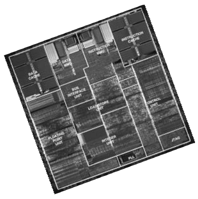
Luento 6

Tietokoneen rakenne

Tietokone-aritmetiikka

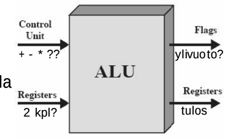


Stallings: Ch 9

- Kokonaislukuesitys
- Kokonaislukuaritmetiikka
- Liukulukuesitys
- Liukulukuaritmetiikka

ALU: Aritmeettis-Looginen Yksikkö

- ALU = Aritmetic Logic Unit
- Suorittava yksikkö, tiedon käsittely
  - Kokonaisluku ja liukulukuaritmetiikkaa
  - Vertailut, sivuttaissiirrot
  - Bittien kopiointi rekisteristä toiseen
  - Osoitelaskenta: Hypyt, muistiviittaukset
- Input
  - Yleensä kaksi operandia sisään
  - Rekistereistä (ja muistista)
- Operatio
  - Usein käskyrekisterin perusteella
- Output
  - Rekisteriin/Muistiin/PSW:hen



(Sta06 Fig 9.1)

Tietokoneen rakenne

Kokonaislukujen esitys

Kokonaislukuesitys (Integer Representation)

- Arvo binäärimuodossa, bittijonona
- ”Merkin” paino määräytyy paikan mukaan

57 = $5*10^1 + 7*10^0$
= 32 + 16 + 8 + 1
= $1*2^5 + 1*2^4 + 1*2^3 + 0*2^2 + 0*2^1 + 1*2^0$
= 0011 1001
= <u>0x</u>39
= $3*16^1 + 9*16^0$ **heksadesimaaliesitys**

- Eniten merkitsevä bitti / vähiten merkitsevä bitti
  - MSB, most significant bit
  - LSB, Least significant bit

Kokonaislukuesitys (Integer Representation)

- Entä negatiiviset arvot?
  - Etumerkki-suuruus
  - 2:n komplemettimuoto

-57 = <u>1</u>011 1001

etumerkki

-57 = <u>1</u>100 0111

- Tietokoneet käyttävät 2:n komplementtia
  - Ei erikseen +0 ja -0
  - Laskuissa ei tarvitse erikseen huomioida etumerkkiä
  - Vähennyslasku voidaan suorittaa yhteenlaskuna!
  - Helpompi laitteistolle

+2 = 0000 0010
+1 = 0000 0001
0 = 0000 0000
-1 = 1111 1111
-2 = 1111 1110

2:n komplementti

- Esimerkki
  - 8-bittinen esitys, esitä arvo -57

57 = 0011 1001	itseisarvo
1100 0110	invertoi bitit (1:n komplementti)
1100 0110	
1	lisää 1
1100 0111	2:n komplementtimuoto

Hylkää mahd. ylivuotava bitti

- Laajentuu helposti esim. 16-bittiseksi

57 = <u>0</u>011 1001 = <u>0000 0000</u> <u>0</u>011 1001
-57 = <u>1</u>100 0111 = <u>1111 1111</u> <u>1</u>100 0111

sign extension

2:n komplementti

- Arvoalue: $-2^{n-1} \dots 2^{n-1} - 1$

8 bits: $-2^7 \dots 2^7-1 = -128 \dots 127$
32 bits: $-2^{31} \dots 2^{31}-1 = -2\,147\,483\,648 \dots 2\,147\,483\,647$

- Yhteenlaskun ylivuoto helppo havaita
  - Ei ylivuotoa, jos erimerkkiset yhteenlaskettavat
  - Ylivuoto, jos samanmerkkiset yhteenlaskettavat ja tuloksen merkki eri kuin yhteenlaskettavien merkki

```
  57 = 0011 1001
+ 80 = 0101 0000
----------------
 137 = 1000 1001    Ylivuoto!
```


2:n komplementti

- Vähennyslasku yhteenlaskuna!
  - Unohda etumerkki, käsittele etumerkittöminä!
  - Ensin 2:n komplementti vähennettävästä, sitten add
  - Helppo laitteisto

```
-3 =  1101          3= 0011
+1 =  0001
-2 =  1110          1100
                       1
                    1101   -3 2:n komplementtiesityksessä
```

