SOS

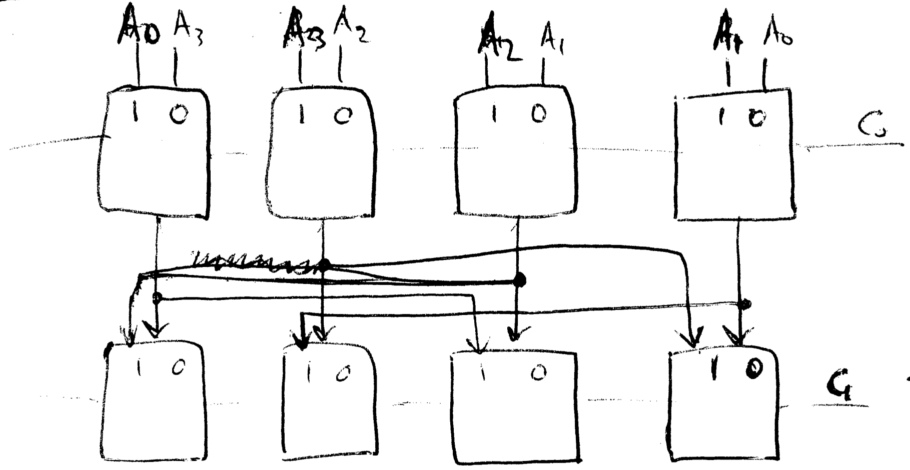
3.2

πολυπλέκτες
ολισθητές 4 bits

κυκλική ολίσθηση
αριθμητική ολίσθηση) προς τα δεξιά

Λύση

κυκλική ολίσθηση



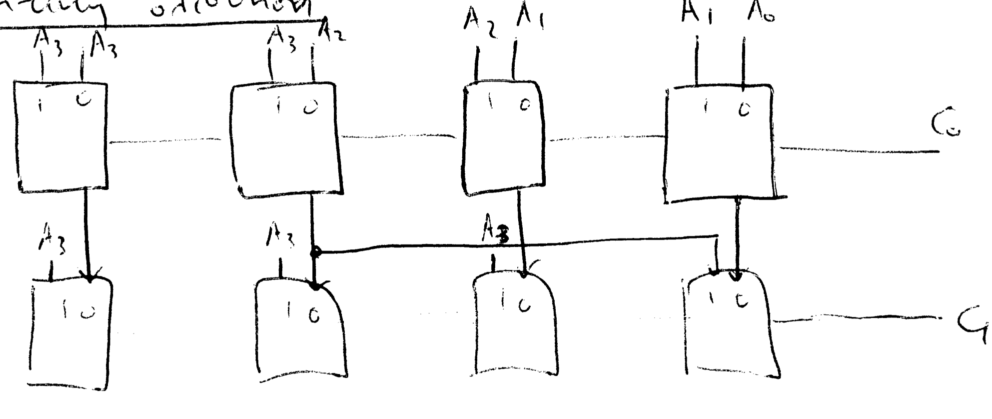
4 bits
⇓
3 ολισθήσ…

1 θέση ολίσθησης

2 θέσεις ολίσθησης

επίπεδα ολίσθησης [1, 2, 4

αριθμητική ολίσθηση



→

2/ $\boxed{\text{16 bit adders}}$

<u>a+b</u>

α) 1111 1111 1000 0110

1111 1111 1100 0000

$\cancel{1}$ 0000 0000 0100 0110

2's

(επειδή 2's το κρατούμενο αγνοείται)

$k_{16} = 1$, $k_{15} = 1$

$Y = 1 \oplus 1 = 0$ (<u>όχι</u> υπερχείλιση)

<u>c+d</u>

0000 0000 0111 1100

\+ 0000 0000 0100 0001

$\cancel{0}$ 0000 0000 1011 1101

$Y = 0 \oplus 0 = 0$ (<u>όχι</u> υπερχείλιση)

<u>a+c</u>

1111 1111 1000 0110

\+ 0000 0000 0111 1100

0000 0000 0000 0010

$Y = 1 \oplus 1 = 0$

<u>b+d</u>

1111 1111 1100 0000

\+ 0000 0000 0100 0001

0000 0000 0000 0001

$Y = 1 \oplus 1 = 0$

→

Μέσος # κύκλων ρολογιού ανά εντολή

$$\text{MKE (ή MKPE)} = \frac{A \cdot 300 + B \cdot 20 + \Gamma \cdot 50}{300+20+50}$$

$$= \frac{XE}{5_{ns}(300+50+20)}$$

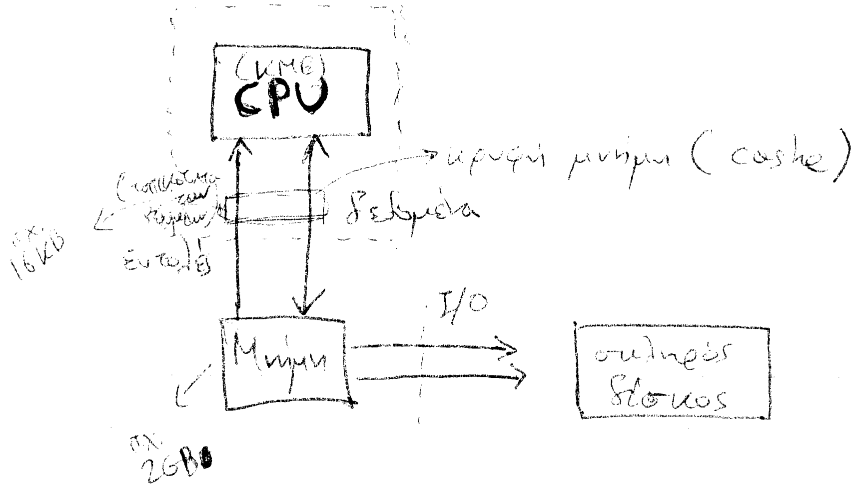
Μέσος χρόνος προσπέλασης του συστήματος μνήμης

$$\text{MX}\Pi = \frac{\left[300\left(\frac{XEE_A}{5}-4\right) + \left(\frac{XEE_B}{5}-7\right)\cdot 20 + 50\left(\frac{XEE_\Gamma}{5}-10\right)\right]5}{300(1+0)+20(2+1)+50(3+2)}$$

Vast or Fast?

17/10/2008

Αρχιτεκτονική Υπολογιστών Ι



KPE_i → κύκλοι ρολογιού μιας εντολής εκτέλεσης i

$KPE_i * T_{clock}$ → η μία εντολή

Ενώ

→ πλήθος που δίνουμε την εντολή

$(KPE_i * \Pi_i) * T_{clock}$

Ενώ για πολλές εντολές

$XE = \sum_{i=1} (KPE_i * \Pi_i) \cdot T_{clock}$ → Χωρίς την επιβάρυνση της μνήμης

execution time

$KPE_i(KME) + KPE_i(\text{κύρια μνήμη})$ → Όταν λαμβάνω υπόψη τη κύρια μνήμη ευστοχίας → E %

(συνέχεια τύπου...)

$KPE_i(KME) + \underbrace{KPE_i(\text{κύρια μνήμη})}$

$E * KPE_i(\text{κορυφή}) + (1-E) * KPE_i(\text{κύρια})$

cashe : α) τύπος εντολών
β) τύπος δεδομένων
γ) τύπος και των δύο

execution time

XE (αυτό που βρήκαμε προηγουμένως)

$XE_{\text{response time}} = XE_{\text{περιφ}} + XE_{\text{εντολών συστήματος}} + \underbrace{I/O_{\text{χρόνος}}}$

(+ πολλά προγράμματα που εκτελούνται)

OS

στέλνουμε προσβάσεις στο δίσκο

Απόδοση

MIPS: Million instructions per secs ⟷ MFLOPS million Floating Operations per secs

C++

XBG

Power

πόσους κύκλους θέλει ο επεξεργαστής να την επεξεργαστεί

Άσκηση 1

	κύκλοι εντολής	πόσες φορές εκτελέστηκε
A	1	1500
B	2	2000
C	3	1000
D	5	500

Να βρεθεί ο μέσος όρος κύκλων για μία εντολή.

$$\text{ΜΚΡΕ} = \frac{1500*1 + 2000*2 + 3*1000 + 5*500}{\underbrace{1500+2000+1000+500}_{\text{πλήθος εντολών}}}$$

