Αρχιτεκτονική Υπολογιστών

Ενότητα 3: Τρόπος αποθήκευσης Byte (endianess).
Αρχιτεκτονική συνόλου εντολών.
Οργάνωση & Διευθυνσιοδότηση μνήμης.
Ρυθμοί Λειτουργίας x86

Δρ. Μηνάς Δασυγένης
mdasyg@ieee.org
Εργαστήριο Ψηφιακών Συστημάτων και Αρχιτεκτονικής Υπολογιστών
http://arch.ece.uowm.gr/mdasyg

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ψηφιακά Μαθήματα στο Πανεπιστήμιο Δυτικής Μακεδονίας**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σκοπός ενότητας

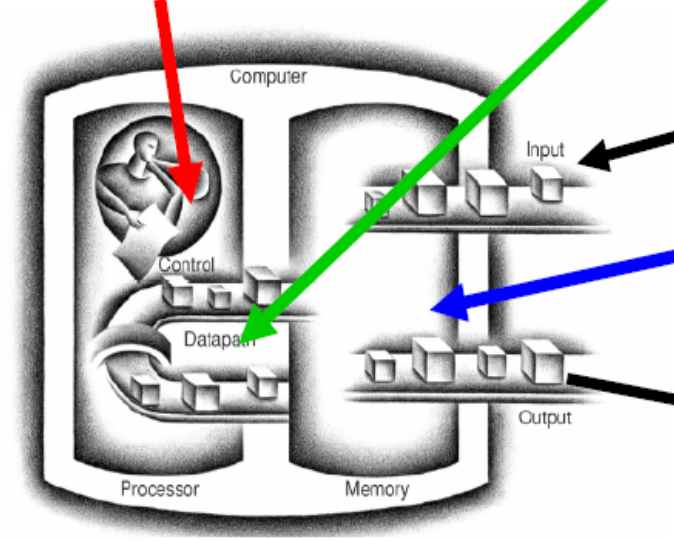
- Η κατανόηση των διαφορετικών τρόπων αποθήκευσης των Byte στην εξωτερική μνήμη.
- Η κατανόηση των τρόπων διευθυνσιοδότησης στους σύγχρονους επεξεργαστές.
- Η κατανόηση ότι κάθε επεξεργαστής συνοδεύεται από τη δικιά του ISA, και κάθε ISA ορίζει μοναδικά την αρχιτεκτονική ενός επεξεργαστή.

Προηγούμενη διάλεξη

Πέντε «κλασικές» μονάδες

DATAPATH :
Εδώ εκτελούνται λειτουργίες

CONTROL: ο μεγάλος «μαέστρος»



ΣΥΣΚΕΥΕΣ ΕΙΣΟΔΟΥ:
Πληκτρολόγιο, ποντίκι, scanner ,αισθητήρες, κλπ

ΜΝΗΜΗ: αποθηκεύει εντολές και δεδομένα

ΣΥΣΚΕΥΕΣ ΕΞΟΔΟΥ:
Οθόνη, εκτυπωτής, ηχεία κλπ

Επεξεργαστής (ΚΜΕ)

- Καλείται και Κεντρική Μονάδα Επεξεργασίας *(ΚΜΕ)* ή CPU.
- Το ενεργό μέρος ενός υπολογιστή.
  - Καθορίζει ανά πάσα στιγμή το τι συμβαίνει στον υπολογιστή.
- Ο επεξεργαστής αποτελείται από δύο μέρη:
  - **Datapath** *(Διαδρομή δεδομένων).*
    - Εκτελεί όλες τις λειτουργίες *(αριθμητικές, λογικές, μεταφορές, κλπ).*
  - **Control** *(Μονάδα ελέγχου).*
    - "Αντιλαμβάνεται" το τι σημαίνουν οι εντολές *(αποκωδικοποίηση εντολών)* και ακολούθως.
    - Κατευθύνει το datapath.
    - Αναλαμβάνει την επικοινωνία με τις συσκευές I/O.

Μνήμη και τρόπος αποθήκευσης των Byte.

Η εξωτερική μνήμη του υπολογιστή

- Η κύρια μνήμη αποτελείται από ένα συνεχόμενο αριθμό θέσεων, κάθε μία από τις οποίες περιέχει το **ίδιο πλήθος** δυαδικών ψηφίων (ψηφίων ή bits).
- Σε κάθε θέση αντιστοιχεί μία **διεύθυνση** (address) μέσω της οποίας πραγματοποιείται η αναφορά στο περιεχόμενο της θέσης.
- Η κάθε θέση είναι ικανή να αποθηκεύσει μία ψηφιοσυλλαβή, ή ένα byte (8 δυαδικά ψηφία).

Παράδειγμα διευθυνσιοδότησης μνήμης στο x86

- Στο listing στη γραμμή 5 του πηγαίου κώδικα ορίζονται κάποια δεδομένα.
- Αυτά τοποθετούνται στη μνήμη ξεκινώντας από τη διεύθυνση 00 .
- Τα δεδομένα είναι 17 (=11h) Byte.
- Το επόμενο τμήμα δε μπορεί να τοποθετηθεί στη θέση 11h λόγω περιορισμών διευθυνσιοδότησης της αρχιτεκτονικής x86. Τοποθετείται στη θέση 20h.

```
[   4]          :                                       ; add your data here!
[   5]    0000: 70 72 65 73 73 20 61 6E 79 20 6B 65     pkey db "press any key...$"
                79 2E 2E 2E 24
[   6]          :                                       ends
[   7]          :
[   8]          :                                       stack segment
[   9]    0020: 00 00 00 00 00 00 00 00 00 00 00 00     dw   128  dup(0)
```

Ορισμός της μνήμης

- Τι είναι η μνήμη;
  - Η μνήμη είναι η βασική μονάδα του υπολογιστή όπου αποθηκεύονται τα δεδομένα και οι εντολές. χρησιμοποιώντας δυαδική αναπαράσταση.
    - Ουσιαστικά είναι ένα (μεγάλο) σύνολο από 0 και 1.
- Οι προγραμματιστές πάντα ζητούσαν απεριόριστη και γρήγορη μνήμη.
  - Πρόβλημα.
    - Δυστυχώς, η “γρήγορη” μνήμη κοστίζει ακριβά...
  - Λύση.
    - Δημιουργία της ψευδαίσθησης ότι υπάρχει διαθέσιμη απεριόριστη και γρήγορη μνήμη.

Endianness

- Ως **Endianness** ορίζεται το σχετικό βάρος των επιμέρους bytes σε τύπους δεδομένων μεγαλύτερους του ενός byte.
- Καθορίζεται από τον κατασκευαστή.
- **Big Endian.**
  - Όταν η διεύθυνση της προσπέλασης προσδιορίζεται από την διεύθυνση του **αριστερότερου byte** ή του byte στο **“μεγάλο άκρο”.**
    - IBM 360/370, Motorola 68000, Sun SPARC, PowerPC 970.
- **Little Endian.**
  - Όταν η διεύθυνση της προσπέλασης προσδιορίζονται από την διεύθυνση του **δεξιότερου byte** ή του byte στο **“μικρό άκρο”.**
    - Intel x86, AMD64, DEC Vax.
- Bi-Endian (Διαμορφώσιμοι).
  - Power PC, PA RISC, MIPS, IA64 (default **Big Endian**).
  - DEC Alpha, ARM (default **Little Endian**).

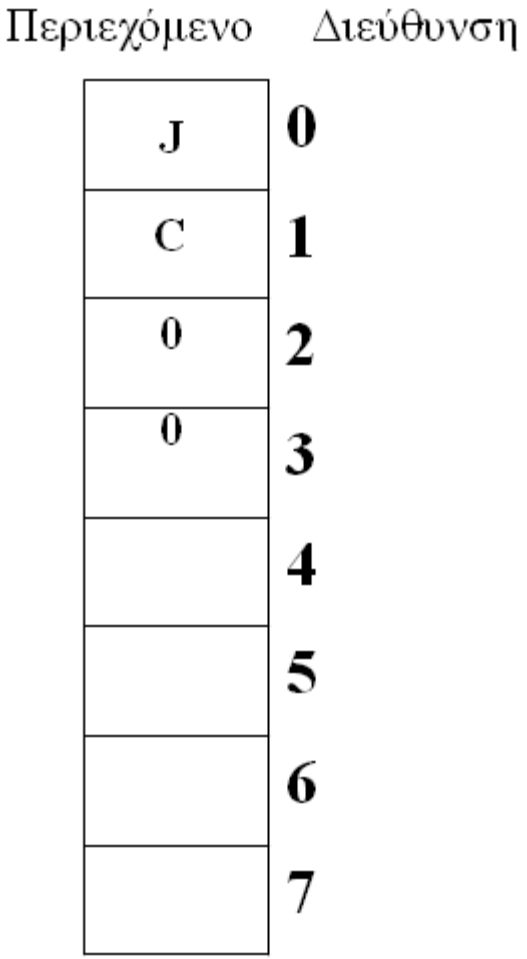
Αποθήκευση Byte στη μνήμη 1

- Κάθε διευθυνσιοδοτούμενο κελί μνήμης αποθηκεύει 1 Byte.
- Όταν θέλουμε να αποθηκεύσουμε πολλαπλά συνεχόμενα Byte στη μνήμη (π.χ. Λέξεις που αποτελούνται από 2 Byte), τότε αυτά μπορούν να αποθηκευτούν με 2 τρόπους:
  - Είτε, στη **χαμηλότερη διεύθυνση** μνήμης αποθηκεύεται **πρώτα το MSB** και στην επόμενη (μεγαλύτερη) το LSB (big endian).
  - Είτε, στη **χαμηλότερη διεύθυνση** μνήμης αποθηκεύεται **πρώτα το LSB** και στην επόμενη (μεγαλύτερη) το MSB (little endian).
- Ο τρόπος αποθήκευσης ονομάζεται «endianess».

Αποθήκευση Byte στη μνήμη 2

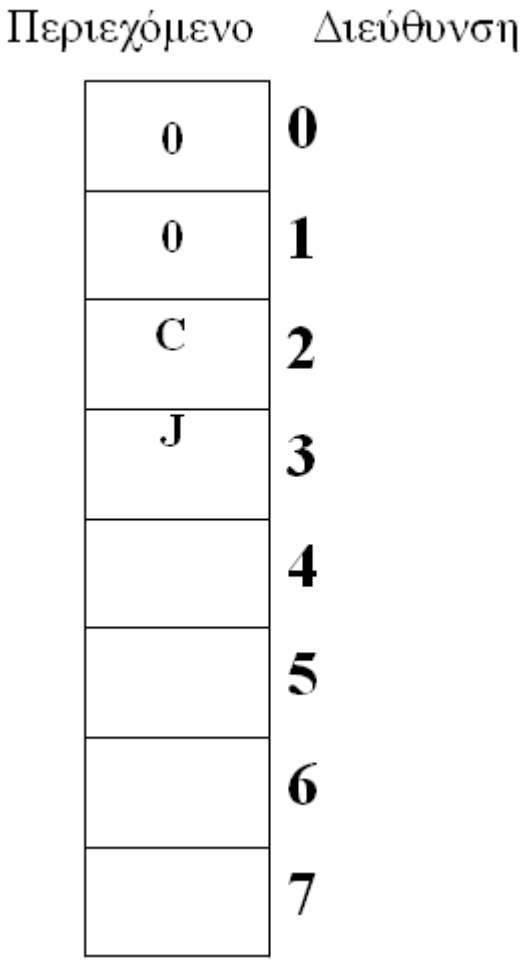
- Το πως αποθηκεύονται στη μνήμη τα Byte ονομάζεται και '**byte ordering**'.
- Είναι πολύ σημαντικό στοιχείο όταν επικοινωνούν υπολογιστές με διαφορετική οργάνωση των byte στη μνήμη. Συνήθως χρησιμοποιούνται ενδιάμεσες απεικονίσεις (π.χ. **XDR** = eXternal Data Representation).
- Αν ένας προγραμματιστής δε λάβει υπόψιν του τη διαφορετικότητα τότε θα δημιουργηθούν bugs που είναι δύσκολο να ανιχνευθούν.
- Οι όροι little,big endian δημιουργήθηκαν το 1980. Συνδέθηκαν με μια ιστορία από τα ταξίδια του Guliver (όταν οι liliputians μάλωναν για το που να σπάσει ένα αβγό little end / big end).
- Έγινε σημαντικό πρόβλημα όταν διασυνδέθηκαν οι υπολογιστές.

Η big-endian οργάνωση μνήμης



- Έστω ότι σε έναν υπολογιστή με μήκος λέξης 32 bit (= 4 byte), θέλουμε να αποθηκεύσουμε τα περιεχόμενα ενός καταχωρητή (**JC00**) στη μνήμη, στις θέσεις 0 – 3.
- **Big endian οργάνωση ("μεγάλου άκρου).**
- **Η αποθήκευση ξεκινά από το σημαντικότερο (αριστερότερο) byte της λέξης.**
- Επεξεργαστές: Motorola.
- Υπολογιστές: Apple Macintosh, Atari, Amiga, workstations Sun, HP.

Η little-endian οργάνωση μνήμης



- Έστω ότι σε έναν υπολογιστή με μήκος λέξης 32 bit (= 4 byte), θέλουμε να αποθηκεύσουμε τα περιεχόμενα ενός καταχωρητή (**JC00**) στη μνήμη, στις θέσεις 0 – 3.
- **Little endian οργάνωση ("μικρού άκρου).**
- **Η αποθήκευση ξεκινά από το λιγότερο σημαντικό (δεξιότερο) byte της λέξης.**
- Επεξεργαστές: Intel.
- Υπολογιστές: IBM και συμβατοί.
- Οι όροι big και small endian προέρχονται από τα "Ταξίδια του Gulliver" όπου σατιρίζονται οι πολιτικοί που διαφωνούν αν πρέπει τα αυγά να σπάζονται από το μεγάλο ή μικρό άκρο τους.

Το ΛΣ πρέπει να υποστηρίζει την endian του επεξεργαστή

- **Little-endian operating systems:**
  - Linux on x86, x64, Alpha and Itanium.
  - Mac OS X on x86, x64.
  - OpenVMS on VAX, Alpha and Itanium.
  - Solaris on x86, x64, PowerPC.
  - Tru64 UNIX on Alpha.
  - Windows on x86, x64 and Itanium.
- **Big-endian operating systems:**
  - AIX on POWER.
  - AmigaOS on PowerPC and 680x0.
  - HP-UX on Itanium and PA-RISC.
  - Linux on MIPS, SPARC, PA-RISC, POWER, PowerPC, 680x0, ESA/390, and z/Architecture.
  - Mac OS on PowerPC and 680x0.
  - Mac OS X on PowerPC.
  - MVS and DOS/VSE on ESA/390, and z/VSE and z/OS on z/Architecture.
  - Solaris on SPARC.

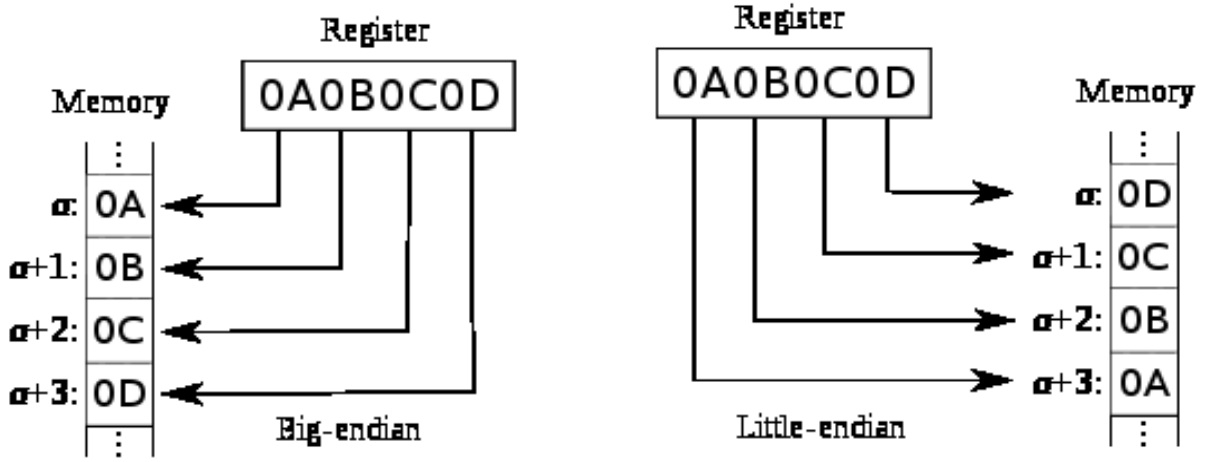
Υπάρχει και μια ακόμη endianess

- Κάποιοι υπολογιστές (π.χ. ARM) χρησιμοποιούν **bi-endian.**
  - Μπορούν να ρυθμιστούν κατά την εκκίνηση να χρησιμοποιούν είτε little είτε big endian.
  - Η ρύθμιση γίνεται είτε με software κατά την εκκίνηση, είτε με hardware (jumper).

Πλεονεκτήματα-Μειονεκτήματα endianness

- Η **little-endian** μπορεί να διαβάσει την ίδια λέξη μερικές φορές σε διάφορα bit width (π.χ. Το 4A 00 00 00) είτε διαβαστεί ως 8bit (4A) είτε ως 16 bit (004A) είναι το ίδιο (χρησιμοποιείται στους compilers).
- Η **big-endian** βοηθάει στο να βρεθεί πόσος μεγάλος είναι ο αριθμός διαβάζοντας το πρώτο κελί μόνο.
- Η **little-endian** απλοποιεί το hardware σε πράξεις multibyte (απαιτείται μια μόνο εντολή inc).

Σύνοψη endianess



Στα παραδείγματα φαίνεται η τιμή που έχει ένας καταχωρητής και πως θα γραφεί στη μνήμη.

Παραδείγματα αποθήκευσης Byte (1/3)

Διεύθυνση byte

Διεύθυνση word	**+0**	**+1**	**+2**	**+3**
0x0000 0000				
0x0000 0004	0xDE	0xAD	0xBE	0xEF
0x0000 0008				
0x0000 000C				

Παραδείγματα αποθήκευσης Byte (2/3)

Παραδείγματα:

Τι θα συμβεί με την εκτέλεση των παρακάτω εντολών εάν ο επεξεργαστής:

α) Ακολουθεί την σύμβαση **Big Endian**.

β) Ακολουθεί την σύμβαση **Little Endian.**

1. LOAD, WORD, 0x00000004

Big Endian: Επιστρέφει 0χDEADBEEF.

Little Endian: Επιστρέφει 0xEFBEADDE.

2. LOAD, HALFWORD 0x00000006

Big Endian: Επιστρέφει 0xBEEF.

Little Endian: Επιστρέφει 0xEFBE.