  - Tarkistus
    - Tuliko ylivuoto?
    - Merkki = 1, siis negatiivinen
    - Itseisarvo: invertoi bitit ja lisää 1

(Sta06 Table 9.1)

Tietokoneen rakenne

Kokonaisluku-aritmetiikkaa

- Negaatio
- Yhteen/vähennyslasku
- Kertolasku
- Jakolasku

Negaatio = 2:n komplementti

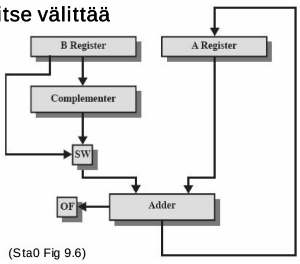
- 1: invertoi kaikki bitit
- 2: lisää 1
- 3: tarkista erikoistilanteet
  - Jätä ylivuotobitti huomiotta
  - Muuttuiko merkki?
    - Pienimmälle luvulle ei negaatiota
    - Ellei, aiheuta poikkeus
- Helppo laitteisto

```
-57 = 1100 0111
      0011 1000
              1
      0011 1001
= 57

-128 = 1000 0000
       0111 1111
               1
       1000 0000
```


Yhteenlasku (ja vähennyslasku)

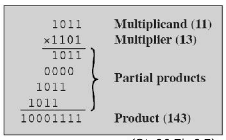
- Normaali binääriyhteenlasku
  - Jos vähennyslasku, muodosta vähennettävästä ensin komplementti, sitten yhteenlaskuna
- Ylivuotobitistä ei tarvitse välittää
  - Tarkkaile sensijaan summan merkkiä
- Helppo laitetoiminto
  - 2:n komplementtipiiri ja yhteenlaskupiiri



(Sta0 Fig 9.6)

Kokonaislukujen kertolasku

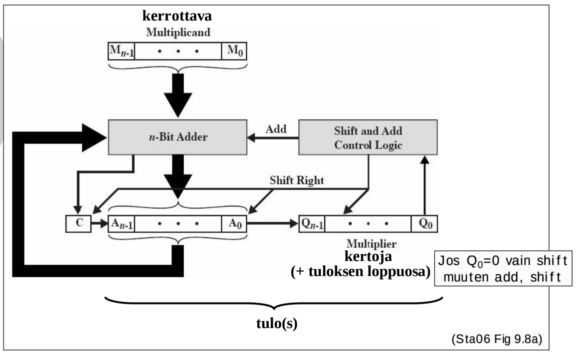
- Binääriluvuillakin kuten koulussa opittu
  - Helppo kertoa 0:lla tai 1:llä
- Laitteistolla?
  - Monimutkainen
  - Tarjolla useita algoritmeja
- Ylivuoto?
  - 32 b operandit ž tulos 64 b?



(Sta06 Fig 9.7)

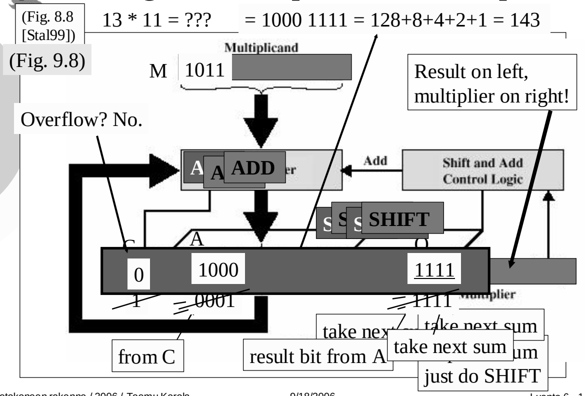
- Helppo laitteisto, jos etumerkittömiä
  - Vain monta yhteenlaskua
  - Tai sivuttaissiirtoa ja yhteenlaskua
    - siirto vasemmalle = kerro 2:lla
    - esim: 5 * => add, shift, shift, add