$= 2.2$ cycles

Άσκηση 2

Y_1	2 GHz	ΜΚΡΕ_1 1.2 cycles
Y_2	2.5 GHz	ΜΚΡΕ_2 1.6 cycles

Ποιος είναι ταχύτερος Η/Υ

$T = \frac{1}{f}$

$T_1 =$ αριθμός εντολών $* \text{ΜΚΡΕ}_1 * 0{,}5 \cdot 10^{-9} =$

$=$ αριθμός εντολών $* 0{,}6 \cdot 10^{-9}$ sec

Όμοια:

$T_2 =$ αριθμός εντολών $* 1{,}6 * 0{,}4 * 10^{-9}$ sec $=$

$=$ αριθμός εντολών $* 0{,}64 \cdot 10^{-9}$ sec

$T_2 > T_1 \rightarrow$ Άρα, ο 1ος Η/Υ πιο γρήγορος αφού εκτελεί τις εντολές σε λιγότερο χρόνο (λιγότερους κύκλους)

$T_2 > T_1 \Rightarrow f_2 < f_1$

make the common case fast

Amdahl

Άσκηση 3

30% χρόνου $\xrightarrow{\text{πολ/σμού}}$ 2x (T_A)

10% χρόνου $\xrightarrow{\text{διαίρεσης}}$ 4x (T_B)

Ποια επιλογή;

Λύση

Έστω T ο χρόνος προγράμματος

$$T = T_{πολ} + T_{διαιρ.} + T_{\text{άλλων εντολών}}$$

$$T = 0.3\,T + 0.1\,T + [1-(0,3+0,1)]\,T_{\text{άλλων εντολών}}$$

$$T_A = T_{A\,πολ} + T_{διαιρ.} + T_{\text{άλλων εντολών}}$$

0.6

Θα μείνει ίδιος όπως είναι.

$\frac{0.3T}{2} \to 0.15\,T$

$$T_A = (0.15 + 0.1 + 0.6)\,T = 0,85\,T$$

Όμοια:

$$T_B = \left(0,3T + \frac{0.1}{4}T + 0.6\right)T = 0,925\,T$$

$\Rightarrow$

$\Rightarrow T_B > T_A \Rightarrow f_B < f_A \Rightarrow$ ο A πιο γρήγορος (με το μάτι παράδοξο!)

8

24/10/20..

$\boxed{\text{Αρχιτ. 1}}$

$$XE = \sum_{i=1}^{N} [(KPE_i(KME) + KPE_i(\text{κύρια μνήμη}$$

$$\sum_{i=1}^{N} [(KPE_i(KME) * \Pi_i + \sum_{i=1}^{N} KPE_i(\text{κύρια μνήμη}) \cdot \Pi_i$$

διάρκεια κύκλων ρολογιού

εντολή	κύκλοι ρολογιού	bytes ρολογιού	bytes data	Πλήθος
Α	4	1	0	600
Β	7	1	1	300
Γ	7	2	1	10
Δ	10	3	0	100
Ε	10	3	2	10

T = 5 n sec

σελ. *

$\xrightarrow{i)}$ cache

$\xrightarrow{ii)}$ κύρια μνήμη

$\xrightarrow{iii)}$ OS

4η άσκηση

Y_1	30 κύκλοι
Y_2	20 κύκλοι

$$XE_{KME,Y_1} = 4\cdot 600 + 7\cdot 300 + 7\cdot 10 + 10\cdot 100 + 10\cdot 10 = $$
$$= 5670 \text{ κύκλοι}$$

$$XE_{KME,Y_2} = XE_{KME,Y_1}$$

KPE_i (μνήμης) = πλήθος προσπελάσεων * καθυστέρηση μνήμης

$$\sum_{i=1}^{5} KPE_i(\text{κύρια μνήμη}) * \Pi_i = 1 * 600 + 2\cdot 300 + 3\cdot 10 + 3\cdot 100 + 5\cdot 10 =$$

π.χ. ADD R, MEM

$$XE_{,Y_1} = (5670 + 1580 * 30) * \text{διάρκεια κύκλου} = 265,35 \mu s$$

$$XE_{,Y_2} = \quad '' \quad * 20 \qquad = 186,35 \mu s$$

$$XE = XE_{\text{προγ.}} + XE_{\substack{\text{εντολών} \\ \text{συστήματος}}} + I/O_{\text{χρόνο}}$$

→ MIPS

MFLOPS

i) διαφορετικής πολυπλοκότητας εντολές

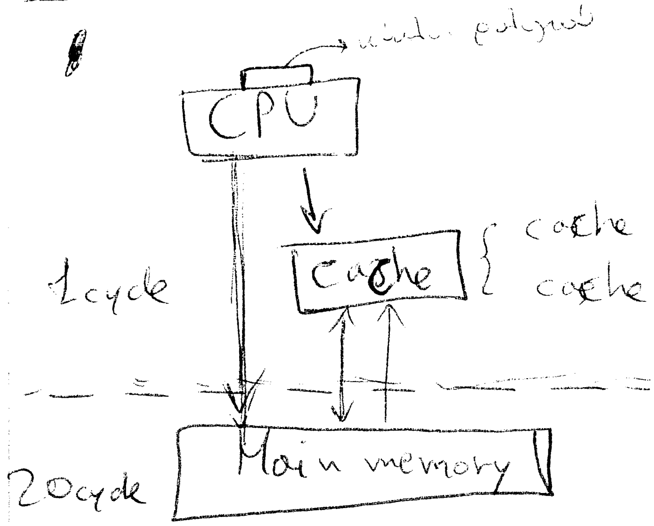
ii) έχουμε διαφορετικά σύνολα εντολών

---

benchmarks

$$\begin{tikzcd} X \arrow[r] \arrow[dr] & A \rightarrow \\ & B \rightarrow \end{tikzcd}$$

5η άσκηση



{ cache για δεδομένα / cache για εντολές } ουσιαστικά υπάρχουν δύο διαφορετικές cache για αυτό.

$\lambda = 0{,}9$ ρυθμός επιτυχίας αναζήτησης μνήμης

$T_{cache} = 1$ cycle

$T_{μνήμη} = 20$ ″

$T_{ΚΜΕ} = 5$ nsec

$$XEE_A = 4 + \underbrace{1}_{\text{πλήθος bytes}} * \lambda * \underbrace{1}_{\text{cache}} + 1 * (1-\lambda) * (\underbrace{20}_{\text{μνήμη}} + \underbrace{1}_{\text{cache}}) = 7 \text{ κύκλοι}$$

$$XEE_B = 7 + \underbrace{2}_{\substack{1 \;+\; 1 \\ \text{byte Data} \;\; \text{byte εντολής}}} \cdot \lambda * 1 + 2 * (1-\lambda) \cdot (20+1) = 13 \text{ ″}$$

B1 πίνακα σελ. *31

$XEE_{\Gamma} = 16$ κύκλοι

$XEE_{\Delta} = 19$ κύκλοι

Άρα, $XEE_E = 25$ κύκλοι

$$XE = (600 \cdot 7 + 300 \cdot 13 + 10 \cdot 16 + 100 \cdot 19 + 10 \cdot 25) \cdot 5ns$$

Άσκηση 6η

	cycles	bytes εντολή	bytes Data	Π πλήθος
A	4	1	1	300
B	4	2	1	10
Γ	16	3	2	100

	clock
Y_1	5nsec
Y_2	2,5nsec

50nsec η κύρια μνήμη

Λύση

$$XE(Y_i) = \left(\sum_{i=1}^{3}\left[KPE_i(KME) + KPE_i(\text{κύρια}) \right] * \Pi_i\right) * \text{κύκλος ρολογιού}$$

$$XEY_1 = \left(\left(4 + 2*10\right)\cdot 300 + \left(4+3*10\right)\cdot 10 + \left(10+5*10\right)\cdot 100\right)\cdot 5ns =$$

$$\boxed{Y_1 = \frac{50}{5} = 10\text{κ} \qquad Y_2 = \frac{50}{2,5} = 20\text{κ}}$$

