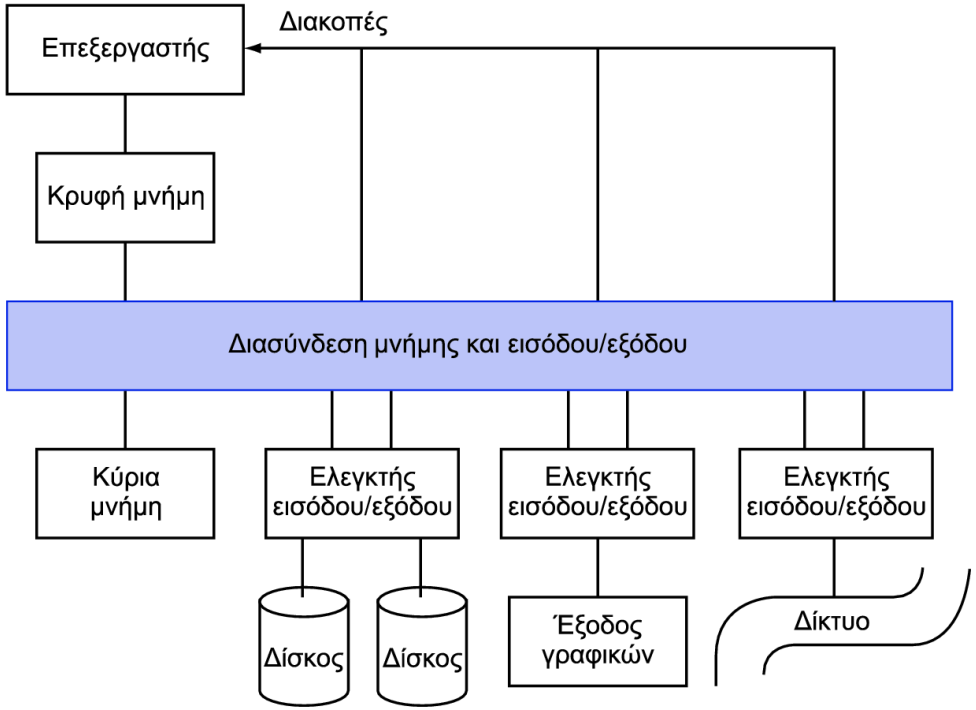
# Εισαγωγή

- Οι συσκευές εισόδου/εξόδου χαρακτηρίζονται από
  - Συμπεριφορά: είσοδος, έξοδος, αποθήκευση
  - Εταίρο: άνθρωπος ή μηχανή
  - Ρυθμό δεδομένων: byte/sec, transfers/sec
- Συνδέσεις διαύλου εισόδου/εξόδου



# Χαρακτηριστικά συστήματος Ε/Ε

- Η φερεγγυότητα (dependability) είναι σημαντική
  - Ειδικά για συσκευές αποθήκευσης
- Μέτρα απόδοσης
  - Λανθάνων χρόνος (latency) ή χρόνος απόκρισης (response time)
  - Διεκπεραιωτική ικανότητα (throughput) ή εύρος ζώνης (bandwidth)
  - Επιτραπέζια και ενσωματωμένα συστήματα
    - Ενδιαφέρονται κυρίως για το χρόνο απόκρισης και την ποικιλομορφία των συσκευών
  - Διακομιστές
    - Ενδιαφέρονται κυρίως για τη διεκπεραιωτική ικανότητα και την επεκτασιμότητα των συσκευών

# Απόδοση συσκευών Ε/Ε

Κριτήρια απόδοσης; Σύνθετα!

- **access latency** – Πόσο χρόνο χρειάζεται για να ξεκινήσει η μεταφορά δεδομένων – μετράται σε χρόνο (sec)

  «Για να παίξουμε Quake 3 θέλουμε όσο το δυνατόν μικρότερο latency»

- **throughput** – Διεκπεραιωτική ικανότητα, πόσο γρήγορα μεταφέρονται δεδομένα στη μονάδα του χρόνου – μετράται σε bytes/sec

  «Για να κατεβάσουμε μια ταινία, θέλουμε όσο το δυνατόν μεγαλύτερο throughput»

# Απόδοση συσκευών Ε/Ε

- Η απόδοση μιας συσκευής Ε/Ε εξαρτάται από:
  - χαρακτηριστικά της συσκευής
  - σύνδεση συσκευής με υπόλοιπο σύστημα
  - ιεραρχία μνήμης
  - λειτουργικό σύστημα

# Οργάνωση συσκευών Ε/Ε

- Βάσει των ακόλουθων χαρακτηριστικών:
  - Συμπεριφορά (behavior)
    - read only, write only, read/write
  - Εταίρος (partner)
    - άνθρωπος ή μηχανή τροφοδοτεί δεδομένα
  - Ρυθμός δεδομένων (data rate)
    - μέγιστος ρυθμός μεταφοράς δεδομένων
    - μεταξύ συσκευής και μνήμης ή CPU

# Οργάνωση συσκευών Ε/Ε

- Βάσει των ακόλουθων χαρακτηριστικών:
  - Συμπεριφορά (behavior)
    - read only, write only, read/write
  - Εταίρος (partner)
    - άνθρωπος ή μηχανή τροφοδοτεί δεδομένα
  - Ρυθμός δεδομένων (data rate)
    - μέγιστος ρυθμός μεταφοράς δεδομένων
      - μεταξύ συσκευής και μνήμης ή CPU
  - *Π.χ. πληκτρολόγιο – συσκευή read only,*
- *χρησιμοποιείται από άνθρωπο,με data rate 10 bytes/sec*

# Οργάνωση συσκευών Ε/Ε

| Συσκευή | Συμπεριφορά | Εταίρος | Ρυθμός μεταφοράς |
|---|---|---|---|
| Πληκτρολόγιο | Input | Άνθρωπος | 0,0001 |
| Ποντίκι | Input | Άνθρωπος | 0,0038 |
| Είσοδος Φωνής | Input | Άνθρωπος | 0,2640 |
| Είσοδος Ήχου | Input | Μηχανή | 3 |
| Scanner | Input | Άνθρωπος | 3,2 |
| Έξοδος φωνής | Output | Άνθρωπος | 0,2640 |
| Έξοδος ήχου | Output | Άνθρωπος | 8 |
| Laser printer | Output | Άνθρωπος | 3,2 |
| Οθόνη γραφικών | Output | Άνθρωπος | 800 – 8000, |
| Modem | Input/Output | Μηχανή | 0,0160– 0,064 |
| Network/LAN | Input/Output | Μηχανή | 100 – 1000 |
| Network/Wireless | Input/Output | Μηχανή | 11 – 54 |
| Optical Disk | Input/Output | Μηχανή | 80 |
| Magnetic Tape | Input/Output | Μηχανή | 32 |
| Magnetic Disk | Input/Output | Μηχανή | 240 - 2560 |

**(Mbit/sec)**

# Τρόπος Αξιολόγησης Απόδοσης Ε/Ε

- Εξαρτάται από την εφαρμογή
- Σε μερικά περιβάλλοντα μας νοιάζει το throughput
- Τότε το bandwidth είναι το πιο σημαντικό.

- <u>Μετράται με 2 τρόπους:</u>
  - Data που διακινούμε τη μονάδα του χρόνου

    ή
  - Λειτουργίες Ε/Ε που πραγματοποιούμε τη μονάδα του χρόνου

# Τρόπος Αξιολόγησης Απόδοσης Ε/Ε

- Σε άλλες εφαρμογές μας νοιάζει ο χρόνος απόκρισης (response time)

- Εξαρτάται από bandwidth και access latency

**Για μεγάλες αιτήσεις Ε/Ε**

**Για μικρές αιτήσεις Ε/Ε**

# Τρόπος Αξιολόγησης Απόδοσης Ε/Ε

Πλήθος εφαρμογών απαιτούν
**υψηλή διεκπεραιωτική ικανότητα**
και
**μικρό χρόνο απόκρισης**

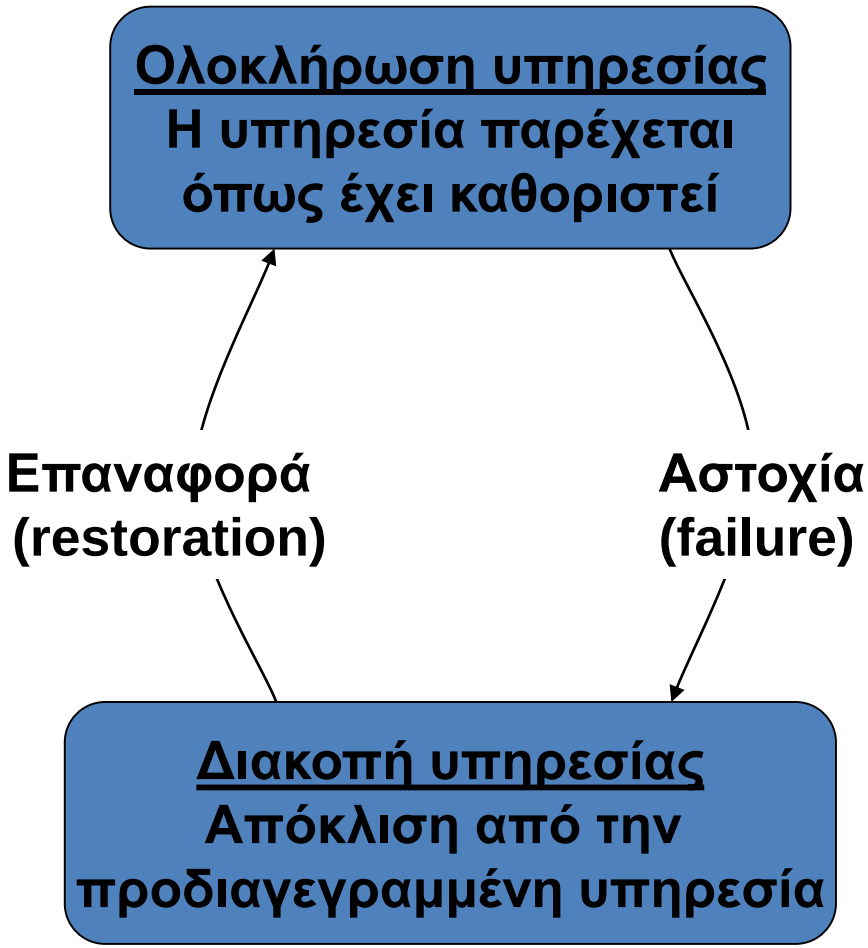
- ATM
- File servers
- Web servers
- κ.α.

*«Μας ενδιαφέρει και η διάρκεια κάθε εργασίας αλλά και πόσες διεργασίες εκτελούνται το δευτερόλεπτο»*

# Σύνοψη Απόδοσης Ε/Ε

- Desktop, Servers, Embedded Systems
  - φερεγγυότητα + κόστος Ε/Ε
- Desktop , Embedded Systems
  - χρόνος απόκρισης + ποικιλία συσκευών Ε/Ε
- Servers
  - διεκπεραιωτική ικανότητα συσκευών Ε/Ε

# Φερεγγυότητα (dependability)




- Ελάττωμα (fault): αστοχία ενός συστατικού
  - Μπορεί να οδηγήσει ή να μην οδηγήσει σε αστοχία του συστήματος

# Μέτρα φερεγγυότητας

- Αξιοπιστία (reliability): μέσος χρόνος πρώτης αστοχίας (mean time to failure – MTTF)
- Διακοπή υπηρεσίας (service interruption): μέσος χρόνος επιδιόρθωσης (mean time to repair – MTTR)
- Μέσος χρόνος μεταξύ αστοχιών (mean time between failures - MTBF)
  - MTBF = MTTF + MTTR
- Διαθεσιμότητα (availability) = MTTF / (MTTF + MTTR)
- Βελτίωση διαθεσιμότητας
  - Αύξηση MTTF: αποφυγή ελαττώματος (fault avoidance), ανοχή ελαττωμάτων (fault tolerance), πρόβλεψη ελαττωμάτων (fault forecasting)
  - Μείωση MTTR: βελτιωμένα εργαλεία και διαδικασίες διάγνωσης και επιδιόρθωσης

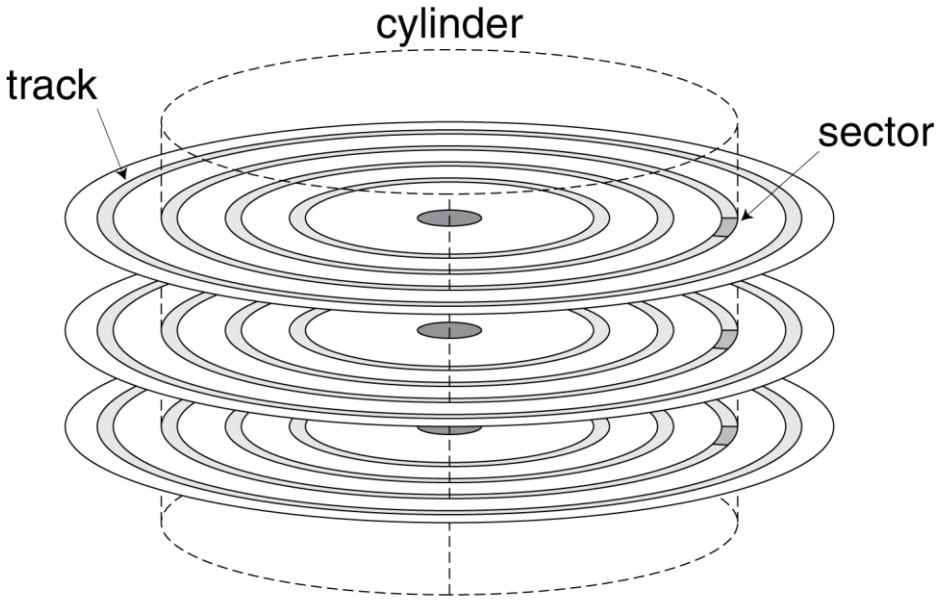
# Αποθήκευση στο δίσκο

Ο Δίσκος αποτελείται από:

- Στοίβα πλακών (1 – 4) με 2 επιφάνειες εγγραφής
- Στοίβα πλακών περιστρέφεται (5400 – 15000 RPM)
- Κάθε επιφάνεια διαιρείται σε ομόκεντρους κύκλους – τροχιές/tracks (10K – 50K tracks/επιφάνεια)
- Κάθε τροχιά διαιρείται σε τομείς/sectors (100 – 500 sectors/track)
- Τυπικό μέγεθος τομέα 512bytes (με τάση προς τα 4KB)

# Αποθήκευση στο δίσκο

- Μη πτητική (nonvolatile), περιστρεφόμενη μαγνητική αποθήκευση



# Τομείς δίσκου και προσπέλαση

- Κάθε τομέας (sector) καταγράφει
  - Την ταυτότητα τομέα (sector ID)
  - Δεδομένα (512 byte, 4096 byte προτεινόμενη τιμή)
  - Κώδικα διόρθωσης σφαλμάτων (error correcting code – ECC)
    - Για απόκρυψη ατελειών και καταγραφή σφαλμάτων
  - Πεδία συγχρονισμού και κενά (gaps)
- Η προσπέλαση ενός τομέα περιλαμβάνει
  - Καθυστέρηση αναμονής σε ουρά αν εκκρεμούν άλλες προσπελάσεις
  - Αναζήτηση (seek): μετακίνηση των κεφαλών
  - Λανθάνων χρόνος περιστροφής (rotational latency)
  - Μεταφορά δεδομένων (transfer time)
  - Επιβάρυνση ελεγκτή (controller)

# Παράδειγμα προσπέλασης δίσκου

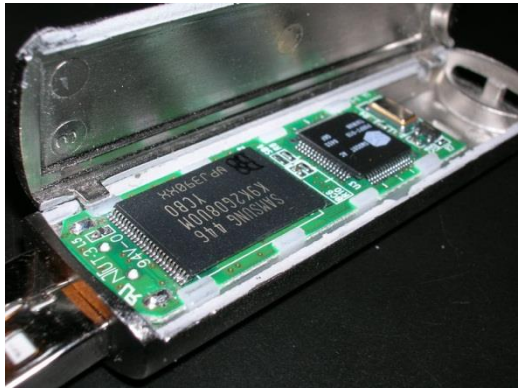
- Δεδομένα
  - Τομέας των 512B, 15000rpm (περιστροφές ανά λεπτό), μέσος χρόνος αναζήτησης 4ms, ρυθμός μεταφοράς 100MB/s, επιβάρυνση ελεγκτή 0.2ms, δίσκος αδρανής
- Μέσος χρόνος ανάγνωσης
  - 4ms χρόνος αναζήτησης
    + ½ / (15,000/60) = 2ms λανθάνων χρόνος περιστροφής
    + 512 / 100MB/s = 0.005ms χρόνος μεταφοράς
    + 0.2ms καθυστέρηση ελεγκτή
    = 6.2ms
- Αν ο πραγματικός μέσος χρόνος αναζήτησης είναι is 1ms
  - Μέσος χρόνος ανάγνωσης = 3.2ms

# Ζητήματα απόδοσης δίσκου

- Οι κατασκευαστές αναφέρουν το μέσο χρόνο αναζήτησης
  - Με βάση όλες τις πιθανές αναζητήσεις
  - Η τοπικότητα και ο χρονοπρογραμματισμός του ΛΣ οδηγούν σε μικρότερους μέσους χρόνους αναζήτησης
- «Έξυπνος» ελεγκτής δίσκου κατανέμει τους φυσικούς τομείς του δίσκου
  - Εμφανίζει τη διασύνδεση των λογικών τομέων (logical sector interface) στον υπολογιστή
  - SCSI, ATA, SATA ελεγκτές
- Οι μονάδες δίσκου περιλαμβάνουν και κρυφές μνήμες
  - Εκ των προτέρων προσκόμιση (prefetch) τομέων με αναμονή προσπέλασής τους σύντομα
  - Αποφυγή αναζήτησης και καθυστέρησης περιστροφής

# Αποθήκευση σε μνήμη φλας

- Μη πτητική ημιαγωγική αποθήκευση
  - 100× – 1000× ταχύτερη από το δίσκο
  - Μικρότερη, χαμηλότερη κατανάλωση, πιο εύρωστη
  - Αλλά κοστίζει περισσότερα $/GB (ανάμεσα στο δίσκο και την DRAM)

# Τύποι μνήμης φλας

- NOR flash: κελί bit μοιάζει με πύλη NOR
  - Τυχαία προσπέλαση ανάγνωσης/εγγραφής
  - Χρησιμοποιείται για μνήμη εντολών σε ενσωματωμένα συστήματα
- NAND flash: κελί bit μοιάζει με πύλη NAND
  - Πιο πυκνή (bit/επιφάνεια), αλλά προσπέλαση ενός ολόκληρου μπλοκ τη φορά
  - Φθηνότερη ανά GB
  - Χρήση σε USB keys, αποθήκευση μέσων (ήχος, εικόνα), ...
- Τα bit της μνήμης φλας φθείρονται μετά από χιλιάδες προσπελάσεις
  - Δεν είναι κατάλληλη για να αντικαταστήσει τη RAM ή το δίσκο
  - Εξισορρόπηση φθοράς (wear leveling): επαναχαρτογράφηση δεδομένων σε λιγότερο χρησιμοποιημένα μπλοκ της

# Συστατικά διασύνδεσης

- Ανάγκη διασύνδεσης μεταξύ
  - CPU, μνήμης, ελεγκτών Ε/Ε
- Δίαυλος: κοινόχρηστο κανάλι επικοινωνίας
  - Παράλληλο σύνολο αγωγών για δεδομένα και συγχρονισμό της μεταφοράς τους
  - Μπορεί να αποτελέσει σημείο συμφόρησης
- Η απόδοση περιορίζεται από φυσικούς παράγοντες
  - Μήκος αγωγού, αριθμός συνδέσεων
- Πιο πρόσφατη εναλλακτική: σειριακές συνδέσεις υψηλής ταχύτητας με μεταγωγούς
  - Όπως στα δίκτυα

# Τύποι διαύλου

- Δίαυλοι επεξεργαστή-μνήμης
  - Κοντοί, μεγάλη ταχύτητα
  - Η σχεδίαση ταιριάζει με την οργάνωση μνήμης
- Δίαυλοι εισόδου/εξόδου
  - Μακρύτεροι, επιτρέπουν πολλές συνδέσεις
  - Προδιαγράφονται με πρότυπα για λόγους διαλειτουργικότητας (interoperability)
  - Σύνδεση με το δίαυλο επεξεργαστή-μνήμης μέσω μιας γέφυρας (bridge)