Etumerkittömien lukujen kertolasku

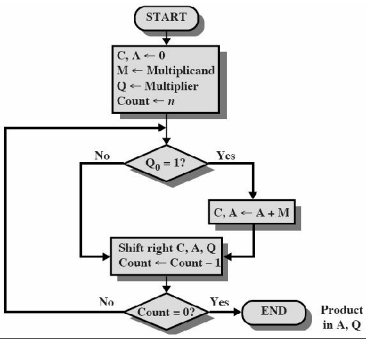


(Sta06 Fig 9.8a)

Unsigned Multiplication Example (19)



Etumerkittömien lukujen kertolasku



(Sta06 Fig 9.9)

Etumerkittömien kertolasku [Sta06 Fig 9.8a]

Q * M = 1101 * 1011 = 1000 1111 eli 13*11 = 143

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	

(b) Example from Figure 9.7 (product in A, Q) (Sta06 Fig 9.8b)

Negatiivisten kertolasku?

- Ed. algoritmi ei toimi negatiivisille luvuille
- Voisi tehdä näin
  - u muuta operandit positiivisiksi kokonaisluvuiksi
  - v käytä ed. algoritmia
  - w tutki operandien merkki, muuta tulos tarvittaessa komplementtimuotoon
- Parempia ja nopeampia tapoja olemassa

Boothin Algoritmi

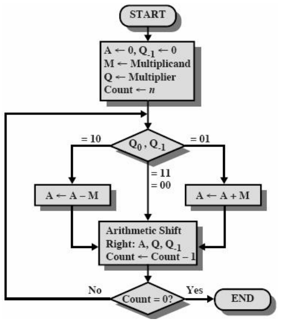
- Huomio edell. algoritmista
  - Yhteenlasku vain (aina), kun kertojassa esiintyy 1
- Boothin algoritmin idea (tehostus)
  - Yhdistä vierekkäiset 1:set yhdeksi köntäksi
  - Tee köntälle yksi yhteenlasku ja yksi vähennyslasku
  - Esim. 7*x = 8*x +(–x)
    111*x = 1000*x +(-x) =
    shift, shift, shift, complement, add

5 * 7 = 0101 * 0111
= 0101 * (1000-0001) ⇨

00101000 40
11111011 -5
100100011 = 35

- Toimii 2:n komplementtimuodoille, myös negatiivisille!

Boothin Algoritmi

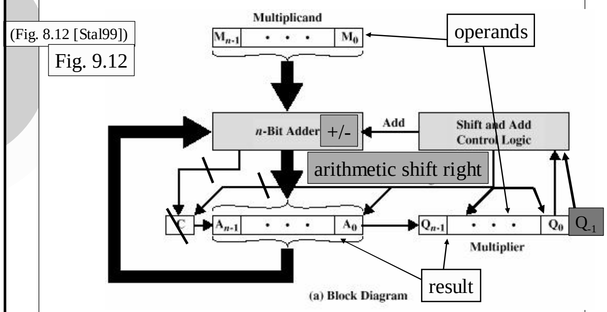


10 = könttä alkoi
11 = könttä jatkuu
01 = könttä loppui

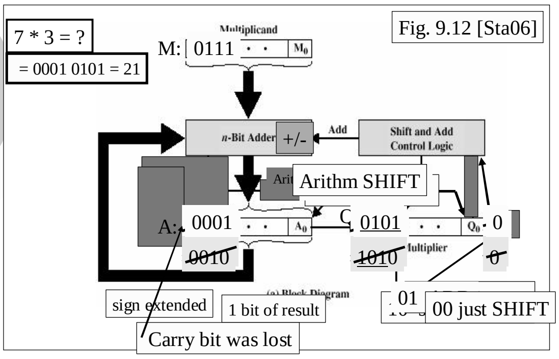
Arithmetic Shift Right:
= täytä etumerkillä

(Sta06 Fig 9.12)

Booth's Algorithm for Twos Complement Multiplication



Booth's Algorithm Example (15)



Boothin Algoritmi, esim.