- Η LOAD.WORD είναι εντολή MIPS και διαβάζει μια λέξη 4 Byte που ξεκινάει από τη συγκεκριμένη διεύθυνση.
- Η LOAD.HALFWORD είναι εντολή MIPS και επιστρέφει μια λέξη 2 Byte που ξεκινάει από τη συγκεκριμένη διεύθυνση.

Παραδείγματα αποθήκευσης Byte (3/3)

Address	Big Endian				Little Endian			
0	**0**	**1**	**2**	**3**	**3**	**2**	**1**	**0**
4	4	5	6	7	7	6	5	4
8	8	9	10	11	11	10	9	8
12	12	13	14	15	15	14	13	12

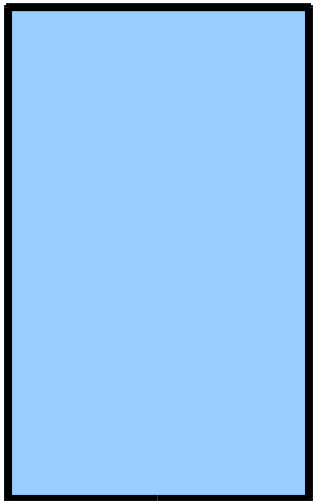
Περί διευθυνσιοδότησης μνήμης

- Συνήθως η μνήμη είναι '**byte addressable**' δηλαδή αντιστοιχεί μια διεύθυνση σε κάθε byte της μνήμης.
- Υπάρχουν μνήμες που είναι '**word addressable**' δηλαδή που αντιστοιχεί μια διεύθυνση σε κάθε λέξη της μνήμης.
  - Η λέξη μπορεί να αποτελείται από:
    - 2 Byte (16bit) ή
    - 4 Byte (32bit) ή
    - 8 Byte (64bit).

Χάρτης Μνήμης

- Η Μνήμη αποτελείται από την Μνήμη Προγράμματος και τη Μνήμη Δεδομένων.
- Υπάρχει και η μνήμη-χαρτογράφησης I/O (memory mapped), η οποία δεν είναι φυσική μνήμη.
- Δεν χρειάζεται να χρησιμοποιηθεί όλη η μνήμη.
- Για Intel μPS, η μνήμη δεδομένων στη χαμηλότερη μνήμη και μνήμη προγράμματος στη ψηλότερη μνήμη.

Ανώτατη Διεύθυνση:
FFFFFh (20),
FFFFFFh (24),
FFFFFFFFh (32)

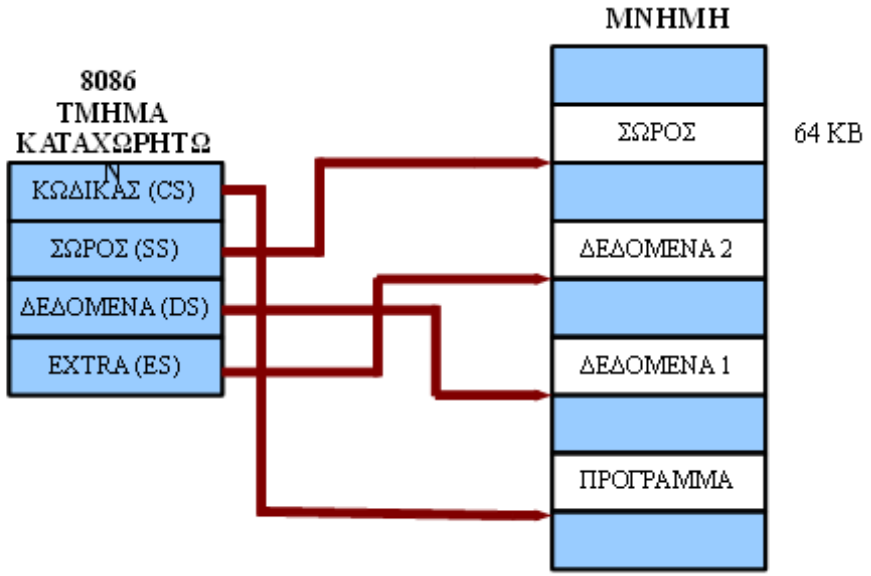


Προσπελάσιμη Μνήμη:
1 Mbyte (20),
16 Mbytes (24),
1 Gbytes (32)

Καταχωρητές Τμημάτων 8086

386/Pentium: Δυο επιπρόσθετα τμήματα καταχωρητών (FS, GS) στη μνήμη δεδομένων.

Τα περιεχόμενα των τμημάτων καταχωρητών δείχνουν στη διεύθυνση βάσης των περιοχών που αντιστοιχούν στη μνήμη.

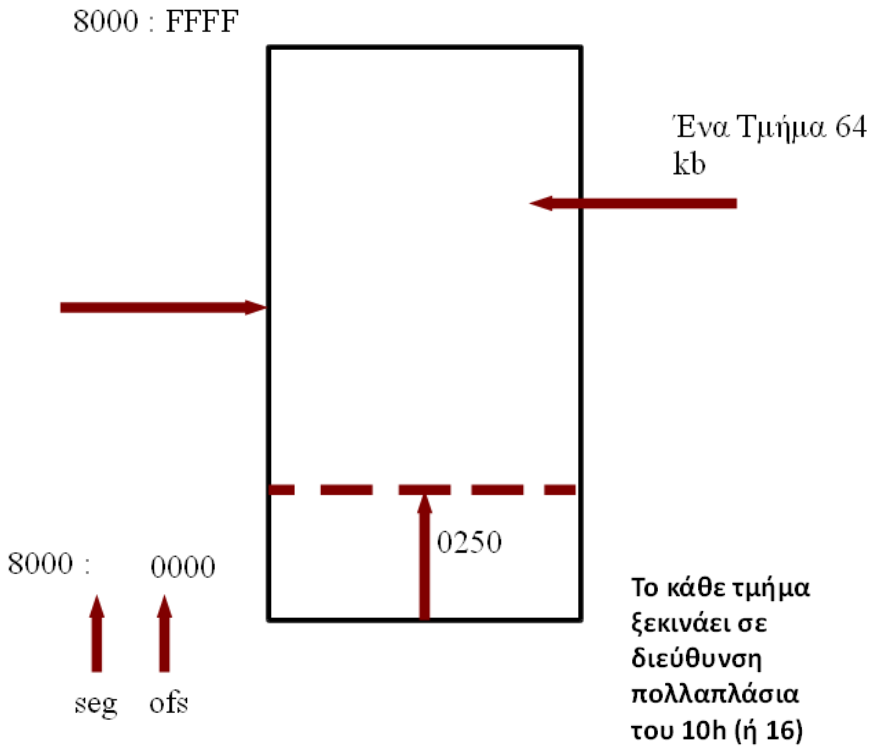


Ο 8086 χρησιμοποιεί τμηματοποιημένη (**segmented**) μνήμη

Γραμμική Διεύθυνση

F0000
E0000
D0000
C0000
B0000
A0000
90000
80000
70000
60000
50000
40000
30000
20000
10000
00000

Τμηματική αντιμετώπιση μνήμης: απόλυτη (γραμμική) διεύθυνση είναι ο συνδυασμός ενός 16-bit τμήματος αξίας προστιθέμενο σε ένα 16-bit offset [CS:IP].

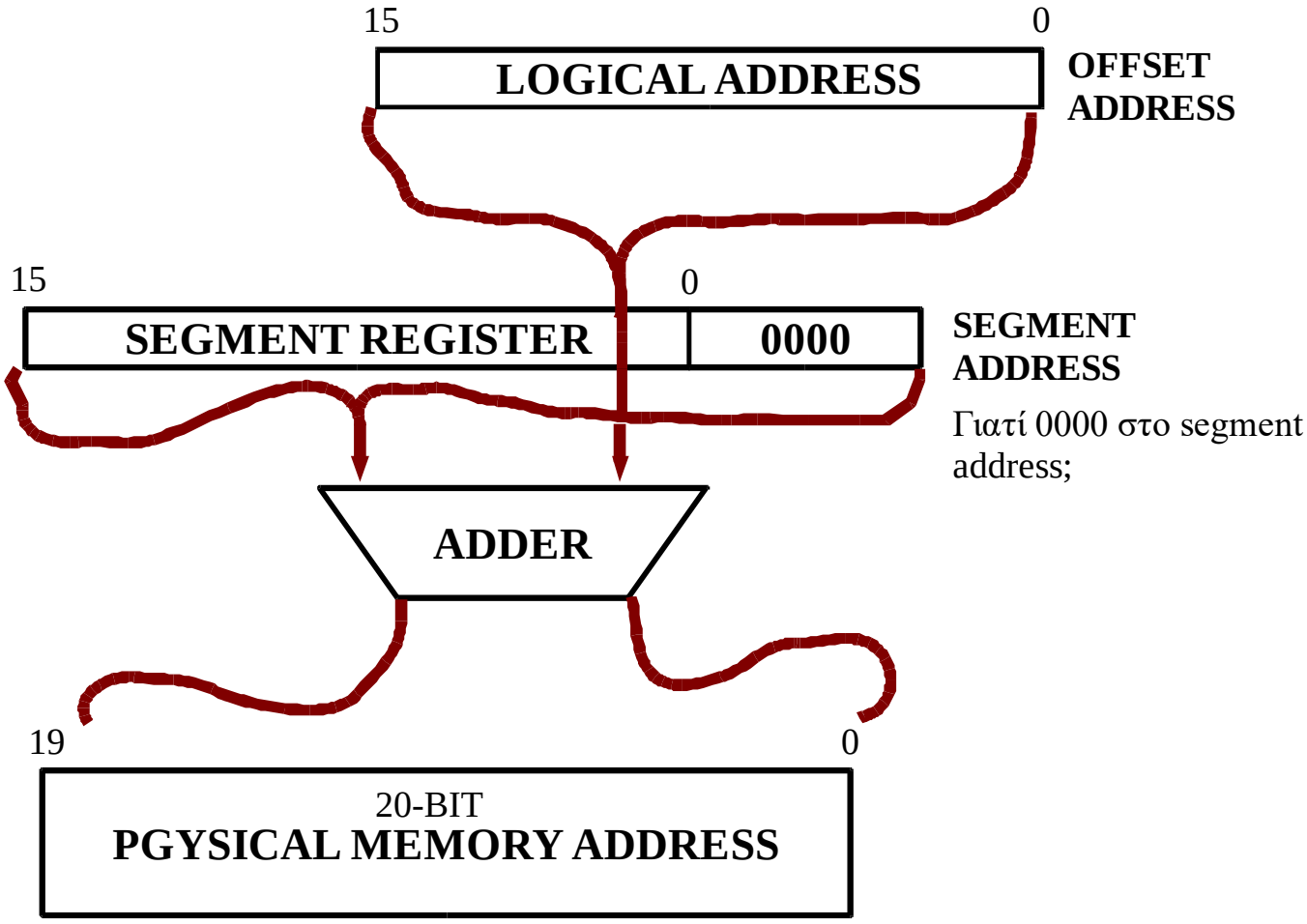


Υπολογισμός της 20Bit διεύθυνσης στο 8086

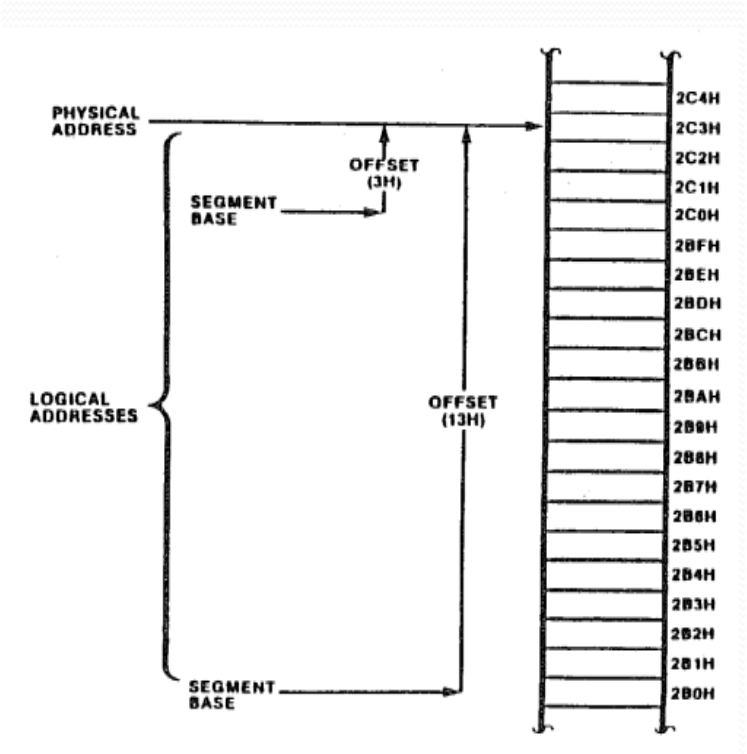
- Για να αποκτήσουμε 20-bit διεύθυνση από 16-bit καταχωρητή:
  - Παίρνουμε μια διεύθυνση τμήματος, την πολλαπλασιάζουμε με το 16 (προσθέτουμε ένα δεκαεξαδικό μηδέν), και το προσθέτουμε το στο αντιστάθμισμα (offset).
  - Παράδειγμα: 20-bit πρόγραμμα διεύθυνσης μνήμης:
  - CS * 10h + IP.
  - Παράδειγμα: 20-bit στοίβα της διεύθυνσης μνήμης:
  - SS * 10h + SP.
- Πρόβλημα: μετατροπή 09F1 : 0100 σε γραμμική διεύθυνση.

- Προσαρμοσμένη αξία του τμήματος: 0 9 F 1 0
- Πρόσθεση του αντισταθμίσματος (offset): 0 1 0 0
- Γραμμική διεύθυνση: 0 A 0 1 0

Η παραγωγή της διεύθυνσης μνήμης στον 8086



Η **λογική** και η **φυσική** διεύθυνση



- Η λογική διεύθυνση είναι η μετατόπιση μέσα στο ίδιο τμήμα.
- Για παράδειγμα:
  - CS=09F1
  - IP=0100
- Λογική διεύθυνση: 0100.
- Φυσική διεύθυνση (προηγούμενη διαφάνεια): A010.

Τρόποι Διευθυνσιοδότησης

- **Τρόποι διευθυνσιοδότησης (Addressing modes).**
  - **Πώς** οι ISAs προσδιορίζουν την διεύθυνση.
- Οι τρόποι διευθυνσιοδότησης προσδιορίζουν σταθερές και καταχωρητές για τον υπολογισμό της διεύθυνσης.
- Η πραγματική διεύθυνση μνήμης η οποία προσδιορίζεται από τον τρόπο διευθυνσιοδότησης καλείται **ενεργός διεύθυνση (effective address).**

Γενικοί Τρόποι διευθυνσιοδότησης μνήμης

Στο σχηματισμό της διεύθυνσης μνήμης μπορούν να συμμετέχουν:

- Απόλυτες τιμές.
- Καταχωρητές.
- Σταθερές τιμές μετατόπισης (offsets).

			Πιθανή χρήση
displacement	mem[offs+reg]	τοπικές	
Register inderect	mem[reg]	δείκτες	
indexed	mem[reg1+reg2]	πίνακες	
direct	mem[addr]	στατικές	
Memory indirect	mem[mem[reg]]	*δείκτες	
auto-increment	mem[reg++]	πίνακες	
scaled	mem[offs+reg1+reg2*d]	πίνακες	

Κάθε ISA υποστηρίζει έναν ή παραπάνω τρόπους διευθυνσιοδότησης

- **Displacement** (μετατόπισης).
- **Register indirect** (ο καταχωρητής φέρει τη δ/νση μνήμης).
- **Indexed** (υπάρχει ένας καταχωρητής δείκτης που δείχνει μετατόπιση από ένα άλλο σημείο).
- **Direct** (απευθείας πρόσβαση στη μνήμη).
- **Memory Indirect** (πρόσβαση σε μια θέση μνήμης η οποία έχει τη διεύθυνση μνήμης που θέλουμε να χρησιμοποιήσουμε).
- **Auto increment** (μετά την πρόσβαση αύξηση ενός δείκτη κατά 1).
- **Scaled** (μετά την πρόσβαση αύξηση ενός δείκτη με πολλαπλάσιο τρόπο).

Αρχιτεκτονική Συνόλου Εντολών
Instruction Set Architecture (ISA)

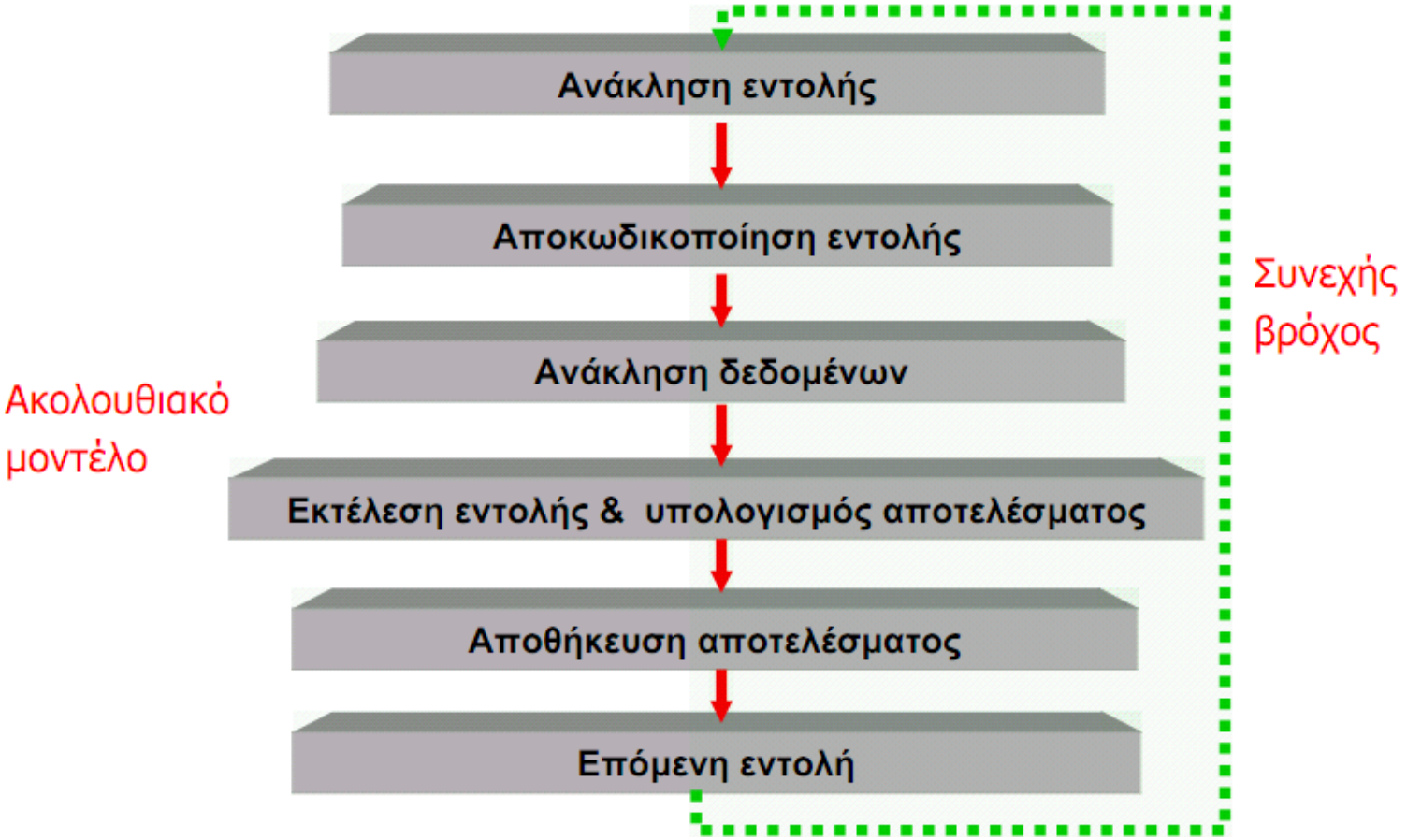
Λειτουργία του Κεντρικού Επεξεργαστή

- **Ανακαλεί αυτόματα** εντολές από τη μνήμη και τις εκτελεί ακολουθιακά.
- Η ΚΜΕ περιέχει έναν αριθμό **καταχωρητών**.
- Κάθε καταχωρητής αποθηκεύει μία ακολουθία ψηφιοσυλλαβών (bytes) ορισμένου μήκους που ονομάζεται λέξη (word).
- Ένα σύστημα με κύρια μνήμη και ΚΜΕ, είναι η απλούστερη μορφή υπολογιστικού συστήματος (εξαιρουμένου του λογισμικού).
- Υπολογιστής = Επεξεργαστής + Κύρια Μνήμη.