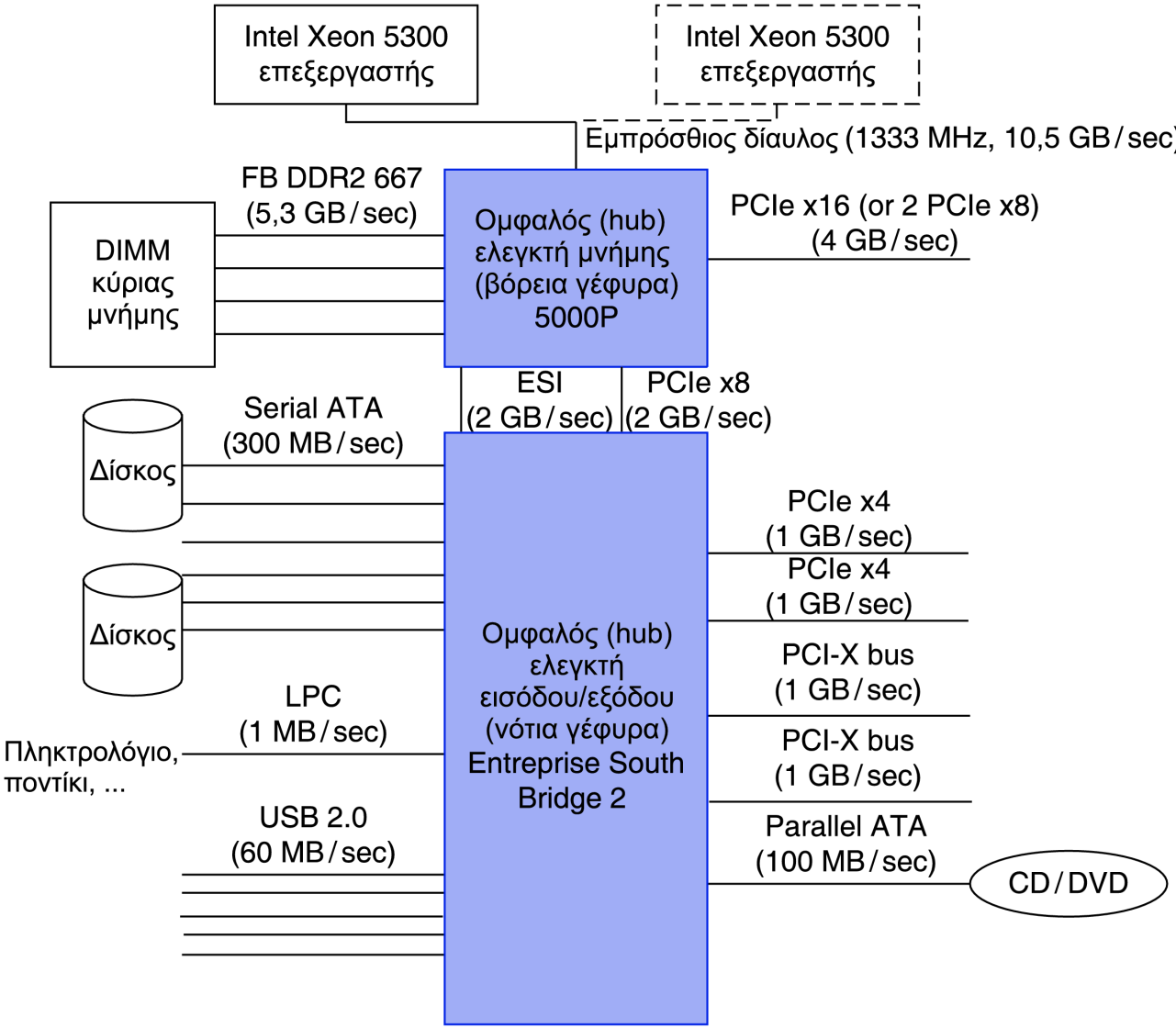
# Σήματα διαύλου και συγχρονισμός

- Γραμμές δεδομένων
  - Μεταφέρουν διεύθυνση και δεδομένα
  - Με πολύπλεξη ή ξεχωριστά
- Γραμμές ελέγχου
  - Δείχνουν τον τύπο δεδομένων, συγχρονίζουν τις συναλλαγές (transactions)
- Σύγχρονη
  - Χρησιμοποιεί ρολόι διαύλου
- Ασύγχρονη
  - Χρησιμοποιεί γραμμές ελέγχου αίτησης/επιβεβαίωσης (request/acknowledge) για χειραψία (handshaking)

# Παραδείγματα διαύλου Ε/Ε

| | Firewire | USB 2.0 | PCI Express | Serial ATA | Serial Attached SCSI |
|---|---|---|---|---|---|
| **Πρόθεση χρήσης** | Εξωτερική | Εξωτερική | Εσωτερική | Εσωτερική | Εξωτερική |
| **Συσκευές ανά κανάλι** | 63 | 127 | 1 | 1 | 4 |
| **Εύρος δεδομένων** | 4 | 2 | 2/lane | 4 | 4 |
| **Μέγιστο εύρος ζώνης** | 50MB/s ή 100MB/s | 0.2MB/s, 1.5MB/s, ή 60MB/s | 250MB/s/lane 1×, 2×, 4×, 8×, 16×, 32× | 300MB/s | 300MB/s |
| **Σύνδεση «εν θερμώ»** | Ναι | Ναι | Εξαρτάται | Ναι | Ναι |
| **Μέγιστο μήκος** | 4.5m | 5m | 0.5m | 1m | 8m |
| **Πρότυπο** | IEEE 1394 | USB Implementers Forum | PCI-SIG | SATA-IO | INCITS TC T10 |

# Τυπικό σύστημα Ε/Ε x86 PC



# Διαχείριση εισόδου/εξόδου

- Το ΛΣ είναι ο ενδιάμεσος για την Ε/Ε
  - Πολλά προγράμματα μοιράζονται πόρους εισόδου/εξόδου
    - Χρειάζεται προστασία και χρονοπρογραμματισμός
  - Η Ε/Ε προκαλεί ασύγχρονες διακοπές
    - Ίδιος μηχανισμός με τις εξαιρέσεις
  - Ο προγραμματισμός Ε/Ε είναι δύσκολος
    - Το ΛΣ παρέχει αφαιρέσεις στα προγράμματα

# Διαταγές εισόδου/εξόδου

- Τις συσκευές Ε/Ε διαχειρίζεται το υλικό των ελεγκτών Ε/Ε
  - Μεταφέρουν δεδομένα από/προς τη συσκευή
  - Συγχρονίζουν τις λειτουργίες με λογισμικό
- Καταχωρητές διαταγών (command registers)
  - Αναγκάζουν τη συσκευή να κάνει κάτι
- Καταχωρητές κατάστασης (status registers)
  - Δείχνουν τι κάνει η συσκευή και την εμφάνιση σφαλμάτων
- Καταχωρητές δεδομένων (data registers)
  - Εγγραφής: μεταφέρουν δεδομένα σε μια συσκευή
  - Ανάγνωσης: μεταφέρουν δεδομένα από μια συσκευή

# Χαρτογράφηση καταχωρητών Ε/Ε

- Ε/Ε με χαρτογράφηση μνήμης (memory mapped I/O)
  - Οι καταχωρητές προσπελάζονται στον ίδιο χώρο δ/νσεων με τη μνήμη
  - Ο αποκωδικοποιητής δ/νσεων κάνει το διαχωρισμό
  - Το ΛΣ χρησιμοποιεί μηχανισμό μετάφρασης δ/νσεων ώστε να τους κάνει προσπελάσιμους μόνο στον πυρήνα (kernel) του ΛΣ
- Εντολές Ε/Ε
  - Ξεχωριστές εντολές για προσπέλαση καταχωρητών Ε/Ε
  - Μπορούν να εκτελεστούν μόνο σε κατάσταση πυρήνα
  - Παράδειγμα: x86

# Περιόδευση (polling)

- Περιοδικός έλεγχος του καταχωρητή κατάστασης (status register) της Ε/Ε
  - Αν η συσκευή είναι έτοιμη, καμία λειτουργία
  - Αν υπάρχει σφάλμα, ανάληψη δράσης
- Συνήθης σε μικρά ή χαμηλών επιδόσεων ενσωματωμένα συστήματα πραγματικού χρόνου
  - Προβλέψιμος χρονισμός
  - Χαμηλό κόστος υλικού
- Σε άλλα συστήματα, σπατάλη χρόνου CPU

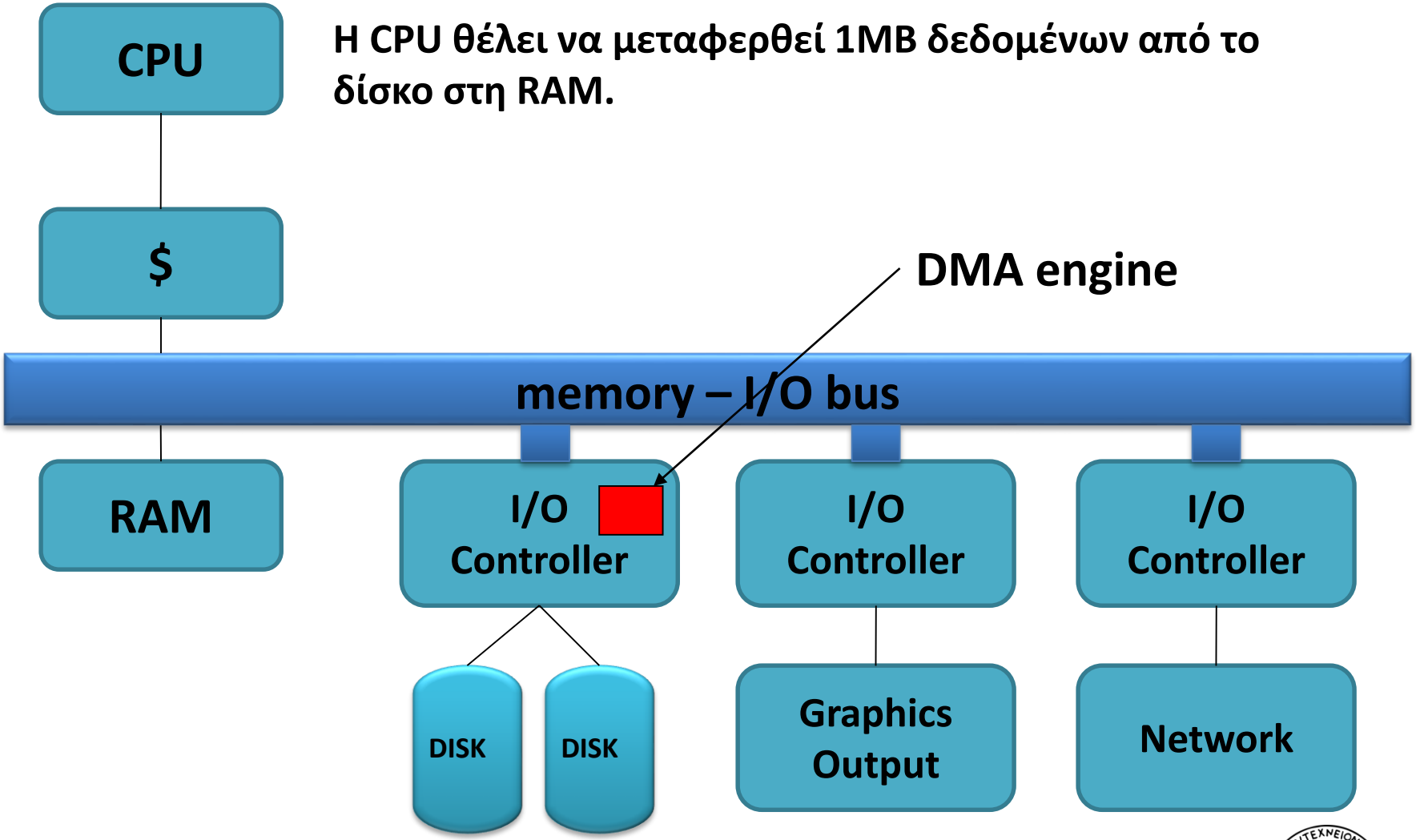
# Διακοπές (interrupts)

- Όταν μια συσκευή είναι έτοιμη ή όταν συμβεί σφάλμα
  - Ο ελεγκτής διακόπτει τη CPU
- Η διακοπή είναι σαν εξαίρεση (exception)
  - Αλλά δε συγχρονίζεται με την εκτέλεση των εντολών
  - Μπορεί να καλέσει το χειριστή (handler) μεταξύ εντολών
  - Πληροφορία αιτίου (cause) προσδιορίζει συχνά τη συσκευή που προκαλεί διακοπή
- Διακοπές με προτεραιότητες
  - Οι συσκευές που χρειάζονται πιο επείγουσα προσοχή λαμβάνουν υψηλότερη προτεραιότητα
  - Μπορούν να διακόψουν το χειριστή μιας διακοπής χαμηλότερης προτεραιότητας

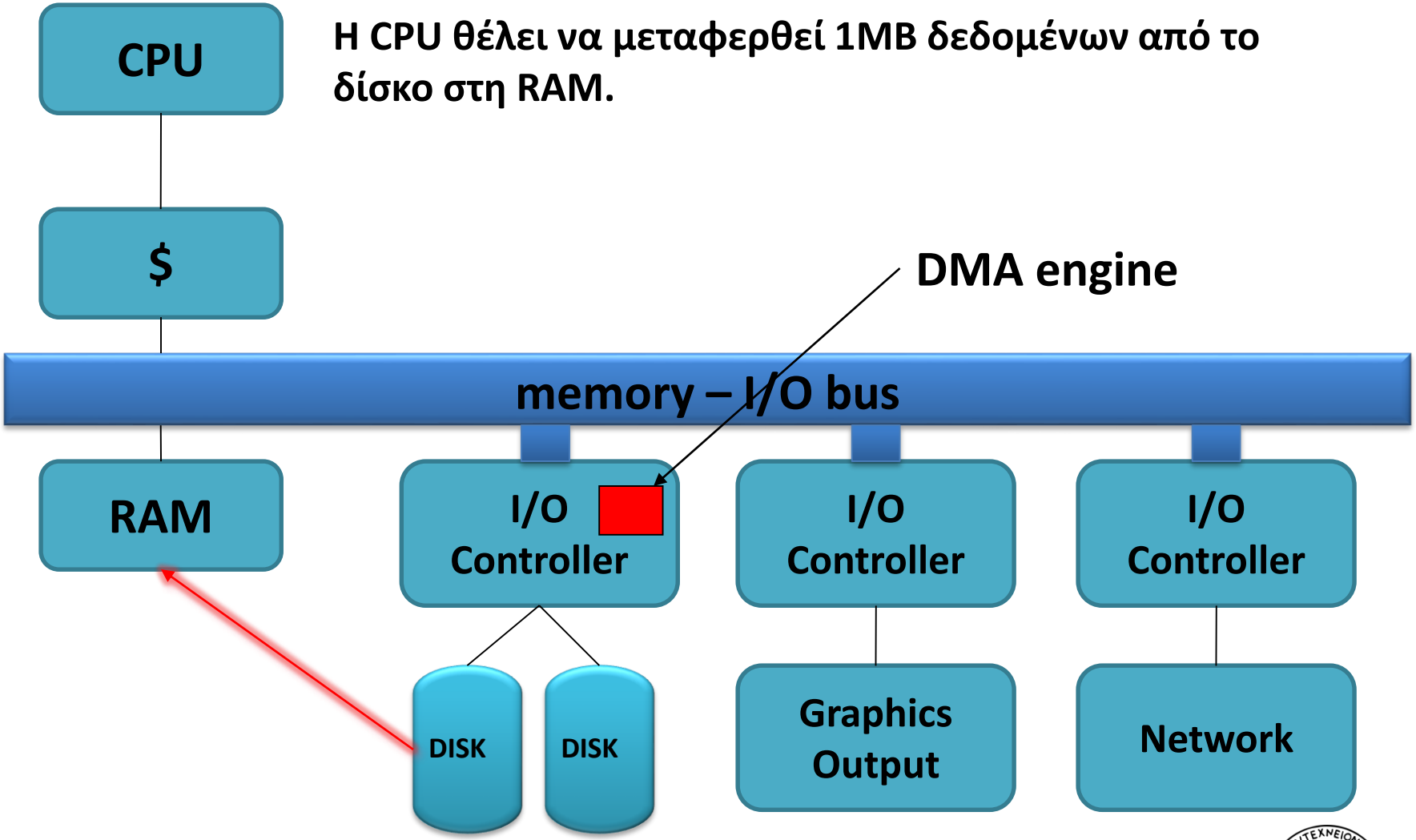
# Μεταφορά δεδομένων Ε/Ε

- Περιόδευση και Ε/Ε οδηγούμενη από διακοπές
  - Η CPU μεταφέρει δεδομένα μεταξύ μνήμης και καταχωρητών δεδομένων Ε/Ε
  - Χρονοβόρα διαδικασία για συσκευές υψηλής ταχύτητας
- Άμεση προσπέλαση μνήμης (direct memory access – DMA)
  - Το ΛΣ παρέχει την αρχική δ/νση μνήμης
  - Ο ελεγκτής Ε/Ε κάνει μεταφορά προς/από τη μνήμη αυτόνομα
  - Ο ελεγκτής προκαλεί διακοπή όταν ολοκληρώσει τη μεταφορά ή σε περίπτωση σφάλματος

