ORGANISASI dan ARSITEKTUR KOMPUTER

Representasi Informasi

YANG AKAN DIPELAJARI

- Representasi Informasi
  - Sign/Magnitude
  - Komplemen 1
  - Komplemen 2
  - Aritmetik untuk Komplemen 1 dan Komplemen 2
  - Floating Point

Key Points

- Dua prinsip utama untuk aritmatika komputer adalah cara di mana bilangan direpresentasikan (format biner) dan algoritma yang digunakan untuk operasi aritmatika dasar (menambah, mengurangi, mengalikan, membagi). Kedua pertimbangan ini berlaku baik untuk aritmatika integer dan floating-point.
- Bilangan Floating-point dinyatakan sebagai bilangan (significand) yang dikalikan dengan sebuah konstanta (basis) dengan kenaikan beberapa pangkat bilangan bulat (eksponen). Floating point dapat digunakan untuk merepresentasikan jumlah yang sangat besar dan sangat kecil.
- Sebagian besar prosesor menerapkan standar IEEE 754 untuk representasi floating-point dan aritmatika floating-point. IEEE 754 mendefinisikan kedua 32-bit dan format 64-bit.

ALU (Arithmetic and Logic Unit)

- ALU merupakan bagian komputer yang berfungsi membentuk operasi-operasi aritmatika dan logik terhadap data
- Semua elemen lain sistem komputer (control unit, register, memori, I/O) berfungsi untuk membawa data ke ALU untuk selanjutnya di proses dan kemudian mengambil kembali hasilnya.

ALU (Arithmetic and Logic Unit)

- Sebuah ALU dan semua komponen elektronik di komputer didasarkan pada penggunaan perangkat logika digital sederhana yang dapat menyimpan digit biner dan melakukan operasi logika Boolean sederhana.

ALU (Arithmetic and Logic Unit)

- Gambar 9.1 menunjukkan secara umum, bagaimana ALU saling berhubungan dengan seluruh prosesor.
- Data diberikan ke ALU dalam register, dan hasil operasi disimpan dalam register-register.
- Register-register ini adalah lokasi penyimpanan sementara dalam prosesor yang dihubungkan oleh jalur sinyal ke ALU.
- ALU juga dapat mengatur flag sebagai hasil dari operasi. Misalnya, flag overflow di set=1 jika hasil perhitungan yang melebihi panjang dari register.
- Nilai-nilai flag juga disimpan dalam register dalam unit kontrol processor.
- Control unit memberikan sinyal yang mengontrol pengoperasian ALU dan pergerakan data ke dalam dan keluar dari ALU.

ALU
(Arithmetic and Logic Unit)

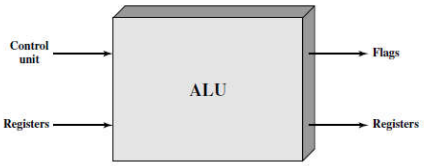


Figure 9.1 ALU Inputs and Outputs

ALU
(Arithmetic and Logic Unit)

- 0 & 1 untuk merepresentasikan apapun.
- Bilangan-bilangan Positif disimpan dalam bentuk biner
  - Contoh: 41=00101001

Representasi Nilai Tanda

- Ada beberapa cara alternatif yang digunakan untuk merepresentasikan bilangan bulat negatif maupun positif, yang melibatkan bit MSB (paling kiri) dalam word yang disebut sebagai bit tanda.
- Jika bit tanda adalah 0, jumlah tersebut secara positif, jika tanda bit adalah 1, nomor tersebut adalah negatif.

Sign-Magnitude

- Bentuk yang paling sederhana representasi yang memakai bit tanda adalah representasi nilai tanda.
- Misal :
  +18 = 00010010
  -18 = 10010010
  (sign magnitute/nilai tanda)
- Terdapat kekurangan pada cara diatas
- Masalah:
  Perlu mempertimbangkan baik tanda dan besarnya dalam aritmatika
  Dua representasi dari nol (+0 dan -0)

Twos Complement Representation

- Seperti sign magnitude, representasi berpasangan komplemen dua menggunakan bit MSB sebagai bit tanda, sehingga mudah untuk menguji apakah integer positif atau negatif.
- Ini memiliki cara yang berbeda dari penggunaan representasi sign-magnitude.

Twos Complement Representation

+18	=	00010010 (twos complement)
bitwise complement	=	11101101
		+ 1
		11101110 = −18

Twos Complement Representation

+7 = 0111 +18 = 00010010

-7 = 1001 - 18 = 11101101

- Dapat di simpulkan bahwa hasil akan berbeda dengan nilai tanda

Keuntungan

- Satu representasi mengenai nilai 0
- Operasi aritmatika lebih mudah
- Menegasikan cukup mudah.
  - 3 = 00000011
  - Boolean complement gives 11111100
  - Add 1 to LSB 11111101

Representasi fixed point

- Semua representasi di atas dapat pula disebut dengan fixed point, karena radix pointnya (binary pointnya) tetap dan di asumsikan akan berada di sebelah kanan.