4 είναι οι βασικές λειτουργίες του υπολογιστή

- Ανάκληση Εντολής.
- Αποκωδικοποίηση Εντολής.
- Εκτέλεση Εντολής.
- Εγγραφή αποτελεσμάτων.

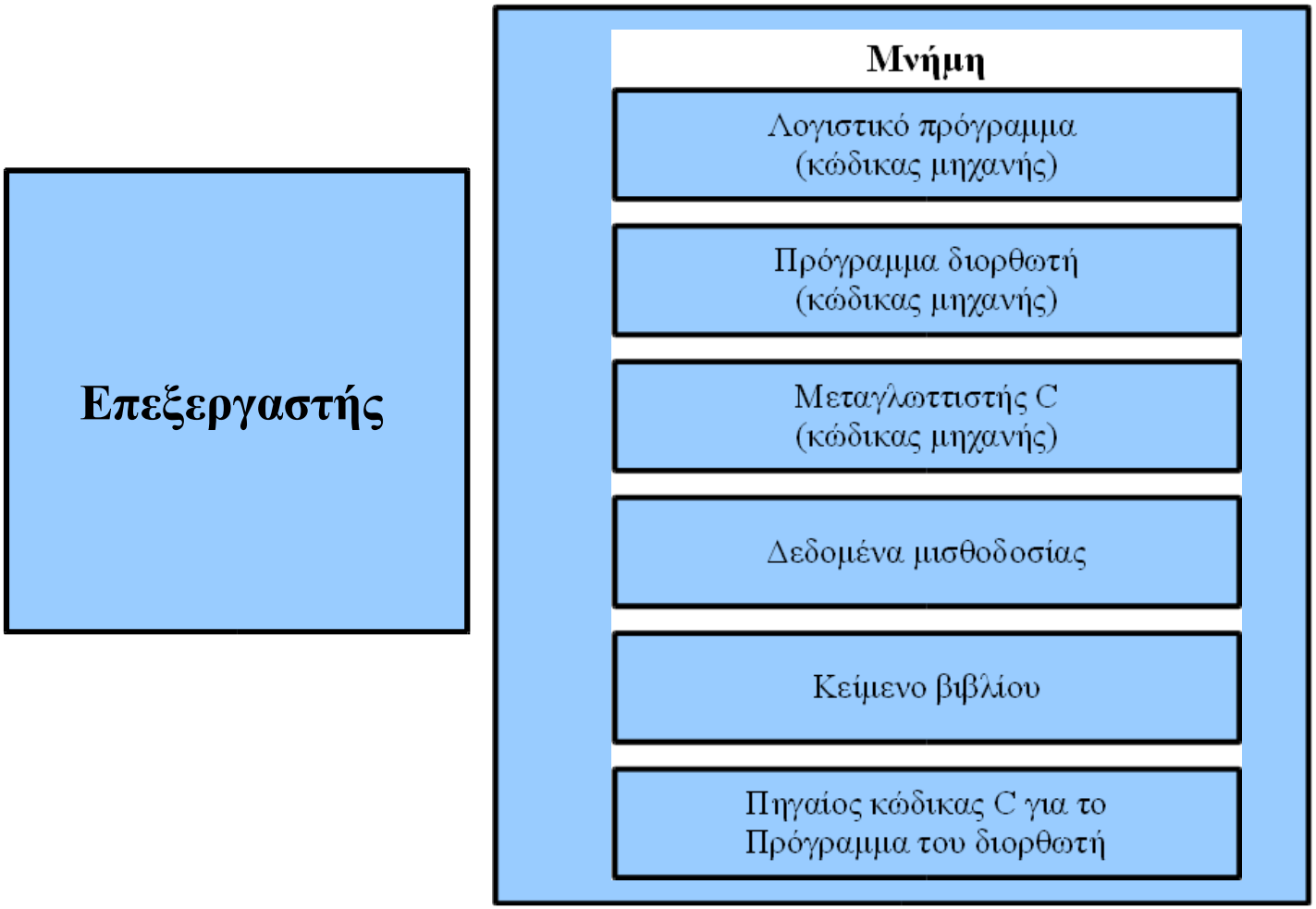
Το ακολουθιακό μοντέλο λειτουργίας του Η/Υ



Η έννοια του αποθηκευμένου προγράμματος (1/4)

- Οι σημερινοί υπολογιστές κτίζονται με βάση δυο αρχές.
  - Οι εντολές αναπαρίστανται με αριθμούς.
  - Τα προγράμματα αποθηκεύονται στην μνήμη για να διαβαστούν ή να γραφούν όπως και οι αριθμοί.
- Οι παραπάνω αρχές οδηγούν στην έννοια του **αποθηκευμένου προγράμματος**.
  - Εντολές και δεδομένα πολλών τύπων μπορούν να αποθηκευτούν στη μνήμη ως αριθμοί
  - Οδήγησε στον **υπολογιστή αποθηκευμένου προγράμματος (stored program computer).**

Η έννοια του αποθηκευμένου προγράμματος (2/4)



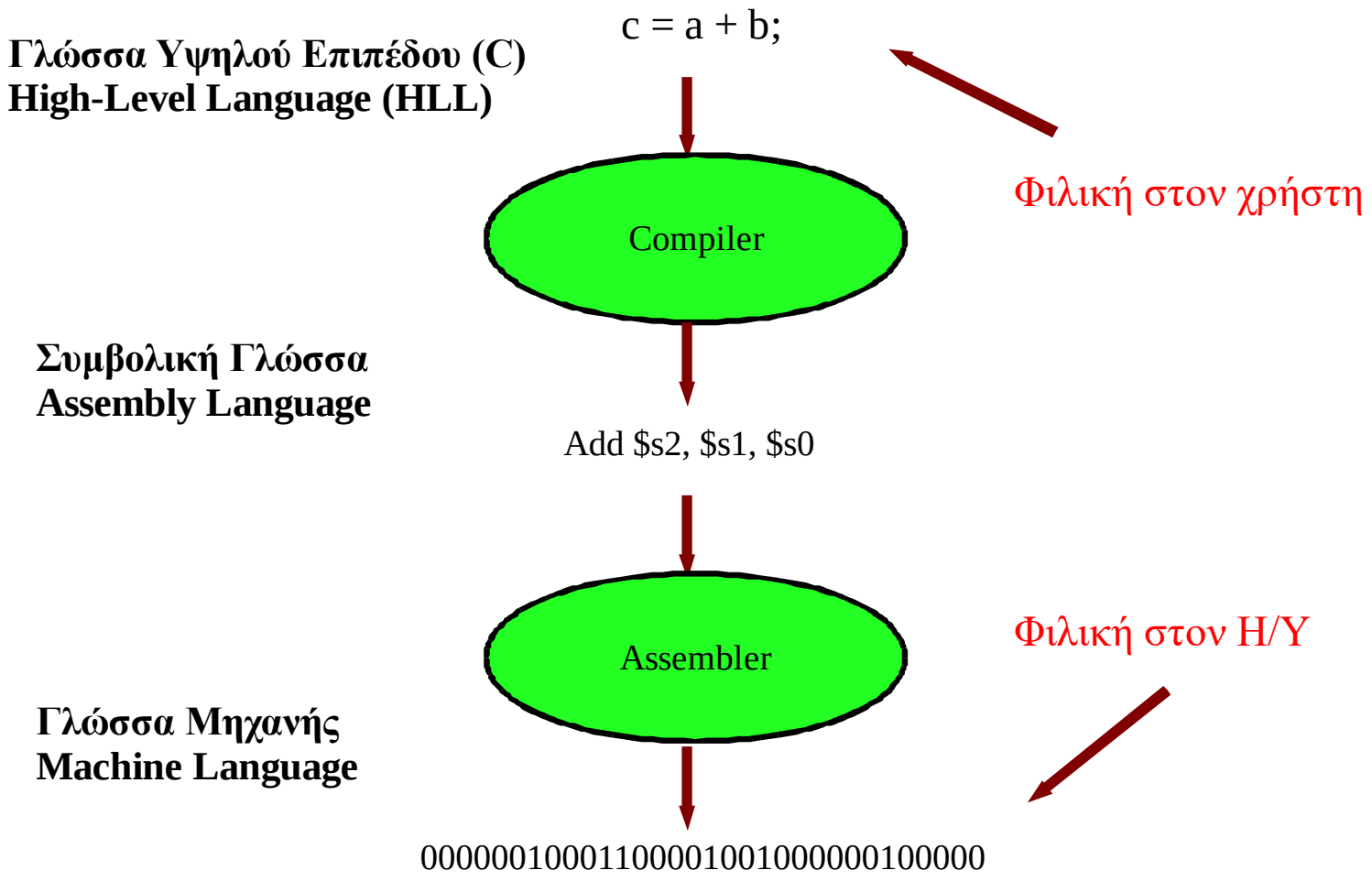
Η έννοια του αποθηκευμένου προγράμματος (3/4)

- Χρήση κοινής τεχνολογίας μνήμης για δεδομένα και προγράμματα.
- Η μνήμη μπορεί να περιέχει:
  - Το πηγαίο κώδικα ενός editor.
  - Τον αντίστοιχο μεταγλωττισμένο κώδικα μηχανής.
  - Το κείμενο που χρησιμοποιεί ο μεταγλωττισμένος editor.
  - Τον μεταγλωττιστή που παρήγαγε τον κώδικα μηχανής.

Η έννοια του αποθηκευμένου προγράμματος (4/4)

- Συνέπειες αναπαράστασης των εντολών με αριθμούς.
  - Τα προγράμματα κυκλοφορούν ως δυαδικά (binary) αρχεία.
  - Επαναχρησιμοποίηση έτοιμου (συμβατού) λογισμικού.
  - Δυαδική συμβατότητα (binary compatibility).
- Ευθυγράμμιση της βιομηχανίας γύρω από ένα μικρό αριθμό αρχιτεκτονικών συνόλου εντολών.

Επίπεδα αναπαράστασης εντολών



Γλώσσα υψηλού επιπέδου

Παράδειγμα γλώσσας υψηλού επιπέδου (HLL)

```
if (i == j)
f = g + h;
 else
f = g – h;
```

- Υπάρχουν πολλές γλώσσες υψηλού επιπέδου.
  - C, C++, Java, Fortran, Basic, Pascal, Lisp, Ada, κλπ.
- Βασικά πλεονεκτήματα γλωσσών υψηλού επιπέδου (HLLs).
  - Είναι εύκολα κατανοητές από τον χρήστη αφού αναπτύχθηκαν επάνω στην ανθρώπινη γλώσσα (Αγγλική).
  - Συμβάλλουν στην αυξημένη παραγωγικότητα .
  - Το πρόγραμμα που γράφεται μπορεί να είναι ανεξάρτητο από τον υπολογιστή που θα εκτελεστεί (portability).

Επίπεδα αναπαράστασης: **ASSEMBLY**

Παράδειγμα γλώσσας υψηλού επιπέδου (HLL)

```
if (i == j)
  f = g + h;
else
   f = g – h;
```

Compiler

Συμβολική γλώσσα (Assembly)

```
bne $s3, $s4, Else
add $s0, $s1, $s2
j Exit
Else:
Sub $s0, $s1, $s2
Exit:
```

- Κάθε επεξεργαστής έχει την δικιά του συμβολική γλώσσα (Assembly).
- Η συμβολική γλώσσα είναι **γλώσσα χαμηλού επιπέδου**.
  - Είναι όμως κατανοητή από τον χρήστη που μπορεί να γράψει πιο συμπαγές και αποδοτικό κώδικα αλλά σε περισσότερο χρόνο σε σχέση από τις άλλες HLLs.
- Συνήθως, ο κώδικας γράφεται σε HLLs και περνάει από μεταγλώττιση.
  - Ο μεταγλωττιστής (**Compiler**) είναι ένα πρόγραμμα που αυτόματα μεταφράζει τις εντολές μιας HLL σε εντολές συμβολικής γλώσσας.

Επίπεδα αναπαράστασης: **Γλώσσα Μηχανής**

Παράδειγμα γλώσσας υψηλού επιπέδου (HLL)

Compiler

```
if (i == j)
  f = g + h;
else
   f = g – h;
```

Συμβολική γλώσσα (Assembly)

Assembler

```
    bne $s3, $s4, Else
    add $s0, $s1, $s2
    j Exit
Else:
Exit:
    Sub $s0, $s1, $s2
```

Γλώσσα Μηχανής (Machine Language)

```
00010110011101000000000000000010
00000010001100101000000000100000
00001000000000000000000000000100
00000010001100101000000000100000
```

- Ο συμβολομεταφραστής (**Assembler)** μεταφράζει τον κώδικα από την συμβολική γλώσσα (Assembly) σε **γλώσσα μηχανής** (δυαδικό).
  - Δηλαδή στην μόνη μορφή την οποία ο επεξεργαστής κατανοεί και εκτελεί.

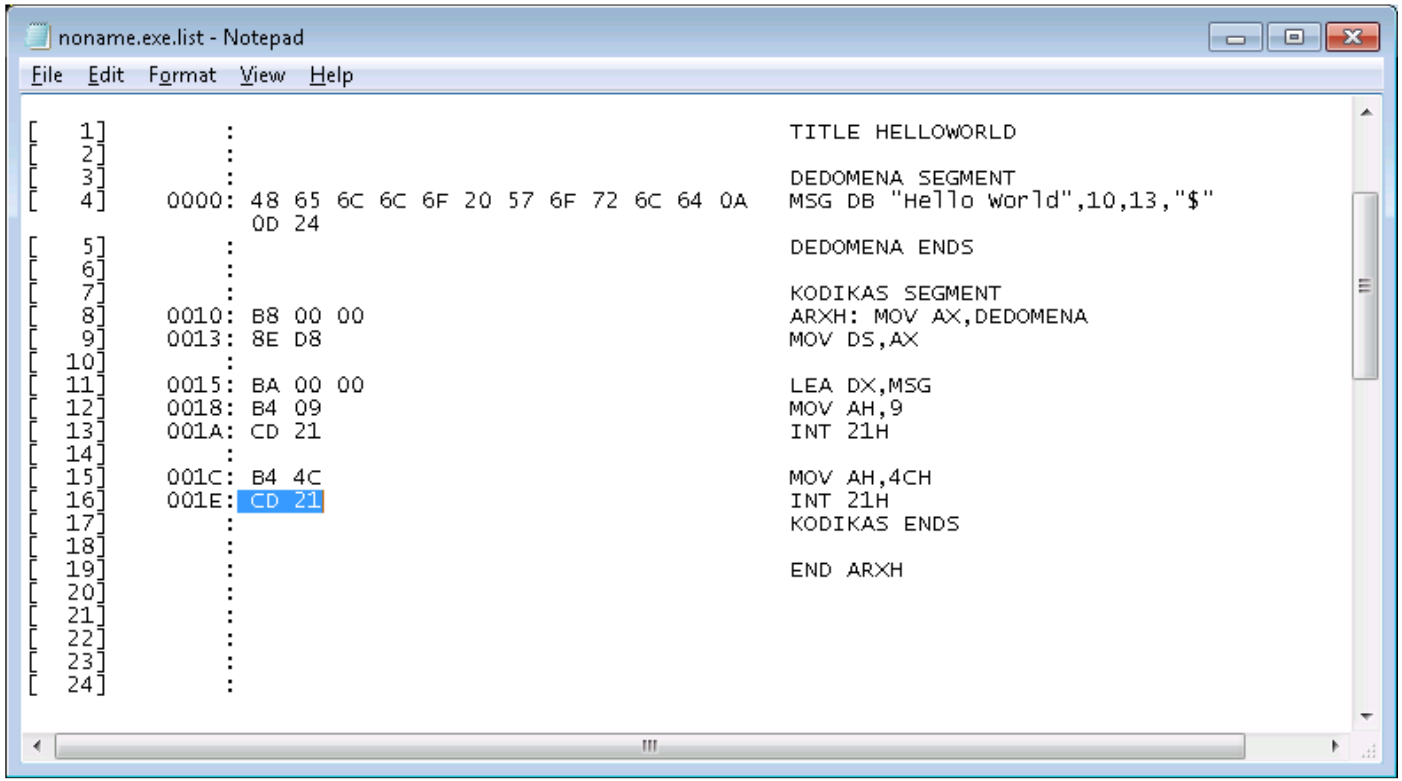
Επίπεδα προγραμματισμού Γλώσσα C, Γλώσσα Assembly

```
int main(void)
{
puts("Hello World");
return;
}
```

```
TITLE HELLOWORLD
DEDOMENA SEGMENT
MSG DB "Hello  World",10,13,"$"
DEDOMENA ENDS
KODIKAS SEGMENT
ARXH: MOV
AX,DEDOMENA
      MOV DS,AX
      LEA DX,MSG
      MOV AH,9
      INT 21H
      MOV AH,4CH
      INT 21H
KODIKAS ENDS
END ARXH
```