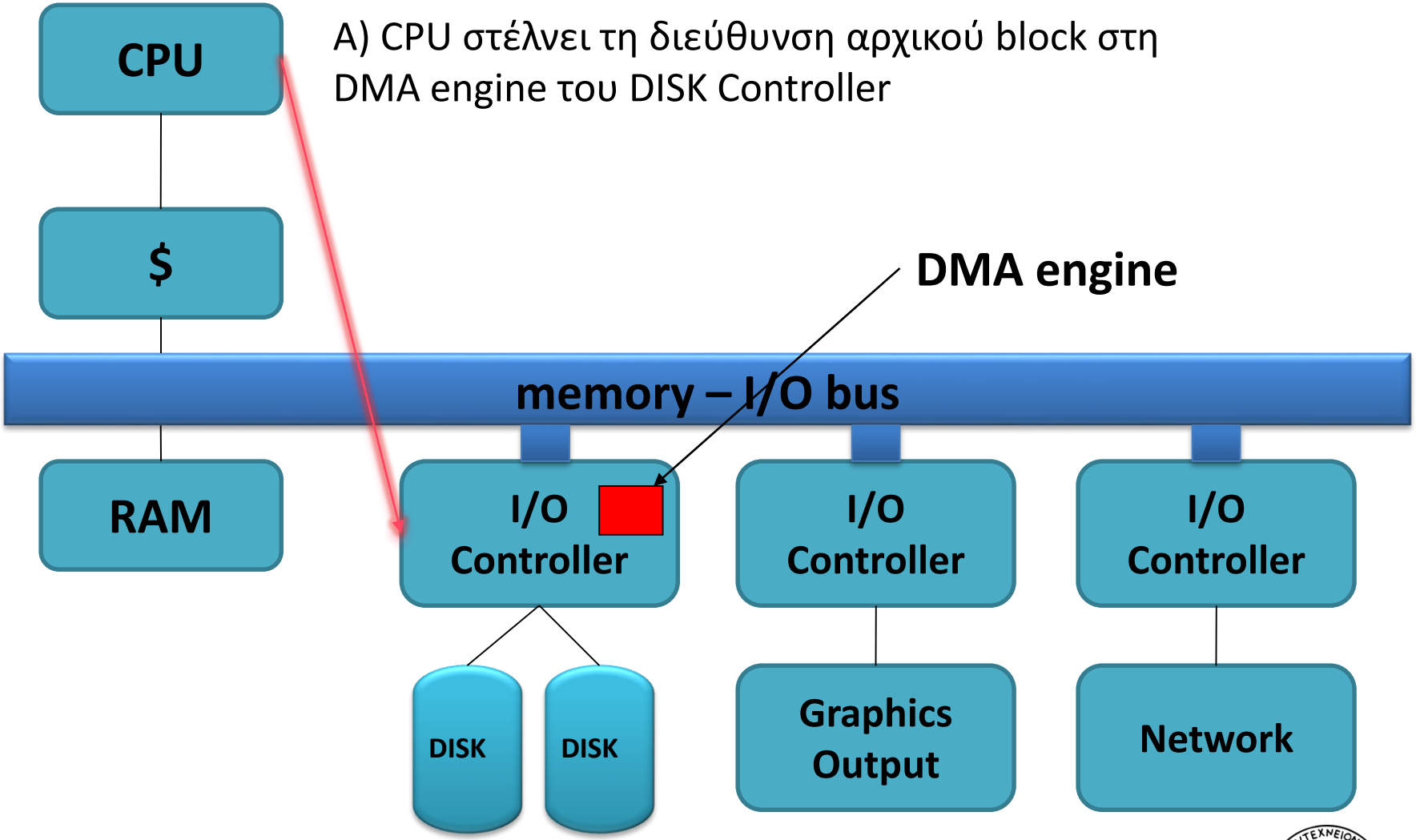
# Direct Memory Access



# Direct Memory Access

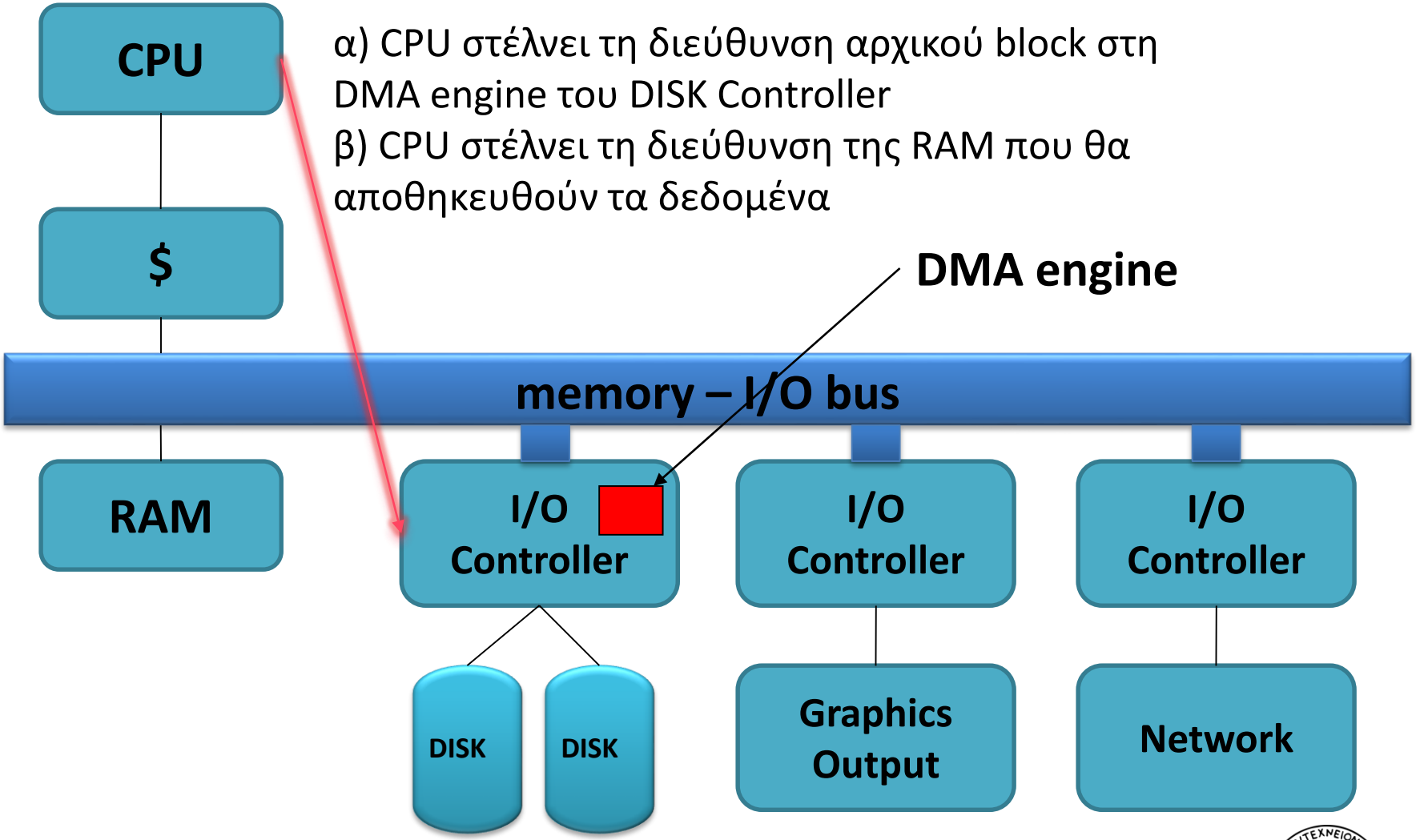


# Direct Memory Access



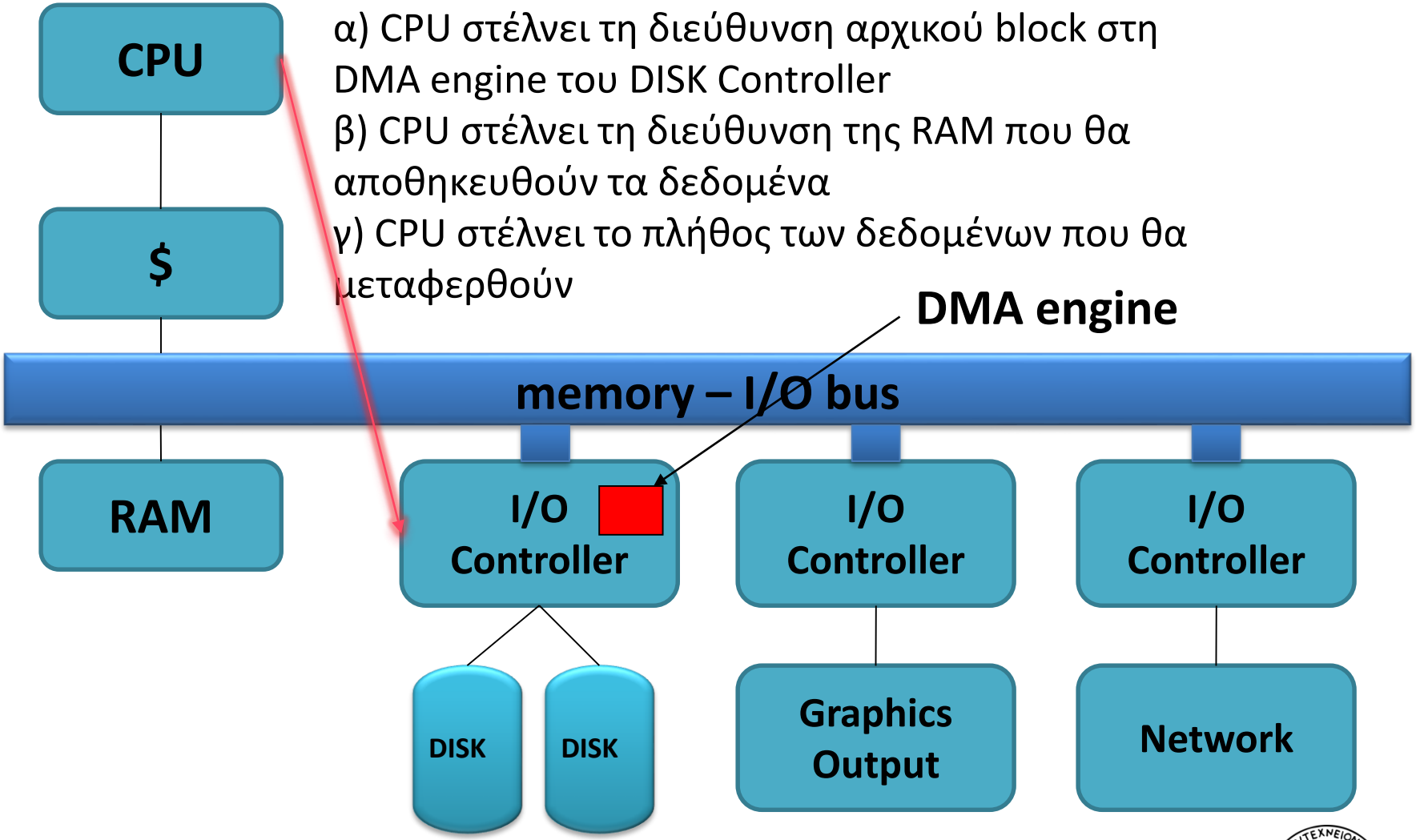
# Direct Memory Access

α) CPU στέλνει τη διεύθυνση αρχικού block στη DMA engine του DISK Controller
β) CPU στέλνει τη διεύθυνση της RAM που θα αποθηκευθούν τα δεδομένα

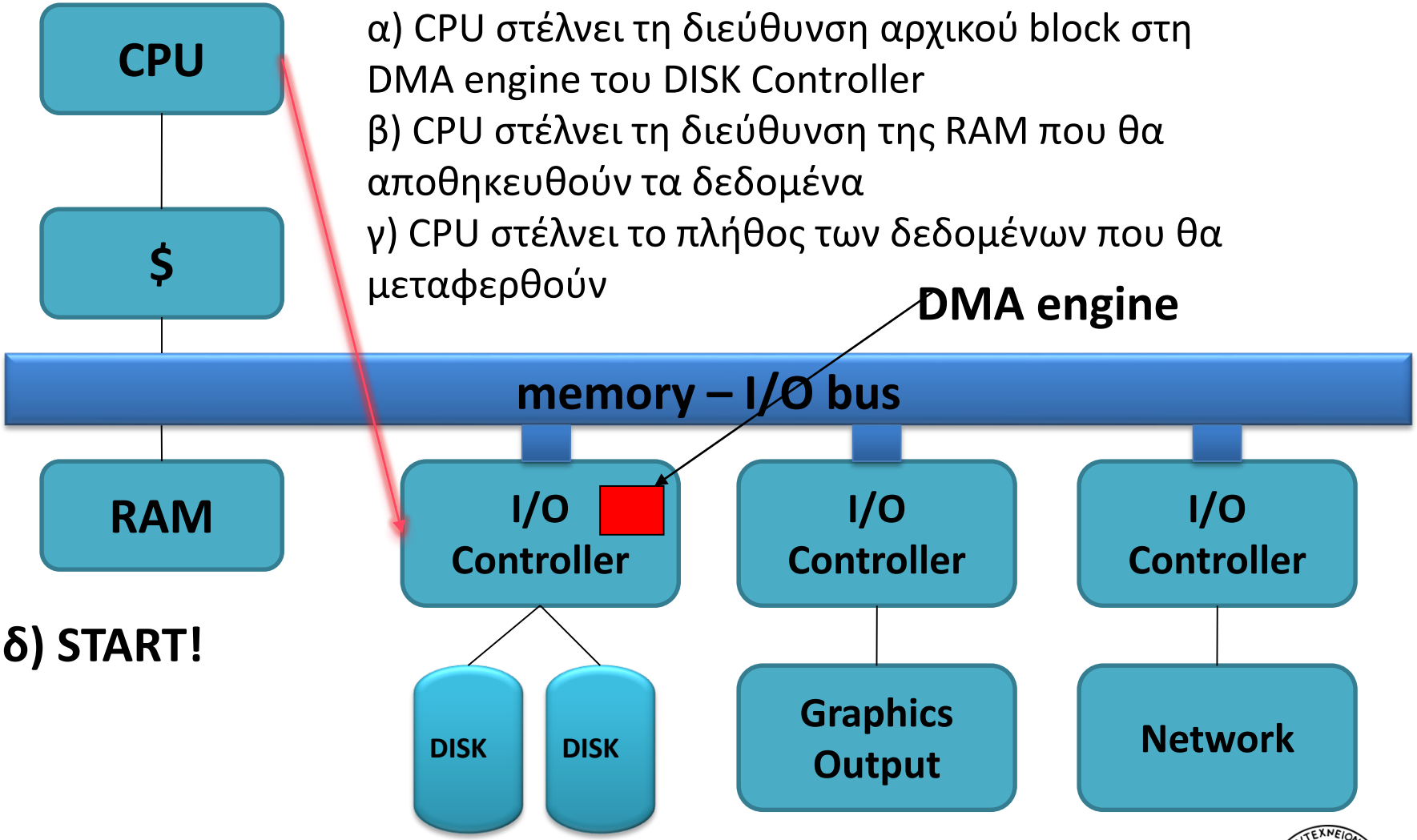


# Direct Memory Access

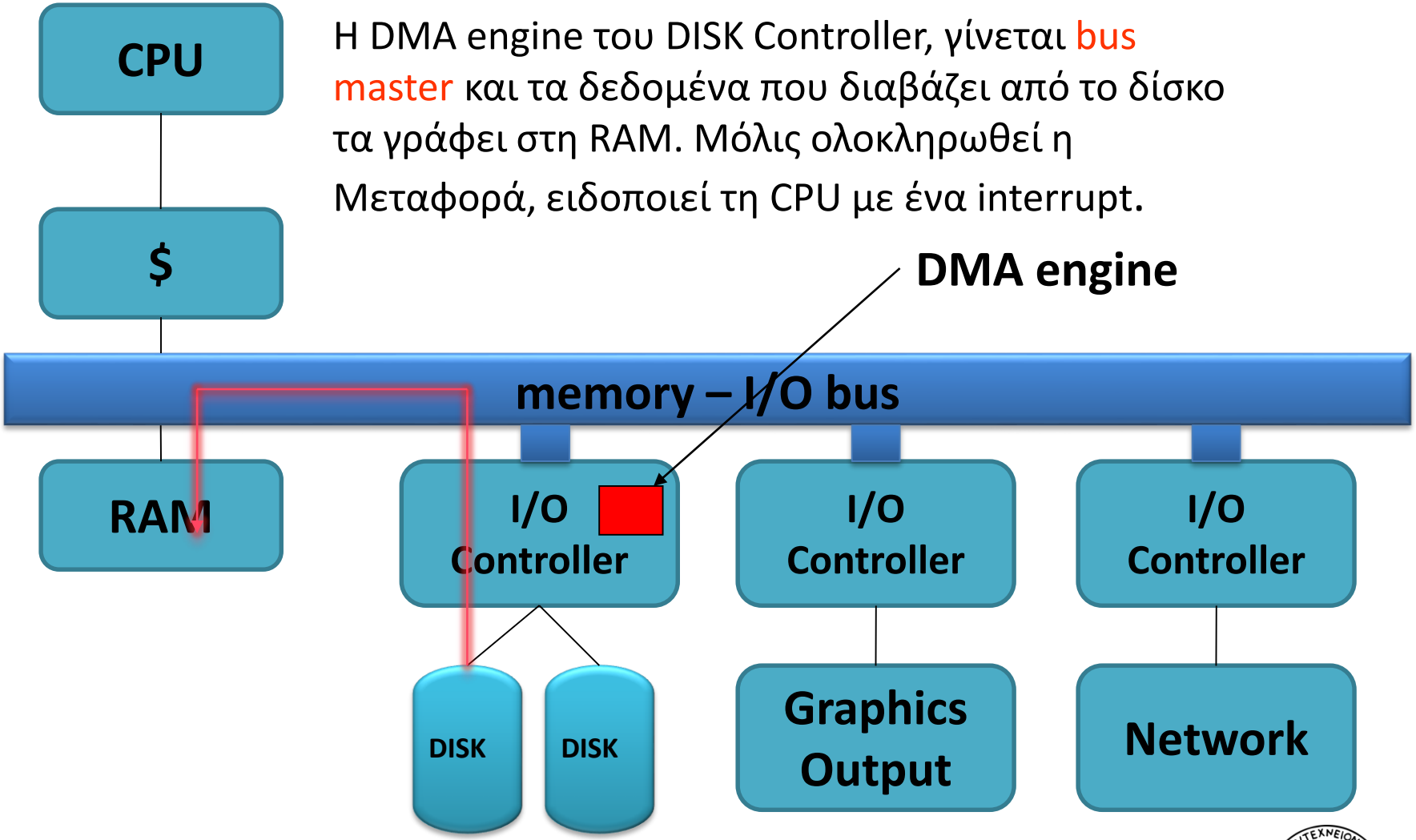
α) CPU στέλνει τη διεύθυνση αρχικού block στη DMA engine του DISK Controller
β) CPU στέλνει τη διεύθυνση της RAM που θα αποθηκευθούν τα δεδομένα
γ) CPU στέλνει το πλήθος των δεδομένων που θα μεταφερθούν



# Direct Memory Access



# Direct Memory Access




Η DMA engine του DISK Controller, γίνεται bus master και τα δεδομένα που διαβάζει από το δίσκο τα γράφει στη RAM. Μόλις ολοκληρωθεί η Μεταφορά, ειδοποιεί τη CPU με ένα interrupt.

# Αλληλεπίδραση DMA/Cache

- Αν το DMA γράφει σε ένα μπλοκ μνήμης που βρίσκεται στην κρυφή μνήμη
  - Το αντίγραφο της κρυφής μνήμης γίνεται «παλιό»
- Αν η κρυφή μνήμη είναι ετερόχρονης εγγραφής και το μπλοκ είναι «ακάθαρτο», και το DMA διαβάζει το μπλοκ της μνήμης
  - Διαβάζει τα «παλιά» δεδομένα
- Πρέπει να εγγυηθούμε τη συνοχή (coherence) της κρυφής μνήμης
  - «Εκκένωση» (flush) των μπλοκ από τη κρυφή μνήμη αν πρόκειται να χρησιμοποιηθούν σε DMA
  - Ή χρήση θέσεων μνήμης που δεν αποθηκεύονται στη κρυφή μνήμη (non-cacheable) για τις λειτουργίες Ε/Ε

# Αλληλεπίδραση DMA/εικονικής μνήμης

- Το ΛΣ χρησιμοποιεί εικονικές δ/νσεις για τη μνήμη
  - Τα μπλοκ του DMA μπορεί να μην είναι συνεχόμενα στη φυσική μνήμη
- Πρέπει το DMA να χρησιμοποιεί εικονικές δ/νσεις;
  - Θα απαιτούσε να κάνει τη μετάφραση ο ελεγκτής
- Αν το DMA χρησιμοποιεί φυσικές δ/νσεις
  - Μπορεί να χρειάζεται να «σπάσει» τις μεταφορές σε τμήματα μεγέθους σελίδας
  - Ή να βάζει στη σειρά πολλές μεταφορές
  - Ή να κατανέμει συνεχόμενες φυσικές σελίδες για DMA

# Μέτρηση απόδοσης Ε/Ε

- Η απόδοση Ε/Ε εξαρτάται από
  - Υλικό: CPU, μνήμη, ελεγκτές, δίαυλοι
  - Λογισμικό: λειτουργικό σύστημα, σύστημα διαχείρισης βάσης δεδομένων, εφαρμογή
  - Φορτίο εργασίας: ρυθμοί και μοτίβα αιτήσεων
- Η σχεδίαση του συστήματος Ε/Ε μπορεί να κάνει συμβιβασμούς μεταξύ χρόνου απόκρισης και ρυθμού διεκπεραίωσης
  - Οι μετρήσεις ρυθμού διεκπεραίωσης γίνονται συχνά με περιορισμένο χρόνο απόκρισης

# Μετροπρογράμματα επεξεργασίας συναλλαγών

- Συναλλαγές (Transactions)
  - Μικρές προσπελάσεις δεδομένων σε ένα σύστημα διαχείρισης βάσης δεδομένων (DBMS)
  - Το ενδιαφέρον είναι στο ρυθμό Ε/Ε, όχι το ρυθμό δεδομένων
- Μέτρηση ρυθμού διεκπεραίωσης (throughput)
  - Υπόκειται σε περιορισμούς χρόνου απόκρισης και χειρισμό αστοχιών
  - ACID (Atomicity/Ατομικότητα, Consistency/Συνέπεια, Isolation/Απομόνωση, Durability/Αντοχή)
  - Συνολικό κόστος ανά συναλλαγή
- Μετροπρογράμματα του Transaction Processing Council (TPC, www.tcp.org)
  - TPC-APP: διακομιστής εφαρμογών και υπηρεσιών ιστού
  - TCP-C: περιβάλλον καταχώρισης παραγγελιών
  - TCP-E: επεξεργασία συναλλαγών μεσιτικού γραφείου
  - TPC-H: υποστήριξη αποφάσεων — κατά περίπτωση (ad-hoc) ερωτήματα με προσανατολισμό επιχειρήσεις

# Μετροπρογράμματα συστήματος αρχείων και Ιστού

- SPEC System File System (SFS)
  - Συνθετικό φορτίο εργασίας για διακομιστή NFS, με βάση παρακολούθηση πραγματικών συστημάτων
  - Αποτελέσματα
    - Ρυθμός διεκπεραίωσης, throughput (λειτουργίες/sec)
    - Χρόνος απόκρισης (μέσο ms/λειτουργία)
- SPEC Web Server benchmark
  - Μετράει τις ταυτόχρονες συνεδρίες (sessions) χρηστών, με βάση τον απαιτούμενο ρυθμό διεκπεραίωσης ανά συνεδρία
  - Τρία φορτία εργασίας: Τραπεζική, Ηλεκτρονικό εμπόριο, και Υποστήριξη

# Ε/Ε έναντι απόδοσης CPU

- Νόμος του Amdahl
  - Μην αγνοείς την απόδοση της Ε/Ε καθώς η παραλληλία αυξάνει την απόδοση των υπολογισμών
- Παράδειγμα
  - Το μετροπρόγραμμα διαρκεί 90s χρόνο CPU, 10s χρόνο Ε/Ε
  - Διπλάσιες CPU κάθε 2 χρόνια
    - Ε/Ε αμετάβλητη

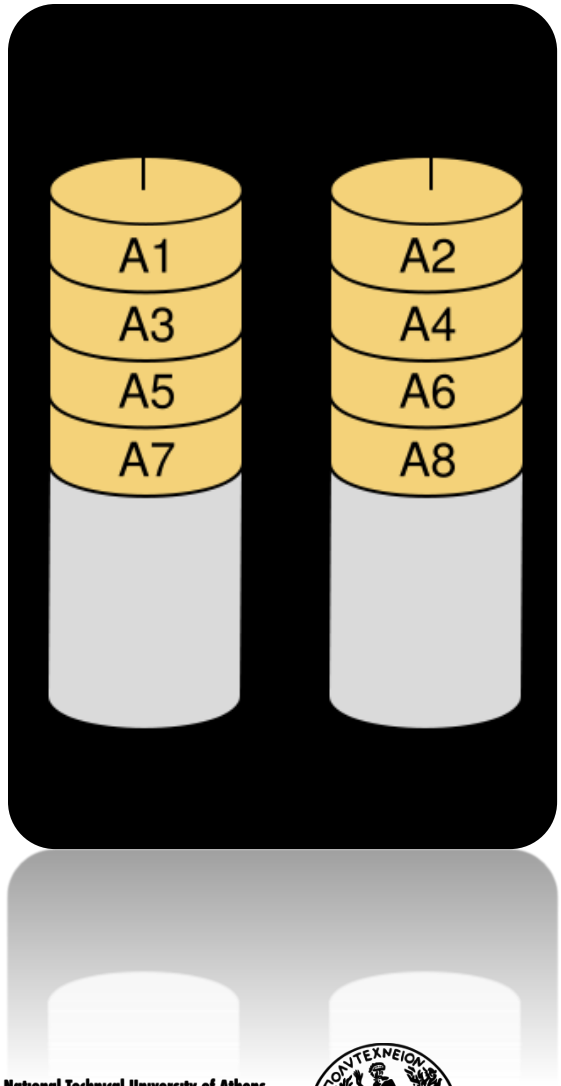
| Έτος | Χρόνος CPU | Χρόνος Ε/Ε | Παρελθών χρόνος | % Χρόνος Ε/Ε |
|---|---|---|---|---|
| Τώρα | 90s | 10s | 100s | 10% |
| +2 | 45s | 10s | 55s | 18% |
| +4 | 23s | 10s | 33s | 31% |
| +6 | 11s | 10s | 21s | 47% |

# Πλεονασματικές Συστοιχίες Φθηνών Δίσκων - RAID

- Redundant Array of Inexpensive Disks
  - Χρήση συστοιχίας “μικρών” και “φτηνών” δίσκων
    - Χρήση πολλών μικρότερων δίσκων (αντί ένα μεγάλο)
    - Αύξηση απόδοσης - Πολλαπλά read “heads”
  - Αξιοπιστία;
    - Χρειάζεται πλεονασμός (redundancy) γιατί οι “μικροί” & “φθηνοί” δίσκοι δεν είναι αξιόπιστοι
    - Δημιουργία συστήματος αποθήκευσης με ανοχή σε ελαττώματα (fault tolerant)
    - Ειδικά αν οι δίσκοι που αστοχούν δεν μπορούν να αντικατασταθούν «εν θερμώ»

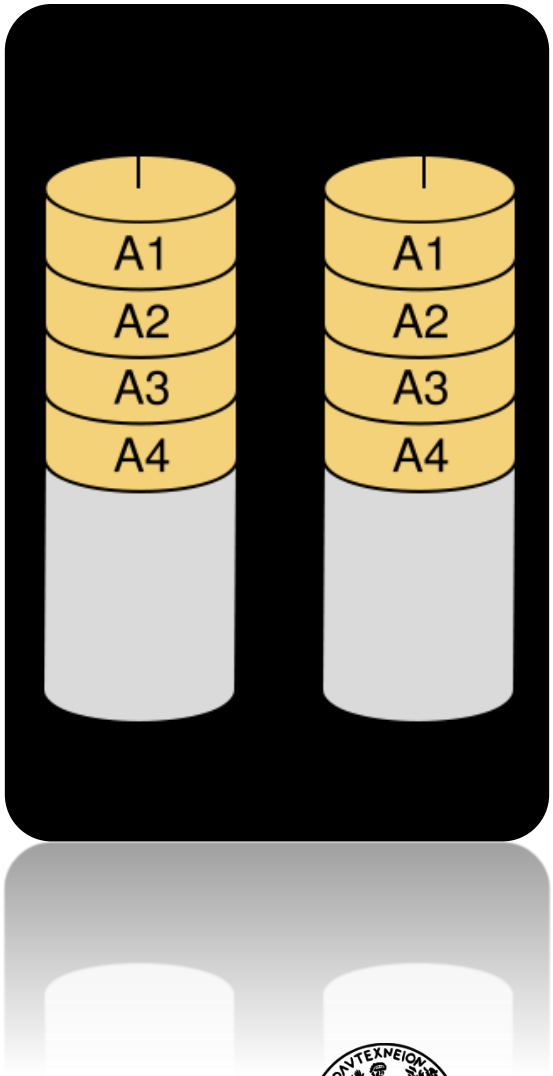
# RAID 0 - Striping

- Συνεχόμενα blocks γράφονται στους 2 δίσκους εναλλάξ.
- Παρέχει υψηλή απόδοση αφού οι εγγραφές και οι αναγνώσεις γίνονται συνεχώς σε διαφορετικούς δίσκους «διπλασιάζοντας» τις επιδόσεις.
- Όμως δεν παρέχει αξιοπιστία, αφού η βλάβη ενός δίσκους καταστρέφει ολόκληρη τη συστοιχία.
- Ο όρος RAID στην περίπτωση αυτή χρησιμοποιείται *καταχρηστικά* (AID)



