakan fondasi vital dalam arsitektur komputer modern yang memungkinkan komputer melakukan perhitungan presisi tinggi, dari transaksi perbankan hingga pembelajaran mesin (AI), sehingga pemahaman mendalam tentang konsep ini menjadi kunci bagi siapapun yang ingin memahami dan mengembangkan teknologi masa depan

TUGAS