$= 67700nsec$

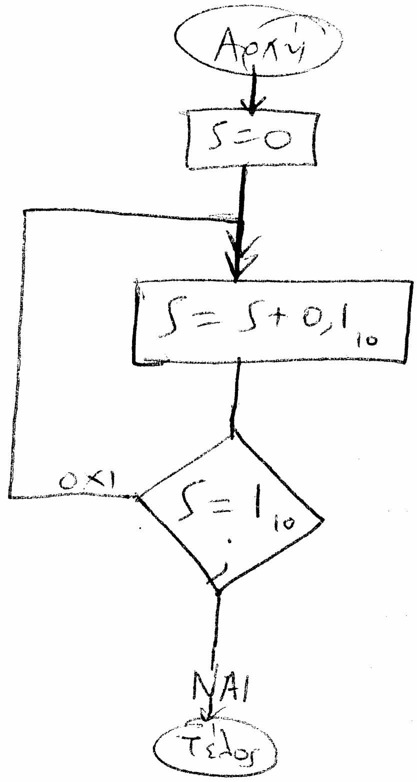
$XEY_2 = 62.100nsec$

$XEE_A = XE(KME) + XE(\text{κύρια})$

29/10/2008

Αρχιτ Υπολ I

$\underbrace{\_\,,\,\_\,\_\,\_\,\_\,\_\,\_\,\_}_{8\ bit}$



$0{,}1_{10} = 0{,}0001100$

$\downarrow$

$0{,}09375_{10}$

$\Downarrow$ Loop

$1{,}03125_{10}$

$$\frac{\beta^{\nu}-1}{\beta^{\lambda+1}} = \frac{\beta^{\nu}-1}{\beta^{\lambda}\beta} = \frac{\beta^{\nu}}{\beta^{\lambda+1}} - \frac{1}{\beta^{\lambda+1}} = \beta^{\nu-(\lambda+1)} - \beta^{-(\lambda+1)}$$

31/10/2008

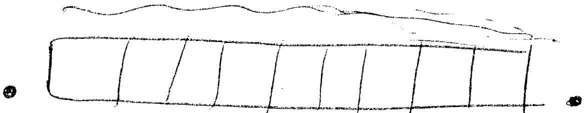
$\boxed{\text{Αρχιτ. Υπολ. Ι}}$

Αναπαράσταση Αριθμών

→ fixed point representation → πεπερασμένο πλήθος στοιχείων που χωρίζονται με υποδιαστολή

→ Floating point representation

ν bits



Έστω υποδιαστολή:
πριν LSB: (ακέραιο αριθμό)

β=2

$$0 < N < \beta^{\nu}-1$$

Έστω υποδιαστολή:
μετά MSB: (δεκαδικό αριθμό)

$$0 < N < 1-\beta^{-\nu}$$

π.χ.

01010.0100

βάρος 16 8 4 $2^1=2$ $2^0=1$ $2^{-1}=\frac{1}{2}$ $1/4$ $1/8$ $1/16$

$$0 \leq N \leq B^{\nu-(\lambda+1)} - B^{-(\lambda+1)}$$

$a^{\lambda} \longrightarrow$ ψηφίο που έχει την υποδιαστολή αριστερά.

πριν από το a^{λ}, βάζω την υποδιαστολή

~~$a_k \; a_{k-1} \; \ldots \; a_3 \; a_2 \; a_1 \; a_0 \bullet$~~

$$a_k \; a_{k-1} \; \ldots \; a_3 \; a_2 \; a_1 \; a_0 \bullet a_{-1} \; a_{-2} \; \ldots \; a_{-(p-1)} \; a_{-p}$$

ν bits

ν = 4

	1111.	111.1	11.11	0.1111
λ	-1	0	1	3
max	$2^{4-0}-2^{0}$	$2^{4-1}-2^{-1}$	$2^{4-2}-2^{-2}$	$2^{4-4}-2^{-4}$
	15	7,5	3,75	0,0375
ακρίβεια ή μικρότερος αριθμός	1	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{16}$

ο μικρότερος μη μηδενικός αριθμός

SOS

Όσο πιο δεξιά, αυξάνεται το εύρος των αριθμών αλλά μειώνεται η ακρίβεια των αριθμών.

10 bits } δεδομένα
└→ 8 bits } επεξεργαστής

Αν δεν χρειαζόμαστε τα 2 τελευταία ψηφία τότε απλά τα αποκόπτουμε.

10101110|01 ⟶ 10101110

π.χ.
$$\begin{array}{r} 123|00 \\ \times\ 154|00 \\ \hline \end{array}$$
0000

Όσα x bit κόβουμε μετά πρέπει να πολλαπλασιαστεί ο αριθμός με 2^x.

Στη συγκεκριμένη περίπτωση, αφού αποκόψαμε 2 bits τότε ο τελικός αριθμός θα είναι ο αποκομμένος πολλαπλασιασμένος επί $4 (=2^2)$.

<u>BCD</u>

Decimal	4 bits
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

όταν πατήσουμε στο keyboard
το 12 → 11~~00~~ → 0001 001

σε δυαδικό

σε BCD

Άσκηση 1η

999 $\rightarrow$ BCD $3*4 = 12$ bits

$\lceil \log_2 999 \rceil = 10$ bits

$\lceil \; \rceil$ ← ακέραιο

Παρατηρούμε ότι χρειαστήκαμε 20% περισσότερα bits

M decimal $\longrightarrow$ BCD : $4*K$ bits

K decimal ψηφία $\Longrightarrow$ 10^K ~~αριθμούς~~ αριθμούς 10 δυαδικά συνεχόμενους

$\lceil \log_2 10^K \rceil$ bits σε δυαδική μορφή

$$\left\lceil \frac{\log_{10} 10^K}{\log_{10} 2} \right\rceil = \left\lceil \frac{K}{0,3} \right\rceil$$

περισσότερα bits σε BCD $4K - \lceil K/0,3 \rceil$

$K/0,3$ περισσότερα bits σε BCD

δυαδικοί - BCD

$4K - \lceil K/0,3 \rceil$

100 $\longrightarrow$ Z;

$$\frac{\frac{K}{0,3}}{100} = \frac{4K - \lceil K/0,3 \rceil}{Z} \Rightarrow$$

$\longrightarrow$

$$\Rightarrow Z = \left(4k - \left\lceil \frac{k}{0,3} \right\rceil\right) \cdot \frac{100}{\left\lceil \frac{k}{0,3} \right\rceil} = \left(\frac{4k}{\left\lceil \frac{k}{0,3} \right\rceil} - 1\right) \cdot 100 \simeq$$

$$\simeq \left(\frac{4k}{k/0,3} - 1\right) \cdot 100 = (1,2-1) * 100 = \underline{\underline{20\%}}$$

BCD

0011	(BCD)		1000	(BCD)
+ 0100	(")		+ 0011	(")
0111	(")		1011	X

↳ δεν υπάρχει σε BCD

(12)

α)

~~0010~~ 0010 0100 0100 (244 → BCD)

ολίσθηση κατά μία θέση αριστερά.

1η (τ.κ.)

~~0100 1000~~
0100 1000 1000 (488 → BCD)

α' τρόπος:

0010 0111 0100 (BSD : 274)

2η ολίσθηση (τ.κ.)

0100 1110 1000 ↳ υπερέβη το 9

+1 ~~0~~ $-1010_2 = -10_{10}$

0101 0100 1000 (274 * 2 : BSD)

17
5
12
↳ −10

ΣΟΣ

Όταν κάνουμε ολίσθηση στα αριστερά τότε ο αρχικός αριθμός πολ/αζε επί 2. Ενώ όταν ολισθαίνει προς τα δεξιά, ο αρχικός αριθμός διαιρείται δια 2.

Β' τρόπος:

2^4 2^3 2^2 2^1 2^0

```
  0100 1110 1000
     + 0110        <- 6_10
  --------------
  0101 0100 1000
```

$$
\begin{array}{rl}
10000 & = 16_{10} \\
-1010 & = -10_{10} \\
\hline
 & +6_{10}
\end{array}
$$

7/11/2008

$\boxed{\text{Αρχιτεκτονική I}}$

ολίσθηση

πριν 0010 0111 0100

μετά 0100 1110 1000 (BCD) λάθος

+0110

0101 0100 1000 σωστό

ολίσθηση

πριν 0010 0010 1001

μετά 0100 0101 0010 λάθος

+0110

0100 0101 1000 σωστό

Από BCD σε πολλαπλασιαστή : προβλήματα

→ όταν δημιουργείται κρατούμε προσθέτουμε +6 στο προηγούμενο

→ από και 10_{10} και πάνω προσθέτουμε +6

2.2

binary → BCD

$100111010_2 = 314_{10}$

αρχικά

BCD				binary
0000	0000	0000	←	100111010

ολισθαίνουμε κατά 1 θέση αριστερά

0000 0000 0001 ← 00111010 0

ολισθαίνουμε κατά 1 θέση αριστερά

0000 0000 0010 ← 01110100 0

ολισθαίνουμε κατά 1 θέση αριστερά κ.ο.κ.