Sta06 Fig 9.12

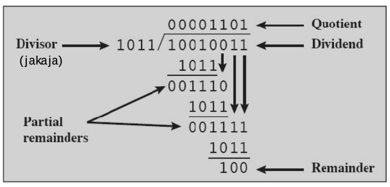
Q * M = 0011 * 0111 = 0001 0101 eli 3*7 = 21

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

(Sta06 Fig 9.13)

Kokonaislukujen jakolasku

- Binääriluvuillakin kuten koulussa opittu
  - Helppo: osamäärään tulee vain 0:ia ja 1:siä

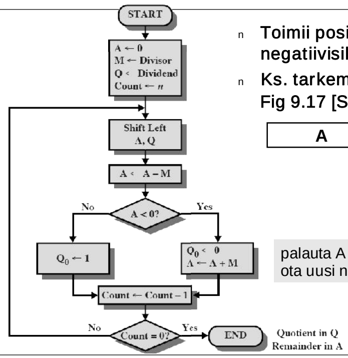


osamäärä
jaettava
jako-
jäännös

- Laitteistototeutus vastaavasti kuin kertolaskussa
  - Siirto vasemmalle = uusi numero mukaan

(Sta06 Fig 9.15)

Kokonaislukujen jakolasku



- Toimii positiivisilla luvuilla, negatiivisille lisävirittelyjä
- Ks. tarkemmin kirjan esimerkki Fig 9.17 [Sta06]

A	Q	Q_0

SHL

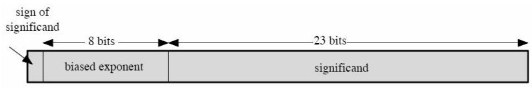
palauta A ennalleen ja ota uusi numero "alas"

Sta06 Fig 9.16

Tietokoneen rakenne

Liukulukuesitys

Liukulukuesitys



- Merkitsevät numerot ja suuruusluokka
- Normeerattu muoto
  - pistettä edeltävä numero > 0

$-0.000\ 000\ 000\ 123 = -1.23 * 10^{-10}$

$0.123 = +1.23 * 10^{-1}$

$123.0 = +1.23 * 10^{2}$

$123\ 000\ 000\ 000\ 000 = +1.23 * 10^{14}$

IEEE 754 Liukulukuformaatit

Parameter	Single	Single Extended	Double	Double Extended
Word width (bits)	32	≥ 43	64	≥ 79
Exponent width (bits)	8	≥ 11	11	≥ 15
Exponent bias	127	unspecified	1023	unspecified
Maximum exponent	127	≥ 1023	1023	≥ 16383
Minimum exponent	–126	≤ –1022	–1022	≤ –16382
Number range (base 10)	10^{-38}, 10^{+38}	unspecified	10^{-308}, 10^{+308}	unspecified
Significand width (bits)*	23	≥ 31	52	≥ 63
Number of exponents	254	unspecified	2046	unspecified
Number of fractions	2^{23}	unspecified	2^{52}	unspecified
Number of values	1.98×2^{31}	unspecified	1.99×2^{63}	unspecified

* not including implied bit

(Sta06 Table 9.3)

32-bittinen liukulukuesitys

- 1 b etumerkille
  - 1 = "-", 0 = "+"
- 8 b exponentille
  - Ei erikseen etumerkkiä, vaan erillinen nollataso (bias)
    - Esim. Exp=5 g talleta 127+5, Exp=-5 g talleta 127-5
- 23 b mantissalle (significant)
  - Normeeratussa muodossa binääripistettä edeltävä numero aina 1, ei talleteta (piilobitti, Zuse Z3 1939)
- Binäärimuodossa esitetyn liukuluvun arvo

  $-1^{\text{Sign}} * 1.\text{Mantissa} * 2^{\text{Exponent}-127}$

Esimerkkejä

$23.0 = +10111.0 * 2^0 = +1.0111 * 2^4 = ?$

127+4=131

0	1000 0011	011 1000 0000 0000 0000 0000
sign	exponent	mantissa

$1.0 = +1.0000 * 2^0 = ?$

0+127 = 127

0	0111 1111	000 0000 0000 0000 0000 0000
sign	exponent	mantissa

Esimerkkejä

0	1000 0000	111 1000 0000 0000 0000 0000
sign	exponent	mantissa

X = ?