Επίπεδα προγραμματισμού
Δεκαεξαδική αναπαράσταση

48 65 6C 6C 6F 20 57 6F 72 6C 64 0A 0D 24 B8 00 00 8E D8 BA 00 00 B4 09 CD 21 B4 4C CD 21



Επίπεδα προγραμματισμού
Δυαδική αναπαράσταση

01001000011001010110110001101100011011110010000001010111011011
11011100100110110001100100000010100000110100101011100000000000
00000000100011101101100010111010000000000000000010110100000010
01110011010010000110110100110011001101001000001

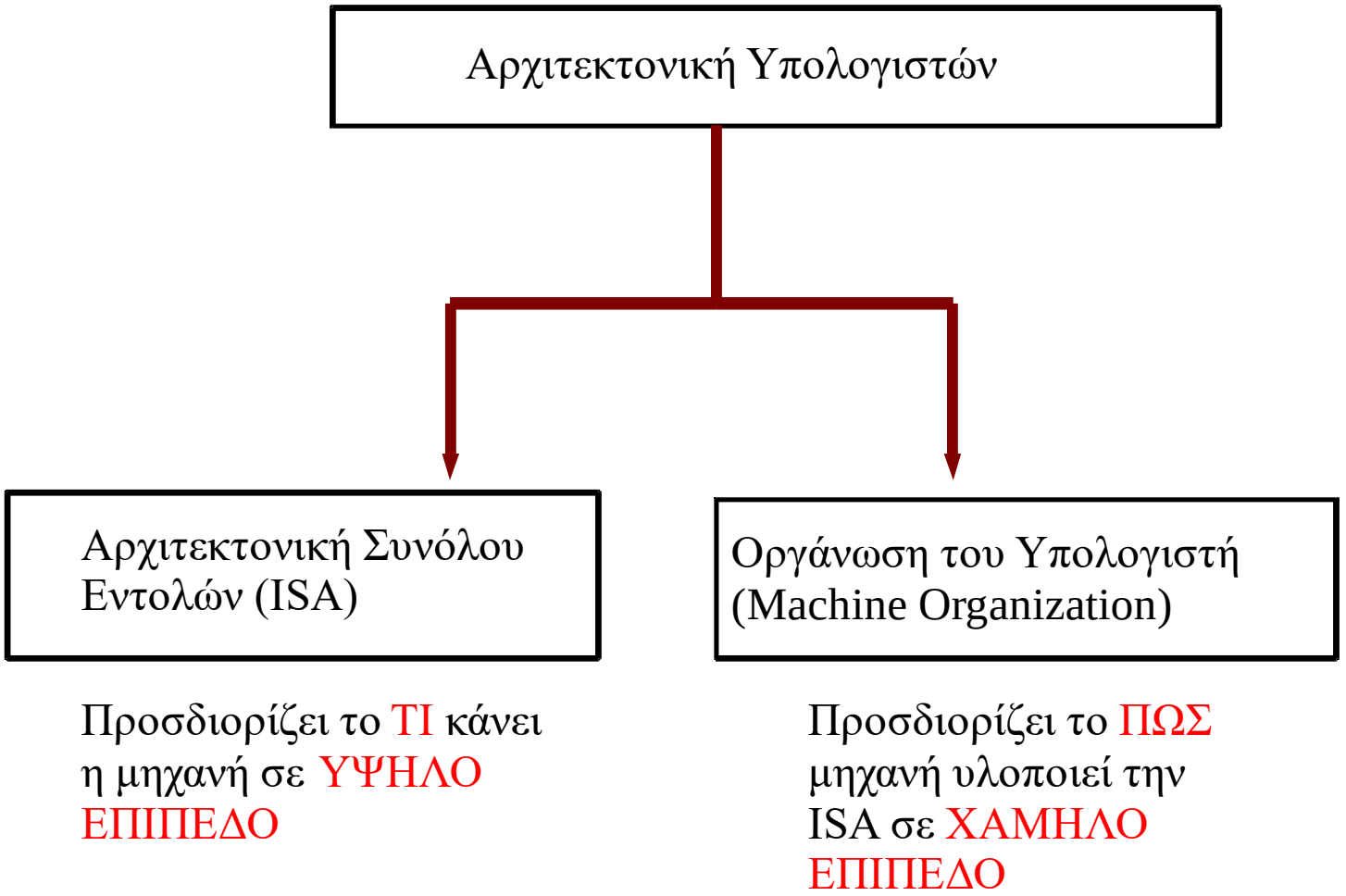
48 65 6C ..=> 0100 1000 0110 0101 0110 1100 ..

Σημείωση: Ο υπολογιστής καταλαβαίνει μόνο το δυαδικό σύστημα. Στη μνήμη αποθηκεύονται μόνο 1 και 0. Για λόγους συμπύκνωσης ο assembler εμφανίζει τις πληροφορίες στο δεκαεξαδικό ή στο δεκαδικό σύστημα.

Εντολές - (Instructions)

- Οι εντολές ανήκουν σε πολλές κατηγορίες:
  - Αριθμητικές: πρόσθεση, αφαίρεση, πολλαπλασιασμός, διαίρεση, κλπ.
  - Ανάγνωσης/εγγραφής από/προς την μνήμη.
  - Διακλάδωσης υπό συνθήκη.
    - Εάν η συνθήκη είναι αληθής, τότε μεταφέρεται η ροή του προγράμματος αλλιώς συνεχίζεται η ακολουθιακή ροή.
  - Ποιες άλλες εντολές θα ήταν χρήσιμες;
- Πώς αναπαρίστανται οι εντολές που καταλαβαίνει ο επεξεργαστής;
  - Φυσικά στην γνωστή **δυαδική μορφή** (0 ή 1).

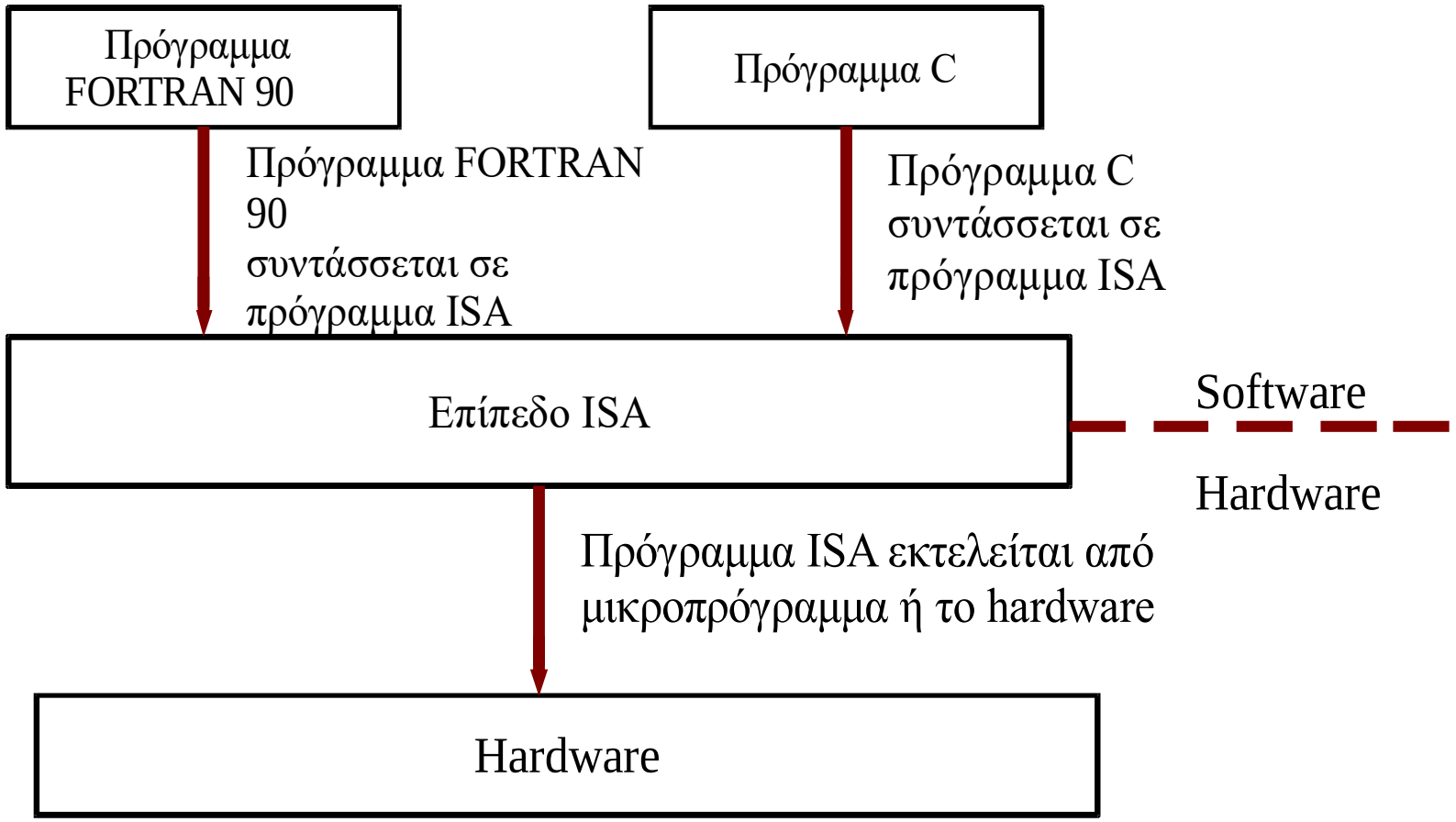
Η αρχιτεκτονική Η/Υ αποτελείται από 2 αντικείμενα



Αρχιτεκτονική συνόλου εντολών

- Αποτελεί το "**συμβόλαιο**" ανάμεσα στο Λογισμικό και στο Υλικό
  - "εάν το Λογισμικό κάνει το Χ, το Υλικό υπόσχεται να κάνει το Υ".
  - Λειτουργική περιγραφή σε υψηλό επίπεδο του **Τι** κάνει το υλικό.
    - Επιτρέπει στον προγραμματιστή να επικεντρώνεται στις λειτουργίες που υλοποιούνται ανεξάρτητα από το πως τις υλοποιεί το υλικό.
    - Επιτρέπει πολλές διαφορετικές υλοποιήσεις με διαφορετικό κόστος και απόδοση να τρέχουν το ίδιο λογισμικό (binary compatible).
  - Ακριβής περιγραφή του πώς το λογισμικό μπορεί να ενεργοποιήσει τις λειτουργίες και να προσπελάσει τα αποτελέσματα.

Που βρίσκεται η ISA



Πως σχεδιάζεται μια ISA;

- Ποιο θα ήταν το μοντέλο της διασύνδεσης Υλικού/Λογισμικού;
  - Πόσα και ποια είδη εντολών υποστηρίζονται;
  - Πώς αναπαρίστανται οι εντολές;
  - Πώς μεταβάλλεται η ακολουθιακή εκτέλεση των εντολών;
    - Δηλαδή πώς αλλάζει η ροή του προγράμματος.
  - Πού αποθηκεύονται οι μεταβλητές ενός προγράμματος;
  - Πόσες και ποιες είναι οι μονάδες αποθήκευσης;
    - Ποιες και πώς μπορεί να τις “δει” το λογισμικό;
  - Υποστηρίζει το υλικό δυνατότητα κλήσης υπορουτινών για δομημένο προγραμματισμό; εάν ναι, πώς γίνεται αυτό;

Μικροαρχιτεκτονική

- Η ISA προσδιορίζει το **ΤΙ** κάνει το υλικό, όχι πώς το κάνει
  - Δεν προσδιορίζει για παράδειγμα τα εξής:
  - Πώς υλοποιούνται οι λειτουργίες.
  - Ποιες λειτουργίες υλοποιούνται “γρήγορα” και ποιες “αργά”.
  - Ποιες λειτουργίες καταναλώνουν περισσότερη-λιγότερη ισχύ.
- Τα παραπάνω προσδιορίζονται από την **μικροαρχιτεκτονική (microarchitecture).**
  - Μικροαρχιτεκτονική: **ΠΩΣ** το υλικό **υλοποιεί** την ISA
  - Μια ISA μπορεί ναι υλοποιείται με πολλές διαφορετικές μικροαρχιτεκτονικές.
    - Οι Pentium και Pentium 3 υλοποιούν σχεδόν την ίδια ISA (IA32 x86) αλλά διαφέρουν σημαντικά στην μικροαρχιτεκτονική.

Βασικές Αρχές Σχεδίασης ISA

- Αρχή 1: Η απλότητα ευνοεί την κανονικότητα.
- Αρχή 2: Το μικρότερο είναι ταχύτερο.
- Αρχή 3: Κάνε τη συνηθισμένη περίπτωση γρήγορα.
- Αρχή 4: Η καλή σχεδίαση απαιτεί καλούς συμβιβασμούς.

Υπάρχει ένα μεγάλο πλήθος από ISA

- Κάθε επεξεργαστής έχει τη **δικιά του** αρχιτεκτονική εντολών.
- Οικογένειες των επεξεργαστών μοιράζονται πολλά στοιχεία της ίδιας αρχιτεκτονικής, αλλά υπάρχουν και κάποιες διαφορές (π.χ. **8086** και **pentium**).
- Οι ISA μπορούν να **κατηγοριοποιηθούν** σε κάποιες βασικές κατηγορίες ως προς την εσωτερική αποθήκευση και χρήση των εντολών.
- Οι ISA μπορούν να κατηγοριοποιηθούν σε σταθερού μήκους εντολές (μαζί με ορίσματα) και σε μεταβλητού μήκους. Π.χ. Ο x86 είναι μεταβλητού μήκους, ενώ ο ARM είναι σταθερού.

Στοιχεία που χαρακτηρίζουν μια ISA

- Αριθμός εντολών.
- Μορφή Εντολών:
  - μεταβλητό ή σταθερό μέγεθος bytes για κάθε εντολή; (8086 1-17 bytes, MIPS 4 bytes).
- Πώς γίνεται η αποκωδικοποίηση (ID);
- Που βρίσκονται τα ορίσματα (operands) και το
- αποτέλεσμα;
- Μνήμη - καταχωρητές, πόσα ορίσματα, τι μεγέθους;
- Ποια είναι στη μνήμη και ποια όχι;
- Πόσοι κύκλοι για κάθε εντολή;

3 είναι οι βασικές Αρχιτεκτονικές Συνόλου Εντολών

Όσον αφορά στον τρόπο αποθήκευσης και προσπέλασης των πληροφοριών για τη διεύθυνση ή την τιμή των τελεστέων, οι κύριες αρχιτεκτονικές είναι:

- Ταξινόμηση των Αρχιτεκτονικών Συνόλου Εντολών ανάλογα με το που βρίσκονται αποθηκευμένοι οι τελεστέοι.
- Αρχιτεκτονικές συσσωρευτών (accumulator).
- Αρχιτεκτονικές στοίβας (stack).
- Αρχιτεκτονικές καταχωρητών γενικού σκοπού.
  - Αρχιτεκτονικές καταχωρητή - μνήμης (register – memory).
  - Αρχιτεκτονικές καταχωρητή – καταχωρητή (register – register) ή φόρτωσης – αποθήκευσης (Load - Store).

1 από 3
Αρχιτεκτονική Συσσωρευτή

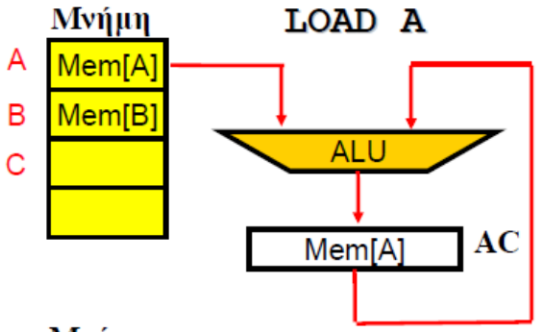
Αρχιτεκτονική Συσσωρευτή (1/2)

- 1η γενιά υπολογιστών: h/w ακριβό, μεγάλο μέγεθος καταχωρητή.
- Ένας καταχωρητής για όλες τις αριθμητικές εντολές (συσσώρευε όλες τις λειτουργίες → Συσσωρευτής (Accum).
- Σύνηθες: 1ο όρισμα είναι ο Accum, 2ο η μνήμη, αποτέλεσμα στον Accum π.χ. add 200.
- Παράδειγμα: A = B + C.
  - Accum = Memory(AddressB);
  - Accum = Accum + Memory(AddressC);
  - Memory(AddressA) = Accum;
- Όλες οι μεταβλητές αποθηκεύονται στη μνήμη. Δεν υπάρχουν βοηθητικοί καταχωρητές.

Αρχιτεκτονική Συσσωρευτή (2/2)

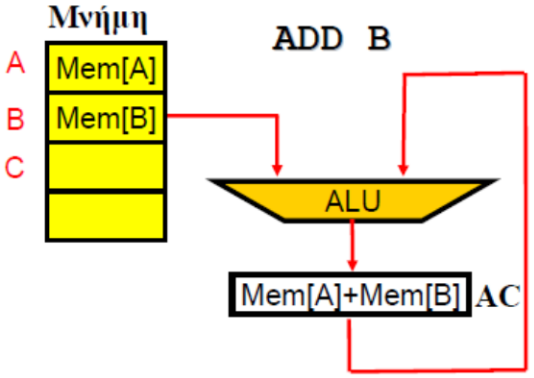
Το περιεχόμενο της θέσης μνήμης με διεύθυνση Α αποθηκεύεται στον συσσωρευτή.

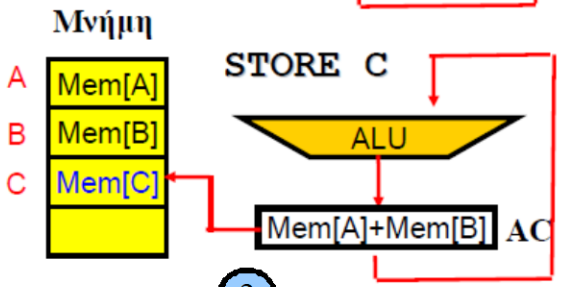
1



Εκτελείται πρόσθεση ανάμεσα στο περιεχόμενο του συσσωρευτή και στο στοιχείο που βρίσκεται στη θέση μνήμης με διεύθυνση Β. Το αποτέλεσμα αποθηκεύεται στο συσσωρευτή.

2





$Mem[C] = Mem[A] + Mem[B]$

3 **Το περιεχόμενο του συσσωρευτή αποθηκεύεται στη θέση μνήμης με διεύθυνση C.**

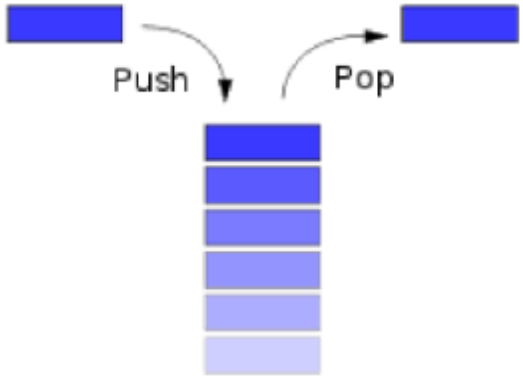
Πλεονεκτήματα / Μειονεκτήματα αρχιτεκτονικής συσσωρευτή

- **Κατά**:
  - Χρειάζονται πολλές εντολές για ένα πρόγραμμα.
  - Κάθε φορά πολλαπλές μεταφορές από τη μνήμη.
  - Bottleneck (σημείο συμφόρησης) ο Accum!
- **Υπέρ**:
  - Εύκολοι compilers, κατανοητός προγραμματισμός, εύκολη σχεδίαση h/w.
  - Λύση; Πρόσθεση καταχωρητών για συγκεκριμένες λειτουργίες (ISAs καταχωρητών ειδικού σκοπού).