0000 0000 0100 ← 111010000

0000 0000 1001 ← 110100000

STOP → βλέπω πρόβλημα

0000 0001 0011 ← 101000000

$\quad\quad\quad\quad\quad\quad +0110$

0000 0001 1001

συνεχίζουμε να ολισθαίνουμε →

~~0001 0011~~

→

22

0000 0011 0011 ← 010 . . .

πάλι πρόβλημα

+0110

συνεχίζουμε ολίσθηση 0000 0011 1001 ← 010

0000 0111 0010 ← 10 . . .

πάλι πρόβλημα άρα +6 στο σφαλμένο

+0110

0000 0111 1000 ← 10 . . .

συνεχίζουμε ολίσθηση

1ο πρόβλημα

0000 1110 0001 ← 0 . . .

2ο πρόβλημα

άρα

+0110 +0110

0001 0101 0111 ← 0 . . .

τελευταία ολίσθηση

1ο 2ο

0010 1010 1110 ← ~~X~~

πρόβλημα

οπότε +0110 +0110

0011 0001 0100

3 1 4 (BCD)

τα κρατούμενα αυτά προκύπτουν από πρόσθεση και όχι από ολίσθηση, επομένως, ~~[illegible]~~ δεν προσθέτουμε +6

BCD BCD

$$\begin{array}{cc} 16 & 8\,4\,2\,1 \\ 0000 \xleftarrow{+10} & 1000 \\ \text{ολίσθηση}\ 0001 & 0000 \end{array}$$

οπότε λείπει ένα +6 αφού από $8 \xrightarrow{\text{πάει}} 16$

2.3

0110 1000 (68_{10} BCD)

ολίσθηση δεξιά

0011 0100 (34_{10} BCD)

π.χ. 2:

10 / 8 4 2 1

0101 1000 (58 BCD)

0010 1100 δεν υπάρχει BCD ! Λάθος

από 10 $\xrightarrow{\text{σε}}$ 8 ενώ εμείς θέλαμε από 10 $\xrightarrow{\text{σε}}$ 5 αφού διαιρούμε δια 2.

επομένως, το κάνουμε 8 και αφαιρούμε 3 ώστε να γίνει 5.

Επομένως,

0010 1100^{12} $-3 = 9$

−0011

0010 1001 (29 BCD)

Από BCD σε δυαδικό (ολίσθ. δεξιά) : ⟶ αν πάει από τη μία τετράδα στην επόμενη τότε , η επόμενη −3

2.5 (δεν υπάρχει 2.4)

0101 0001 1001

BCD ⟶ binary

1ο προβλ. 2ο

LSB

ολίσθηση 1 δεξιά

0010 1000 1100

δάνεια

−0011 −0011

ολίσθηση 1 δεξιά

0010 0101 1001

0001 0010 1100 1

−0011

0001 0010 1001 1

11

διόρθωση

0000 1001 0100

0011

0000 0110 0100 → 0 011

0000 0011 0010 → 0011

0000 0001 1001

− 0011

// // 0110 → 00011

// // 1011

− 0011

// // 1000 → 0 0011

// // 0000 → 000011

// // 0010 → 0000011

// // 0001 → 1 0000011

MSB LSB

2.5 Ⓐ

β)

→ περικοπή

→ στρογγυλοποίηση

βάση β

κ κλασματικά ψηφία

$0 \le a_i \le \beta - 1$

$N = a_\mu \, a_{\mu-1} \ldots a_1 \, a_0 \, a_{-1} \ldots a_{-\kappa} \, a_{-(\kappa+1)} \ldots$

υποδιαστολή

• Μέθοδος περικοπής: έστω ότι επιτρέπει μέχρι κ κλασμ. ψηφία

$N = a_\mu \, a_{\mu-1} \ldots a_1 \, a_0 \, a_{-1} \ldots a_{-\kappa} \Big\} a_{-(\kappa+1)} \ldots$

μέχρι κ κλασματικά ψηφία

υποστηρίζει 4 κλασμ. ψηφία.

τότε,

$0.1435 | \times \times \times \times \times \times \ldots$

μέγιστο σφάλμα

βαρύτητα $2^{-(k+1)}$

$$0.00\ldots a_{-k}\, a_{-(k+1)}\, a_{-(k+2)}\ldots$$

$$\Sigma = (\beta-1)\cdot\beta^{-(k+1)} + (\beta-1)\cdot\beta^{-(k+2)} + (\beta-1)\cdot\beta^{-(k+3)} + \ldots =$$

$$= (\beta-1)\beta^{-(k+1)}\left(1 + \frac{1}{\beta} + \frac{1}{\beta^2} + \frac{1}{\beta^3} + \ldots\right)$$

$$(\beta-1)\,\beta^{-(k+1)}\; 1/(1-\tfrac{1}{\beta}) =$$

$$= (\beta-1)\,\beta^{-(k+1)}\left(1/1 - 1/\beta\right) = \beta^{-k}$$

$$\frac{a_0}{1-\omega}, \quad \omega = \frac{a_{k+1}}{a_k}$$

$$\omega = \frac{\frac{1}{\beta}}{1} = \frac{1}{\beta}$$

→

• μέθοδος στρογγυλοποίησης

π.χ.

$$\begin{array}{r} 0.12347 \\ +\ 0.00005 \\ \hline \end{array}$$

ό,τι βγει ~~ψ~~ και κρατάμε μόνο 5 ψηφία

π.χ.$_2$

$$\begin{array}{r} 0.124 \\ +\ 0.005 \\ \hline 0.129 \end{array}$$

αν όμως

$$\begin{array}{r} 0.125 \\ 0.005 \\ 0.130 \end{array}$$

14/11/2008

$\boxed{\text{Αρχιτ. Υπολ. Ι}}$

Ασκ. 256

β : βάση

$N = a_\mu a_{\mu-1} \ldots a_2 a_1 a_0 . a_{-1} a_{-2} \ldots a_{-\kappa} \mid a_{-(\kappa+1)} a_{-(\kappa+2)} \longrightarrow$

$x = 1.14 \mid x_1 x_2 x_3 \ldots \longrightarrow$

505 { ο μικρότερος αριθμός που θα προσθέτουμε ώστε το 1.14 να γίνει 1.15, είναι το μέγιστο σφάλμα. } 505

Δηλαδή,

α) 0,005 0 0 0 0 ....

ή

β) το 0,004 999999....

α)

συντελεστής

$N' = a_\mu a_{\mu-1} a_{\mu-2} \ldots a_2 a_1 a_0 . a_{-1} a_{-2} \ldots a_{-\kappa} \boxed{\beta/2} 0000$

βαρύτητα: $\beta^{-(\kappa+1)}$

$$E = (\beta/2) * \beta^{-(\kappa+1)} = \frac{\beta^{-\kappa}}{2}$$

β) $\left(\frac{\beta}{2} - 1\right)(\beta-1)(\beta-1)(\beta-1) \ldots$

$$\Sigma = \left((B/2)-1\right) B^{-(k+1)} + (B-1)\, B^{-(k+2)} + (B-1)\, B^{-(k+3)} + \dots =$$

$$= \left[(B/2)-1\right] B^{-k-1} + (B-1) B^{-k-2} \underbrace{\left(1 + \frac{1}{B} + \frac{1}{B^2} + \dots\right)}_{\text{γεωμετρική πρόοδος}} =$$

$$= \left[(B/2)-1\right] B^{-k-1} + B^{-k-1} = \frac{B^{-k}}{2}$$

Άθροισμα
Γεωμ. προόδου:

α : πρώτος όρος
ω : βήμα.

$$\frac{\alpha}{1-\omega}$$

Αριθμοί ~~Κινητής~~ Υποδιαστολής

$$Z = \underset{\text{συντελεστής}}{\Sigma} * B^{E}$$

B → βάση, E → εκθέτης

αναπαράσταση: $\nu + 1$ bits

- 1 bit το πρόσημο
- μ bits συντελεστή
- $\nu - \mu$ bits εκθέτη

- κ bits αριστερά της υποδιαστολής
- λ bits δεξιά

Το πρόβλημα !