# RAID 1 - Mirroring

- Χρήση N + N δίσκων ειδώλων
- Κάθε block γράφεται και στους N δίσκους (αντίγραφο/είδωλο)
- Συνολικός χρήσιμος χώρος: N
- Επιτυγχάνει υψηλή απόδοση στις αναγνώσεις, αφού αυτές μπορούν να γίνουν από 2 δίσκους εναλλάξ (και παράλληλα)
- Παρέχει αξιοπιστία, αφού αν πάθει βλάβη ένας δίσκος, τα δεδομένα υπάρχουν στον 2ο

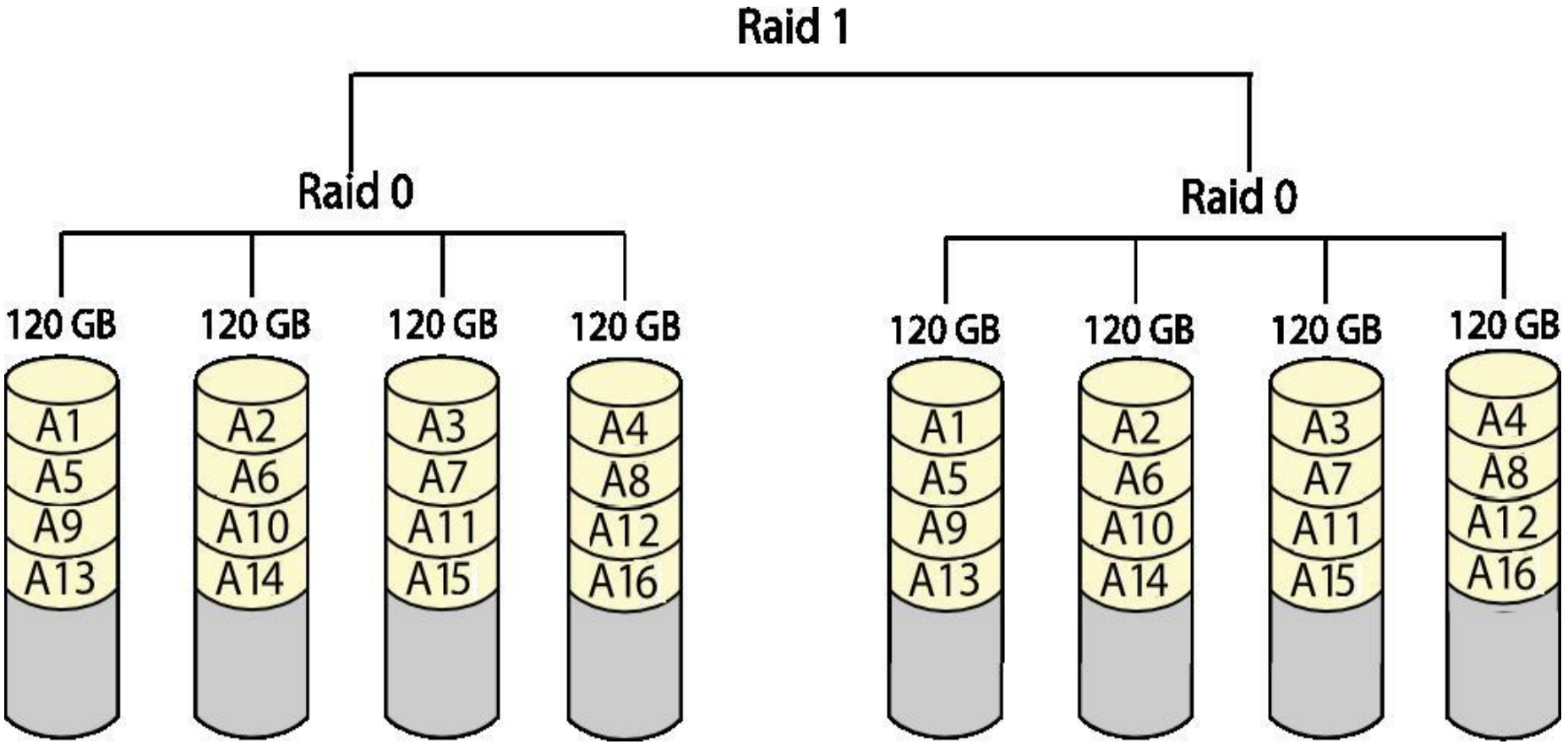


# Χρήση RAID 0 & 1

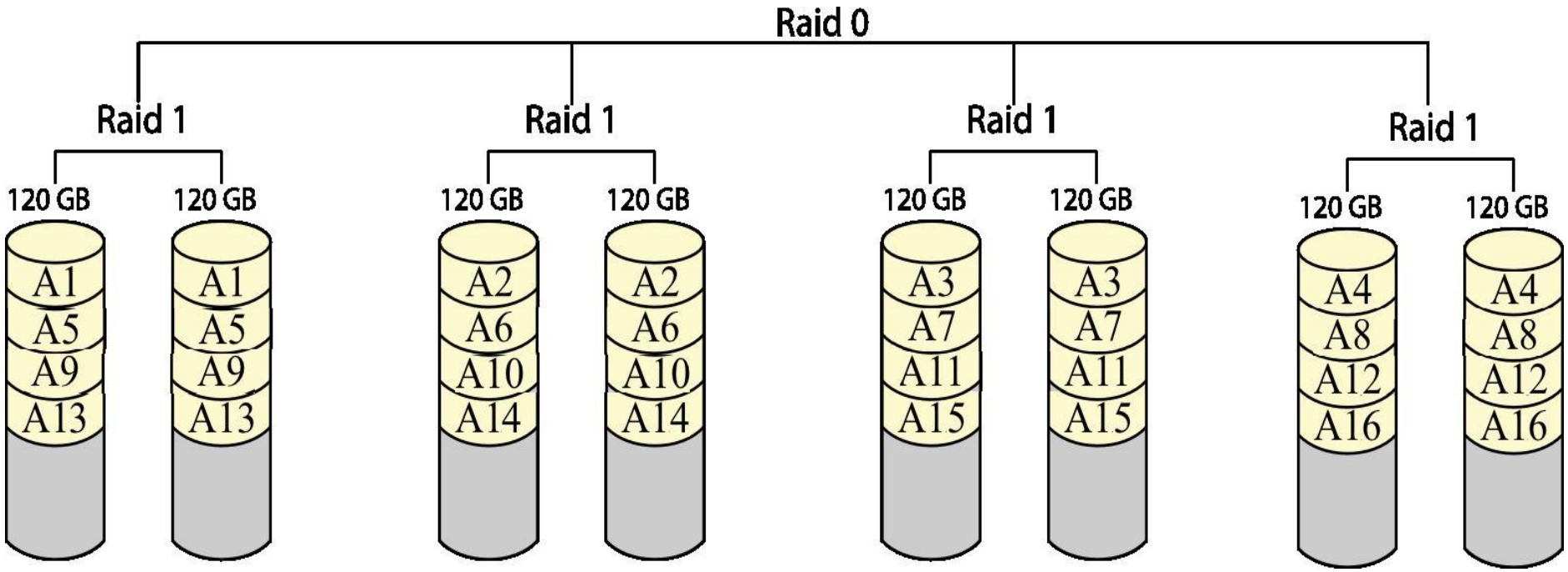
Έχουμε όγκο δεδομένων που χωράει σε 4 δίσκους Αγοράζουμε 8 φυσικούς δίσκους. Πώς θα τους οργανώσουμε για mirroring και stripping;

- RAID 0+1 (RAID 01)
  - Φτιάχνουμε 2 σύνολα των 4 δίσκων, το κάθε σύνολο το οργανώνουμε σε RAID 0 (stripping) και τα 2 σύνολα είναι mirror το ένα του άλλου (RAID-1)
- RAID 1+0 (RAID 10)
  - Φτιάχνουμε 4 σύνολα των 2 δίσκων, το κάθε σύνολο το οργανώνουμε σε RAID-1 (mirroring) και τα 4 σύνολα σε RAID-0 (stripping)

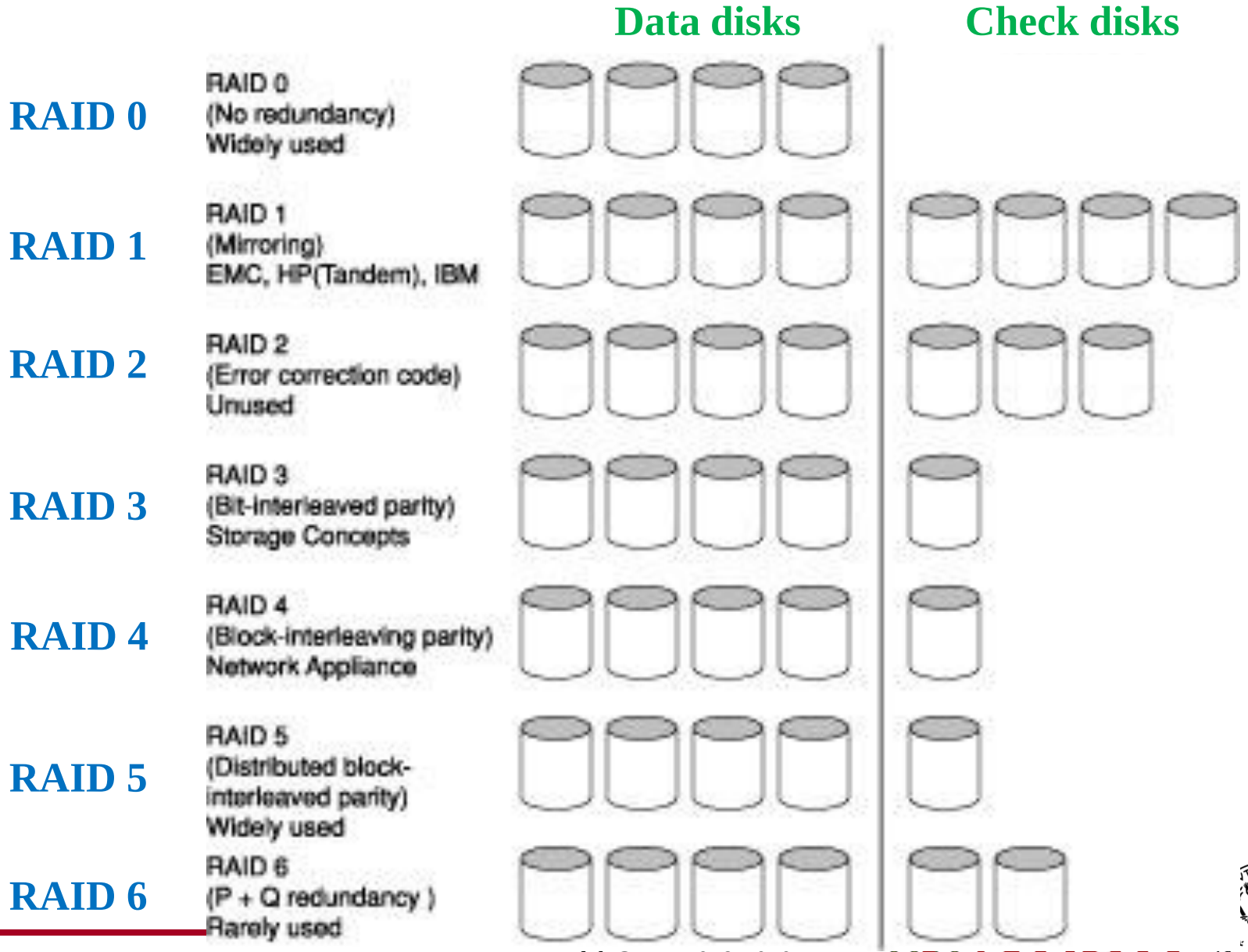
# RAID 0+1 (RAID 01)



# RAID 1+0 (RAID 10)



# Επίπεδα RAID



# RAID 2

- RAID 2: Χρήση κώδικα διόρθωσης σφαλμάτων (Error correcting code – ECC)
  - N + E δίσκοι (π.χ., 10 + 4)
  - Χωρισμός δεδομένων σε επίπεδο bit στους N δίσκους
  - Δημιουργία ECC των E bit
  - Υπερβολικά πολύπλοκο, δε χρησιμοποιείται στην πράξη

# RAID 3: Ισοτιμία πλέξης bit

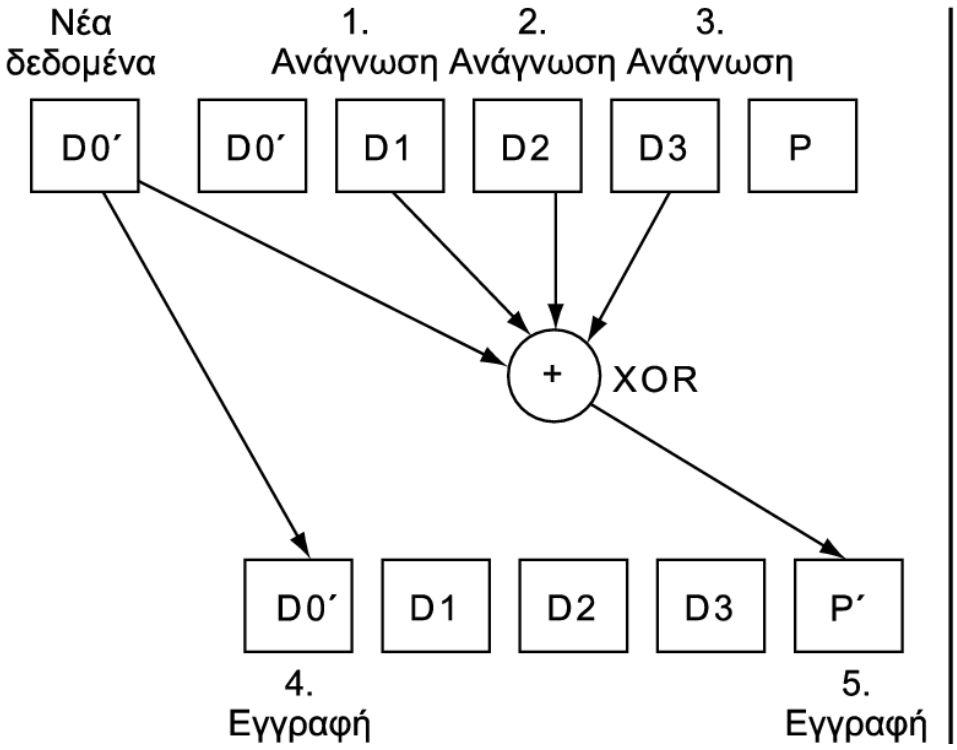
- Bit-Interleaved Parity
- N + 1 δίσκοι
  - Δεδομένα μοιράζονται σε N δίσκους σε επίπεδο byte
  - Πλεονασματικός δίσκος αποθηκεύει την ισοτιμία
  - Προσπέλαση ανάγνωσης
    - Ανάγνωση όλων των δίσκων
  - Προσπέλαση εγγραφής
    - Δημιουργία νέας ισοτιμίας και ενημέρωση όλων των δίσκων
  - Σε περίπτωση αστοχίας
    - Χρήση ισοτιμίας για επανασύσταση των χαμένων δεδομένων
- Δε χρησιμοποιείται ευρέως

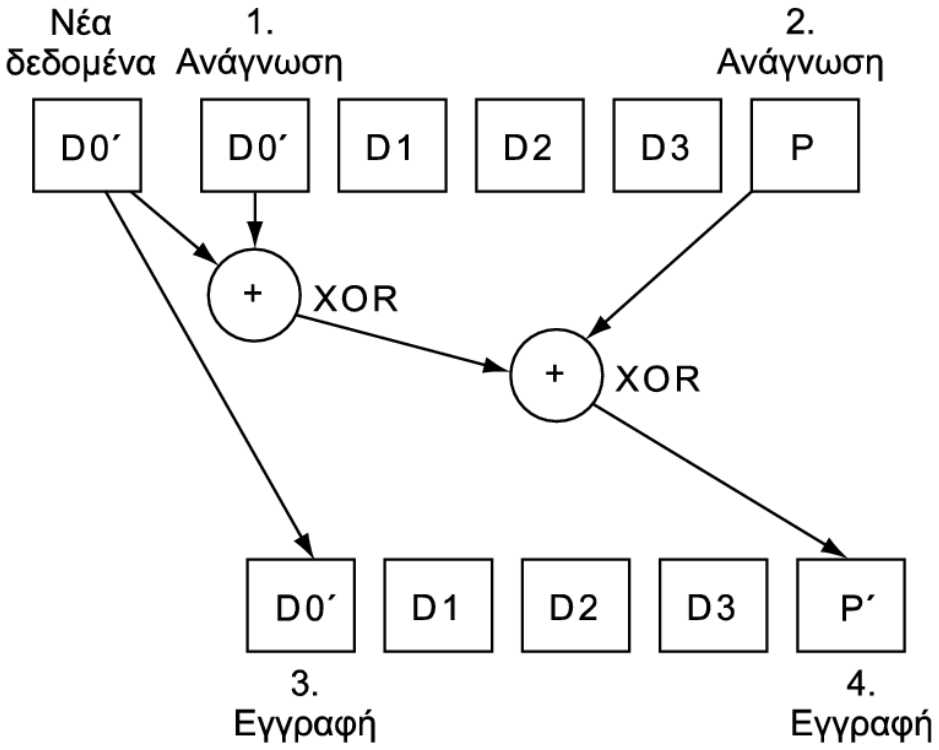
# RAID 4: Ισοτιμία πλέξης μπλοκ

- Block-Interleaved Parity
- N + 1 δίσκοι
  - Τα δεδομένα μοιράζονται σε N δίσκους σε επίπεδο μπλοκ
  - Πλεονασματικός δίσκος αποθηκεύει την ισοτιμία για μια ομάδα μπλοκ
  - Προσπέλαση ανάγνωσης
    - Διαβάζει μόνο το δίσκο που περιέχει το ζητούμενο μπλοκ
  - Προσπέλαση εγγραφής
    - Απλώς διαβάζει το δίσκο οι οποίος περιέχει το μπλοκ που τροποποιείται, και το δίσκο ισοτιμίας
    - Υπολογισμός νέας ισοτιμίας, ενημέρωση δίσκου δεδομένων και δίσκου ισοτιμίας
  - Σε περίπτωση αστοχίας
    - Χρήση ισοτιμίας για την επανασύσταση των χαμένων δεδομένων
- Δε χρησιμοποιείται ευρέως

# RAID 3 έναντι RAID 4 (small write updates)

## Τι γίνεται όταν γράφεται ένα block με νέα τιμή D0’



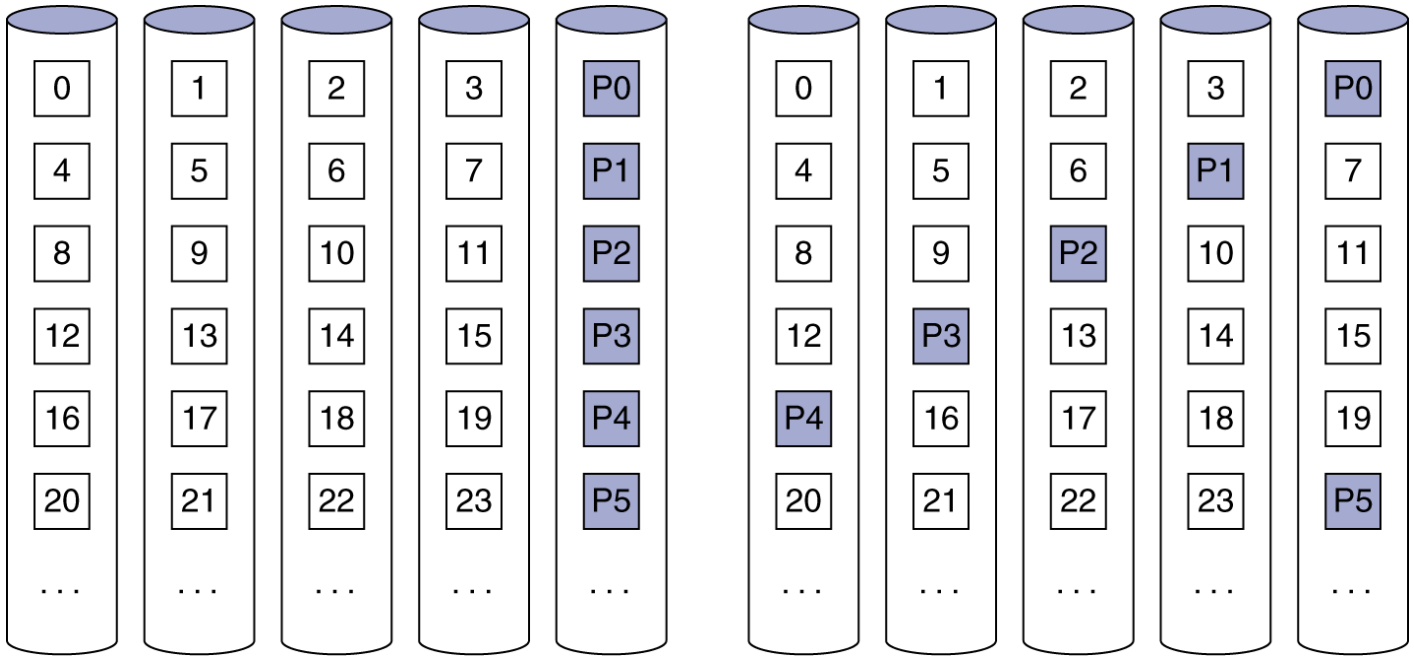




**RAID-3:** Υπολογισμός νέου Parity P’ αναγκάζει σε ανάγνωση όλους τους δίσκους (3 disk reads (D1, D2, D3) και 2 disk writes (D0’, P’)