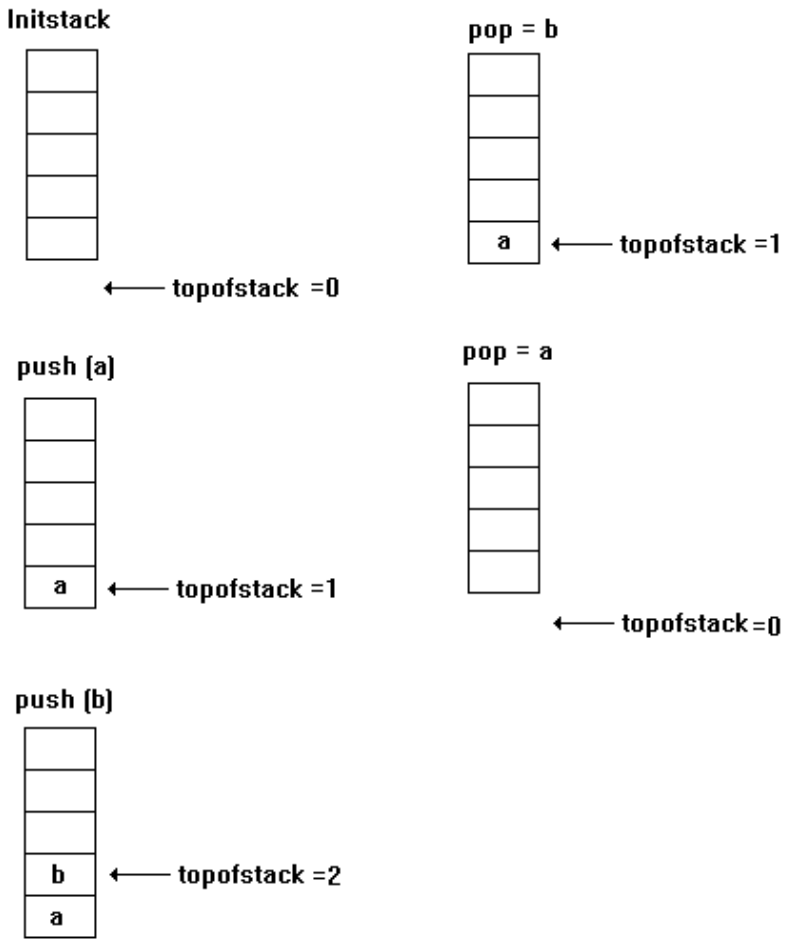
2 από 3
Αρχιτεκτονική Στοίβας

Τι είναι σορός

- Ο σορός (stack) είναι μια θεμελιώδη δομή τύπου LIFO (Last In First Out).
- Καταλαμβάνει ένα μέρος της μνήμης.
- Δύο είναι οι βασικές λειτουργίες:
  - Ώθηση (push), τοποθέτηση στοιχείου.
  - Εξαγωγή (pop), απομάκρυνση στοιχείου.
- Οι λειτουργίες γίνονται είτε στην κορυφή είτε στη βάση.



Παράδειγμα λειτουργιών pop/push

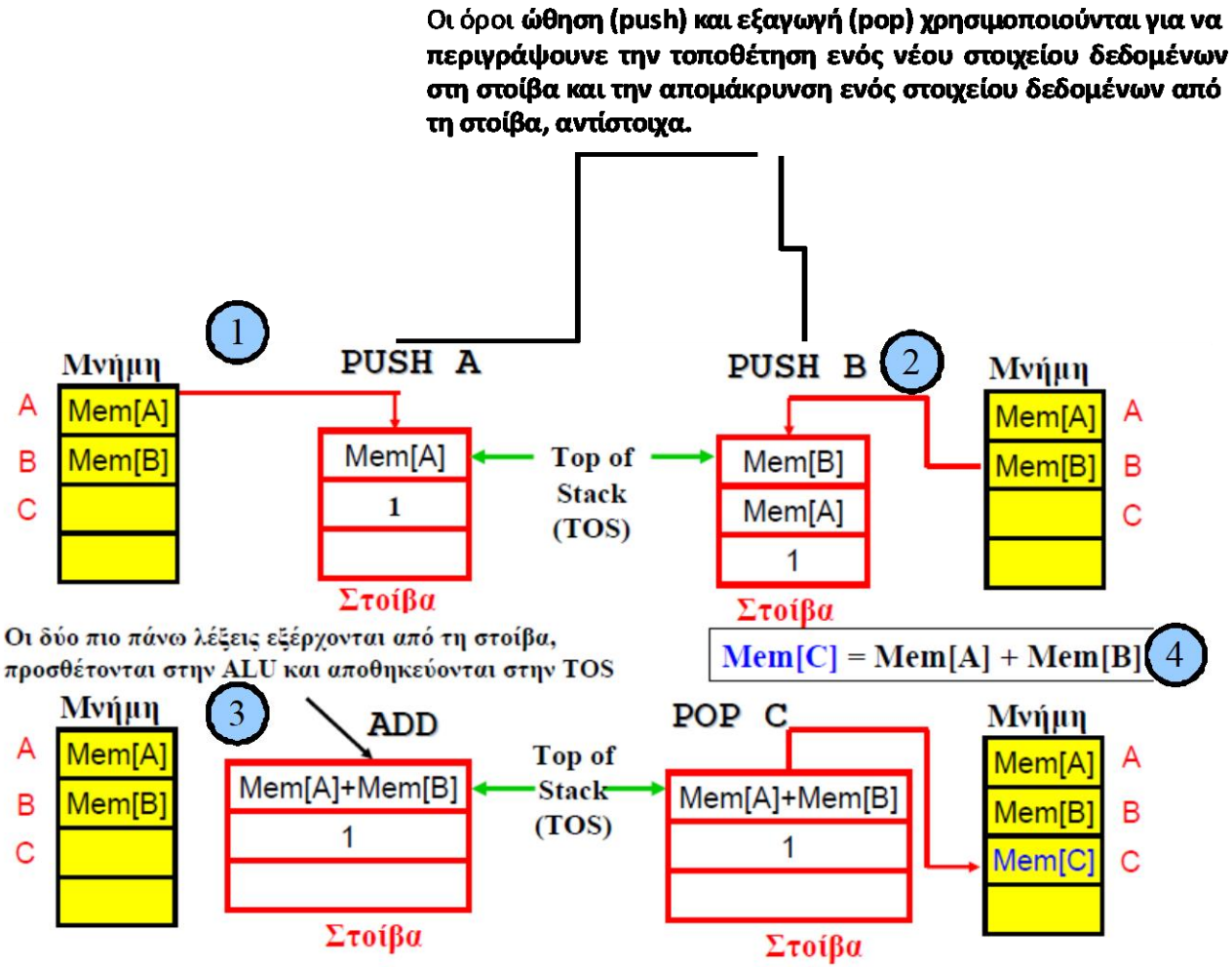


Βασικά Στοιχεία Αρχιτεκτονικής Στοίβας

- Καθόλου registers! Stack model ~ 1960!!!
- Στοίβα που μεταφέρονται τα ορίσματα που αρχικά βρίσκονται στη μνήμη. Καθώς βγαίνουν γίνονται οι πράξεις και το αποτέλεσμα ξαναμπαίνει στη στοίβα.
- Παράδειγμα τα HP calculators με reverse polish notation.
- Παράδειγμα: A=B+C

```
push Address C
push AddressB
add
pop AddressA
```

Αρχιτεκτονική Στοίβας (1/2)



Παράδειγμα αρχιτεκτονικής στοίβας (1/2)

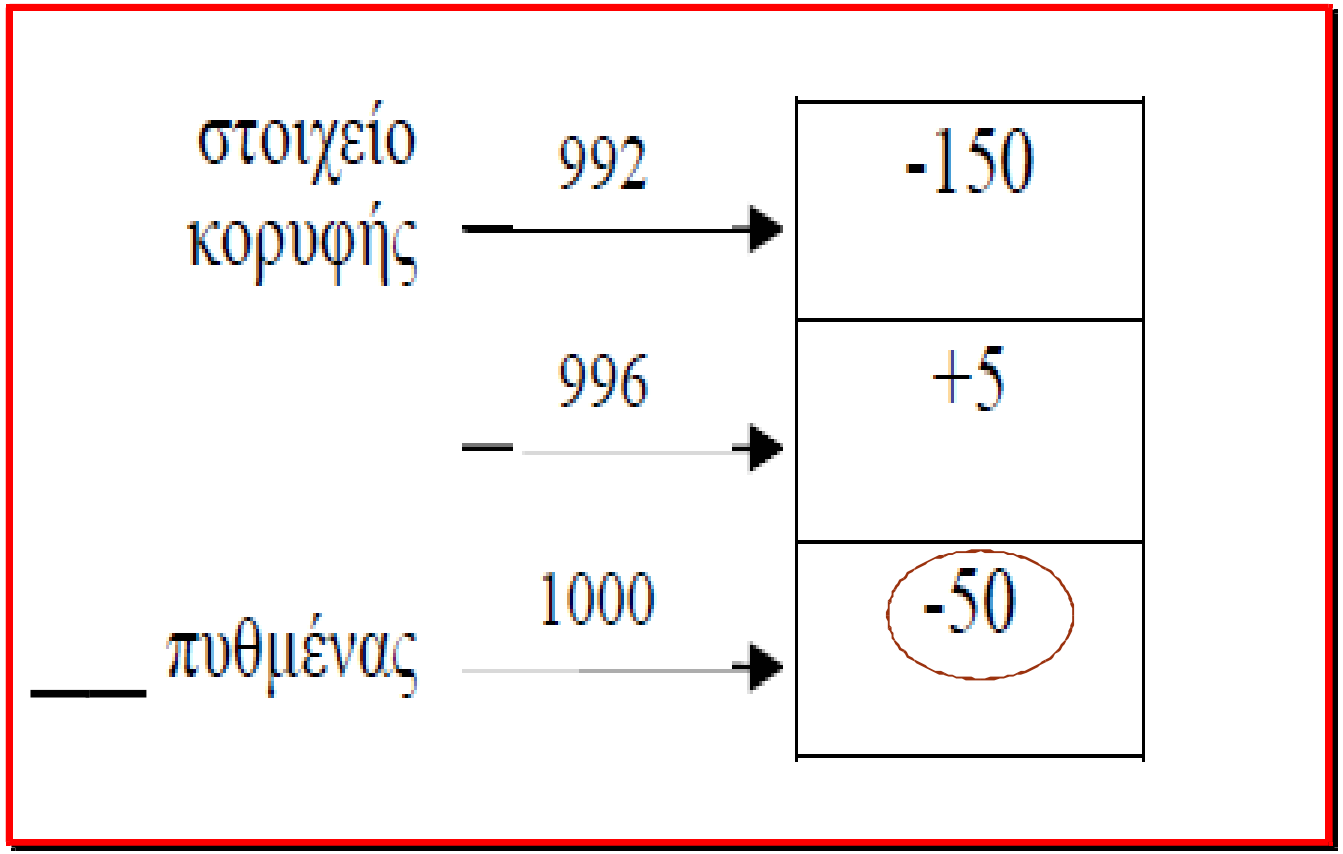
Έστω ότι σε μια δομή στοίβας πρόκειται να αποθηκευτούν διαδοχικά τρία στοιχεία με τιμές -50, +5 και -150 αντίστοιχα και κάθε στοιχείο καταλαμβάνει 4 bytes. Εάν ο πυθμένας βρίσκεται στη διεύθυνση 1000, σε ποιες διευθύνσεις θα αποθηκευτούν αυτά τα τρία στοιχεία στην περίπτωση που:

α) τα στοιχεία αποθηκεύονται διαδοχικά σε θέσεις μνήμης με μικρότερη διεύθυνση,

β) τα στοιχεία αποθηκεύονται διαδοχικά σε θέσεις μνήμης με μεγαλύτερη διεύθυνση.

Παράδειγμα αρχιτεκτονικής στοίβας (2/2)

Α)



Εντολές (Instructions)

- Πώς ένας επεξεργαστής μπορεί να εκτελεί προγράμματα που γράφηκαν από τον άνθρωπο/χρήστη;
  - Ο επεξεργαστής καταλαβαίνει μόνο δυαδικές τιμές.
  - Πολύ λίγοι άνθρωποι/χρήστες μπορούν να γραφούν προγράμματα σε γλώσσα μηχανής (0 ή 1).
- Χρησιμοποιούνται **διαφορετικά επίπεδα αναπαράστασης** ενός προγράμματος.

Αρχιτεκτονική Στοίβας (2/2)

Οι δυο βασικές λειτουργίες που μπορούν να εκτελεστούν σε μια στοίβα είναι η ώθηση και η εξαγωγή, οι οποίες μπορούν να υλοποιηθούν ακολουθώντας τα παρακάτω βήματα (υποθέτουμε πως τα στοιχεία τοποθετούνται σε θέσης μνήμης με μεγαλύτερη διεύθυνση και κάθε στοιχείο καταλαμβάνει 4 bytes):

Ώθηση:

- Αύξηση της τιμής (του περιεχομένου) του δείκτη της στοίβας κατά 4.
- Ώθηση του επόμενου στοιχείου, στην κορυφή της στοίβας. Ο δείκτης στοίβας δείχνει το νέο στοιχείο κορυφής.

Εξαγωγή:

- Εξαγωγή του πρώτου στοιχείου που υπάρχει στην κορυφή της στοίβας.
- Μείωση της τιμής (του περιεχομένου) του δείκτη στοίβας κατά 4. ο δείκτης στοίβας δείχνει στο προηγούμενο στοιχείο, το οποίο είναι τώρα το νέο στοιχείο κορυφής

Παράδειγμα αρχιτεκτονικής στοίβας

Έστω η στοίβα του παρακάτω σχήματος στην οποία ο δείκτης στοίβας βρίσκεται στη διεύθυνση 1000 και δείχνει το στοιχείο με τιμή -28.

Δείκτης στοίβας **1000---->**

-28
17
739
.
.
.
43

Να εξηγήσετε τον τρόπο με τον οποίο θα εκτελεστεί:

i) ώθηση στη στοίβα του στοιχείου με τιμή 19,

ii) εξαγωγή από τη στοίβα του στοιχείου κορυφής.

• Ποια θα είναι η διεύθυνση του δείκτη στοίβας σε κάθε μία από αυτές τις περιπτώσεις;

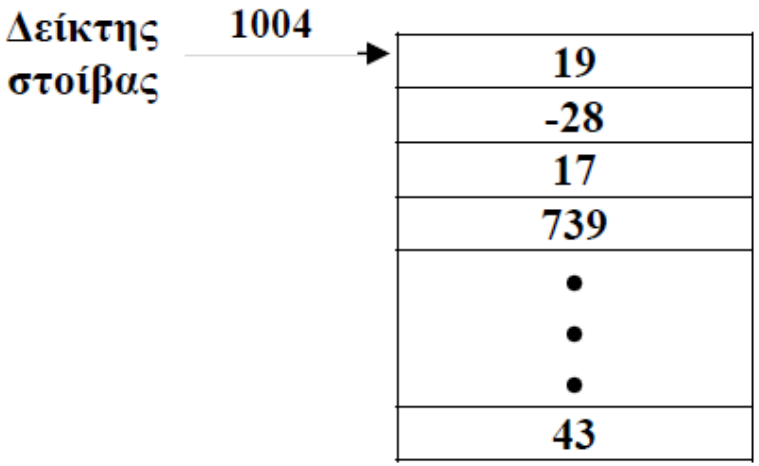
Να θεωρήσετε ότι κάθε στοιχείο καταλαμβάνει 4 bytes και πως τα στοιχεία τοποθετούνται διαδοχικά σε θέσεις μνήμης με μεγαλύτερη διεύθυνση.

• Να σχεδιάσετε τη μορφή που θα έχει η νέα στοίβα μετά την ώθηση και την εξαγωγή στοιχείων αντίστοιχα.

PUSH
(ώθηση στοιχείου στη στοίβα)

i. Ώθηση:

- Αύξηση της τιμής (του περιεχομένου) του δείκτη της στοίβας κατά 4. Η νέα τιμή του δείκτη στοίβας είναι το 1004.
- Ώθηση του επόμενου στοιχείου, που έχει τιμή 19, στην κορυφή της στοίβας. Ο δείκτης δείχνει το νέο στοιχείο κορυφής, με τιμή 19.

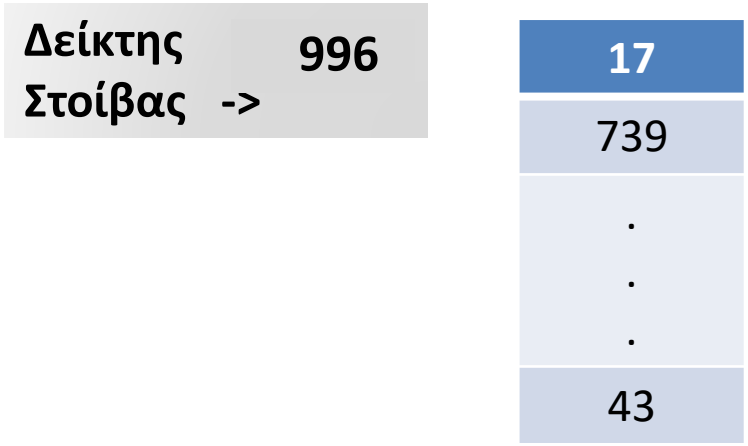


POP
(εξαγωγή στοιχείου από τη στοίβα)

ii. Εξαγωγή:

• Εξαγωγή του πρώτου στοιχείου που υπάρχει στην κορυφή της στοίβας, το οποίο είναι το -28.

• Μείωση της τιμής (του περιεχομένου) του δείκτη στοίβας κατά 4. Η νέα τιμή του δείκτη στοίβας είναι το 996. Ο δείκτης στοίβας δείχνει στο προηγούμενο στοιχείο, το οποίο έχει τιμή 17 και είναι τώρα το νέο στοιχείο κορυφής.