$X = (-1)^0 * 1.1111 * 2^{(128-127)}$

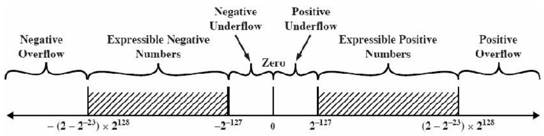
$= 1.1111_2 * 2$

$= (1 + 1/2 + 1/4 + 1/8 + 1/16) * 2$

$= (1 + 0.5 + 0.25 + 0.125 + 0.0625) * 2$

$= 1.9375 * 2$ $\boxed{= 3.875}$

Liukulukujen tarkkuudesta (32b)



- Arvoalue
  - 8 b eksponentti g 2^{-126} ... 2^{127} ~ -10^{-38} ... 10^{38}
- Tarkkuus
  - 24 b mantissa g 2^{24} ~ 1.7 * 10^{-7} ~ 6 desimaalia
  - Parempi tarkkuus pienille luvuille ilman normalisointia

IEEE 754 Erityismerkitykset

	Single Precision (32 bits)			
	Sign	Biased exponent	Fraction	Value
positive zero	0	0	0	0
negative zero	1	0	0	–0
plus infinity	0	255 (all 1s)	0	∞
minus infinity	1	255 (all 1s)	0	$-\infty$
quiet NaN	0 or 1	255 (all 1s)	$\neq 0$	NaN
signaling NaN	0 or 1	255 (all 1s)	$\neq 0$	NaN
positive normalized nonzero	0	$0 < e < 255$	f	$2^{e-127}(1.f)$
negative normalized nonzero	1	$0 < e < 255$	f	$2^{e\ 127}(1.f)$
positive denormalized	0	0	$f \neq 0$	$2^{e-126}(0.f)$
negative denormalized	1	0	$f \neq 0$	$-2^{e-126}(0.f)$

Not a Number

Double Precision vastaavasti

(Sta06 Table 9.4)

NaN: Not a Number

Operation	Quiet NaN Produced by
Any	Any operation on a signaling NaN
Add or subtract	Magnitude subtraction of infinities: $(+\infty)+(-\infty)$ $(-\infty)+(+\infty)$ $(+\infty)-(+\infty)$ $(-\infty)-(-\infty)$
Multiply	$0 \times \infty$
Division	$\frac{0}{0}$ or $\frac{\infty}{\infty}$
Remainder	x REM 0 or ∞ REM y
Square root	$\sqrt{x}$ where $x < 0$

(Sta06 Table 9.6)

Tietokoneen rakenne

Liukulukuaritmetiikkaa

- IEEE-754 Standardi
- Yhteen/vähennyslasku
- Kertolasku
- Jakolasku

Liukulukuaritmetiikka

- Laskentaa varten leveämpiä työrekistereitä
  - Guard bits
  - Enemmän merkitseviä bittejä mm. mantissalle
  - Käytetään myös normeeraamattomia muotoja
- Yhteen- ja vähennyslasku
  - Enemmän välivaiheita kuin kerto/jakolaskussa
  - Operandeille ensin sama eksponentti
    - Toisen normeeraus "purettava" – tarkkuutta häviää
  - Tulos voi vaatia normeerauksen
- Kerto- ja jakolasku
  - Mantissa ja eksponentti käsiteltävä erikseen

Liukulukuaritmetiikka

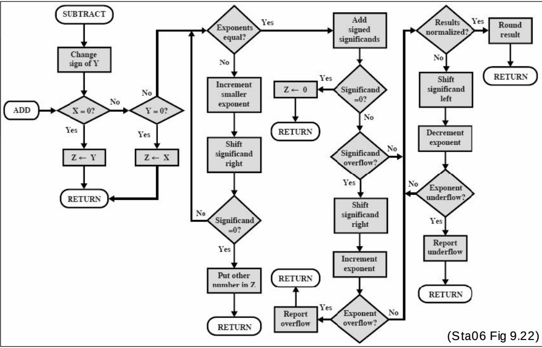
Floating Point Numbers	Arithmetic Operations
$X = X_S \times B^{X_E}$ $Y = Y_S \times B^{Y_E}$	$\left.\begin{array}{l} X + Y = \left(X_s \times B^{X_E - Y_E} + Y_s\right) \times B^{Y_E} \\ X - Y = \left(X_s \times B^{X_E - Y_E} - Y_s\right) \times B^{Y_E} \end{array}\right\} X_E \leq Y_E$ $X \times Y = \left(X_s \times Y_s\right) \times B^{X_E + Y_E}$ $\frac{X}{Y} = \left(\frac{X_s}{Y_s}\right) \times B^{X_E - Y_E}$