**RAID-4:** Υπολογισμός νέου Parity P’ κάνει ανάγνωση σε 2 δίσκους (2 disk reads (D0, P) και 2 disk writes (D0’, P’)

# RAID 5: **Κατανεμημένη** ισοτιμία

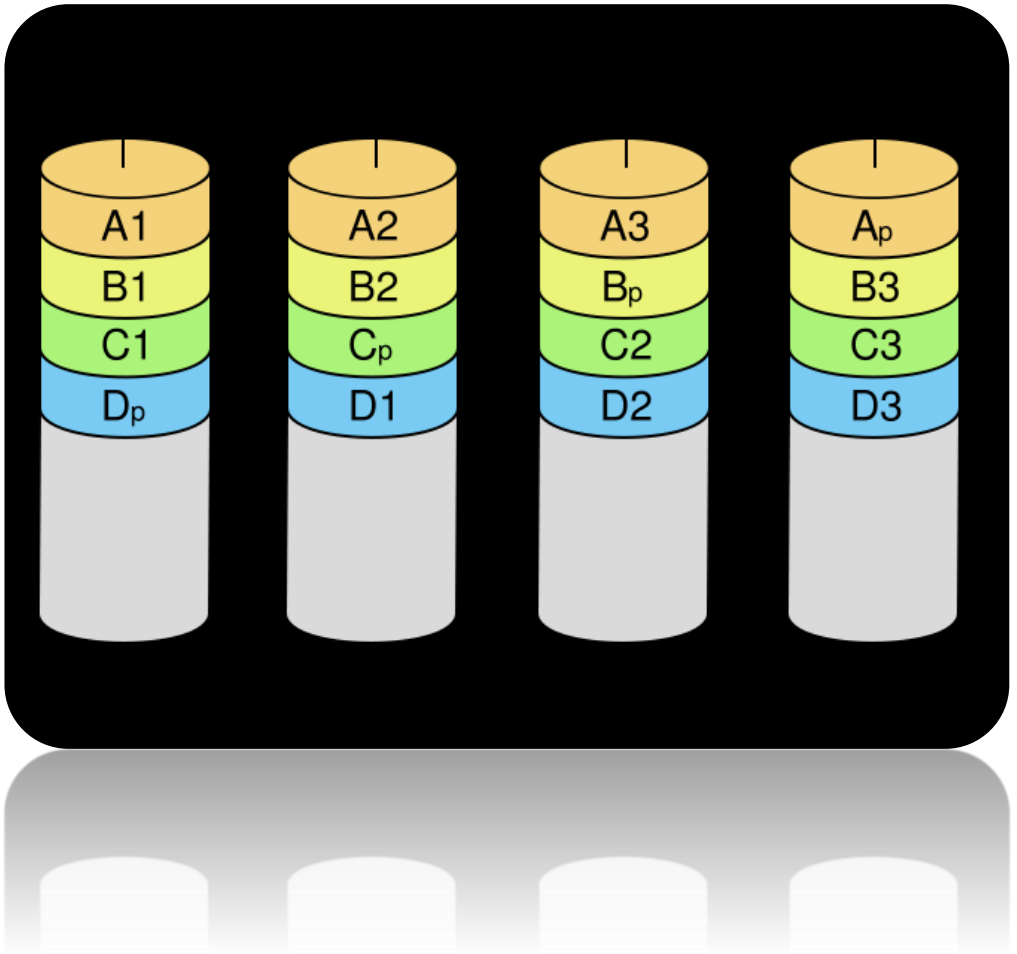
- N + 1 δίσκοι
  - RAID 4 + κατανομή μπλοκ ισοτιμίας σε όλους τους δίσκους
    - Αποφυγή σημείου συμφόρησης (bottleneck) στο δίσκο ισοτιμίας
- Ευρεία χρήση




**RAID 4** **RAID 5**

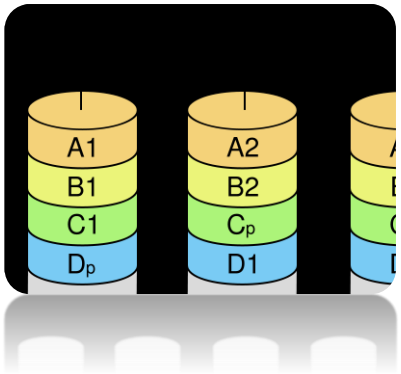
# RAID 5 - Striped set with distributed parity

- Συνεχόμενα blocks γράφονται εναλλάξ στους δίσκους, ενώ κατανέμεται σε αυτούς και ένα block ισοτιμίας
- Υψηλή απόδοση σε αναγνώσεις: μπορούν να γίνουν παράλληλα από πολλούς δίσκους
- Αξιοπιστία: αφού αν πάθει βλάβη ένας δίσκος, τα δεδομένα μπορούν να ανακτηθούν από τους υπόλοιπους



# RAID 5 - Striped set with distributed parity

**Παράδειγμα:** Έστω ότι διαθέτουμε 4 δίσκους.
Πώς δουλεύει το RAID 5;



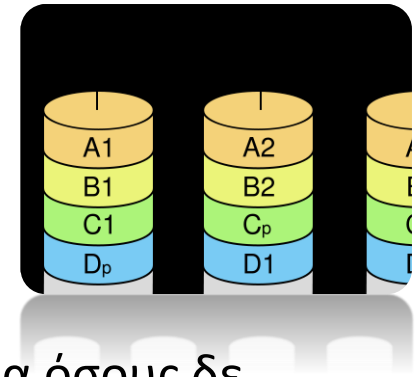

**Απάντηση:** Ας θεωρήσουμε ότι οι 4 δίσκοι έχουν τα Παρακάτω δεδομένα (δυαδικό):

| | DISK0 | DISK1 | DISK2 | DISK3 |
|---|---|---|---|---|
| STRIPE0 | 0100 | 0101 | 0010 | |
| STRIPE1 | 0010 | 0000 | | 0100 |
| STRIPE2 | 0011 | | 1010 | 1000 |
| STRIPE3 | | 0001 | 1101 | 1010 |

Στα κίτρινα σημεία, τοποθετούνται τα δεδομένα ισοτιμίας. Η ισοτιμία υπολογίζεται ως το **Exclusive-OR (XOR)** του ίδιου stripe όλων των δίσκων.

# RAID 5 - Striped set with distributed parity

**Παράδειγμα:** Έστω ότι διαθέτουμε 4 δίσκους.
Πώς δουλεύει το RAID 5;




**Απάντηση:** Ας θεωρήσουμε ότι οι 4 δίσκοι έχουν τα Παρακάτω δεδομένα (δυαδικό):

| | DISK0 | DISK1 | DISK2 | DISK3 |
|---|---|---|---|---|
| STRIPE0 | 0100 | 0101 | 0010 | |
| STRIPE1 | 0010 | 0000 | | 0100 |
| STRIPE2 | 0011 | | 1010 | 1000 |
| STRIPE3 | | 0001 | 1101 | 1010 |

Στα κίτρινα σημεία, τοποθετούνται τα δεδομένα ισοτιμίας. Η ισοτιμία υπολογίζεται ως το **Exclusive-OR (XOR)** του ίδιου stripe όλων των δίσκων.

Για όσους δε θυμούνται της XOR
…
ο πίνακας αληθείας

| XOR | | |
|---|---|---|
| Είσοδος | | Έξοδος |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

# RAID 5 - Striped set with distributed parity

| | DISK0 | DISK1 | DISK2 | DISK3 |
|---|---|---|---|---|
| **STRIPE0** | **0100** | **0101** | **0010** | **0011** |
| STRIPE1 | 0010 | 0000 | | 0100 |
| STRIPE0 | 0011 | | 1010 | 1000 |
| STRIPE3 | | 0001 | 1101 | 1010 |

STRIPE0,DISK3 = 0100 XOR 0101 XOR 0010 = 0011

# RAID 5 - Striped set with distributed parity

| | DISK0 | DISK1 | DISK2 | DISK3 |
|---|---|---|---|---|
| STRIPE0 | 0100 | 0101 | 0010 | 0011 |
| **STRIPE1** | **0010** | **0000** | **0110** | **0100** |
| STRIPE2 | 0011 | | 1010 | 1000 |
| STRIPE3 | | 0001 | 1101 | 1010 |

STRIPE0,DISK3 = 0100 XOR 0101 XOR 0010 = 0011
STRIPE1,DISK2 = 0010 XOR 0000 XOR 0100 = 0110

# RAID 5 - Striped set with distributed parity

| | DISK0 | DISK1 | DISK2 | DISK3 |
|---|---|---|---|---|
| STRIPE0 | 0100 | 0101 | 0010 | 0011 |
| STRIPE1 | 0010 | 0000 | 0110 | 0100 |
| **STRIPE2** | **0011** | **0001** | **1010** | **1000** |
| STRIPE3 | | 0001 | 1101 | 1010 |

STRIPE0,DISK3 = 0100 XOR 0101 XOR 0010 = 0011
STRIPE1,DISK2 = 0010 XOR 0000 XOR 0100 = 0110
STRIPE2,DISK1 = 0011 XOR 1010 XOR 1000 = 0001

# RAID 5 - Striped set with distributed parity

| | DISK0 | DISK1 | DISK2 | DISK3 |
|---|---|---|---|---|
| STRIPE0 | 0100 | 0101 | 0010 | 0011 |
| STRIPE1 | 0010 | 0000 | 0110 | 0100 |
| STRIPE2 | 0011 | 0001 | 1010 | 1000 |
| **STRIPE3** | **0110** | **0001** | **1101** | **1010** |

STRIPE0,DISK3 = 0100 XOR 0101 XOR 0010 = 0011
STRIPE1,DISK2 = 0010 XOR 0000 XOR 0100 = 0110
STRIPE2,DISK1 = 0011 XOR 1010 XOR 1000 = 0001
STRIPE3,DISK0 = 0001 XOR 1101 XOR 1010 = 0110

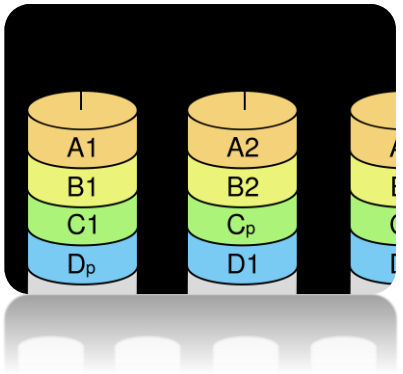
# RAID 5 - Striped set with distributed parity

**Τελική Εικόνα της συστοιχίας ΔΙΣΚΩΝ με διάταξη RAID5**

| | DISK0 | DISK1 | DISK2 | DISK3 |
|---|---|---|---|---|
| STRIPE0 | 0100 | 0101 | 0010 | 0011 |
| STRIPE1 | 0010 | 0000 | 0110 | 0100 |
| STRIPE2 | 0011 | 0001 | 1010 | 1000 |
| STRIPE3 | 0110 | 0001 | 1101 | 1010 |

# RAID 5 - Striped set with distributed parity

**Παράδειγμα:** Τι γίνεται στις εγγραφές;




**Απάντηση:** Ας θεωρήσουμε ότι οι 4 δίσκοι έχουν τα Παρακάτω δεδομένα (δυαδικό):

| | DISK0 | DISK1 | DISK2 | DISK3 |
|---|---|---|---|---|
| STRIPE0 | 0100 | 0101 | 0010 | 0011 |
| STRIPE1 | 0010 | 0000 | 0110 | 0100 |
| STRIPE2 | 0011 | 0001 | 1010 | 1000 |
| STRIPE3 | 0110 | 0001 | 1101 | 1010 |

Έστω ότι γίνεται η εγγραφή του στοιχείου 1101 στο block 2 (αρίθμηση ξεκινάει από block 0).

# RAID 5 - Striped set with distributed parity

Έστω ότι γίνεται η εγγραφή του **στοιχείου 1101 στο block 2**
(αρίθμηση ξεκινάει από block 0)



| | DISK0 | DISK1 | DISK2 | DISK3 |
|---|---|---|---|---|
| STRIPE0 | 0100 | 0101 | 0010 | 0011 |
| STRIPE1 | 0010 | 0000 | 0110 | 0100 |
| STRIPE2 | 0011 | 0001 | 1010 | 1000 |
| STRIPE3 | 0110 | 0001 | 1101 | 1010 |

# RAID 5 - Striped set with distributed parity

Έστω ότι γίνεται η εγγραφή του **στοιχείου 1101 στο block 2**
(αρίθμηση ξεκινάει από block 0)

| | DISK0 | DISK1 | DISK2 | DISK3 |
|---|---|---|---|---|
| STRIPE0 | 0100 | 0101 | ~~0010~~ 1101 | 0011 |
| STRIPE1 | 0010 | 0000 | 0110 | 0100 |
| STRIPE2 | 0011 | 0001 | 1010 | 1000 |
| STRIPE3 | 0110 | 0001 | 1101 | 1010 |

Ο ελεγκτής RAID κάνει την εγγραφή του στοιχείου στο αντίστοιχο block …

# RAID 5 - Striped set with distributed parity

Έστω ότι γίνεται η εγγραφή του **στοιχείου 1101 στο block 2**
(αρίθμηση ξεκινάει από block 0)

| | DISK0 | DISK1 | DISK2 | DISK3 |
|---|---|---|---|---|
| STRIPE0 | 0100 | 0101 | ~~0010~~ 1101 | 0011 |
| STRIPE1 | 0010 | 0000 | 0110 | 0100 |
| STRIPE2 | 0011 | 0001 | 1010 | 1000 |
| STRIPE3 | 0110 | 0001 | 1101 | 1010 |

Ο ελεγκτής RAID κάνει την εγγραφή του στοιχείου στο αντίστοιχο block ... και ταυτόχρονα ξαναδημιουργεί την ισοτιμία για το συγκεκριμένο stripe, χρησιμοποιώντας παλιά τιμή, νέα τιμή και ισοτιμία
STRIPE0,DISK3 = **0010** XOR **1101** XOR **0011** = **1100**

# RAID 5 - Striped set with distributed parity

Έστω ότι γίνεται η εγγραφή του **στοιχείου 1101 στο block 2** (αρίθμηση ξεκινάει από block 0).

| | DISK0 | DISK1 | DISK2 | DISK3 |
|---|---|---|---|---|
| STRIPE0 | 0100 | 0101 | ~~0010~~ 1101 | ~~0011~~ 1100 |
| STRIPE1 | 0010 | 0000 | 0110 | 0100 |
| STRIPE2 | 0011 | 0001 | 1010 | 1000 |
| STRIPE3 | 0110 | 0001 | 1101 | 1010 |

Ο ελεγκτής RAID κάνει την εγγραφή του στοιχείου στο αντίστοιχο block ... και ταυτόχρονα ξαναδημιουργεί την ισοτιμία για το συγκεκριμένο stripe, χρησιμοποιώντας παλιά τιμή, νέα τιμή και ισοτιμία STRIPE0,DISK3 = 0010 XOR 1101 XOR 0011 = **1100**

# RAID 5 - Striped set with distributed parity

Έστω ότι γίνεται η εγγραφή του **στοιχείου 1101 στο block 2**
(αρίθμηση ξεκινάει από block 0)

| | DISK0 | DISK1 | DISK2 | DISK3 |
|---|---|---|---|---|
| STRIPE0 | 0100 | 0101 | **1101** | **1100** |
| STRIPE1 | 0010 | 0000 | 0110 | 0100 |
| STRIPE2 | 0011 | 0001 | 1010 | 1000 |
| STRIPE3 | 0110 | 0001 | 1101 | 1010 |

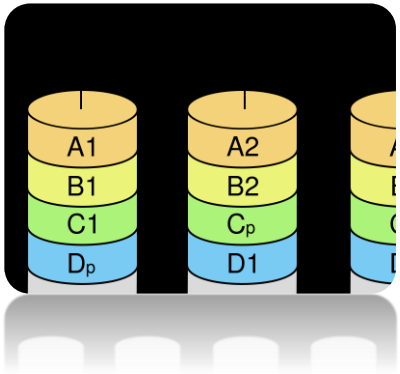
Ο ελεγκτής RAID κάνει την εγγραφή του στοιχείου στο αντίστοιχο block ... και ταυτόχρονα ξαναδημιουργεί την ισοτιμία για το συγκεκριμένο stripe, χρησιμοποιώντας παλιά τιμή, νέα τιμή και ισοτιμία
STRIPE0,DISK3 = 0010 XOR 1101 XOR 0011 = 1100

*Η εγγραφή στο RAID 5, απαιτεί με 2 αναγνώσεις και 2 εγγραφές σε δίσκους*

# RAID 5 - Striped set with distributed parity

**Παράδειγμα:** Τι γίνεται αν χαλάσει ένας δίσκος;

**Απάντηση:** Ας θεωρήσουμε ότι οι 4 δίσκοι έχουν τα Παρακάτω δεδομένα (δυαδικό):



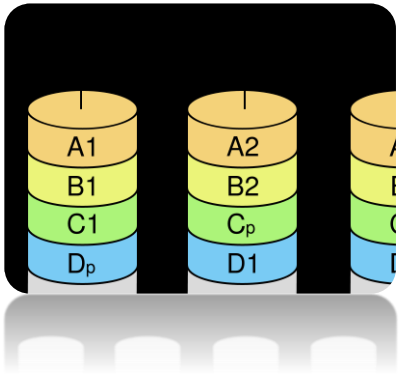

| | DISK0 | DISK1 | DISK2 | DISK3 |
|---|---|---|---|---|
| STRIPE0 | 0100 | 0101 | 1101 | 1100 |
| STRIPE1 | 0010 | 0000 | 0110 | 0100 |
| STRIPE2 | 0011 | 0001 | 1010 | 1000 |
| STRIPE3 | 0110 | 0001 | 1101 | 1010 |

Έστω ότι χαλάει ο DISK2

# RAID 5 - Striped set with distributed parity

**Παράδειγμα:** Τι γίνεται αν χαλάσει ένας δίσκος;

**Απάντηση:** Ας θεωρήσουμε ότι οι 4 δίσκοι έχουν τα Παρακάτω δεδομένα (δυαδικό):




| | DISK0 | DISK1 | DISK2 | DISK3 |
|---|---|---|---|---|
| STRIPE0 | 0100 | 0101 | 1101 | 1100 |
| STRIPE1 | 0010 | 0000 | 0110 | 0100 |
| STRIPE2 | 0011 | 0001 | 1010 | 1000 |
| STRIPE3 | 0110 | 0001 | 1101 | 1010 |

Έστω ότι χαλάει ο DISK2

# RAID 5 - Striped set with distributed parity

Έστω ότι χαλάει ο DISK2

| | DISK0 | DISK1 | DISK2 | DISK3 |
|---|---|---|---|---|
| **STRIPE0** | **0100** | **0101** | 1101 | **1100** |
| STRIPE1 | 0010 | 0000 | 0110 | 0100 |
| STRIPE2 | 0011 | 0001 | 1010 | 1000 |
| STRIPE3 | 0110 | 0001 | 1101 | 1010 |

Ο ελεγκτής RAID εξυπηρετεί τις αιτήσεις για τις πληροφορίες που είχε ο DISK2, χρησιμοποιώντας όλους τους άλλους δίσκους + την ισοτιμία.

# RAID 5 - Striped set with distributed parity

Έστω ότι χαλάει ο DISK2

| | DISK0 | DISK1 | DISK2 | DISK3 |
|---|---|---|---|---|
| **STRIPE0** | **0100** | **0101** | 1101 | **1100** |
| STRIPE1 | 0010 | 0000 | 0110 | 0100 |
| STRIPE2 | 0011 | 0001 | 1010 | 1000 |
| STRIPE3 | 0110 | 0001 | 1101 | 1010 |

Ο ελεγκτής RAID εξυπηρετεί τις αιτήσεις για τις πληροφορίες που είχε ο DISK2, χρησιμοποιώντας όλους τους άλλους δίσκους + την ισοτιμία. Έτσι

STRIPE0,DISK2 = 0100 XOR 0101 XOR 1100 = 1101

# RAID 5 - Striped set with distributed parity

Έστω ότι χαλάει ο DISK2

| | DISK0 | DISK1 | DISK2 | DISK3 |
|---|---|---|---|---|
| STRIPE0 | 0100 | 0101 | 1101 | 1100 |
| STRIPE1 | 0010 | 0000 | 0110 | 0100 |
| **STRIPE2** | **0011** | **0001** | 1010 | **1000** |
| STRIPE3 | 0110 | 0001 | 1101 | 1010 |

Ο ελεγκτής RAID εξυπηρετεί τις αιτήσεις για τις πληροφορίες που είχε ο DISK2, χρησιμοποιώντας όλους τους άλλους δίσκους + την ισοτιμία. Έτσι

STRIPE0,DISK2 = 0100 XOR 0101 XOR 1100 = 1101

STRIPE**2**,DISK2 = 0011 XOR 0001 XOR 1000 = 1010

# RAID 5 - Striped set with distributed parity

Έστω ότι χαλάει ο DISK2

| | DISK0 | DISK1 | DISK2 | DISK3 |
|---|---|---|---|---|
| STRIPE0 | 0100 | 0101 | 1101 | 1100 |
| STRIPE1 | 0010 | 0000 | 0110 | 0100 |
| STRIPE2 | 0011 | 0001 | 1010 | 1000 |
| **STRIPE3** | **0110** | **0001** | 1101 | **1010** |

Ο ελεγκτής RAID εξυπηρετεί τις αιτήσεις για τις πληροφορίες που είχε ο DISK2, χρησιμοποιώντας όλους τους άλλους δίσκους + την ισοτιμία. Έτσι

STRIPE0,DISK2 = 0100 XOR 0101 XOR 1100 = 1101
STRIPE2,DISK2 = 0011 XOR 0001 XOR 1000 = 1010
STRIPE3,DISK2 = 0110 XOR 0001 XOR 1010 = 1101

# RAID 5 - Striped set with distributed parity

Έστω ότι χαλάει ο DISK2

| | DISK0 | DISK1 | DISK2 | DISK3 |
|---|---|---|---|---|
| STRIPE0 | 0100 | 0101 | 1101 | 1100 |
| STRIPE1 | 0010 | 0000 | 0110 | 0100 |
| STRIPE2 | 0011 | 0001 | 1010 | 1000 |
| STRIPE3 | 0110 | 0001 | 1101 | 1010 |

Ο ελεγκτής RAID εξυπηρετεί τις αιτήσεις για τις πληροφορίες που είχε ο DISK2, χρησιμοποιώντας όλους τους άλλους δίσκους + την ισοτιμία.
Έτσι