3 από 3
Αρχιτεκτονική Καταχωρητών

Η αρχιτεκτονική Load-Store (1/2)

- Η εφαρμογή λειτουργιών σε τελεστέους που βρίσκονται στη μνήμη κοστίζει.
  - Αυξημένος χρόνος πρόσβασης στη μνήμη (latency).
- Μικρότερο κόστος εάν οι τελεστέοι είναι αποθηκευμένοι σε καταχωρητές
- Αρχιτεκτονική **Load-Store:**
  - Πολλοί καταχωρητές στον επεξεργαστή.
    - Όλες οι λειτουργίες στον επεξεργαστή εφαρμόζονται σε τελεστέους που βρίσκονται σε καταχωρητές.
  - Η πρόσβαση στη μνήμη γίνεται μόνο με εντολές μεταφοράς δεδομένων (Load-Store).

Η αρχιτεκτονική Load-Store (2/2)

- Πλεονεκτήματα:
  - Μείωση του αριθμού των προσβάσεων στη μνήμη.
  - Απλοποίηση της ISA.
  - Διευκόλυνση του μεταγλωττιστή (compiler).
    - Διαδικασία register allocation.
- Σχεδόν όλοι οι επεξεργαστές που σχεδιάστηκαν μετά το 1980 χρησιμοποιούν αρχιτεκτονική LOAD-STORE.

Ταχύτητα (οι καταχωρητές είναι γρηγορότεροι από τη μνήμη).

Ευκολία χρήσης και αποδοτικότητα.

Παλαιοί υπολογιστές έχουν καταχωρητές συγκεκριμένης χρήσης (μειονέκτημα).

Αυξάνεται ο αριθμός των καταχωρητών από γενιά σε γενιά.

2 είναι οι Υποκατηγορίες: "Αρχιτεκτονική Καταχωρητών"

Register - Memory

Αφήνουν το ένα όρισμα να είναι στη μνήμη (πχ. 80386)

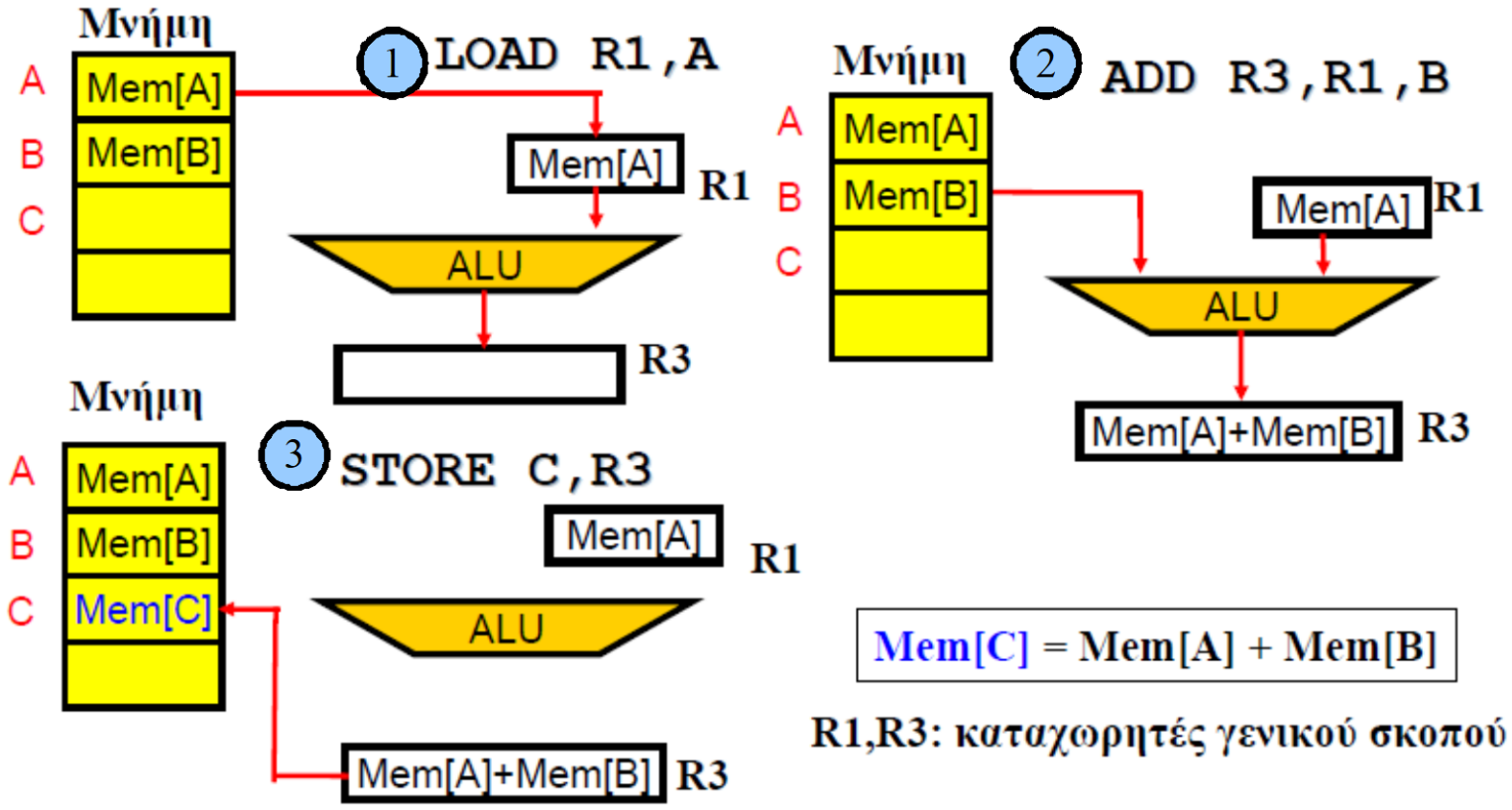
Register – register (load store) (1980+)

A = B + C

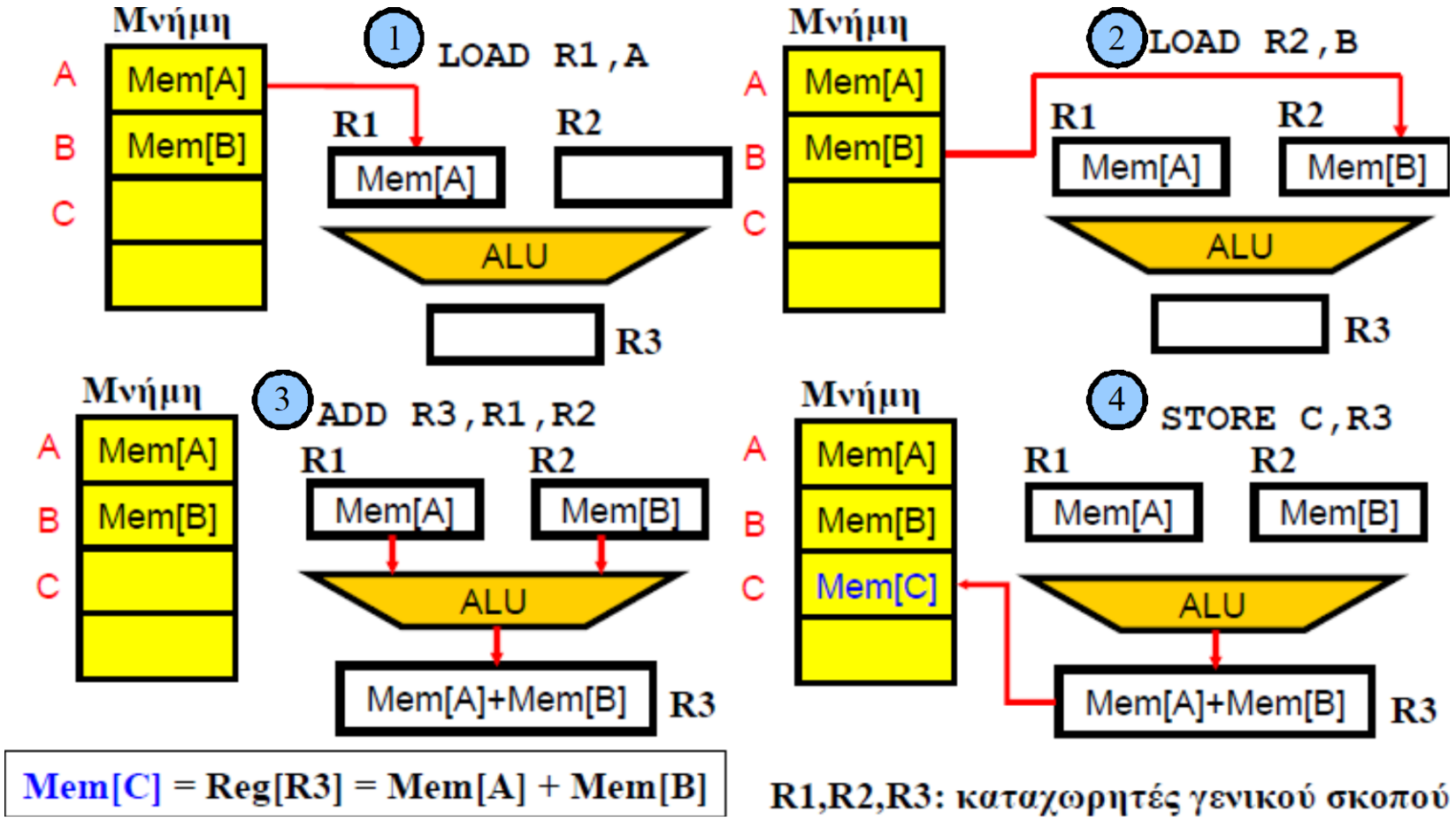
```
Load R1, B
Add R1, C
Store A, R1
```

```
Load R1, B
Load R2, C
Add R3, R1, R2
Store A, R3
```

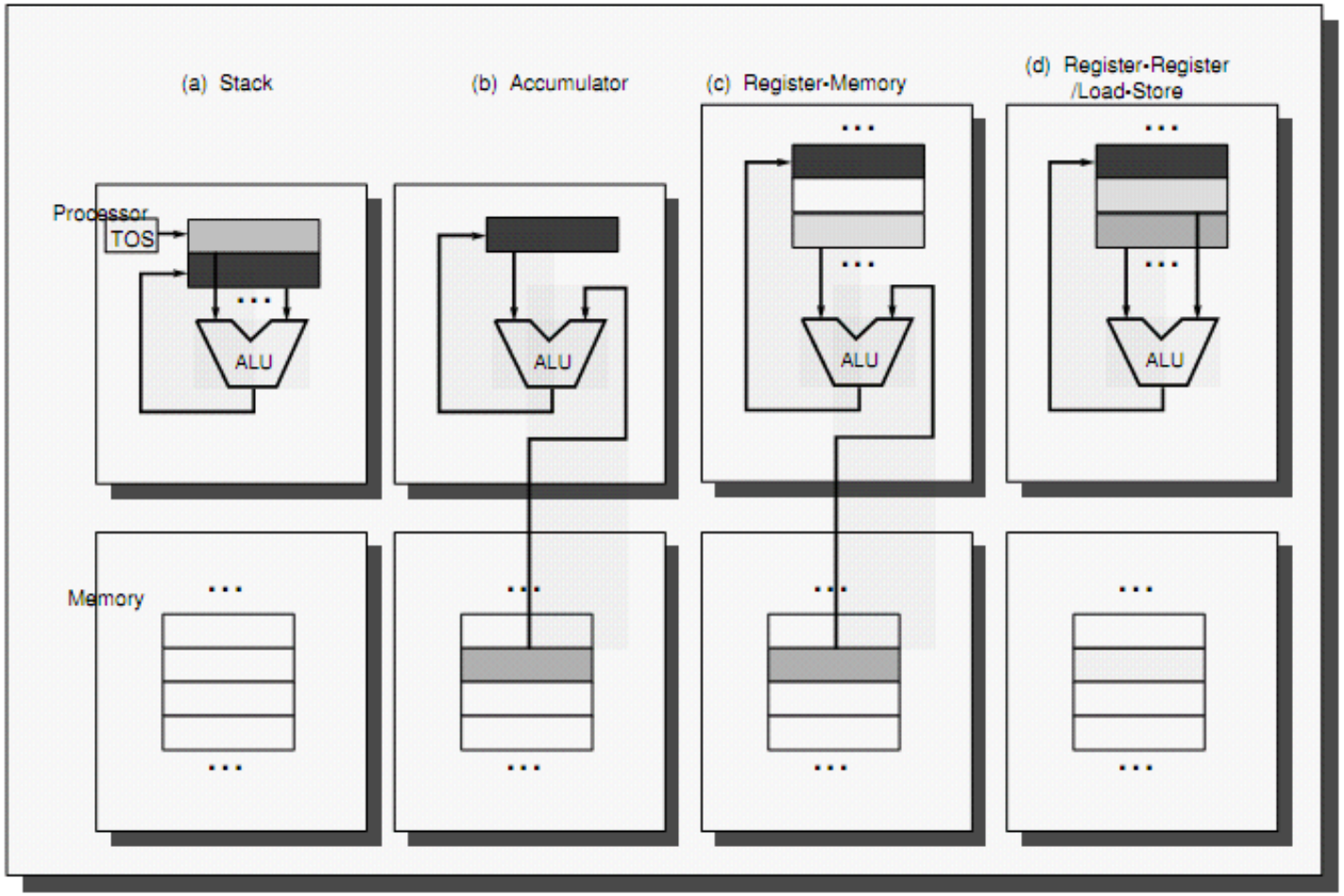
Αρχιτεκτονική Καταχωρητή-Μνήμη (reg-mem)



Αρχιτεκτονική Load-Store



Σύνοψη ISA


(a) Stack
(b) Accumulator
(c) Register-Memory
(d) Register-Register /Load-Store
Processor
TOS
ALU
ALU
ALU
ALU
Memory

Τυπικοί Συνδυασμοί τελεστέων μνήμης και ολικών τελεστών

Αριθμός διευθύνσεων μνήμης	Μέγιστος αριθμός επιτρεπόμενων τελεστών	Τύπος αρχιτεκτονικής	Παραδείγματα
0	3	καταχωρητής-καταχωρητής	Alpha, ARM, MIPS, PowerPC, SPARC, SuperH, Trimedia TM5200
1	2	καταχωρητής-μνήμη	IBM 360/370, Intel 80x86, Motorola 6800, TI TMS320C54x
2	2	Μνήμη - μνήμη	VAX (επίσης έχει τρεις μορφές τελεστέων)
3	3	Μνήμη - μνήμη	VAX (επίσης έχει δυο μορφές τελεστέων)

Συγκρίνοντας τις ISA για την ίδια λειτουργία

Η ακολουθία εντολών στην συμβολική γλώσσα Assembly και για τις τέσσερις Αρχιτεκτονικές Συνόλου Εντολών (ISAs) για την λειτουργία **C = A + B**.

Stack	Accumulator	Register (register-memory)	Register (load-store)
Push A	Load A	Load R1,A	Load R1,A
Push B	Add B	Add R3,R1,B	Load R2,B
Add	Store C	Store R3,C	Add R3,R1,R2
Pop C			Store R3,C

Προσέξτε ότι η εντολή ADD έχει υπονοούμενους τελεστέους για τις 2 πρώτες αρχιτεκτονικές.

Θεωρούμε ότι A,B βρίσκονται στη μνήμη.

Σύγκριση των 3 κύριων κατηγοριών ISA

Τύπος	Πλεονεκτήματα	Μειονεκτήματα
Καταχωρητής-καταχωρητής (0,3)	Απλή, σταθερού μήκους εντολή κωδικοποίησης. Απλό μοντέλο παραγωγής κώδικα. Οι εντολής παίρνουν παρόμοιο αριθμό χτύπων για να εκτελεστούν.	Υψηλότερη εντολή μετράει από αρχιτεκτονικές με αναφορές μνήμης σε εντολές. Περισσότερες εντολές και λιγότερη πυκνότητα εντολής οδηγούν σε μακροσκελή προγράμματα
Καταχωρητής-μνήμη (1,2)	Πρόσβαση στα δεδομένα χωρίς πρώτα να υπάρξει ξεχωριστή εντολή φόρτωσης. Οι μορφές των εντολών τείνουν να είναι εύκολες στην κωδικοποίηση και αποδίδουν καλή πυκνότητα	Οι τελεστές δεν είναι ισοδύναμοι από τη στιγμή που η πηγή τελεστή σε μια δυαδική λειτουργία είναι κατεστραμμένη. Κωδικοποιώντας έναν αριθμό εγγραφής και μια διεύθυνση μνήμης για κάθε εντολή μπορεί να περιορίσει τον αριθμό των καταχωρητών. Χτύποι ανά εντολή ποικίλλουν ανά θέση τελεστή.
Μνήμη-μνήμη (2,2) ή (3,3)	Η πιο συμπαγής. Δεν σπαταλάει καταχωρητές.	Μεγάλη αλλαγή σε μέγεθος εντολής, ειδικά για εντολές τριών-τελεστών. Επίσης, μεγάλη αλλαγή στην εργασία ανά εντολή. Οι προσβάσεις μνήμης δημιουργούν συμφόρηση μνήμης. Δεν χρησιμοποιείται σήμερα

Κατηγοριοποίηση ως προς τον αριθμό των τελεστών

- Οι αρχιτεκτονικές με καταχωρητές γενικού σκοπού (General Purpose Register) μπορεί να έχουν 2 ή 3 τελεστέους.
  - Παράδειγμα: **MIPS** **`add r1,r2,r3`**
  - ΠΑΡΑΔΕΙΓΜΑ: **8086** **`add a1,b1`**
- Στους 3 τελεστέους: Ο ένας προορισμός, οι δύο πηγή.
- Στους 2 τελεστέους: Ο ένας είναι και πηγή και αποτέλεσμα (δηλ. Το αποτέλεσμα αποθηκεύεται πάνω σε αυτόν).

Κατηγοριοποίηση ως προς αν επιτρέπονται διευθύνσεις μνήμης

- Ο αριθμός των τελεστέων μνήμης για μια ALU μπορεί να ποικίλει από κανένα έως τρεις.
  - π.χ. τελεστέος1,τελεστέος2 και προορισμός είναι καταχωρητές (καμία διεύθυνση μνήμης).
  - π.χ. τελεστέος1 διεύθυνση μνήμης, τελεστέος2 και προορισμός καταχωρητές (μια διεύθυνση μνήμης).
  - π.χ. τελεστέος1 και τελεστέος2 διεύθυνση μνήμης, προορισμός καταχωρητής (δυο διευθύνσεις μνήμης).
  - π.χ. τελεστέος1,τελεστέος2 και προορισμός διευθύνσεις μνήμης (τρεις διευθύνσεις μνήμης) .....
- Υπάρχουν επτά πιθανοί συνδυασμοί.
- 3 όμως είναι οι βασικές κατηγορίες.
  - Όλοι καταχωρητές ή ένας καταχωρητής και μια διεύθυνση μνήμης ή δυο διευθύνσεις μνήμης.