Negation Special Case 1

- 0 = 00000000
- Bitwise not 11111111
- Add 1 to LSB +1
- Result 100000000
- Overflow is ignored, so:
- - 0 = 0 √

Negation Special Case 2

- -128 = 10000000
- bitwise not 01111111
- Add 1 to LSB +1
- Result 10000000
- Jadi:

  -(-128) = -128

Range of Numbers

- 8 bit 2s compliment
  - ❑ +127 = 01111111 = 2^7 -1
  - ❑ -128 = 10000000 = -2^7
- 16 bit 2s compliment
  - ❑ +32767 = 011111111 11111111 = $2^{15} – 1$
  - ❑ -32768 = 100000000 00000000 = -2^{15}

Conversion Between Lengths

- Positive number pack with leading zeros
  +18 = 00010010
  +18 = 00000000 00010010
- Negative numbers pack with leading ones
  -18 = 10010010
  -18 = 11111111 10010010

Penjumlahan and Pengurangan

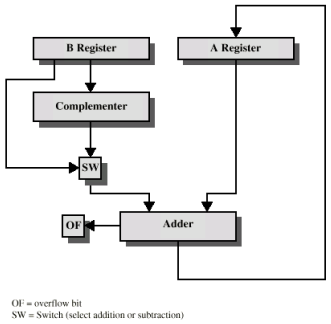
- Memantau bit tanda bit untuk overflow

  Ambil komplemen dua untuk substahend dan tambahkan dengan minuend

  yaitu a - b = a + (-b)

- Jadi kita hanya perlu sirkuit penjumlahan dan komplemen.

Hardware untuk Penjumlahan dan Pengurangan



OF = overflow bit
SW = Switch (select addition or subtraction)

Hardware untuk Penjumlahan dan Pengurangan

- Gambar 9.6 menunjukkan jalur data dan elemen perangkat keras yang diperlukan untuk mencapai penambahan dan pengurangan.
- Unsur utama adalah penjumlah biner, untuk penjumlahan dan menghasilkan jumlah serta indikasi overflow.
- Penjumlah Biner memperlakukan dua angka sebagai unsigned integer.
- Selain itu, dua angka yang akan dijumlahkan diberikan dari dua register, (dalam hal ini register A dan B).
- Hasil mungkin disimpan dalam salah satu register ini atau dalam register ketiga.
- Overflow indikasi disimpan dalam overflow flag 1-bit (0 =NO OVERFLOW 1=OVERFLOW).
- **Untuk pengurangan**, pengurang (register B) dilewatkan melalui complementer sehingga komplemen dua yang diberikan kepada adder.
- Gambar 9.6 hanya menampilkan jalur data.
- Sinyal kontrol yang diperlukan untuk mengontrol apakah ada atau tidak ada komplementer yang digunakan, tergantung pada apakah operasi adalah penambahan atau pengurangan.

Aritmatika Integer

A. Negasi
Untuk membuat negasi gunakan komplement dua (dianjurkan)
Penjumlahan negasi :
+7 = 0111
-7 = 1001

maka bila ada soal (-7) + (+5) = 1001
0101 +
1110

Aritmatika Integer

- Hasil = 1110 adalah bilangan negatif maka positifnya adalah = komplement 2-kan bilangan tersebut : 0010 = +2 maka bilangan 1110 adalah negatif dari 2 atau (-2)
- Aturan overflow = Bila dua buah bilangan di tambahkan, dan keduanya positif atau keduanya negatif maka over flow akan terjadi jika dan hanya jika hasilnya memiliki tanda yang berlawanan.

PENJUMLAHAN

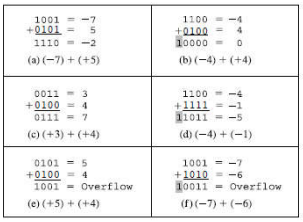


Figure 9.3 Addition of Numbers in Twos Complement Representation

- Overflow ATURAN: Jika dua nomor yang ditambahkan, dan mereka keduanya positif atau keduanya negatif, maka overflow terjadi jika dan hanya jika hasilnya memiliki tanda yang berlawanan.

PENGURANGAN

- Pengurangan ATURAN: Untuk mengurangi satu nomor (pengurang) dari yang lain (minuend), mengambil komplemen twos (negasi) dari pengurang dan menambahkannya ke minuend tersebut.