$$X = 1.000 * 2^{0}$$

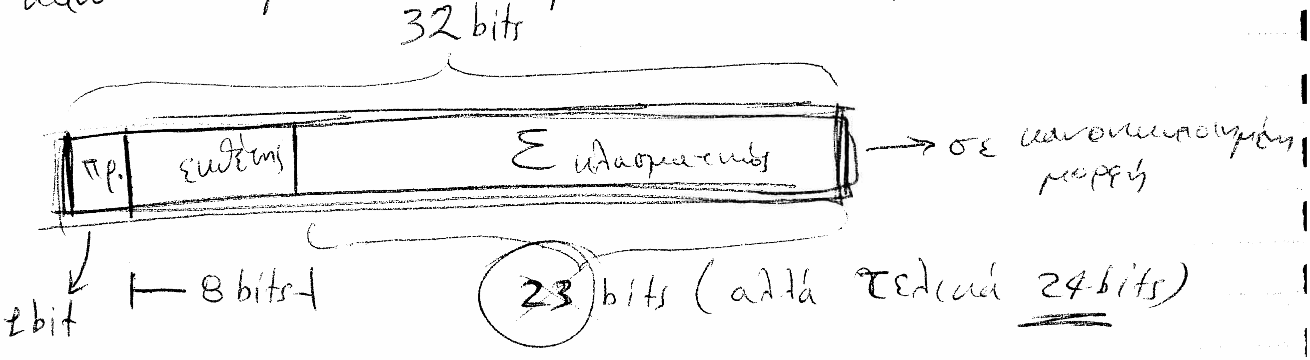
$$0.1000 * 2^{1}$$

} είναι ο ίδιος αριθμός

→ Ακριβώς επειδή είναι οι ίδιοι αριθμοί, με διαφορετική αναπαράσταση, υπάρχει το πρόβλημα της σύγκρισης.

$1.000 * 2^{0}$ — κανονικοποιημένη

$0.1000 * 2^{1}$ — μη κανονικοποιημένη



$1.000 \cdot 2^{-1}$ $\xrightarrow{2's}$ 11111110 (εκθέτης)

$1.000 \cdot 2^{+1}$ → 0000 0001 (εκθέτης)

} δύσκολη σύγκριση

πόλωση

$2^{-1} \rightarrow +127$: 0111.1110

$2^{+1} \rightarrow +127$: 1000.0000

} εύκολη σύγκριση

0000 0000 } min εκθέτης (με βάση την πόλωση)

1111 1111 } max εκθέτης (με βάση την πόλωση)

Πίνακας: IEEE (πρότυπο)

$N = NaN$ (Not a Number), αν $E = 255$ και $Σ_κ \neq 0$

$N = (-1)^{Π} * \infty$, αν $E = 255$ και $Σ_κ = 0$

$N = (-1)^{Π} * 2^{E-127} * (1.Σ_κ)$, αν $0 < E < 255$

$N = (-1)^{Π} * 2^{-126} * (0.Σ_κ)$, αν $E = 0$ και $Σ_κ \neq 0$

$N = (-1)^{Π} * 0$, αν $E = 0$ και $Σ_κ = 0$

Ασκήσεις:

261 εκθ. πεδίο $Σ_κ$

0.1000.0010.1100000....

πρόσημο, 2 bits + 21 bits

$= 130_{10}$

Άρα, παίρνουμε τον τύπο

$N = (-1)^{Π} * 2^{E-127} * (1.Σ_κ) = (-1)^0 \cdot 2^{130-127} \cdot (1.11_2) = 14$ α

β. 400 γ. $-2^{75} \cdot 7$

2.7

	Π	E	Εκ (23 bits)	
i)	0	00000000	0,0000 ...	→ 0
ii)	1	//	//	→ 0
iii)	0	11111111	0,0000 ...	→ $(-1^0)\cdot\infty = \infty$
iv)	1	//	//	→ $(-1)^1 \times \infty = -\infty$
v)	0	1111 1111	0011 ...	→ NaN
vi)	0	$\underbrace{10000000}_{128_{10}}$	1000	→ (κανονικοποιημένη) → 3_{10}
vii)	0	00000000	01000	→ (Μη κανονικοποιημένη μορφή) → $2^{-126} * 0,25$

2.8 a)

τα νούμερα άσχετα μεταξύ τους

32 bits → address bus

32 καταχωρητές

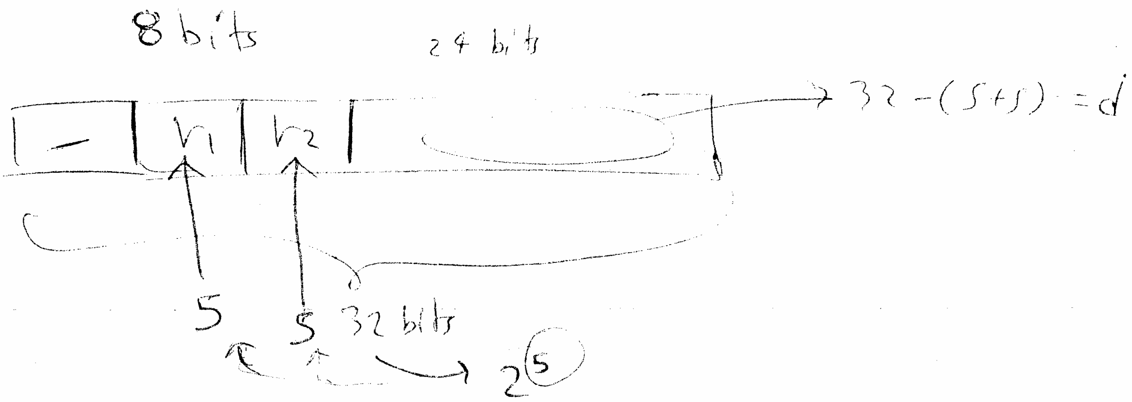
32 bits εντολές

8 bits opcode → κωδικός έτσι ώστε ο επεξεργαστής να καταλάβει τι κάνει η κάθε εντολή. Προφανώς, κάθε εντολή έχει το δικό της.

BRE r_1, r_2, d

if $(r_1 - r_2 = 0)$

$\{ΜΠ = ΜΠ + d;\}$



Επειδή 32 καταχωρητές

$d \leftarrow 14$ bits | ~~$ΜΠ + 2^{14}$~~ | $\boxed{ΜΠ - 2^{13} \quad ΜΠ + 2^{13} - 1}$