(Sta06 Table 9.5)

$$X = 0.3 \times 10^2 = 30$$
$$Y = 0.2 \times 10^3 = 200$$

$$X + Y = (0.3 \times 10^{2-3} + 0.2) \times 10^3 = 0.23 \times 10^3 = 230$$
$$X - Y = (0.3 \times 10^{2-3} - 0.2) \times 10^3 = (-0.17) \times 10^3 = -170$$
$$X \times Y = (0.3 \times 0.2) \times 10^{2+3} = 0.06 \times 10^5 = 6000$$
$$X \div Y = (0.3 \div 0.2) \times 10^{2-3} = 1.5 \times 10^{-1} = 0.15$$

Yhteen- ja vähennyslasku



(Sta06 Fig 9.22)

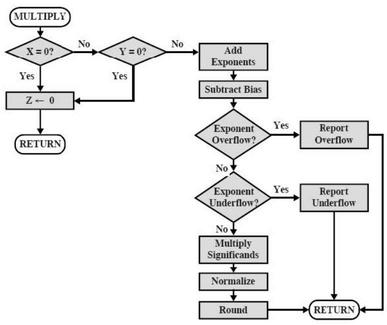
Erikoistilanteita

- **Eksponentin ylivuoto** (Hyvin suuri luku)
  - Arvoksi ∞ tai -∞ vai
  - Aiheuta poikkeus ohjelmoitava optio
- **Eksponentin alivuoto** (Olemattoman pieni luku)
  - Arvoksi 0 (tai aiheuta poikkeus) ohjelmoitava optio
- **Mantissan ylivuoto**
  - Yhteenlaskun tuloksena mantissa, jossa binääripisteen edellä useita numeroita
  - Normeeraa!
- **Mantissan alivuoto**
  - Yhteiseen eksponenttiin siirtyminen voi aiheuttaa merkitsevien bittien katoamista
    (entä, jos kaikki merkitsevät menee?)
  - Pyöristä?

Pyöristys

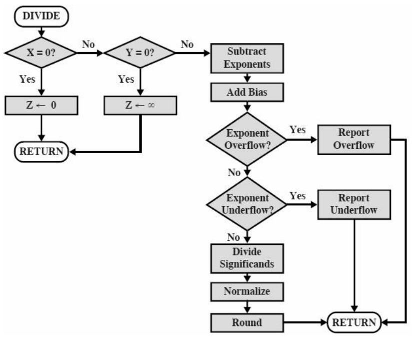
- Esimerkki
  - Arvo neljän desimaalin tarkkuudella 3.1234, -4.5678
  - Esittämiseen käytössä vain 3 desimaalia
  - Normaalien pyöristyssääntöjen mukaan lähimpään esitettävissä olevaan 3.123, -4.568
  - Aina ∞ kohti 3.124, -4.567
  - Aina -∞ kohti 3.123, -4.568
  - Aina 0 kohti 3.123, -4.567
- Esim. Intel Itanium -laitteisto tukee näitä kaikkia

Kertolasku



(Sta06 Fig 9.23)

Jakolasku



(Sta06 Fig 9.24)

Kertauskysymyksiä

- Miksi käytetään 2:n komplementtimuotoa?
- Miten 2:n komplementtiesitys laajenee "suurempaan tilaan" (esim. 8b esitys ž 16 b:n esitys)?
- Millainen on yksinkertaisen tarkkuuden liukuluvun esitysmuoto?
- Milloin tulee liukuluvun alivuoto?