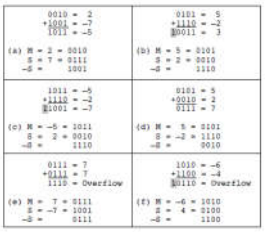


Figure 9.4 Subtraction of Numbers in Twos Complement Representation (M − S)

PERKALIAN

- Kompleks
- Bekerja dengan partial product untuk setiap digit
- Perhatikan penempatan nilai di kolom.
- Jumlahkan partial products

PERKALIAN

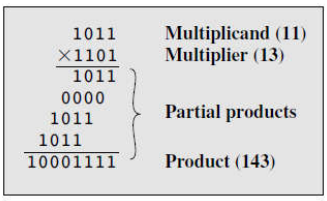
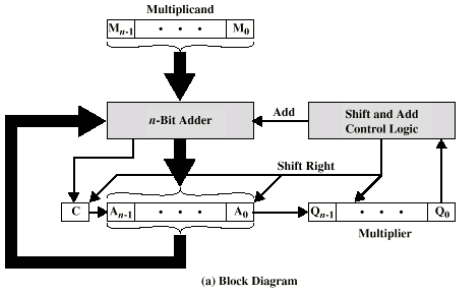


Figure 9.7 Multiplication of Unsigned Binary Integers

Unsigned Binary Multiplication



(a) Block Diagram

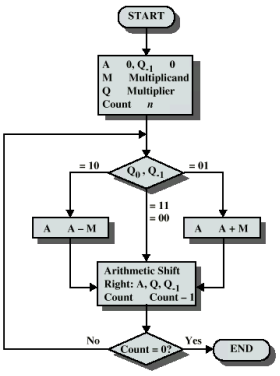
Execution of Example

C	A	Q	M		
0	0000	1101	1011	Initial Values	
0	1011	1101	1011	Add	First Cycle
0	0101	1110	1011	Shift	
0	0010	1111	1011	Shift	Second Cycle
0	1101	1111	1011	Add	Third Cycle
0	0110	1111	1011	Shift	
1	0001	1111	1011	Add	Fourth Cycle
0	1000	1111	1011	Shift	

Multiplying Negative Numbers

- Ini tidak bekerja!
- solusi 1
  - ✓Konversikan ke positif jika diperlukan
  - ✓Kalikan seperti cara pada perkalian
  - ✓Jika tanda-tanda yang berbeda, komplemen 2-kan jawaban.
- Solusi 2
  Algoritma Booth

Booth's Algorithm



Booth's Algorithm

A	Q	Q_{-1}	M		
0000	0011	0	0111	Initial Values	
1001	0011	0	0111	A A - M	First Cycle
1100	1001	1	0111	Shift	
1110	0100	1	0111	Shift	Second Cycle
0101	0100	1	0111	A A + M	Third Cycle
0010	1010	0	0111	Shift	
0001	0101	0	0111	Shift	Fourth Cycle

Aturan:
Jika kedua bit Q_0 dan Q_{-1} sama (1–1 or 0–0), maka geser ke kanan satu kali semua bit yang ada di register A,Q, Q_{-1}.

Jika bit Q_0 dan Q_{-1} (0-1) maka multiplicand dijumlahkan dgn A. Jika bit Q_0 dan Q_{-1} (1-0) → A - M dan hasil disimpan di register A lalu geser 1x.

Booth's Algorithm

Example Consider the multiplication of the two numbers $M = 0111$ (7) and $Q = 1101$ (−3) and assuming that $n = 4$. The steps needed are tabulated below.

M	A	Q	Q(−1)		
0111	0000	1101	0	Initial value	
0111	1001	1101	0	$A = A - M$	
0111	1100	1110	1	ASR	End cycle #1
0111	0011	1110	1	$A = A + M$	
0111	0001	1111	0	ASR	End cycle #2
0111	1010	1111	0	$A = A - M$	
0111	1101	0111	1	ASR	End cycle #3
0111	1110	1011	1	ASR	End cycle #4

−21 (correct result)

PEMBAGIAN

- Pembagian biner dilakukan juga dengan cara yang sama dengan bilangan desimal.
- contoh :

```
101/1111101\11001
 101    -
 ---
  101
  101   -
  ---
   01
   00   -
   --
   10
   00   -
   --
   101
   101  -
   ---
    0
```

PEMBAGIAN

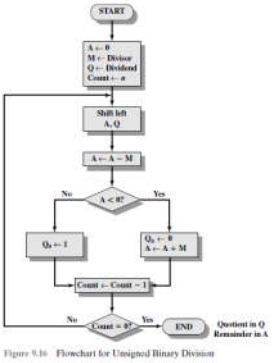


Figure 9.16 Flowchart for Unsigned Binary Division

A	Q	
0000	0111	Initial value
0000	1110	Shift
1101		Use twos complement of 0011 for subtraction
1101		Subtract
0000	1110	Restore, set $Q_0 = 0$
0001	1100	Shift
1101		
1110		Subtract
0001	1100	Restore, set $Q_0 = 0$
0011	1000	Shift
1101		
0000	1001	Subtract, set $Q_0 = 1$
0001	0010	Shift
1101		
1110		Subtract
0001	0010	Restore, set $Q_0 = 0$