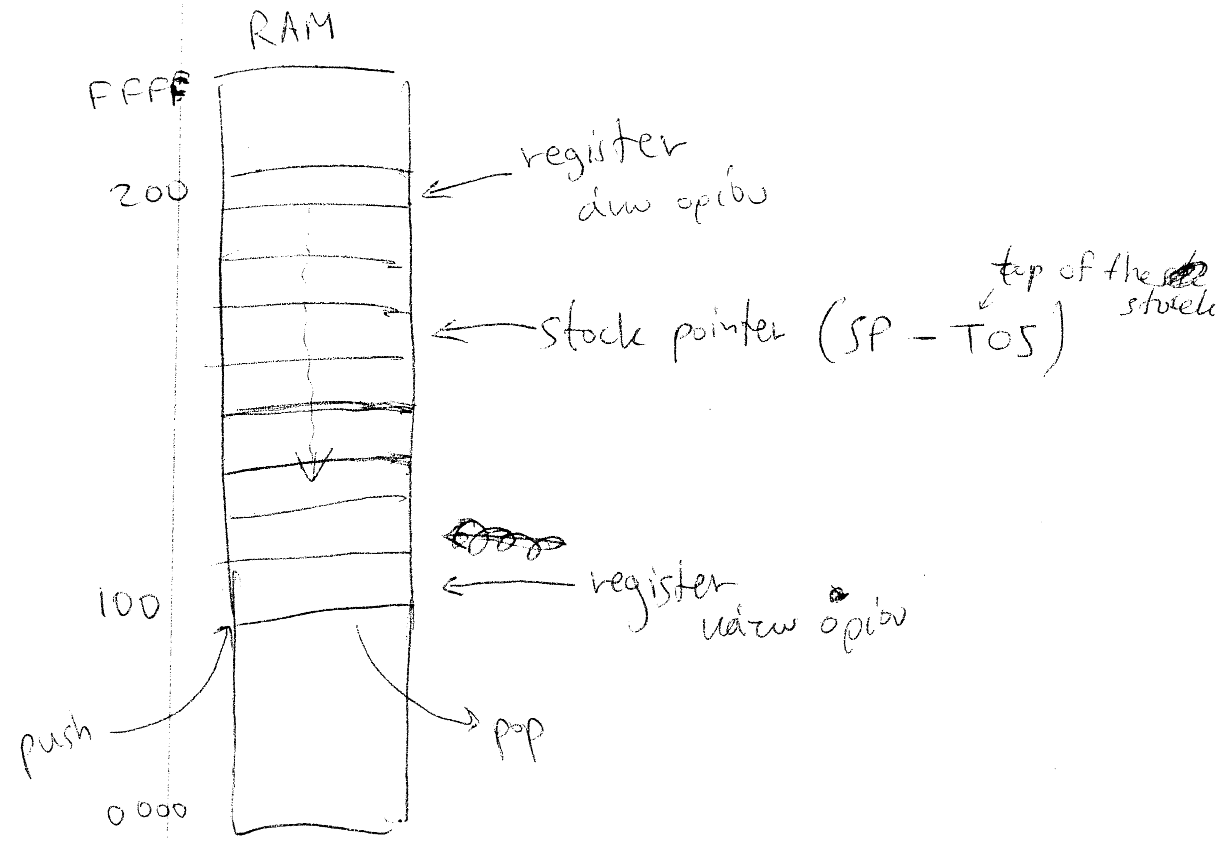
b) BRE d

$ΜΠ = ΜΠ + d$

$d \rightarrow 24$ bits

21/11/2008

$\boxed{\text{Αρχιτ. 1}}$



push A: SP = SP - 1

pop A: sp = sp + 1

main(){

int A[1000000] ~> 4MB<

}

αυτός που αν βάλουμε μπροστά το static

στοίβα

οτιδήποτε μέσα σε block πάει στο stack, ο,τιδήποτε δηλώνεται στην αρχή πάει στο heap → σωρό

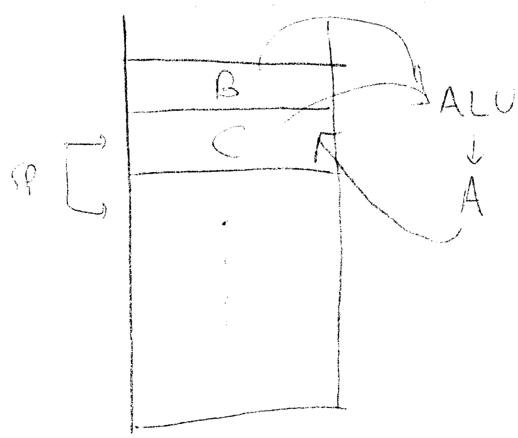
ISA $\rightarrow$ instruction set architecture
(σύνολο εντολών)

stack based $\rightarrow$ virtual machine ("Java")
Accumulator based
General Purpose register

stack based αρχιτεκτονικές

A = B + C

push B
push C
add
pop A



accumulator based αρχιτεκτονικές

αρχικά: 8085
8086
8186
⋮
8586 $\rightarrow$ Pentium

Acc ← A
τότε add B
Acc ← Acc + B (A + B)

Load B ($Acc \leftarrow B$)
Add C ($Acc \leftarrow Acc + C$)
store A

- έχει λιγότερες προσβάσεις μνήμης ακόμα και από τις stack based αρχιτεκτονικές

General Purpose Register αρχιτεκτονικές

1. καταχωρητής μνήμης → CISC (complex instr. set)
2. load-store (καταχωρητή - καταχωρητή) → RISC.

$mem(r_1) = mem(r_2) + mem(r_3)$ ← (δυσκολεύει compiler)

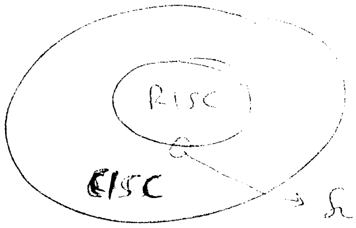
load R_1, $mem(R_2)$
store . _
add R_1, R_2

RISC Vs CISC → [illegible]: 423

ADD R_1, $mem(R_2)$.

από CISC επειδή ο RISC κάνει

πράξεις μόνο σε καταχωρητές.



→ διεπαφή που μεταγράφει το CISC σε εντολές RISC

Παραδείγματα:

$A = B + C$

- register - memory

load R_1, B
~~load R_2, C~~
add R_1, C
store A, R_1

(CISC)

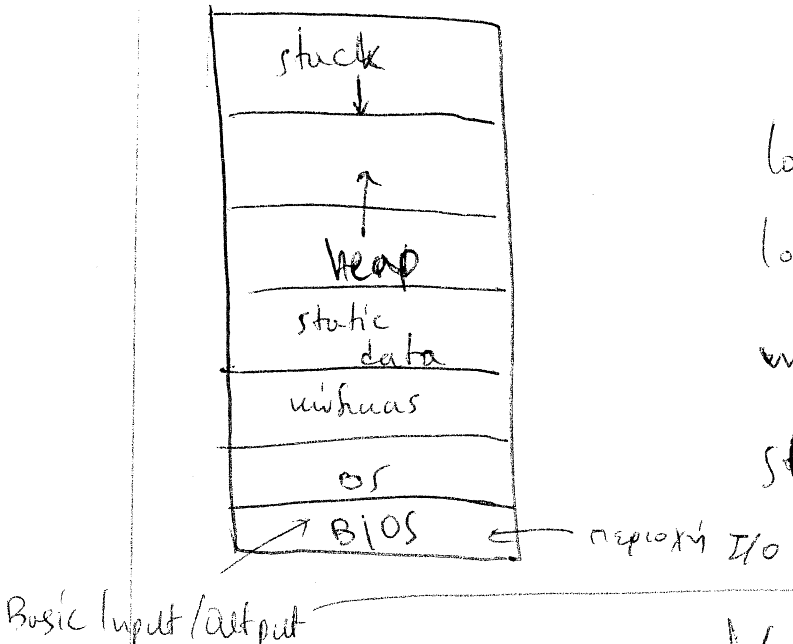
- load store

load R_1, B
load R_2, C
add R_3, R_1, R_2
store A, R_3

(RISC)

Κεραμίδας : keramidas@ee.upatras.gr

$mem(r_1) = mem(r_2 + r_3) * mem(r_4 + disp)$



$R_5 = R_2 + R_3$

load fp_1, (r_5)

load fp_2, $(r_4 + disp)$

multi fp_3, fp_1, fp_2

store (r_1), fp_3

RISC Vs CISC

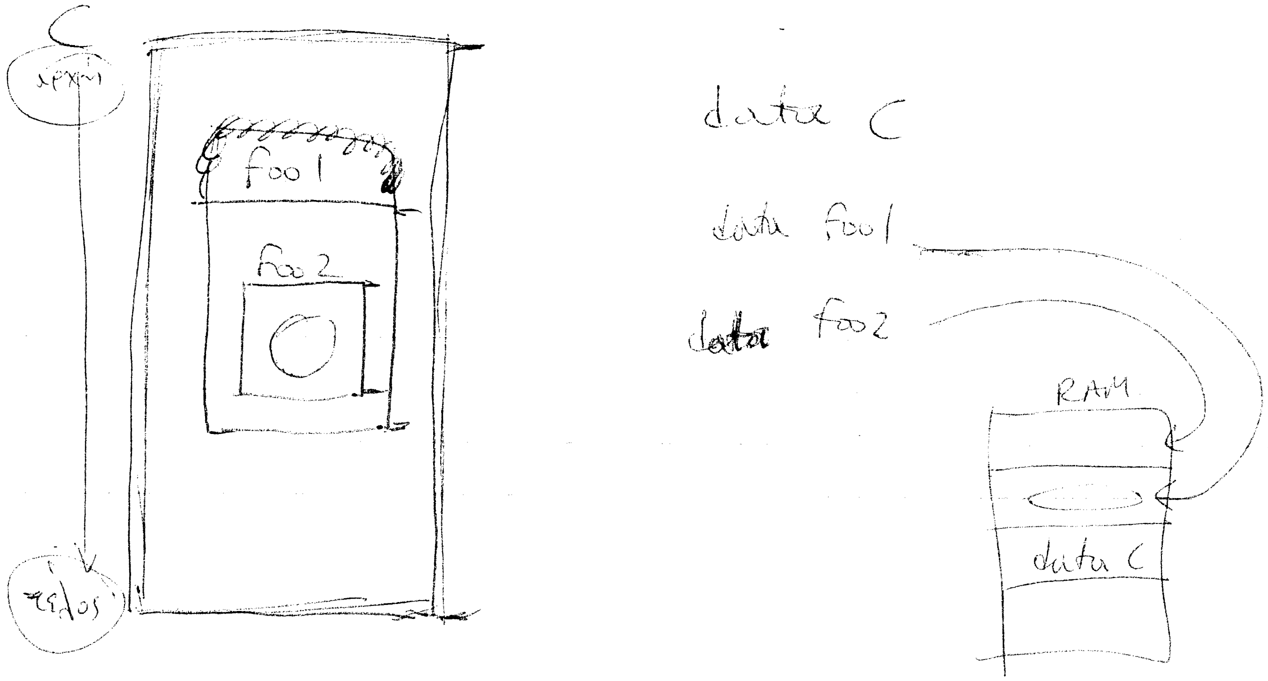
RISC	CISC
• πολλοί καταχωρητές	• ≠
• σταθερό μέγεθος εντολών	• ≠