STRIPE0,DISK2 = 0100 XOR 0101 XOR 1100 = 1101
STRIPE2,DISK2 = 0011 XOR 0001 XOR 1000 = 1010
STRIPE3,DISK2 = 0110 XOR 0001 XOR 1010 = 1101

*Κάθε ανάγνωση του χαλασμένου δίσκου, αντιστοιχεί σε αναγνώσεις σε όλους τους υπόλοιπους δίσκους. Καλό είναι να αντικαταστήσουμε το χαλασμένο δίσκο γρήγορα!*

# RAID 5 - Striped set with distributed parity

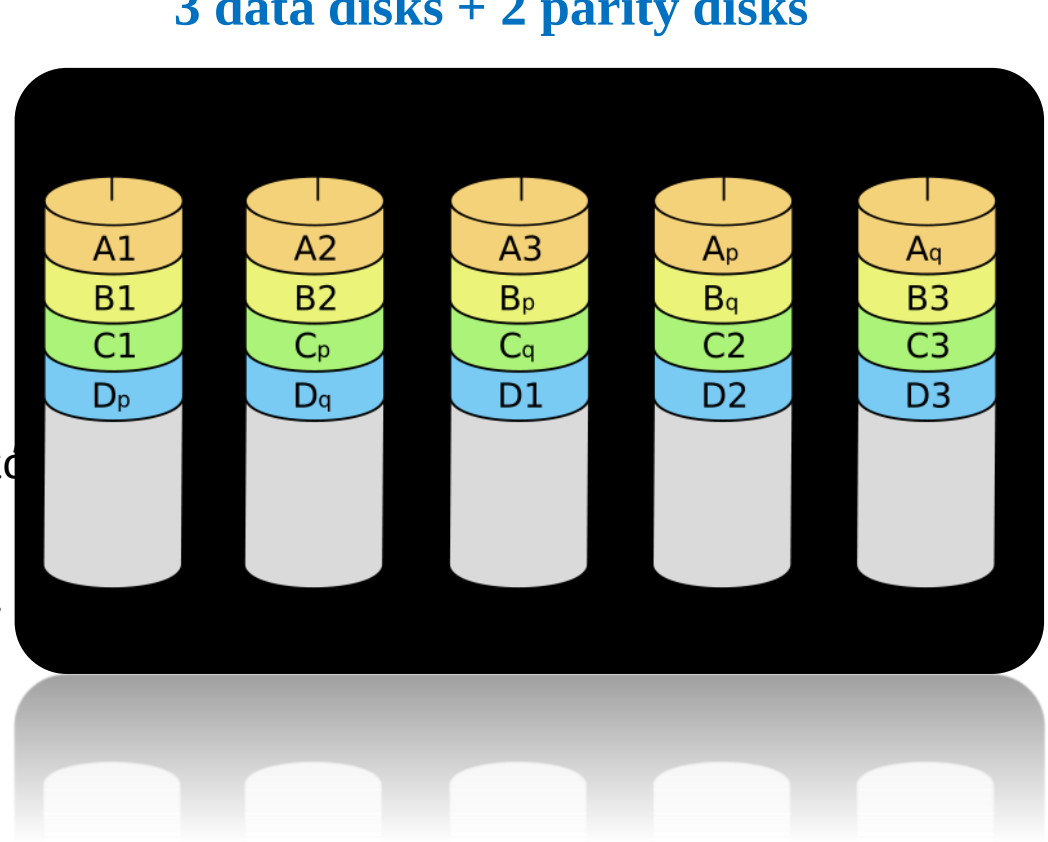
Όταν αντικατασταθεί ο DISK2, ανακατασκευή των περιεχομένων του:

STRIPE0,DISK2 = 0100 XOR 0101 XOR 1100 = 1101
STRIPE1,DISK2 = 0010 XOR 0001 XOR 0100 = 0110
STRIPE2,DISK2 = 0011 XOR 0001 XOR 1000 = 1010
STRIPE3,DISK2 = 0110 XOR 0001 XOR 1010 = 1101

| | DISK0 | DISK1 | DISK2 | DISK3 |
|---|---|---|---|---|
| STRIPE0 | 0100 | 0101 | 1101 | 1100 |
| STRIPE1 | 0010 | 0000 | 0110 | 0100 |
| STRIPE2 | 0011 | 0001 | 1010 | 1000 |
| STRIPE3 | 0110 | 0001 | 1101 | 1010 |

# RAID 6 - Striped set with dual (P+Q) parity

- P + Q Redundancy, N + 2 δίσκοι
  - RAID 5 με *δύο* group ισοτιμίας
- Συνεχόμενα blocks γράφονται εναλλάξ στους δίσκους, ενώ κατανέμεται σε αυτούς και δύο block ισοτιμίας
- Υψηλή απόδοση σε αναγνώσεις:
  - αφού αυτές μπορούν να γίνουν από πολλούς δίσκους εναλλάξ
- Μεγαλύτερη ανοχή σε ελαττώματα (αξιοπιστία) μέσω περισσότερου πλεονασμού:
  - αν χαλάσουν μέχρι 2 δίσκοι, τα δεδομένα ανακτούνται από τους υπόλοιπους



# Περίληψη RAID

- Το RAID μπορεί να βελτιώσει την απόδοση και τη διαθεσιμότητα (availability)
  - Η υψηλή διαθεσιμότητα απαιτεί «εν θερμώ» εναλλαγή (hot swapping)
- Υποθέτει ότι οι αστοχίες δίσκων είναι ανεξάρτητες
  - Μεγάλο πρόβλημα αν καεί όλο το κτήριο!
- Δείτε το “Hard Disk Performance, Quality and Reliability”
  - http://www.pcguide.com/ref/hdd/perf/index.htm

# Σχεδίαση συστήματος Ε/Ε

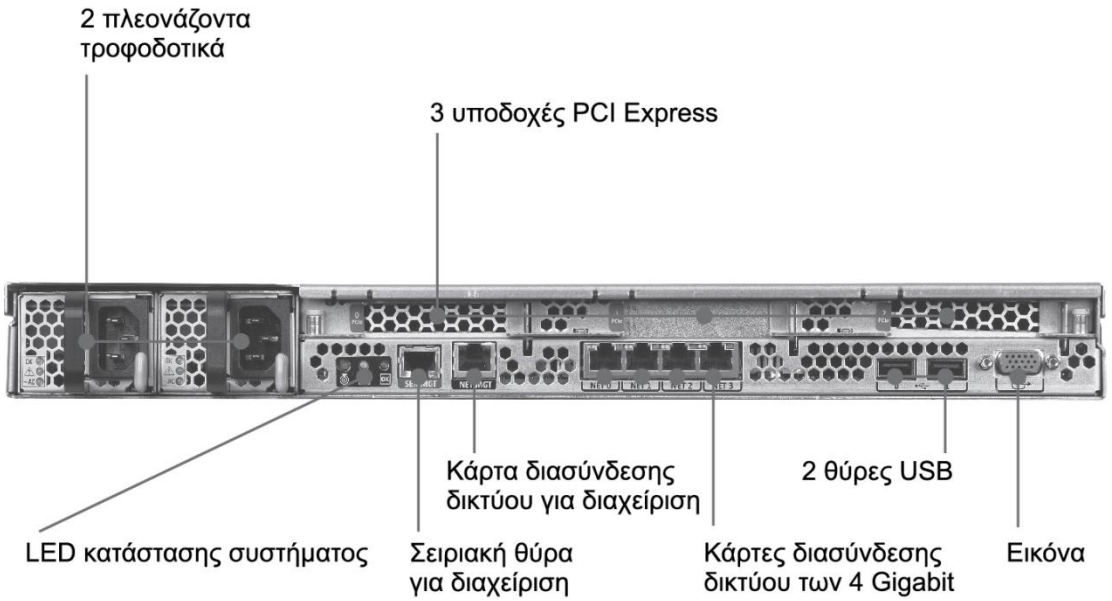
- Ικανοποίηση απαιτήσεων λανθάνοντος χρόνου
  - Για χρονικά κρίσιμες λειτουργίες
  - Αν το σύστημα δεν είναι φορτωμένο
    - Άθροισμα των λανθανόντων χρόνων των συστατικών
- Μεγιστοποίηση ρυθμού διεκπεραίωσης
  - Εύρεση του «πιο αδύναμου κρίκου» (το συστατικό με το χαμηλότερο εύρος ζώνης)
  - Ρύθμιση για λειτουργία στο μέγιστο εύρος ζώνης του
  - Ισορροπία μεταξύ των υπόλοιπων συστατικών του συστήματος
- Αν το σύστημα είναι φορτωμένο, η απλή ανάλυση δεν είναι αρκετή
  - Χρειάζονται μοντέλα ουρών αναμονής ή προσομοίωση

# Διακομιστές

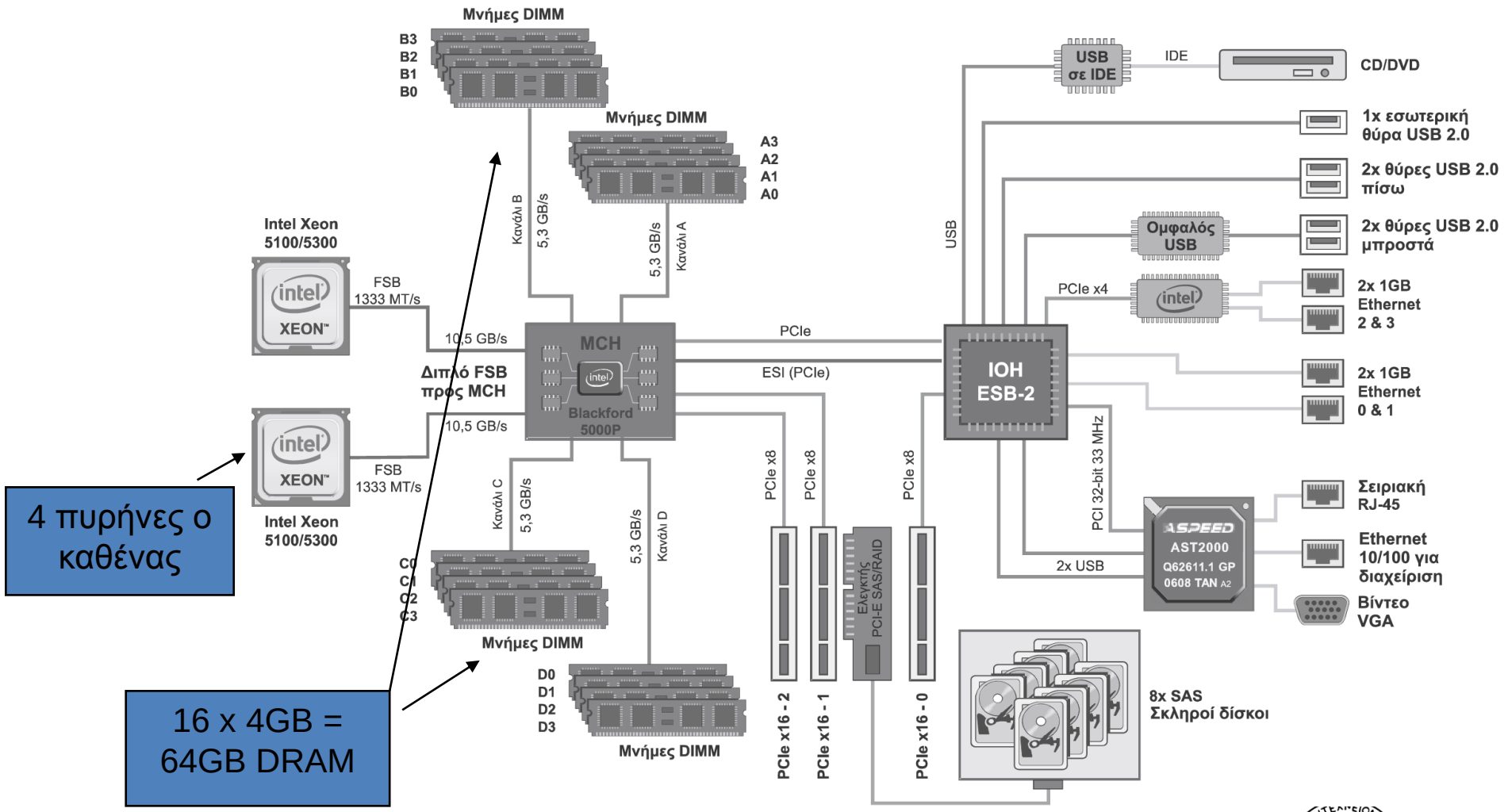
- Οι εφαρμογές εκτελούνται όλο και περισσότερο σε διακομιστές (servers)
  - Αναζήτηση στον Ιστό, εφαρμογές γραφείου, εικονικοί κόσμοι, …
- Απαιτούνται μεγάλοι διακομιστές κέντρων δεδομένων
  - Πολλοί επεξεργαστές, συνδέσεις δικτύου, μαζική αποθήκευση
  - Περιορισμοί χώρου και ηλεκτρικής ισχύος
- Εξοπλισμός διακομιστών για ικριώματα (racks) των 19 ιντσών
  - Ύψος σε πολλαπλάσια 1.75 ιντσών (1U)

# Διακομιστές για ικρίωμα

## Διακομιστής Sun Fire x4150 1U



# Διακομιστής Sun Fire x4150 1U



# Παράδειγμα σχεδίασης συστήματος Ε/Ε

- Δεδομένα: ένα σύστημα Sun Fire x4150 με
  - Φορτίο εργασία: αναγνώσεις δίσκου των 64KB
    - Κάθε λειτουργία Ε/Ε απαιτεί 200000 εντολές κώδικα χρήστη και 100000 εντολές του ΛΣ
  - Κάθε CPU: 109 εντολές/sec
  - FSB (Front Side Bus, Εμπρόσθιος δίαυλος): 10.6 GB/sec μέγιστο
  - DRAM DDR2 στα 667MHz: 5.336 GB/sec
  - PCI-E 8× δίαυλος: 8 × 250MB/sec = 2GB/sec
  - Δίσκοι: 15000 rpm, 2.9ms μέσος χρόνος αναζήτησης, 112MB/sec ρυθμός μεταφοράς
- Ποιος είναι ο ρυθμός Ε/Ε που μπορεί να διατηρηθεί;
  - Για τυχαίες αναγνώσεις, και για ακολουθιακές αναγνώσεις

# Παράδειγμα σχεδίασης (συνέχεια)

- Ρυθμός Ε/Ε για τις CPU
  - Ανά πυρήνα: $10^9/(100000 + 200000) = 3333$
  - 8 πυρήνες: 26667 λειτουργίες/sec
- Τυχαίες αναγνώσεις, ρυθμός Ε/Ε για τους δίσκους
  - Υποθέστε ότι ο πραγματικός χρόνος αναζήτησης είναι το 1/4 μέσου
  - Χρόνος/λειτουργία = αναζήτηση + λανθάνων χρόνος + μεταφορά = 2.9ms/4 + 4ms/2 + 64KB/(112MB/s) = 3.3ms
  - 303 λειτουργίες/sec ανά δίσκο, 2424 λειτουργίες/sec για 8 δίσκους
- Ακολουθιακές αναγνώσεις
  - 112MB/s / 64KB = 1750 λειτουργίες/sec ανά δίσκο
  - 14000 λειτουργίες/sec για 8 δίσκους

# Παράδειγμα σχεδίασης (συνέχεια)

- Ρυθμός Ε/Ε του PCI-E
  - 2GB/sec / 64KB = 31,250 λειτουργίες/sec
- Ρυθμός Ε/Ε της DRAM
  - 5.336 GB/sec / 64KB = 83375 λειτουργίες/sec
- Ρυθμός Ε/Ε του FSB
  - Υποθέστε ότι μπορούμε να διατηρήσουμε το μισό του μέγιστου ρυθμού
  - 5.3 GB/sec / 64KB = 81540 λειτουργίες/sec ανά FSB
  - 163080 λειτουργίες/sec για 2 FSB
- Ο πιο αδύναμος κρίκος: οι δίσκοι
  - 2424 λειτουργίες/sec τυχαίες, 14000 λειτουργίες/sec ακολουθιακές
  - Τα άλλα συστατικά έχουν άφθονο χώρο για να χωρέσουν αυτούς του ρυθμούς

# Πλάνη: Φερεγγυότητα δίσκων

- Αν ένας κατασκευαστής δίσκων δίνει ότι το MTTF είναι 1200000 ώρες (140 χρόνια)
  - Ένας δίσκος θα έχει τόσο μεγάλη διάρκεια
- Λάθος: αυτός είναι ο μέσος χρόνος πρώτης αστοχίας
  - Ποια είναι η κατανομή των αστοχιών;
  - Τι θα γίνει αν έχετε 1000 δίσκους;
    - Πόσοι θα αστοχούν κάθε χρόνο;

$$\text{Annual Failure Rate (AFR)} = \frac{1000 \text{ disks} \times 8760 \text{ hrs/disk}}{1200000 \text{ hrs/failur e}} = 0.73\%$$

# Πλάνες

- Οι ρυθμοί αστοχίας δίσκων είναι αυτοί που λένε οι προδιαγραφές
  - Μελέτες ρυθμών αστοχίας στο πεδίο
    - Schroeder και Gibson: 2% ως 4% έναντι. 0.6% ως 0.8%
    - Pinheiro, *κ.ά.*: 1.7% (πρώτος χρόνος) ως 8.6% (τρίτος χρόνος) έναντι 1.5%
  - Γιατί;
- Μια διασύνδεση του 1GB/s μεταφέρει 1GB σε ένα sec
  - Αλλά τι είναι το GB;
  - Για το εύρος ζώνης, χρήση του 1GB = $10^9$ B
  - Για αποθήκευση, χρήση του 1GB = $2^{30}$ B = $1.075 \times 10^9$ B
  - Άρα το 1GB/sec είναι 0.93GB σε ένα second
    - Περίπου 7% σφάλμα

# Παγίδα: «ξεφόρτωμα» σε επεξεργαστές Ε/Ε

- Η επιβάρυνση της διαχείρισης των αιτήσεων του επεξεργαστή Ε/Ε μπορεί να κυριαρχεί
  - Ταχύτερο να γίνεται η μικρή λειτουργία στη CPU
  - Αλλά η αρχιτεκτονική Ε/Ε μπορεί να μην το επιτρέπει αυτό
- Ο επεξεργαστής Ε/Ε μπορεί να είναι πιο αργός
  - Εφόσον υποτίθεται ότι είναι απλούστερος
- Αν γίνει ταχύτερος καθίσταται σημαντικό συστατικό του συστήματος
  - Μπορεί να χρειάζεται τους δικούς του συνεπεξεργαστές!

# Παγίδα: αντίγραφα ασφαλείας σε ταινία

- Η μαγνητική ταινία είχε πλεονεκτήματα
  - Φορητότητα, μεγάλη χωρητικότητα
- Τα πλεονεκτήματα άρχισαν να χάνονται με τις εξελίξεις της τεχνολογίας δίσκων
- Είναι πιο λογικό να επαναλαμβάνονται τα δεδομένα
  - Π.χ, RAID, απομακρυσμένη δημιουργία ειδώλων (remote mirroring)

# Πλάνη: χρονοπρ/μός δίσκου

- Καλύτερα ο χρονοπρογραμματισμός των προσπελάσεων δίσκου να γίνεται από το ΛΣ
  - Αλλά οι σύγχρονες μονάδες ασχολούνται με δ/νσεις λογικών μπλοκ
    - Χαρτογράφηση σε θέσεις φυσικής τροχιάς, κυλίνδρου, τομέα
    - Επίσης, τα μπλοκ αποθηκεύονται και στην κρυφή μνήμη της μονάδας
  - Το ΛΣ δε γνωρίζει τις φυσικές θέσεις
    - Η αναδιάταξη μπορεί να μειώσει την απόδοση
    - Εξαρτάται από την τοποθέτηση και τη χρήση της κρυφής μνήμης

# Παγίδα: μέγιστη απόδοση

- Οι μέγιστοι ρυθμοί Ε/Ε είναι σχεδόν αδύνατον να επιτευχθούν
  - Συνήθως, κάποια άλλα συστατικά του συστήματος περιορίζουν την απόδοση
  - Π.χ., μεταφορές στη μνήμη μέσω ενός διαύλου
    - Σύγκρουση με την ανανέωση (refresh) της DRAM
    - Συναγωνισμός διαιτησίας με άλλου κύριους (masters) του διαύλου
  - Π.χ., δίαυλος PCI: μέγιστο εύρος ζώνης ~133 MB/sec
    - Στην πράξη, μπορεί να διατηρηθεί το 80MB/sec κατά μέγιστο