Langkah:

1. **Setiap step → A dan Q di geser ke kiri sebanyak 1 bit**
2. **A=A-M**
3. **Jika A positif maka $Q_0 = 1$**
   Jika A negatif maka $Q_0 = 0$ dan restore angka sebelumnya

Representasi Floating Point

- Untuk menuliskan bilangan floating point (bilangan pecahan) dilakukan dengan menuliskan bentuk eksponensial, sehingga bilangan tersebut memiliki bilangan dasar, bilangan pemangkat dan basis bilangan tersebut.

Representasi Floating Point

- Format

$$X = X * B_X^{EX}$$

X= bilangan floating point
Bx= basis dari bilangan X
Ex= Bilangan pemangkat Basis

Misalnya:

$300 = 3 * 10^2$

Representasi Floating Point

Misal :

$976.000.000.000 = 9{,}76 \times 10^{11}$

MENJADI

$0{,}000000000976 = 9{,}76 \times 10^{-10}$

Representasi Floating Point

Konversi

- Konversi bilangan floating point berbasis deka ke basis biner harus dilakukan terlebih dahulu sebelum mengubah kedalam representasi floating point.

Contoh

- 3,75 → 11,11
- Biner 3 = 11
- Mengubah 0.75 menjadi biner:
- 0,75*2= 1,5 → ambil nilai didepan koma (1), lalu sisanya kalikan lagi dengan 2
- 0,5*2 = 1,0 → didapat bilangan didepan koma 1 dan sisanya 0

Representasi Floating Point

- Penulisan bilangan floating point dengan cara eksponensial dapat menyebabkan adanya kemungkinan sebuah bilangan ditulis dengan cara yang bermacam-macam

$300 = 0{,}3 * 10^3 = 3 * 10^2 = 30 * 10^1 = 30^0 * 100 = 3000 * 10^{-1} \ldots$

demikian juga pada bilangan biner

$11{,}11 = 11{,}11 * 2^0 = 1{,}111 * 2^1 = 0{,}1111 * 2^2$

Standarisasi Penulisan Bilangan

- Bentuk normalisasi:

$$\pm 1,bbb...b \times 2^{\pm E}$$

- Bit pertama significand selalu 1 sehingga tidak perlu disimpan dalam field significand. B adalah bilangan biner (1 atau 0).
- Untuk keperluan yang luas makandiadakan standar bagi representasi bilangan floating point ini, yaitu standar IEEE 754. standar ini juga mendefinisikan operasi aritmetikanya.

Format Penulisan Menurut Standar IEEE 754

1 bit	8 bit	23 bit
tanda	biased exponent	Fraction / significand

(a) Format tunggal (single format) 32 bit

1 bit	11 bit	52 bit
	biased exponent	Fraction / significand

(b) Format ganda (double format) 64 bit

- Pada format tunggal, bit paling kiri digunakan untuk representasi tanda 0, jika positif dan 1 jika negatif, sedangkan 8 bit berikutnya adalah pangkat (exponen) yang direpresentasikan dalam bentuk bias.
- Bagian 23 bit terakhir digunakan untuk menunjukkan bit dari bilangan fractionnya.

Contoh Konversi ke format IEEE

Bilangan= -113.3125

$113_{10} = 1110001_2$

sedangkan 0.3125 didapat:

0.3125 × 2=0.625	0
0.625 × 2 =1.25	1
0.25×2=0.5	0
0.5×2=1.0	1

jadi $113.3125_{10} = 1110001.0101_2$.

di normalisasi menjadi: $1110001.0101 \times 2^0 = 1.1100010101 \times 2^6$

Contoh Konversi ke format IEEE

karena mantisa terdiri dari 23 bit maka akan menjadi 11000101010000000000000.

Bilangan pemangkat adalah 6 maka dirubah ke bias dengan cara 6 + 127 = 133 = 10000101_2.

Karena negatif maka bilangan tanda = 1

1	10000101	11000101010000000000000

Sama dengan

$C2E2A000_{16}$.

Aritmetika Floating Point

Penambahan dan pengurangan

- a. periksa bilangan-bilangan nol
- b. ratakan significand
- c. tambahkan atau kurangkan significand
- d. normalisasi hasilnya

contoh :

123×10^2 $\quad\rightarrow\quad$ 123×10^2

456×10^{-2} + $\quad\rightarrow\quad$ $0{,}0456 \times 10^2$

$123{,}0456 \times 10^2$

TERIMA KASIH