Σύγκριση των ISA ως προς το μήκος των εντολών

- Εντολές μεταβλητού μήκους:
  - 1-17 bytes 80x86.
  - 1-54 bytes VAX, IBM.
- Γιατί τέτοιες διαφοροποιήσεις στο μέγεθος;
  - Instruction Memory ακριβή, οικονομία χώρου!!!!
- Εμείς στο μάθημα: register-register ISA! (load- store).
  1. Οι καταχωρητές είναι γρηγορότεροι από τη μνήμη.
  2. Μειώνεται η κίνηση με μνήμη.
  3. Υποστηρίζεται όμως και register-memory ISA.
- Compilers πιο δύσκολοι όταν υπάρχει μεταβλητό μέγεθος!!!

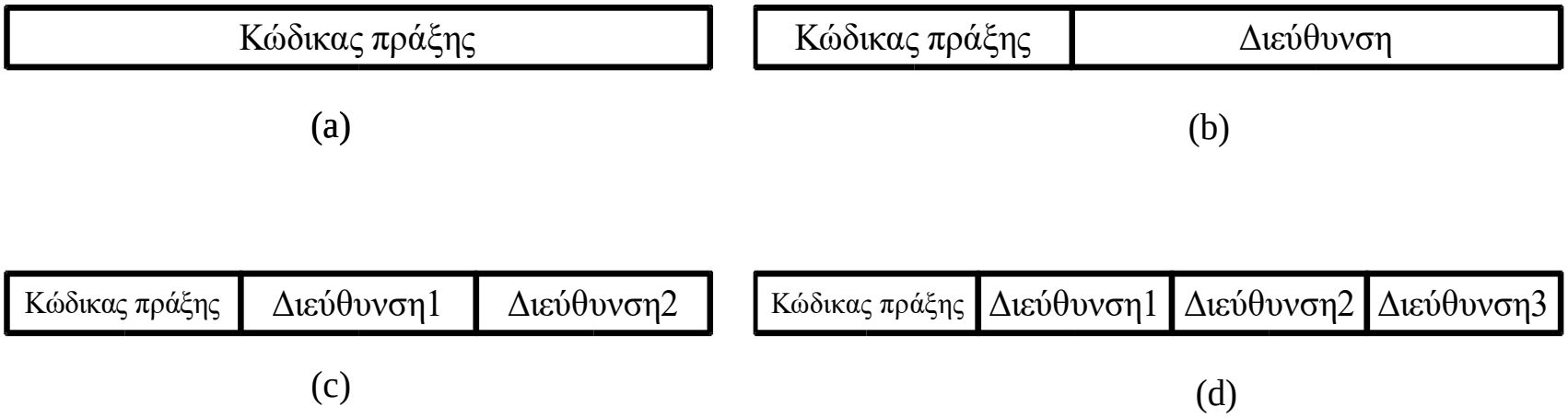
Βασικά Στοιχεία ISA 8086

- Μεταβλητού Μήκους εντολές.
- 16 καταχωρητές των 2Byte.
- Little Endian αρχιτεκτονική (βασική μονάδα: Byte).
- Το πλήθος των ορισμάτων είναι πάντα σταθερό.
- Εντολές.
  - Αριθμητικές (add, sub, mul, div, inc).
  - Λογικές (and, or, xor, κτλ.).
  - Σύγκρισης (cmp, κτλ).
  - Διακλάδωσης (jmp, je, jae, κτλ).
  - Μεταφοράς δεδομένων.

Γιατί να μην έχουμε πολλούς καταχωρητές;

- Αφού οι καταχωρητές είναι τόσο....«γρήγοροι» γιατί να μην μεγαλώσουμε το μέγεθος του register file;
- 2η αρχή: Όσο μικρότερο τόσο ταχύτερο!
- Αν το register file πολύ μεγάλο, πιο πολύπλοκη η αποκωδικοποίηση, πιο μεγάλος ο κύκλος ρολογιού (κατά την κατάσταση ID) άρα .... υπάρχει tradeoff.
- Μνήμη οργανωμένη σε bytes:
  - (Κάθε byte και ξεχωριστή δ/νση).

Συνήθως οι εντολές όλων των ISA χρησιμοποιούν κάποιες από τις δομές



Τέσσερις κοινές μορφές εντολών:

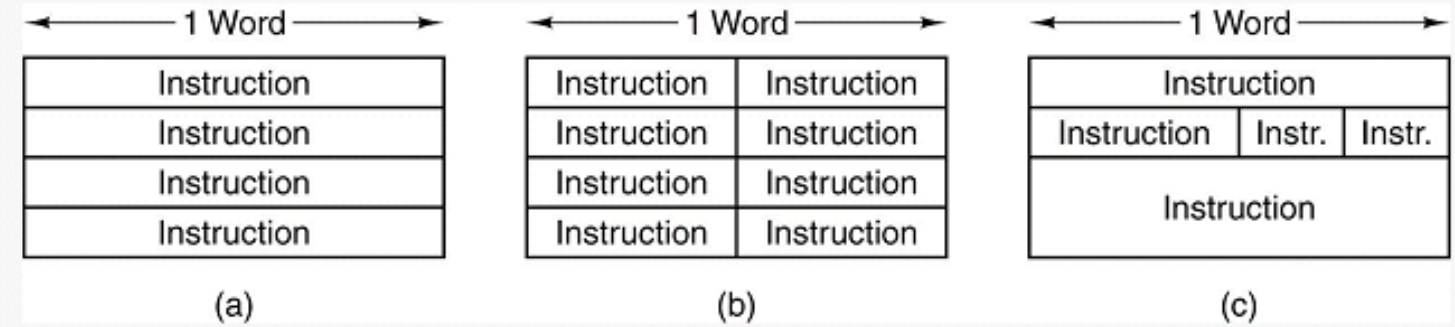
(a) Μηδενική-διεύθυνση εντολών.

(b) Μοναδιαία- διεύθυνση εντολών.

(c) Διπλή-διεύθυνση εντολών.

(d) Τριπλή-διεύθυνση εντολών.

Οι σχέσεις των εντολών ως προς τη word



- **Σε ποια κατηγορία ανήκει ο MIPS;**
- **Σε ποια κατηγορία ανήκει ο 8086;**

Αριθμητικές Πράξεις στο 8086

- add (παράμετρος1), (παράμετρος2).
- π.χ. **add ax, bx** (ax<= ax+ bx).

- a=b+c+d+e (αν a,b,c,d,e είναι σε καταχωρητές).

```
add a,b
add a,c
add a,d
add a,e
```

4 εντολές....

Όλα είναι σε δυαδική αναπαράσταση μέσα στον Η/Υ

- Όλα τα στοιχεία (εντολές και δεδομένα) βρίσκονται σε δυαδική αναπαράσταση.
- Χρησιμοποιείται πλήθος κωδικοποιήσεων στον επεξεργαστή για να ερμηνεύεται κάθε φορά τι σημαίνει μια σειρά από 1 και 0.

- Για παράδειγμα, αν στη μνήμη είναι τοποθετημένα τα παρακάτω στοιχεία:
- 01001000011001... θα έχουν διαφορετική ερμηνεία αν είναι χαρακτήρες ASCII, αριθμοί ή εντολές.

Αναπαράσταση εντολών στον Υπολογιστή (8086)

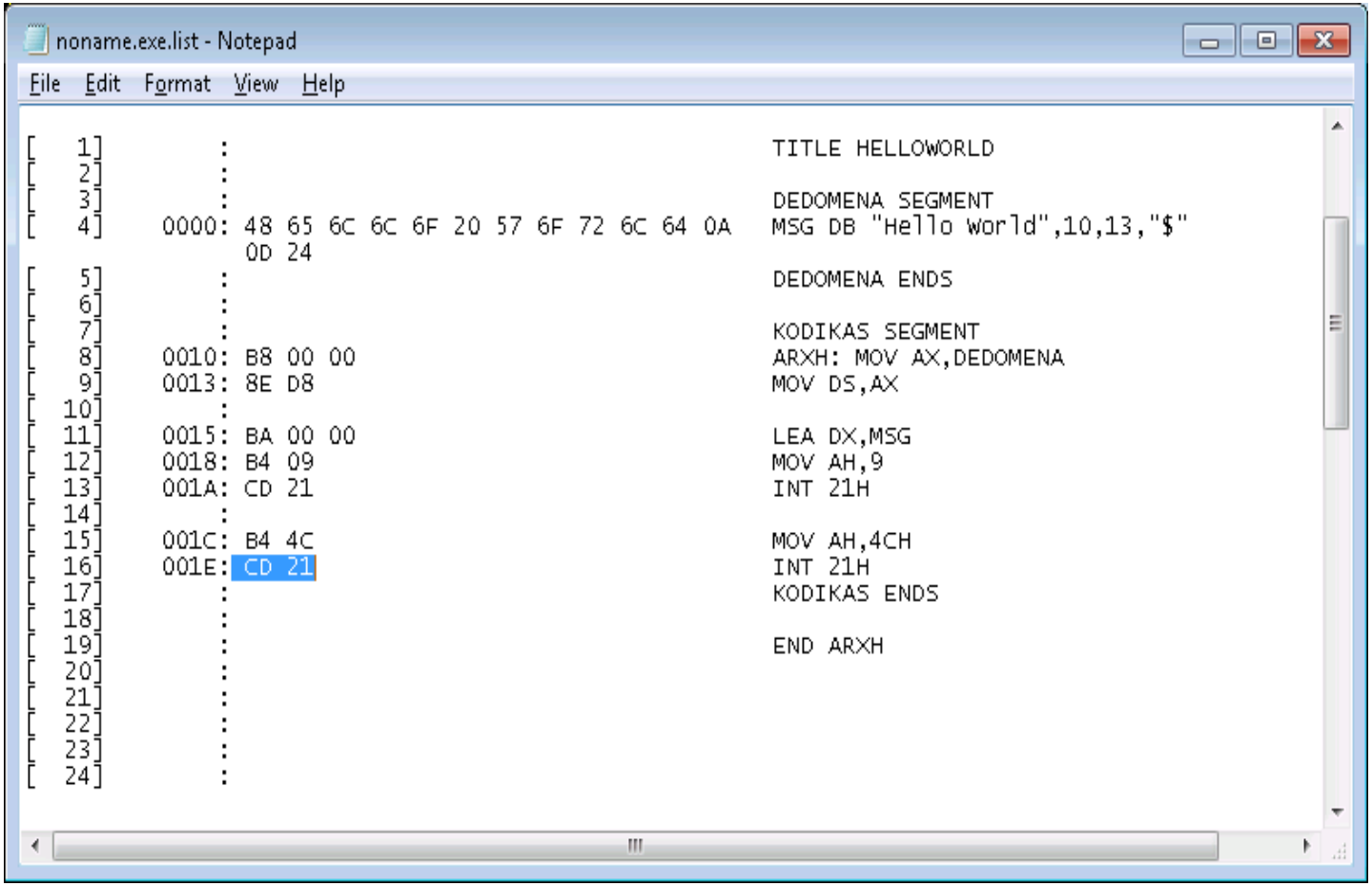
- Μεταβλητού μήκους. Πρέπει να ξέρει για κάθε machine code πόσα byte απαιτούνται.
- Π.χ. Η B4 4C (MOV AH, 4Ch).
  - Το B4 είναι η MOV AH, δηλαδή να μπει μια αριθμητική τιμή στον AH.
  - Μόλις διαβαστεί το B4 ο υπολογιστής το αποκωδικοποιεί και καταλαβαίνει ότι πρέπει να φέρει ακόμη ένα Byte (το επόμενο).
  - Φέρνει το επόμενο 4C και έχοντας τα 2 Byte τότε τα αποκωδικοποιεί και τα εκτελεί.
- Ερώτηση: Η εντολή **B4 00** τι κάνει;

Αναπαράσταση εντολών στον Υπολογιστή (8086) – Παράδειγμα1

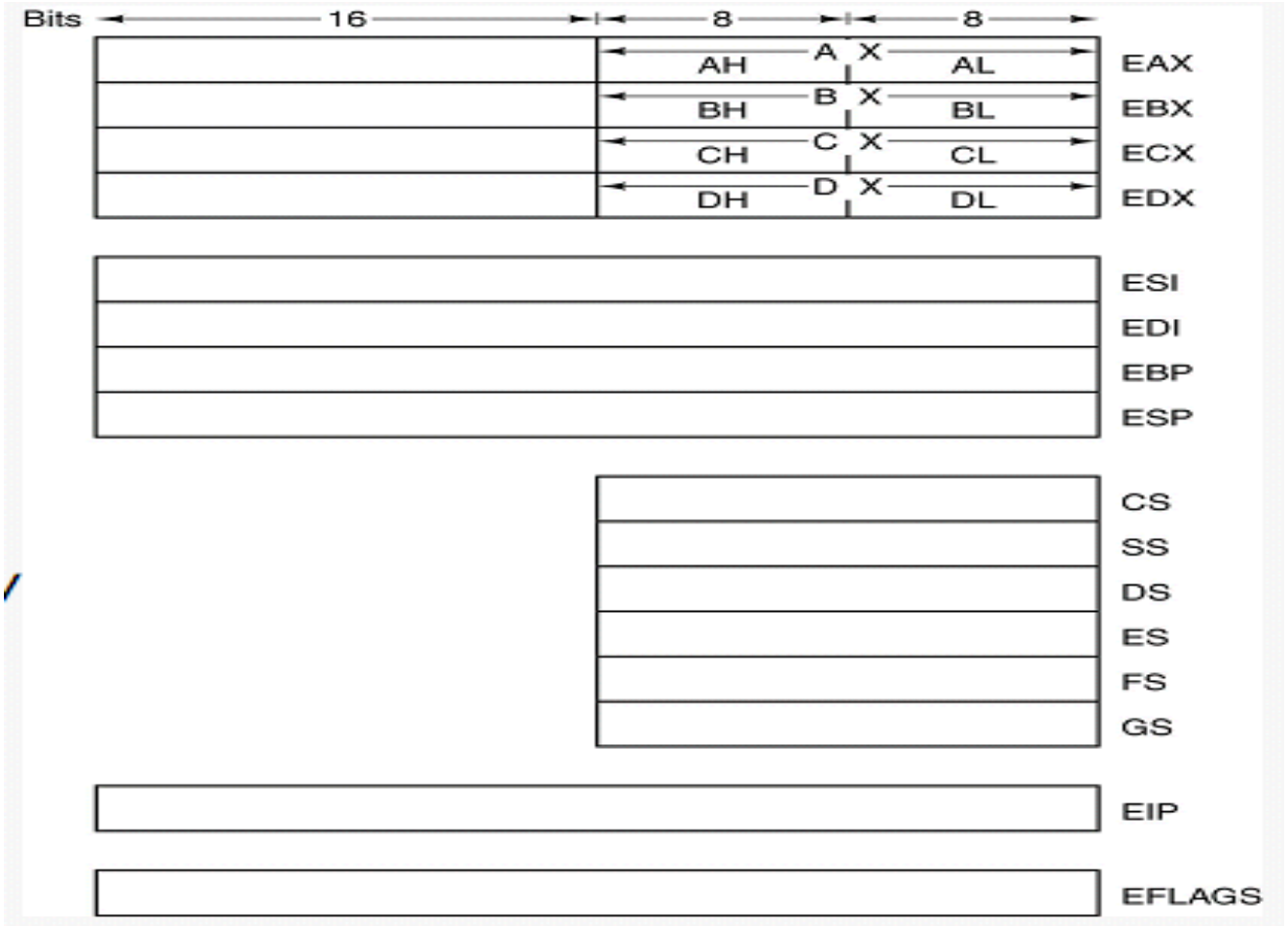
- Στο πρόγραμμα που ακολουθεί βρείτε ποια bits καθορίζουν τη λειτουργία και ποια τις παραμέτρους.

- Εξηγείστε τι θα γίνει στις εντολές CD 71,BA1222.

Αναπαράσταση εντολών στον Υπολογιστή (8086) – Παράδειγμα 2



Οι βασικοί καταχωρητές του Pentium IV



Οι αριθμητικές πράξεις που υποστηρίζει ο IA32 (32bit x86) (1/2)

Type	Lenght	Range
Unsigned char	8 bits	0 to 255
char	8 bits	-128 to 127
enum	16 bits	-32,768 to 32,767
Unsigned int	16 bits	0 to 65,535
short int	16 bits	-32,768 to 32,767
int	16 bits	-32,768 to 32,767
Unsigned long	32 bits	0 to 4,294,967,295
long	32 bits	-2,147,483,648 to 2,147,483,647

Δεν υποστηρίζονται float

Length	Size name
4-bit	nibble
8-bit	byte
16-bit	word
32-bit	Dword (or doubleword)

Ο 8086 έχει 79 εντολές

```
AAA AAD AAM AAS ADC ADD AND
CALL CBW CLC CLD CLI CMC CMP CMPS CMPXCHG
CWD DAA DAS DEC DIV ESC HLT IDIV
IMUL IN INC INT INTO IRET/IRETD
JxxJCXZ/JECXZ JMP
LAHF LDS LEA LES LOCK LODS LOOP
MOV MOVS LOOPE/LOOPZ LOOPNZ/LOOPNE
MUL NEG NOP NOT OR OUT POP POPF/POPFD PUSH
RCL RCR PUSHF/PUSHFD
REP REPE/REPZ RET/RETF REPNE/REPNZ
ROL ROR SAHF SAL/SHL SAR SBB SCAS SHL SHR
STC STD STI STOS SUB
TEST WAIT/FWAIT XCHG XLAT/XLATB XOR
```

Οι αριθμητικές πράξεις που υποστηρίζει ο IA32 (32bit x86) (2/2)

Type	1 Bit	8 Bits	16 Bits	32 Bits	64 Bits	128 Bits
Bit						
Signed integer		x	x	x		
Unsigned integer		x	x	x		
Binary coded decimal integer		x				
Floating point				x	x	

Ρυθμοί λειτουργίας των x86 (1/3): Πραγματικός ρυθμός

- Πραγματικός ρυθμός (real - mode):
  - Χρησιμοποιείται από το MS-DOS.
  - Διαθέσιμα μόνο το πρώτο 1MB RAM.
  - Ο ρυθμός προεπιλογής ύστερα από reset ή power-on.
  - Το 8086 υποστηρίζει μόνο αυτόν το ρυθμό.
  - Υποστηρίζονται όλες οι εντολές του επεξεργαστή.
  - Κάθε φορά εκτελείται μια διεργασία.

Ρυθμοί λειτουργίας των x86 (2/3): Προστατευμένος ρυθμός

- Προστατευμένος ρυθμός (protected mode):
  - Υποστηρίζεται από 80286 και έπειτα.
  - Χρησιμοποιείται από 80386 και μετά.
  - Τα Windows, Linux, FreeBSD χρησιμοποιούν αυτόν το ρυθμό.
  - Χρησιμοποιεί όλη τη μνήμη.
  - Διαφορετική χρήση των τμημάτων ως προς τον πραγματικό ρυθμό.
  - Με την χρήση κατάλληλων καταχωρητών/σημαιών (protection enable bit) ενεργοποιείται αυτός ο ρυθμός από τον πραγματικό ρυθμό. Μπορεί να απενεργοποιηθεί ομοίως.
  - Υποστηρίζονται επίπεδα δικαιωμάτων.