⇓

άρα π.χ.

- αποκωδικοποίηση εντολών πιο εύκολη
- εκτέλεση εντολών πιο εύκολη

C → blocked structure



<u>sos</u> άσκηση 2.9 / β)

Load $r_1, (r_2)$ // $r_1 \leftarrow Mem(r_2)$

add $r_1, r_2, \#m$ // $r_1 \leftarrow r_1 + r_2 + m, \; m \geq 0$

clear r1 // $r_1 = 0$

comp r1 // $r_1 \leftarrow \overline{r_1}$ (1's)

1) ADD r_1, r_2 // $r_1 \rightarrow r_1 + 2$

ADD r_1, r_2 #0

2) ADD r_1, #m // $r_1 \leftarrow r_1 + m$

[redacted]

CLEAR R_2 // $R_2 = 0$

ADD r_1, r_2, #m

3) ADD r_1, m(r_2) // $r_1 \leftarrow r_1 + Mem(m + r_2)$

CLEAR r_3

ADD r_3, r_2, m

load r_4,(r_3)

add r_1, r_4, #0

ή

load r_3,(r_3)

add r_1, r_3, #0

// ο r_3 θα πάρει την τιμή που δείχνει ο καταχωρητής r_3

4) ADD r_1, (r_2) // $r_1 \leftarrow r_1 + Mem(r_2)$

load r_3 (r_2)

add r_1, r_3, #0

immediate: (αριθμός)

direct: άμεσος

indirect: έμμεσος

5) ADD r_1, (r_2+r_3) // $r_1 \leftarrow r_1 + Mem(r_2 + r_3)$

CLEAR r_4

ADD r_4, r_2, #0

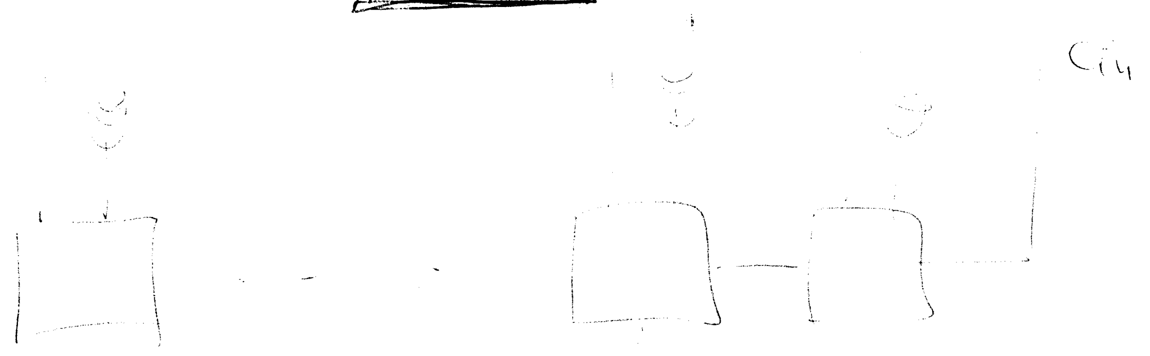
ADD r_4, r_3, #0

load r_4, (r_4)

add r_2, r_4, #0

28/11/200

Αρχιτ. I



$(v-1) \cdot T_c + T_s$

$g_i = A_i B_i$ (carry generator)

$p_i = A_i + B_i$ (carry propagator) προώθηση

$h_i = A_i \oplus B_i$ → ημιάθροισμα { άθροισμα → χωρίς carry }

! λάθος στο βιβλίο

$C_i = g_i + p_i C_{i-1}$ → 2 πύλες

$S_i = h_i \oplus C_{i-1}$ → 2 πύλες + xor = 3 πύλες

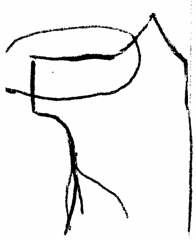
2 πύλες ≃ xor (σε χρονοκαθυστέρηση)

Αθροιστής πρόβλεψης κρατουμένου των 4 δυαδικών ψηφίων

$2(\nu-1)+3 \rightarrow$ πύλες (critical path)

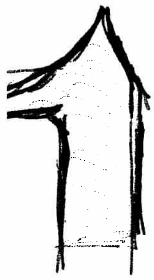
$$\boxed{c_i = g_i + p_i \cdot c_{i-1} \quad // \quad s_i = h_i \oplus c_{i-1}}$$

$$c_1 = g_1 + p_1 g_0 + p_1 \cdot p_0 \cdot c_{i-1}$$



$$c_2 = g_2 + p_2 \cdot g_1 + p_2 \cdot p_1 \cdot g_0 + p_2 \cdot p_1 p_0 \cdot c_{i-1}$$

5 πύλες



HA : 1 χρονοκαθυστέρηση πύλης = χπ

χπ carry

2 χπ sum

FA : 2 χπ carry

3 χπ sum

$$T = T_{AND} + T_{HA_{carry}} + (\nu-1) T_{FA_{sum}} + 2 T_{FA_{carry}} + 2(\nu-3) T_{FA_{carry}} =$$

$$= T_{AND} + T_{HA_{carry}} + (\nu-1) T_{FA_{sum}} + 2(\nu-2) T_{FA_{carry}}$$

44

5/12/2008

$\boxed{\text{Αρχιτ. I}}$

$011110 \rightarrow 30_{10}$

$\Downarrow$

$1000\text{-}10$

$32 \qquad -2 \qquad = 30_{10}$

αναπαράσταση του 30 με λιγότερους "άσσους"

~~δύο προσθέσεις και αφαιρέσεις~~

1 πρόσθεση + 1 αφαίρεση = 2 t πρόσθ.

→ 4 προσθέσεις + 4 ολισθήσεις

011110 0 βάζουμε το 0 → επειδή δεν ξέρουμε μετά αν είναι μηδέν ή άσσο

0111100 → 0

0111100 → −1

0111100 → 0

0111100 → 0

0111100 → 0

0111100 → 1

Αλγ. Booth

00	→	0
01	→	1
10	→	−1
11	→	0

1000-10

01010101010101 ← worst case

Συμπλήρωμα ως προς 2 (2's)

$$X = -X_{v-1} * 2^{v-1} + \sum_{i=0}^{v-2} x_i \cdot 2^i$$

$$\begin{array}{cccccc}
 & & \overline{W_3} & W_2 & W_1 & W_0 \\
 & & -1 & & & \\
 & \overline{X_3} & X_2 & X_1 & X_0 & \\
 & -1 & & & & \\
\overline{Y_3} & Y_2 & Y_1 & Y_0 & & \\
-1 & & & & & \\
Z_3 \; \overline{Z_2} & \overline{Z_1} & \overline{Z_0} & & & \\
-1 & & +1 & & &
\end{array}$$

$$\begin{array}{ccccccc}
 & & & \overline{W_3} & W_2 & W_1 & W_0 \\
 & & \overline{X_3} & X_2 & X_1 & X_0 & \\
 & \overline{Y_3} & Y_2 & Y_1 & Y_0 & & \\
Z_3 & \overline{Z_2} & \overline{Z_1} & \overline{Z_0} & & & \\
 & -1 & -1 & -1 & & & \\
\hline
-1 & 0 & 0 & +1 & & &
\end{array}$$