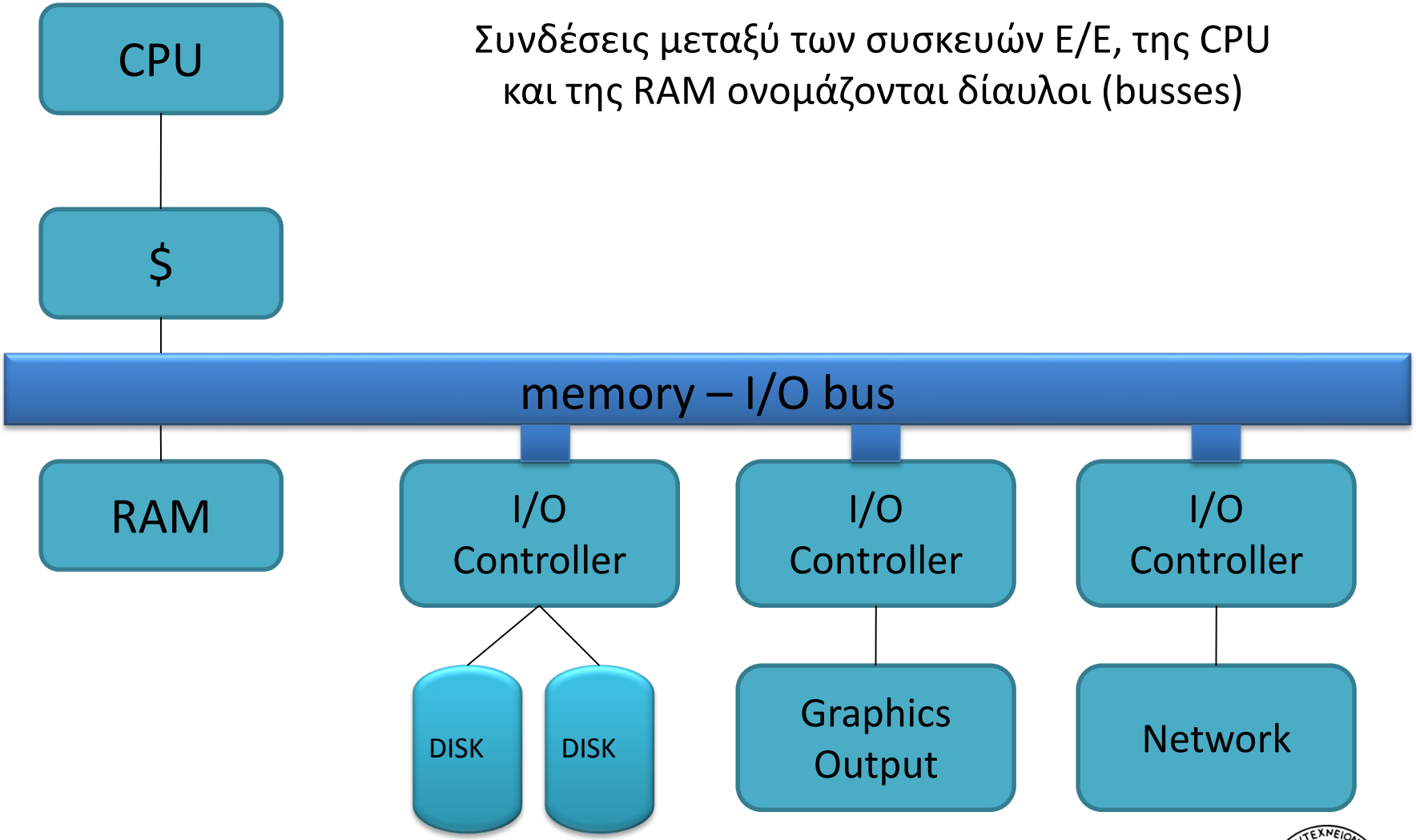
# Συμπερασματικές παρατηρήσεις

- Μέτρα απόδοσης Ε/Ε
  - Ρυθμός διεκπεραίωσης, χρόνος απόκρισης
  - Η φερεγγυότητα και επίσης το κόστος είναι σημαντικά
- Χρησιμοποιούνται δίαυλοι για τη σύνδεση CPU, μνήμη, ελεγκτές Ε/Ε
  - Περιόδευση, διακοπές, DMA
- Μετροπρογράμματα Ε/Ε
  - TPC, SPECSFS, SPECWeb
- RAID
  - Βελτιώνει την απόδοση και τη φερεγγυότητα

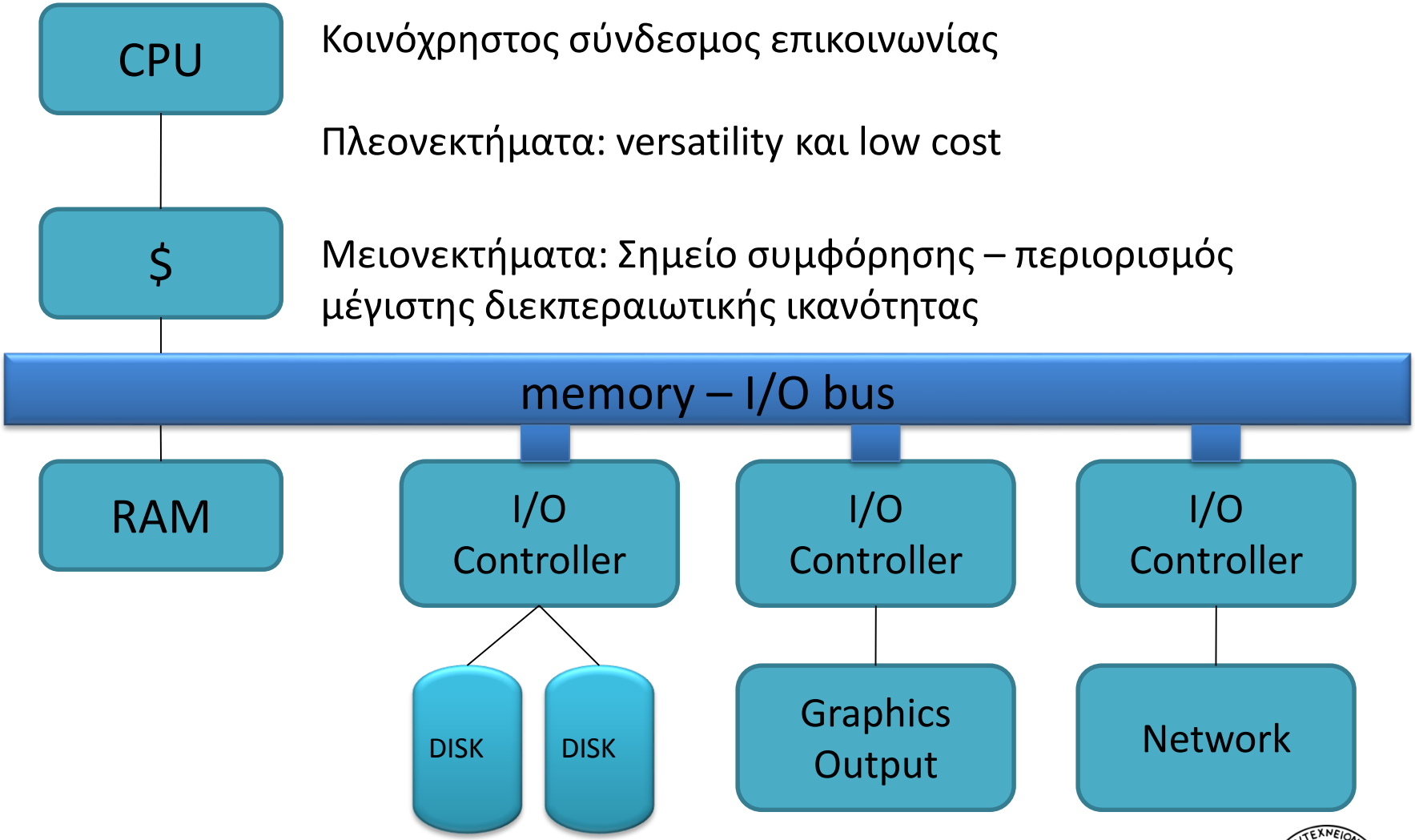
# Επιπλέον Υλικό

# Δίαυλοι

Συνδέσεις μεταξύ των συσκευών Ε/Ε, της CPU και της RAM ονομάζονται δίαυλοι (busses)



# Δίαυλοι

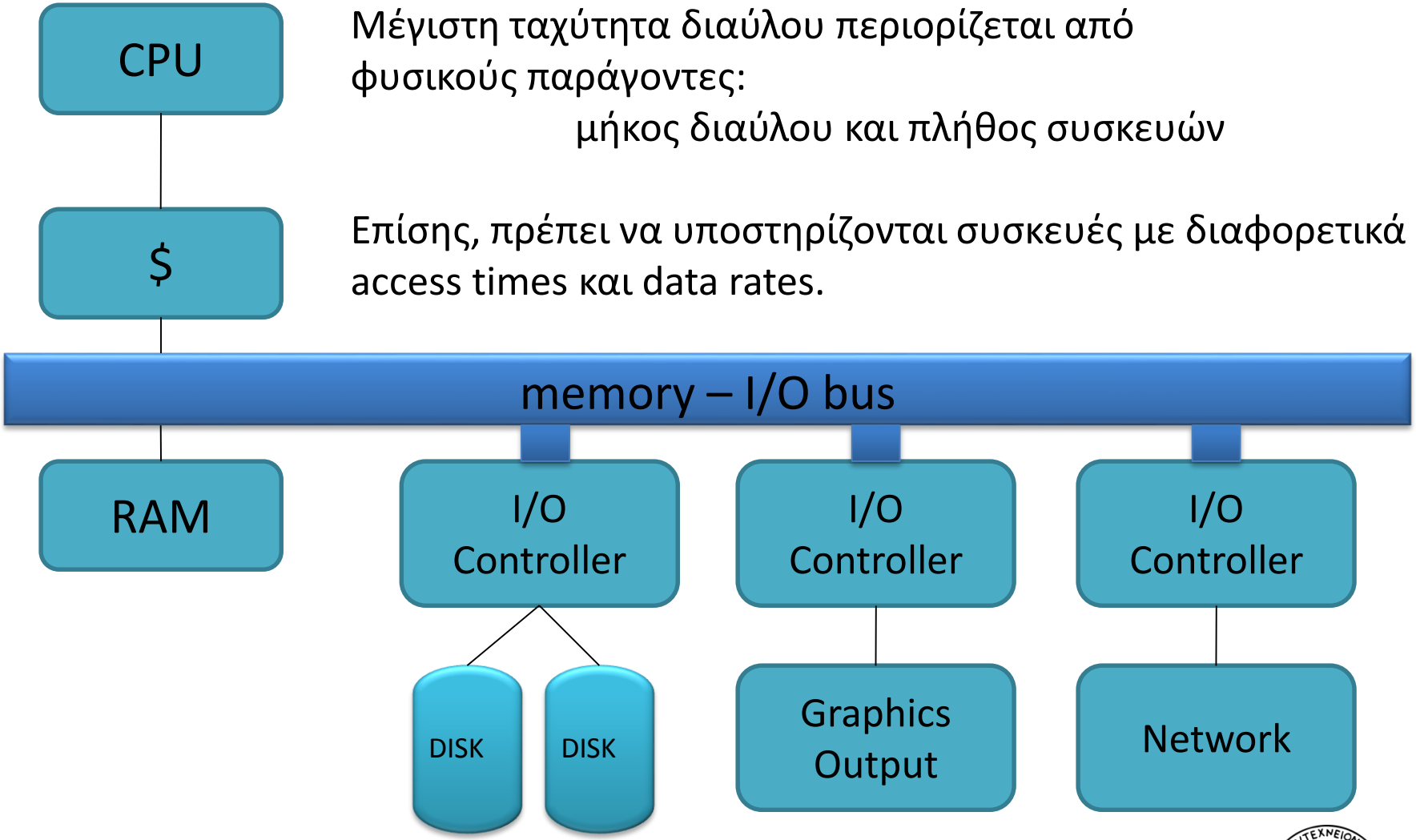



Κοινόχρηστος σύνδεσμος επικοινωνίας

Πλεονεκτήματα: versatility και low cost

Μειονεκτήματα: Σημείο συμφόρησης – περιορισμός μέγιστης διεκπεραιωτικής ικανότητας

# Δίαυλοι




Μέγιστη ταχύτητα διαύλου περιορίζεται από φυσικούς παράγοντες:

μήκος διαύλου και πλήθος συσκευών

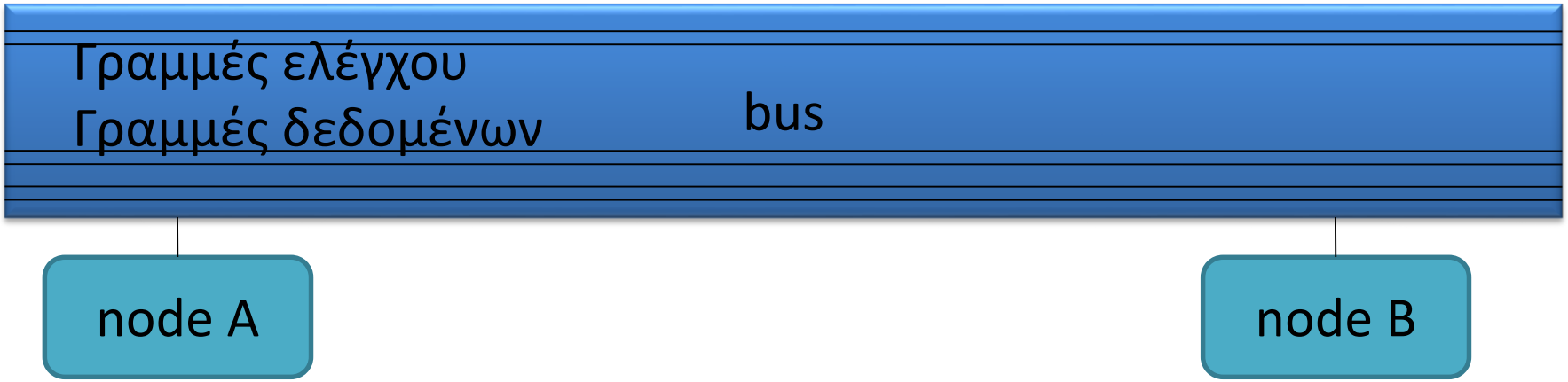
Επίσης, πρέπει να υποστηρίζονται συσκευές με διαφορετικά access times και data rates.

# Βασικά των Διαύλων

**Γραμμές ελέγχου**: Σηματοδοτούν requests και acknowledgements

**Γραμμές δεδομένων**: Μεταφέρουν πληροφορίες (δεδομένα ή διευθύνσεις)

(Πολλές φορές υπάρχουν ξεχωριστές γραμμές διευθύνσεων – address vs data bus)



# Βασικά των Διαύλων

**Bus transaction**: Ακολουθία λειτουργιών διαύλου που περιλαμβάνει αίτηση (+απόκριση) και
μεταφορά δεδομένων

**CPU-memory bus**: Δίαυλος που συνδέει τον επεξεργαστή με τη μνήμη. Μικρό μήκος,
υψηλή ταχύτητα,
μεγιστοποίηση CPU-memory bandwidth

**I/O bus**: Μεγάλο μήκος, πολλές συνδεδεμένες συσκευές, ποικιλία στο εύρος ζώνης δεδομένων συσκευών

# Βασικά των Διαύλων

Οι συσκευές Ε/Ε δεν συνδέονται απευθείας στη μνήμη. Δεδομένα πάνε μέσω του διαύλου CPU-memory (backplane bus).

Λόγω αυξημένων απαιτήσεων, έχουν δημιουργηθεί ειδικοί δίαυλοι (π.χ. γραφικών)

I/O bus χρησιμεύει για την σύνδεση νέων περιφερειακών. Ανάπτυξη προτύπων (π.χ. USB, Firewire) για διασφάλιση λειτουργίας συσκευών σε όλα τα συστήματα.

# Βασικά χαρακτηριστικά USB/Firewire

| Χαρακτηριστικά | Firewire (1394) | USB 2.0 |
|---|---|---|
| Τύπος διαύλου | Εισόδου/Εξόδου | Εισόδου/Εξόδου |
| Βασικό εύρος διαύλου δεδομένων (σήματα) | 4 | 2 |
| Χρονισμός | Ασύγχρονος | Ασύγχρονος |
| Μέγιστο θεωρητικό bandwidth | 50MB/sec (Firewire 400) ή 100MB/sec (Firewire 800) | 0,2MB/sec (χαμηλή ταχύτητα) ή 1,5MB/sec (πλήρης ταχύτητα) ή 60MB/sec (υψηλή ταχύτητα) |
| Hotplug | NAI | NAI |
| Μέγιστος αριθμός συσκευών | 63 | 127 |
| Μέγιστο μήκος διαύλου (χάλκινο καλώδιο) | 4,5μ | 5μ |
| Όνομα προτύπου | IEEE 1394, 1394b | USB Implementers Forum |

# Μέθοδοι επικοινωνίας διαύλων

## Σύγχρονη vs Ασύγχρονη Επικοινωνία

**Σύγχρονη επικοινωνία**: Ρολόι στις γραμμές ελέγχου

**Μειονεκτήματα**: Όλες οι συσκευές πρέπει να έχουν το ίδιο ρολόι – δύσκολο (βλ. clock skew) – εφικτό για μικρούς διαύλους (π.χ. CPU-MEM)