Ενεργοποίηση του προστατευμένου ρυθμού:

- Πρέπει να τεθεί το 0bit του cr0 στην τιμή 1.
  - ΠΡΟΣΟΧΗ: Μόνο για assembly 80386 και έπειτα!
  - Μεταφέρουμε το cr0 στον eax.
  - Κάνουμε την πράξη OR.
  - Μεταφέρουμε το eax στο cr0.

```
; set PE bit
mov eax, cr0
or eax, 1
; far jump (cs = selector of code segment)
jmp cs:@pm
@pm:
; now we are in PM.
```

Ρυθμοί λειτουργίας των x86 (3/3): Ιδεατός 8086 ρυθμός

- Ιδεατός 8086 ρυθμός (virtual 8086 mode).
  - Πολλαπλές και ταυτόχρονες εκδόσεις/εκτελέσεις του πραγματικού ρυθμού μέσα σε ρυθμό εκτέλεσης πραγματικό.
  - Κάθε διεργασία έχει το δικό της περιβάλλον 8086.
  - Υποστηρίζονται σχεδόν όλες οι λειτουργίες του 8086 αλλά με κάποιους περιορισμούς (π.χ. Δεν επιτρέπεται απευθείας πρόσβαση στο υλικό ή προνομιακές εντολές).
  - Γίνεται software εξομοίωση των int οπότε υπάρχει καθυστέρηση κατά την είσοδο/έξοδο.

Παράρτημα

Δεδομένα

- Το στάδιο της εκτέλεσης μιας εντολής από τον επεξεργαστή αφορά στην επεξεργασία των δεδομένων.
- Ο επεξεργαστής ως ψηφιακό σύστημα επεξεργάζεται τα δεδομένα δυαδικά.
    - Αριθμητικά δεδομένα .
        - Δυαδική αναπαράσταση αριθμητικών δεδομένων.
    - Μη αριθμητικά δεδομένα-Χαρακτήρες.
        - Δυαδική αναπαράσταση χαρακτήρων.

Επεξεργαστής

- Ψηφιακό σύστημα (Γενικά).
  - Είναι μια εξειδικευμένη σχεδίαση για συγκεκριμένη εφαρμογή.
- Ο επεξεργαστής είναι ένα πολύπλοκο ψηφιακό σύστημα.
  - **Προγραμματιζόμενος** (μπορεί να εκτελεί πολλές εφαρμογές).
  - Αποτελείται από συνδυαστική και ακολουθιακή λογική.
  - "Η σχεδίαση ενός επεξεργαστή απαιτεί την εφαρμογή της "αφρόκρεμας" των τεχνικών σχεδίασης ψηφιακών συστημάτων" *.
  - Εάν σας ενδιαφέρει να σχεδιάσετε και να υλοποιήσετε έναν επεξεργαστή στο υλικό, μπορείτε να παρακολουθήσετε το προπτυχιακό μάθημα της **Σχεδίασης Ψηφιακών Συστημάτων**.
    - **Προηγμένη Σχεδίαση Ψηφιακών Συστημάτων (Μεταπτυχιακό).**

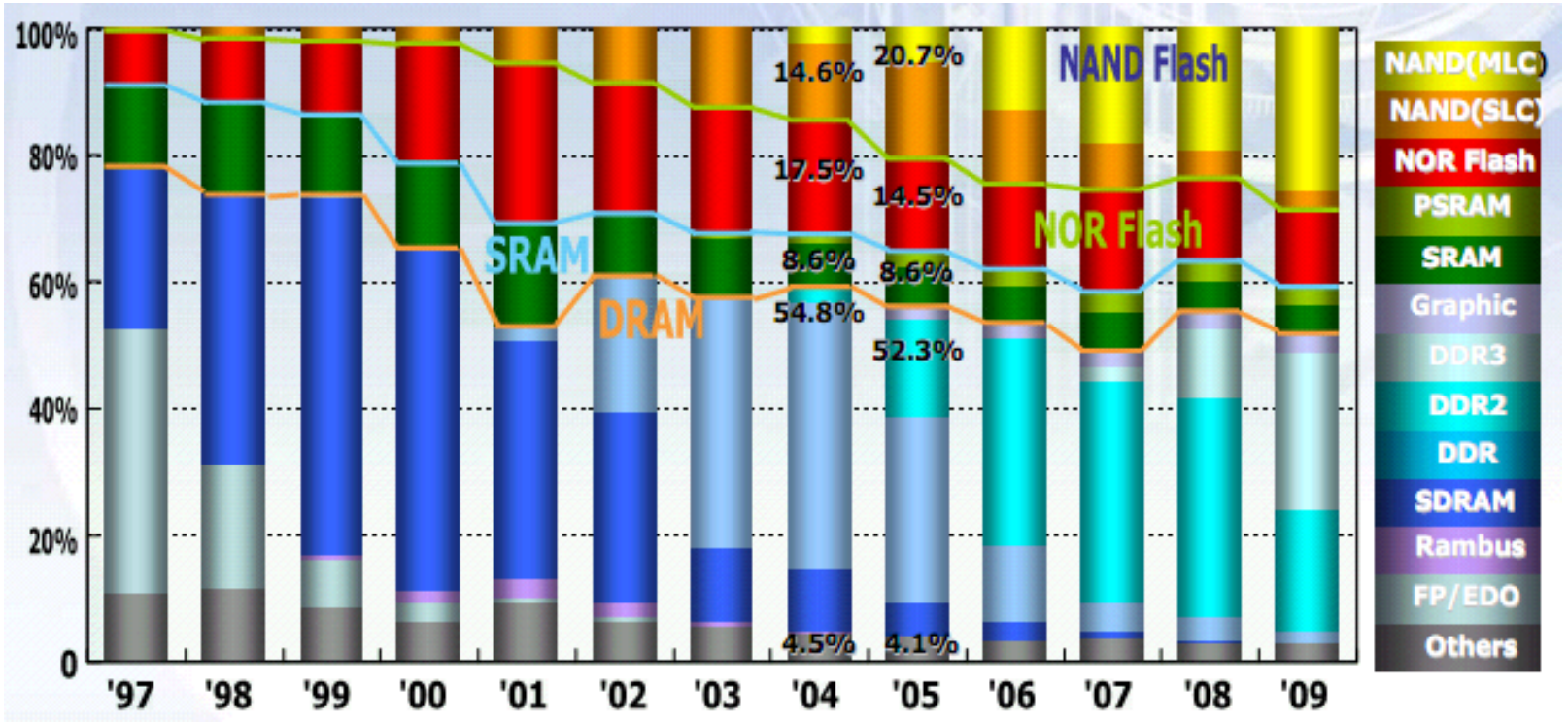
*** Πηγή: Design Automation Conference (SAC'99)**

Βασικές τεχνολογίες μνήμης

- **SRAM** (πολύ γρήγορη, πάρα πολύ ακριβή).
- **DRAM** (γρήγορη, ακριβή).
- **Flash** (NAND, NOR).
- **External Magnetic Storage** (hard disk, tape, dat).
- **External Optical Storage** (dvdrom, cdrom, bluray).

- Για την καλύτερη εκμετάλλευση των τεχνολογιών μνήμης, χρησιμοποιείται η **'Ιεραρχία της μνήμης'** (μελλοντική διάλεξη).

Memory Market Trend



Μεταφορά Δεδομένων

- Είναι απαραίτητη η δυνατότητα μεταφοράς δεδομένων μεταξύ εσωτερικών πόρων αποθήκευσης του επεξεργαστή.
  - Καταχωρητών και μνήμης.
- **Εντολές μεταφοράς δεδομένων (data transfer instructions).**
- Για αν προσπελάσουμε για να διαβάσουμε ή να μεταβάλλουμε τα περιεχόμενα της μνήμης ορίζονται δύο λειτουργίες.
  - Ανάγνωση από τη μνήμη (LOAD).
  - Εγγραφή/Αποθήκευση στη μνήμη (STORE).

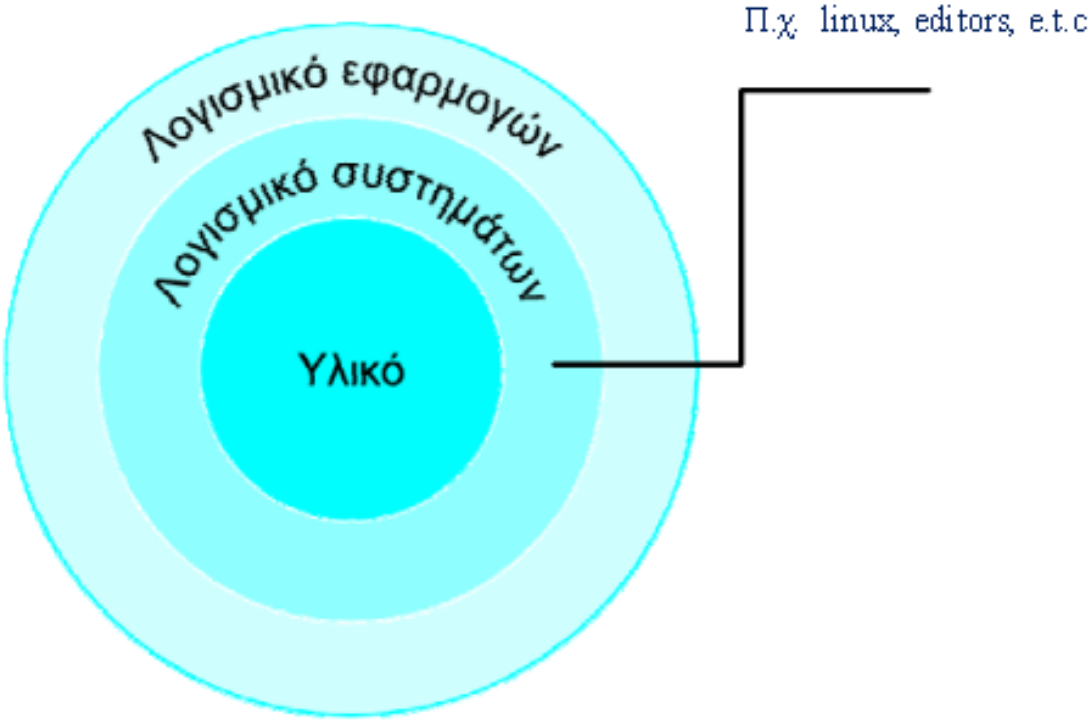
Ο επεξεργαστής καταλαμβάνει συγκεκριμένες εντολές

- Μια τυπική εφαρμογή (πχ. Επεξεργαστές κειμένου) μπορεί να αποτελείται από εκατομμύρια γραμμές κώδικα.
  - Χρήση πολύπλοκων βιβλιοθηκών λογισμικού για υποστήριξη της εφαρμογής.
- Το υλικό μπορεί να εκτελέσει μόνο εξαιρετικά απλές εντολές χαμηλού επιπέδου.
- Η μετάβαση από την πολύπλοκη εφαρμογή στις απλές εντολές περιλαμβάνει πολλά επίπεδα λογισμικού που διερμηνεύουν ή μεταφράζουν λειτουργίες υψηλού επιπέδου σε απλές εντολές του υπολογιστή.

Επίπεδα Λογισμικού

- Τα επίπεδα λογισμικού είναι οργανωμένα με έναν ιεραρχικό τρόπο.
- Ιεραρχία υπό την μορφή ομόκεντρων κύκλων.
- Οι εφαρμογές είναι ο εξωτερικός δακτύλιος.
- Το **λογισμικό συστημάτων (systems software)** βρίσκεται ανάμεσα στο υλικό και το λογισμικό των εφαρμογών.
- Το λογισμικό συστημάτων παρέχει χρήσιμες υπηρεσίες όπως τα λειτουργικά συστήματα, οι μεταγλωττιστές και οι συμβολομεταφραστές.

Η ιεραρχία του λογισμικού



Ιεραρχία υπο την μορφή ομόκεντρων κύκλων

Λογισμικό Συστημάτων

- **Λειτουργικό σύστημα:**
  - Παρέχει υπηρεσίες και λειτουργίες επίβλεψης καθώς διαχειρίζεται τους πόρους ενός Η/Υ προς όφελος των προγραμμάτων που εκτελούνται.
  - Οι πιο σημαντικές λειτουργίες του είναι οι εξής:
    - Διαχείριση βασικών λειτουργιών εισόδου κι εξόδου.
    - Κατανομή χώρου αποθήκευσης και μνήμης.
    - Παροχή δυνατοτήτων κοινής χρήσης του υπολογιστή μεταξύ πολλών εφαρμογών που τον χρησιμοποιούν ταυτόχρονα.
  - Παραδείγματα: Windows, Linux, MacOS.
- **Μεταγλωττιστής:**
  - Μεταφράζει εντολές μιας γλώσσας υψηλού επιπέδου (π.χ. C) σε εντολές συμβολικής γλώσσας (assembly).
- **Συμβολομεταφραστής:**
  - Μεταφράζει εντολές συμβολικής γλώσσας (assembly) σε γλώσσα μηχανής.

Τεχνική της αφαίρεσης

- Η έρευνα στα βάθη της περιγραφής του υλικού ή του λογισμικού αποκαλύπτει περισσότερες πληροφορίες.
- Χαμηλότερου επιπέδου λεπτομέρειες κρύβονται για να παρασχεθεί ένα απλούστερο μοντέλο στα υψηλότερα επίπεδα.
- Η χρήση **αφαιρέσεων** είναι η βασική τεχνική για το σχεδιασμό υπολογιστικών συστημάτων μεγάλης πολυπλοκότητας.
- Μια βασική διασύνδεση ανάμεσα στα επίπεδα αφαίρεσης είναι **η Αρχιτεκτονική Συνόλου Εντολών.**
  - Η διασύνδεση μεταξύ υλικού και λογισμικού χαμηλού επιπέδου.

Βασικά Στοιχεία MIPS

- Λέξεις των 32 bit (μνήμη οργανωμένη σε bytes, ακολουθεί το μοντέλο big Endian).
- 32 καταχωρητές γενικού σκοπού - REGISTER FILE.
  - Εντολές αποθήκευσης στη μνήμη (lw, sw).
  - Αριθμητικές εντολές (add, sub κλπ).
  - Εντολές διακλάδωσης (branch instructions).
  - Δεν αφήνουμε τις εντολές να έχουν μεταβλητό πλήθος ορισμάτων- π.χ. add a,b,c πάντα: a=b+c.
  - Θυμηθείτε την 1ηαρχή: Η ομοιομορφία των λειτουργιών συμβάλλει στην απλότητα του h/w.

Αριθμητικές Πράξεις στο MIPS

Αριθμητικές Πράξεις:

- add a, b, c # a <- b + c

- a = b + c + d + e;

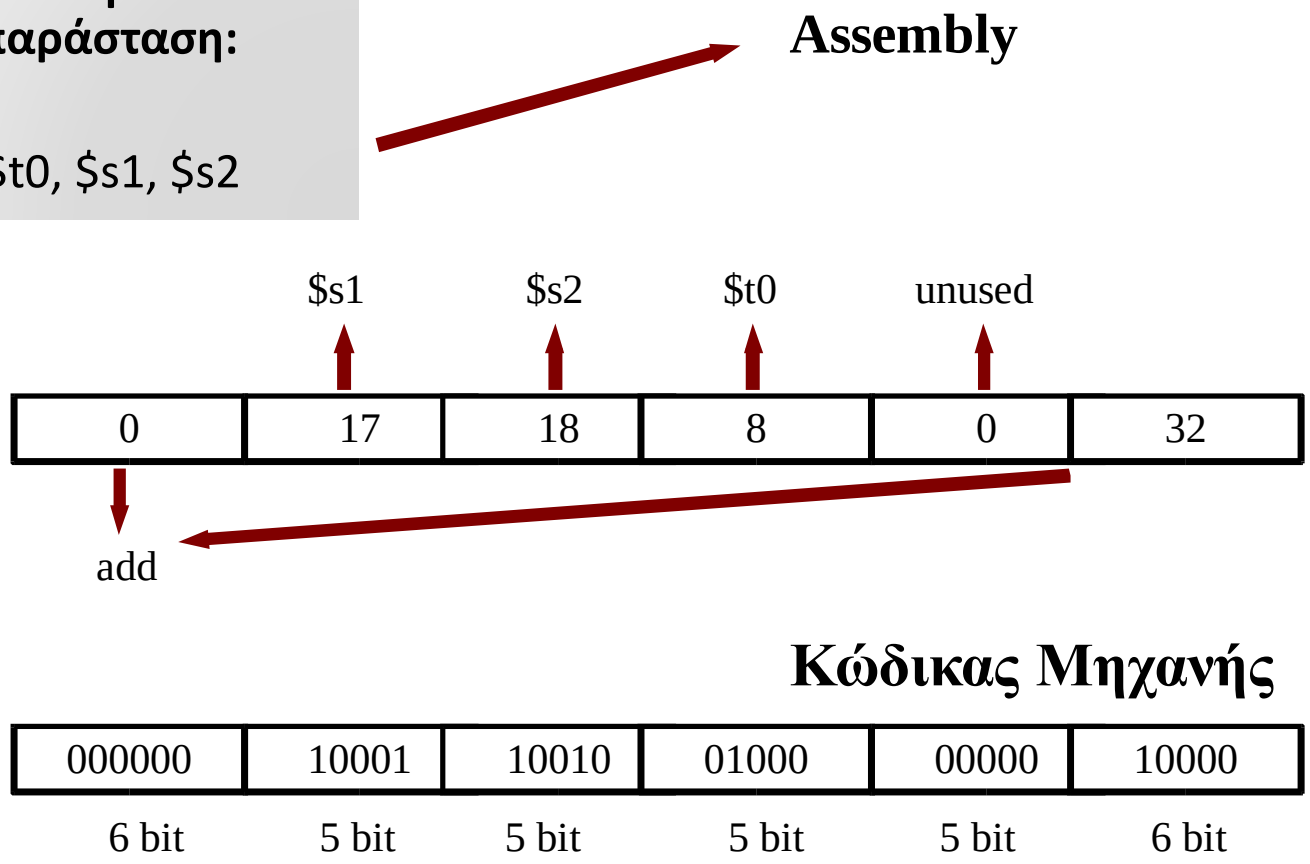
-add a, b, c
-add a, a, d
-add a, a, e

- 3 εντολές για το άθροισμα 4 μεταβλητών.

Αναπαράσταση εντολών στον Υπολογιστή (MIPS)

Συμβολική αναπαράσταση:

add $t0, $s1, $s2



Τέλος Ενότητας