000
001
010
011
100
101
110
111

07/01/2009

Αρχιτεκτονική I

$(x+y)+z = x+(y+z)$? (όχι για κινητής υποδιαστολής)

$Y = 2^3 \cdot 1,00000000$

$x = 2^{-4} \cdot 1,11000000$

$z = -2^{-4} \cdot 1,01100000$

$x - z = 2^{-4} \cdot (1,11000000 - 1,01100000)$

$= 2^{-4} \cdot 0,01100000 = 2^2 \cdot 0,0000000$

$Y + x - z = 2^2 \cdot 1,00000000$

[redacted]

$\mu \rightarrow \lceil \log_2 \mu \rceil$

μ ↓ καταστάσεις

↳ flip-flops

9/1/2009

Αρχιτεκτονική I

3.1 comp 2

a) 1000.0110 → -122_{10}
b) 1100.0000 → -64_{10}
c) 0111.1100 → 24_{10}
d) 0100.0001 → 65_{10}

$$\begin{array}{r} \bar{X} \\ +\ 1 \\ \hline \end{array}$$

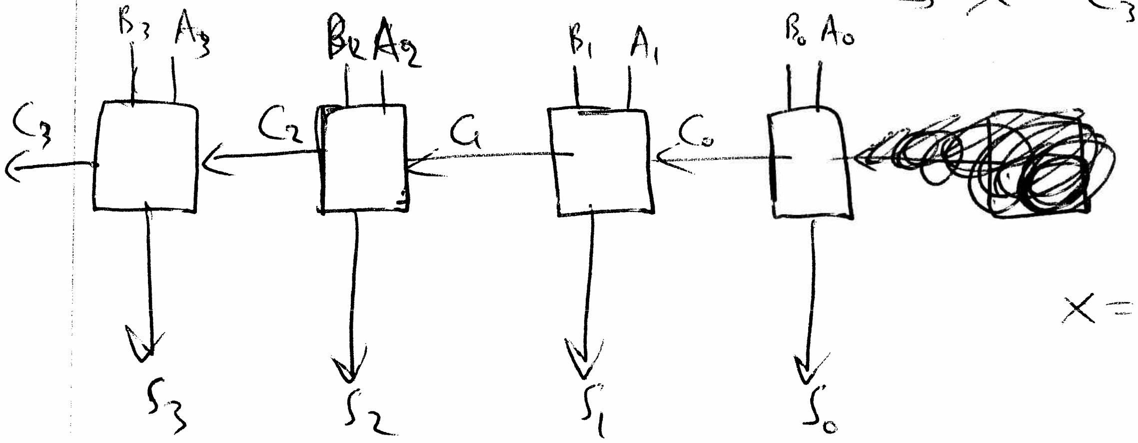
απόλυτη τιμή του X

a+b

$$\begin{array}{r} 1000.0110 \\ +\ 1100.0000 \\ \hline 1)0100.0110 \end{array}$$

σε 2's

$X = C_3 \oplus C_2$



$X = k_8 \oplus k_7$

c+d

$$\begin{array}{r} 0111.1100 \\ 0100.0001 \\ \hline 1011.1101 \end{array} \qquad \begin{array}{r} 124 \\ 65 \\ \hline 189 \end{array} \rightarrow >128$$

άρα, έχω υπερχείλιση

$C_7 = 1$, $C_8 = 0$

$Y = 0 \oplus 1 = 1$

a+c

$$\begin{array}{r} 1000.0110 \\ 0111.1100 \\ \hline 10000.0010 \end{array}$$

$k_8 = 1$, $k_7 = 1$

$Y = 1 \oplus 1 = 0$ (όχι υπερχείλιση)

b+d

$$\begin{array}{r} 1100.0000 \\ + \; 0100.0001 \\ \hline 10000.0001 \end{array}$$

$k_8 = 1$, $k_7 = 1$

$Y = 1 \oplus 1 = 0 \Rightarrow$ (όχι υπερχείλιση)

2/ $\boxed{\text{16 bit adders}}$

a+b

α) 1111 1111 1000 0110
  1111 1111 1100 0000

  1 1111 1111 0100 0110

$K_{16} = 1$, $K_{15} = 1$

$Y = 1 \oplus 1 = 0$ (όχι υπερχείλιση)

2's (επειδή 2's το κρατούμενο αγνοείται)

c+d

  0000 0000 0111 1100
+ 0000 0000 0100 0001

0) 0000 0000 1011 1101

$Y = 0 \oplus 0 = 0$ (όχι υπερχείλιση)

a+c

  1111 1111 1000 0110
+ 0000 0000 0111 1100

  0000 0000 0000 0010

$Y = 1 \oplus 1 = 0$

b+d

  1111 1111 1100 0000
+ 0000 0000 0100 0001

  0000 0000 0000 0001

$Y = 1 \oplus 1 = 0$

→

σας

3.2

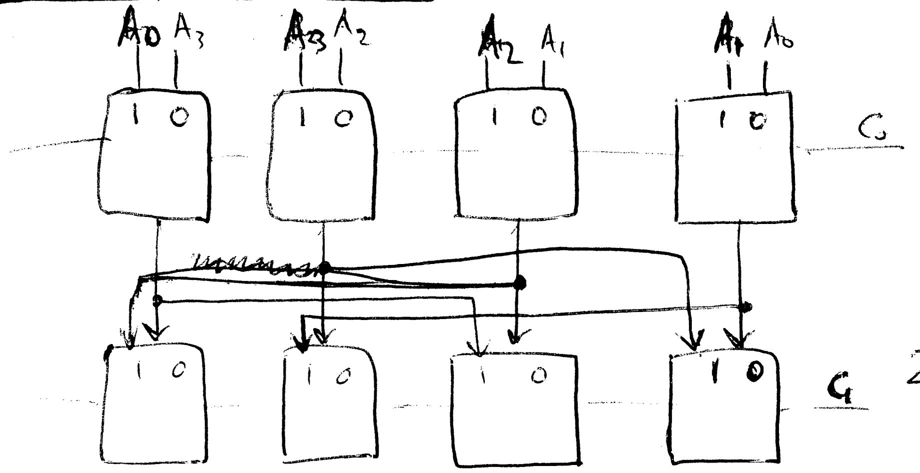
πολυπλέκτες

ο ολισθητές 4 bits

κυκλική ολίσθηση
αριθμητική ολίσθηση
προς τα δεξιά

Λύση

κυκλική ολίσθηση



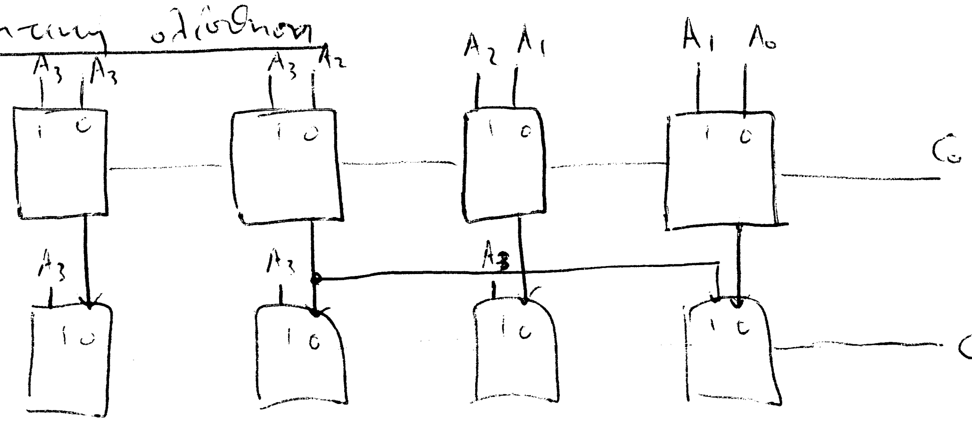
4 bits
⇓
3 ολισθητές

1 θέση ολίσθησης

2 θέσεις ολίσθησης

επίπεδα ολίσθησης
1
2
4

αριθμητική ολίσθηση



→