**Πλεονεκτήματα**: Υψηλή Ταχύτητα

# Μέθοδοι επικοινωνίας διαύλων

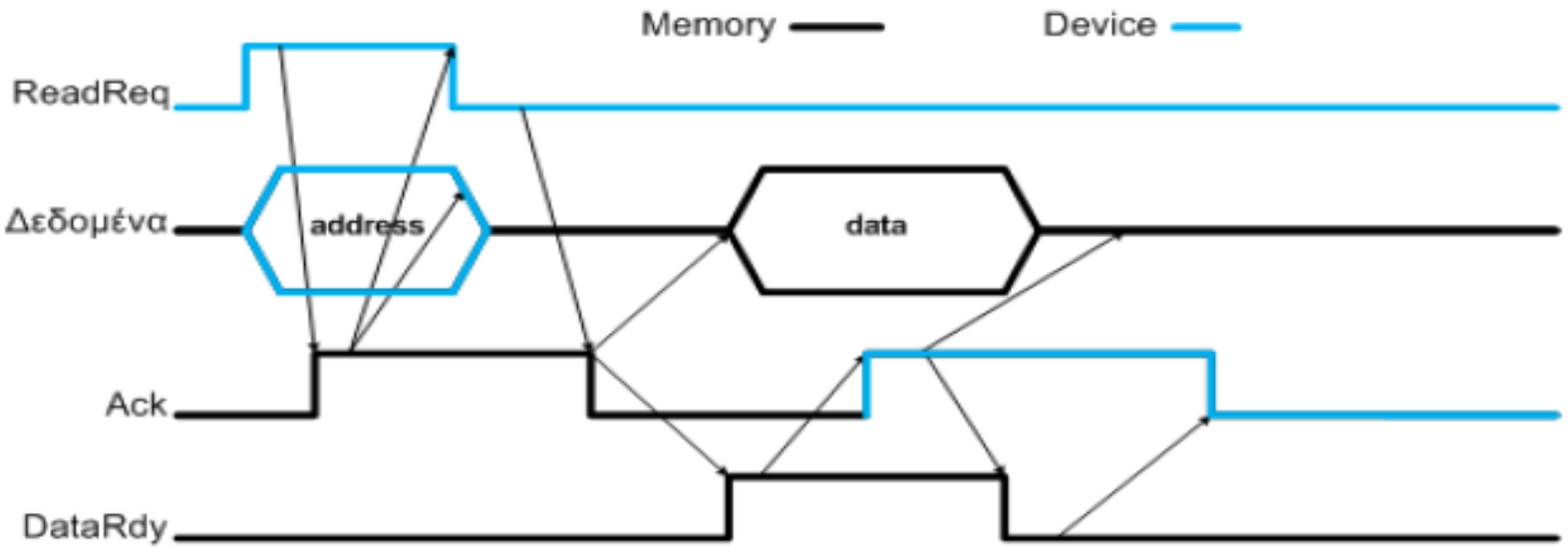
## Σύγχρονη vs Ασύγχρονη Επικοινωνία

**Ασύγχρονη επικοινωνία:** Χωρίς ρολόι

**Πλεονεκτήματα:** Μεγάλη ποικιλία συσκευών, μεγάλο μήκος (USB, Firewire = asynchronous)

**Πλεονεκτήματα:** Χαμηλή Ταχύτητα (σε σχέση με CPU-MEM bus)

# Παράδειγμα ασύγχρονης επικοινωνίας

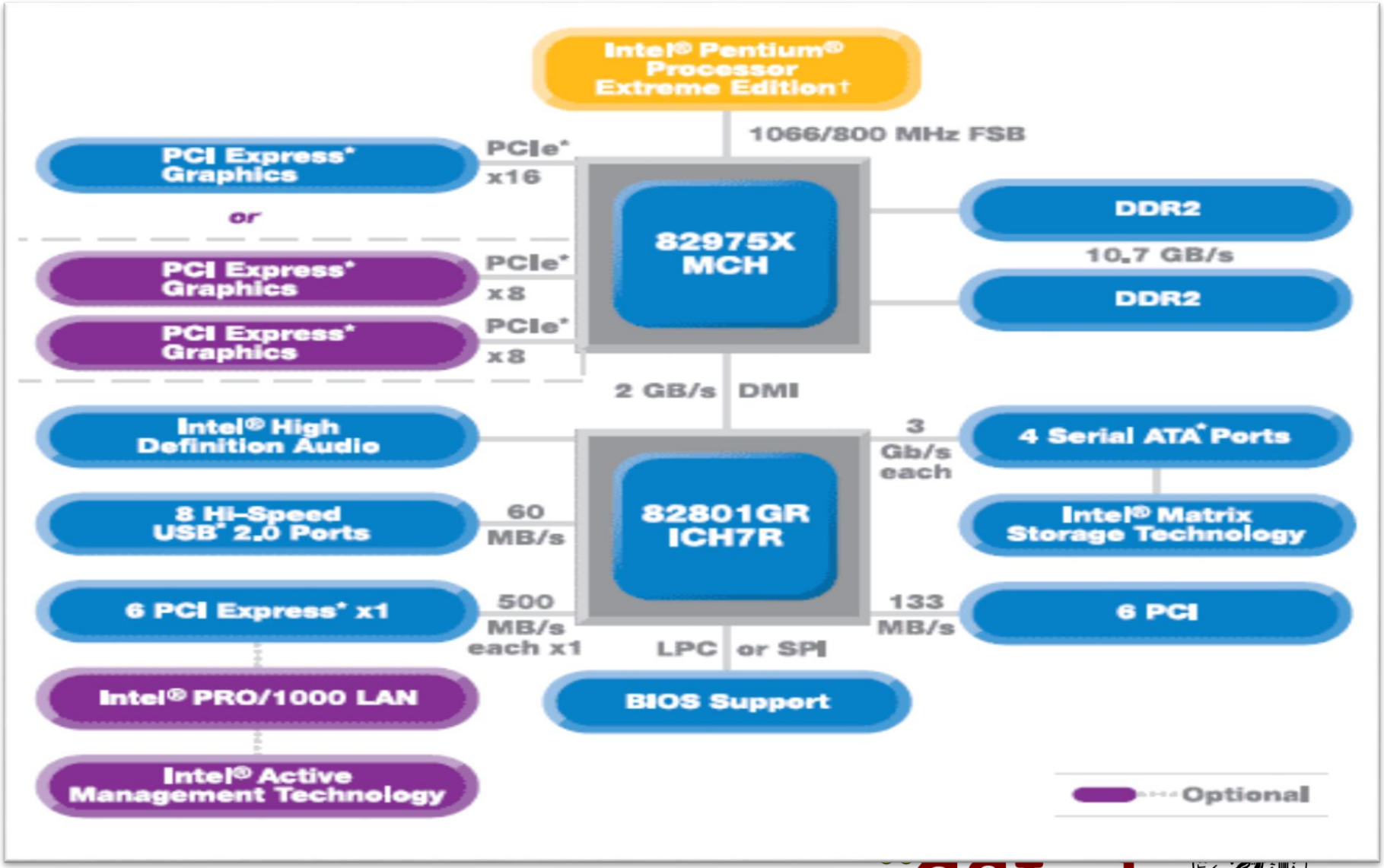


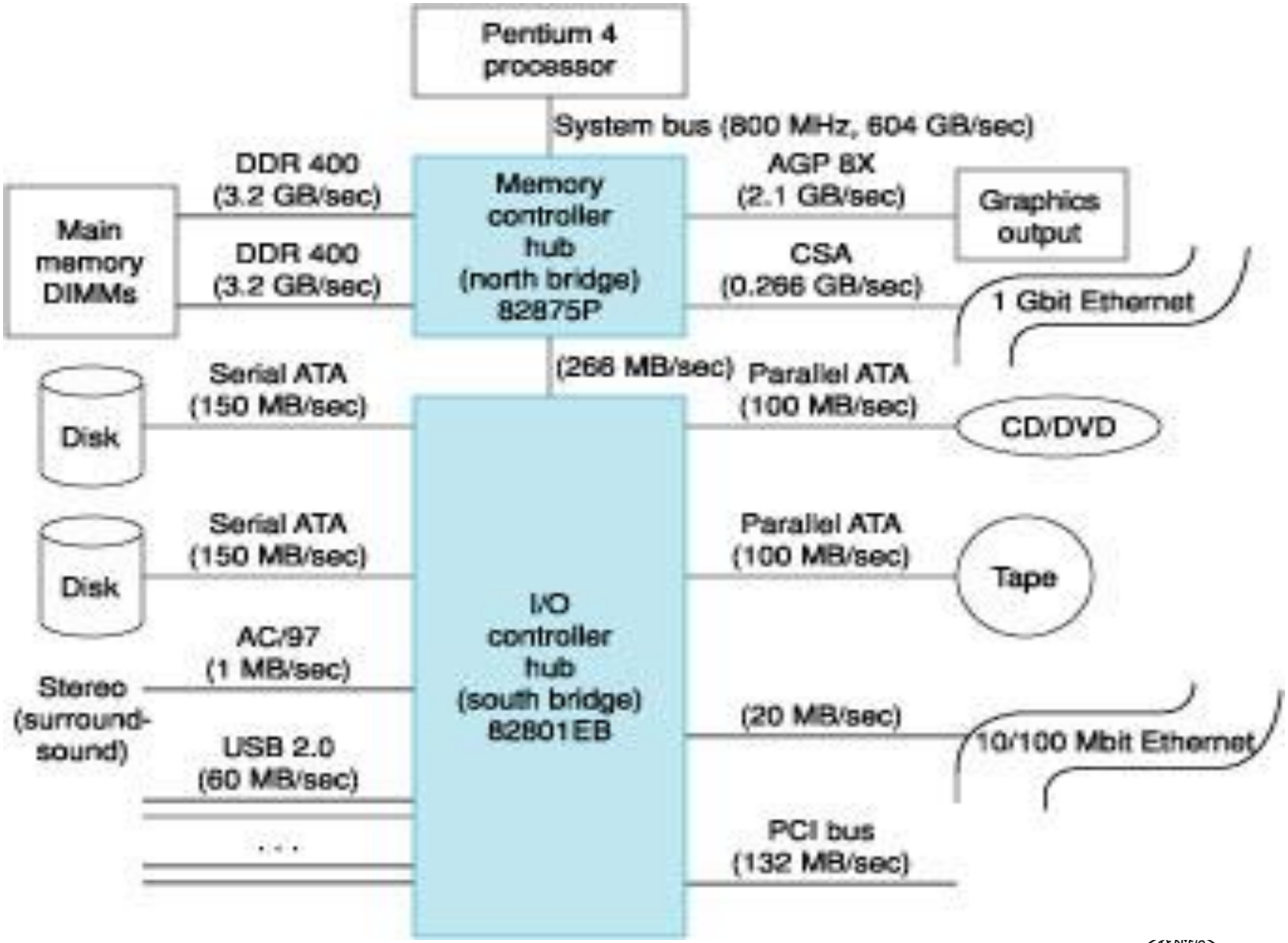

**ReadReq: Δηλώνει αίτηση ανάγνωσης. Διεύθυνση τοποθετείται στη γραμμή δεδομένων.**

**DataRdy: Δηλώνει έγκυρα δεδομένα στη γραμμή δεδομένων**

**Ack: Επιβεβαιώνει το σήμα ReadReq ή DataRdy της άλλης πλευράς**

# Core2Duo busses and interconnects




Pentium 4 processor
System bus (800 MHz, 604 GB/sec)
Main memory DIMMs
DDR 400 (3.2 GB/sec)
DDR 400 (3.2 GB/sec)
Memory controller hub (north bridge) 82875P
AGP 8X (2.1 GB/sec)
Graphics output
CSA (0.266 GB/sec)
1 Gbit Ethernet
(266 MB/sec)
Disk
Serial ATA (150 MB/sec)
Parallel ATA (100 MB/sec)
CD/DVD
Disk
Serial ATA (150 MB/sec)
Parallel ATA (100 MB/sec)
Tape
I/O controller hub (south bridge) 82801EB
AC/97 (1 MB/sec)
Stereo (surround-sound)
USB 2.0 (60 MB/sec)
. . .
(20 MB/sec)
10/100 Mbit Ethernet
PCI bus (132 MB/sec)

# Διαταγές προς συσκευές Ε/Ε

**Για να μιλήσει ο επεξεργαστής με I/O χρησιμοποιούνται 2 μέθοδοι:**

**α) memory mapped I/O**

**π.χ.** Οι διευθύνσεις 0xFFFF0000 – 0xFFFF000F αντιστοιχούν σε 4 command registers 32bit μιας συσκευής I/O. Οι εγγραφές σε αυτές τις διευθύνσεις, αντί για τη μνήμη, γράφουν στους registers.

**β) special Input/Output commands**

# Επικοινωνία με τον επεξεργαστή

Όταν μια συσκευή Ε/Ε ολοκληρώσει μια λειτουργία, πώς ενημερώνει τη CPU;

**α) Polling –** η CPU ελέγχει ανά τακτά χρονικά διαστήματα τη συσκευή για αλλαγές στο state

**β) Interrupt –** η συσκευή ΕΕ στέλνει ένα interrupt (διακοπή) στη CPU για να «τραβήξει» την προσοχή της

# Μεταφορά δεδομένων μεταξύ ΕΕ και μνήμης

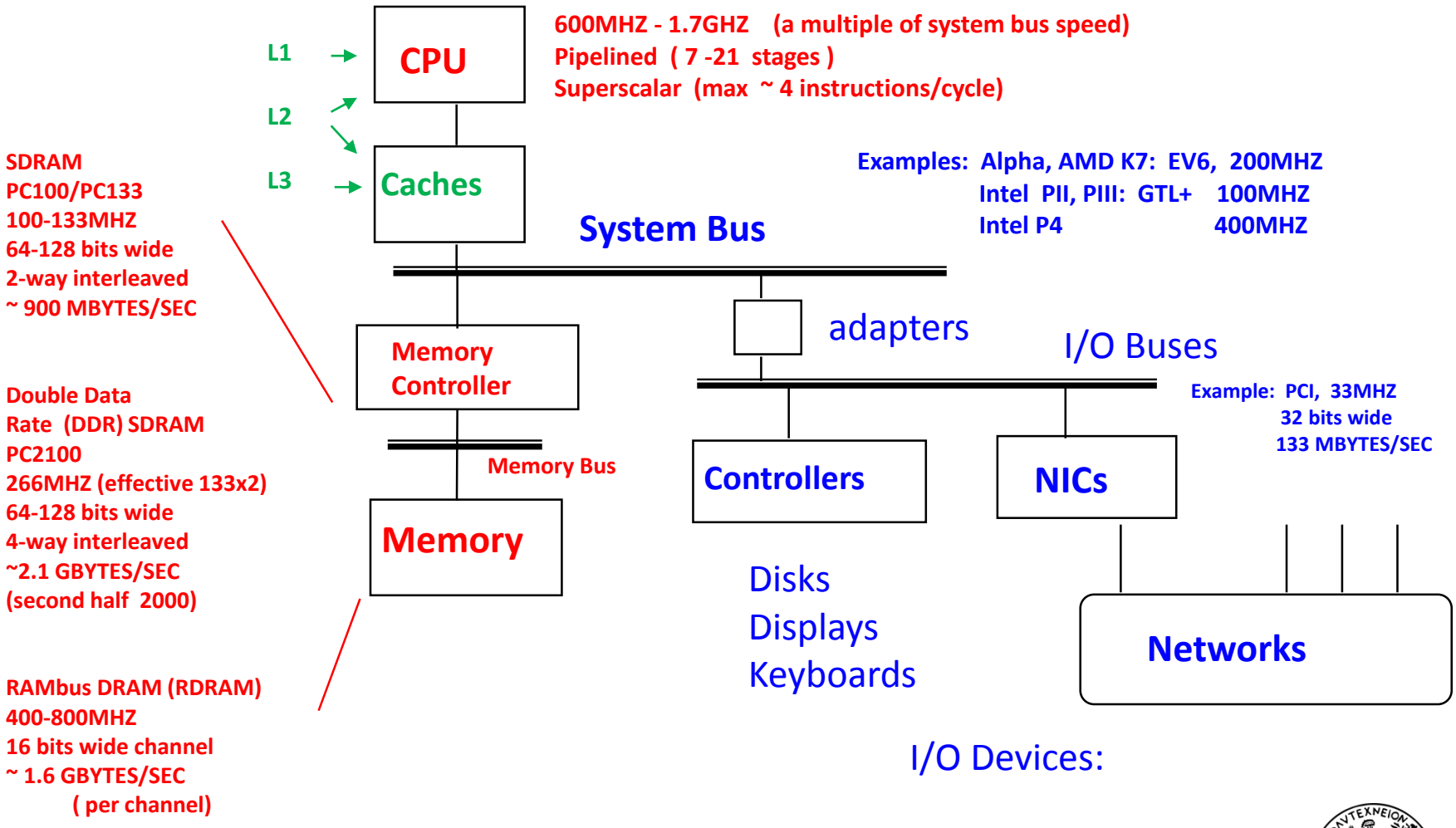
Πώς μεταφέρονται τα δεδομένα από μια συσκευή (π.χ. δίσκο) στη μνήμη του υπολογιστή;

**α) Programmed I/O –** η CPU διαβάζει τα δεδομένα και τα γράφει στη μνήμη (**CPU busy**)

**β) Direct Memory Access –** η CPU προγραμματίζει τη συσκευή DMA μιας συσκευής να γράψει μόνη της τα δεδομένα στη μνήμη (bus master) και όταν ολοκληρώσει να ενημερώσει τη CPU με interrupt (**CPU free**) – προβλήματα

**Η κάθε μονάδα DMA λειτουργεί σαν μια μικρή ειδική CPU για το σκοπό της μεταφοράς δεδομένων απευθείας στη μνήμη χωρίς τη μεσολάβηση του επεξεργαστή**

# Συστατικά ενός Computer System



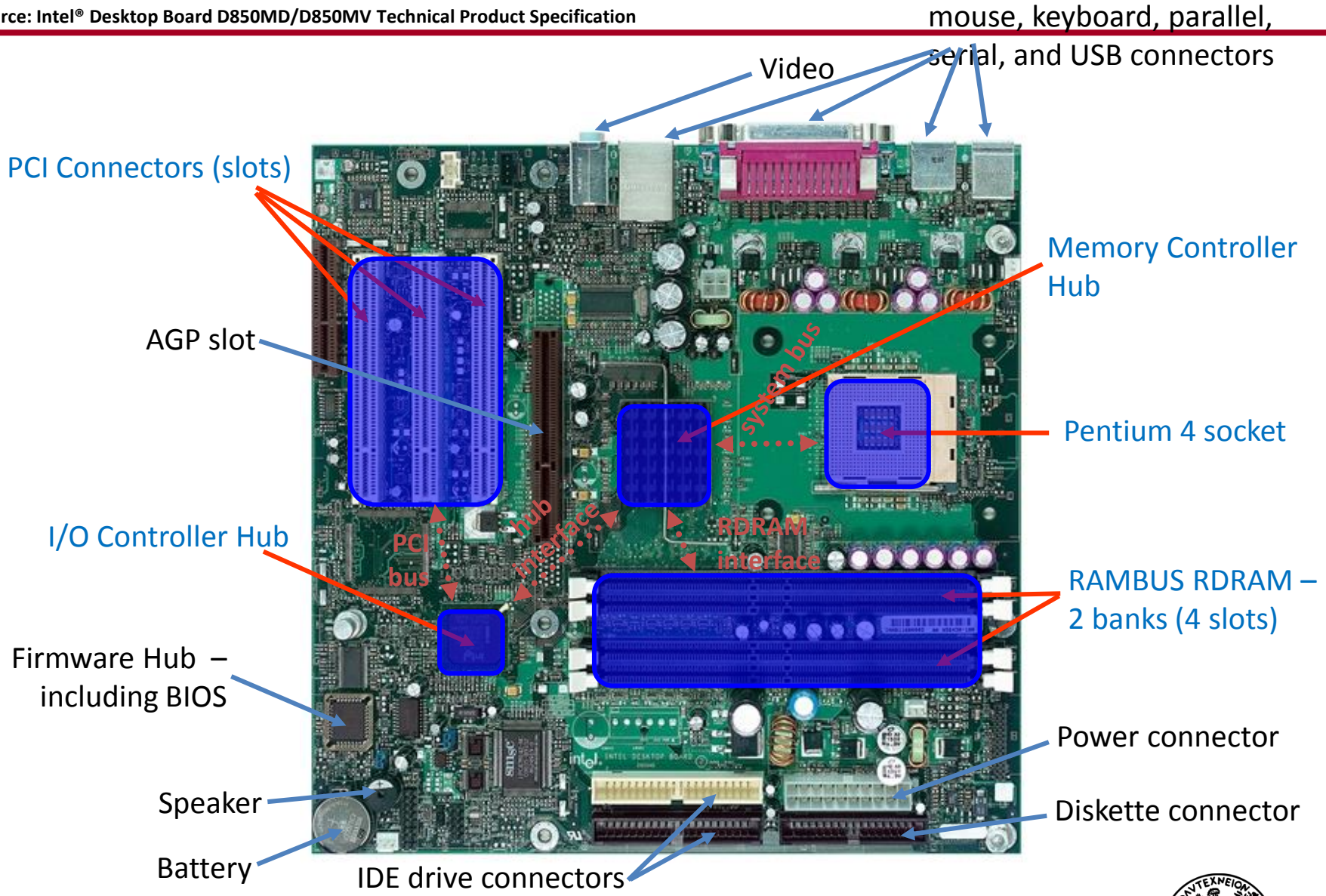
# Intel Hub Architecture (850 Chipset)

Intel D850MD Motherboard:

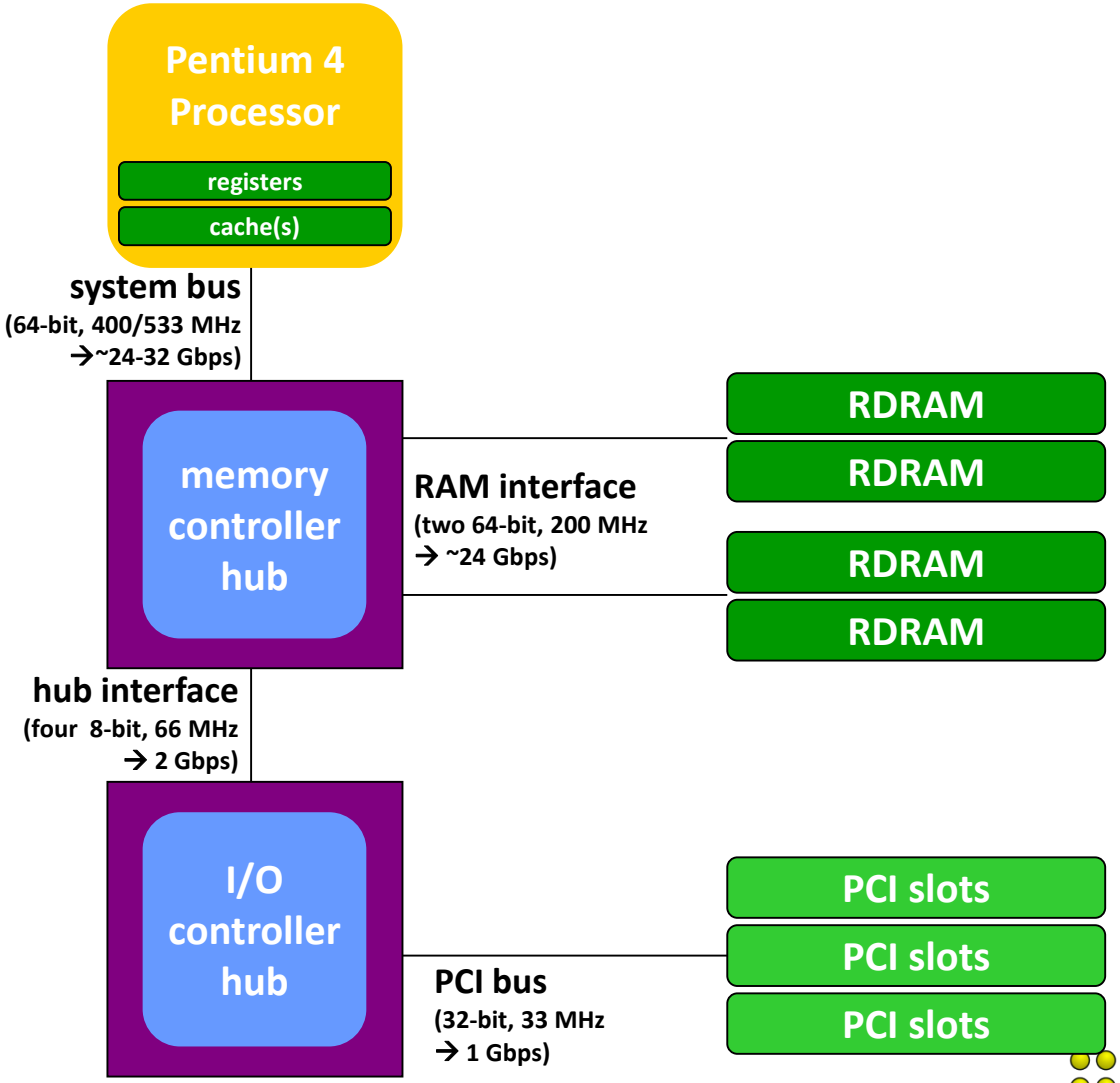
Source: Intel® Desktop Board D850MD/D850MV Technical Product Specification

# Intel D850MD Motherboard:

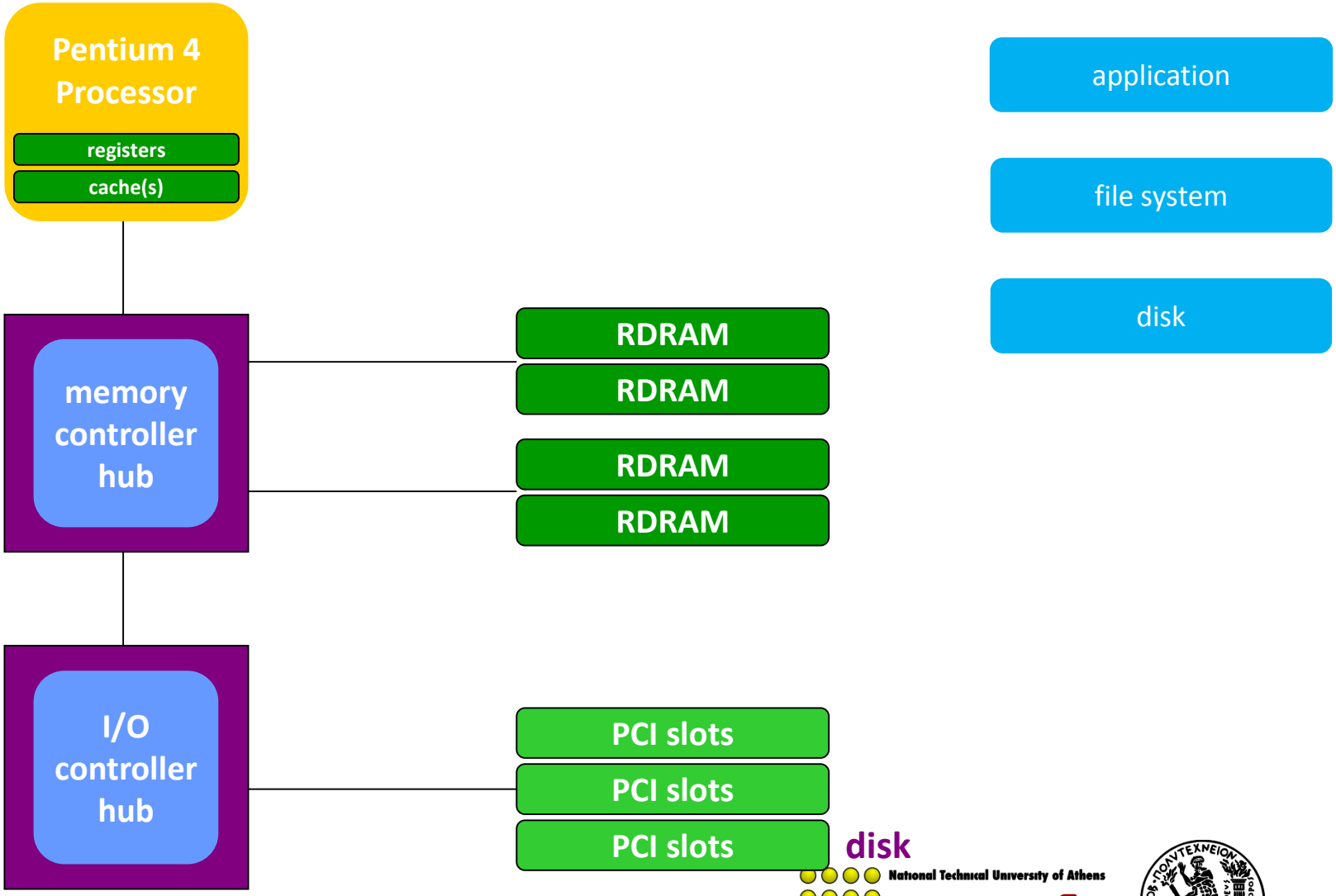
Source: Intel® Desktop Board D850MD/D850MV Technical Product Specification



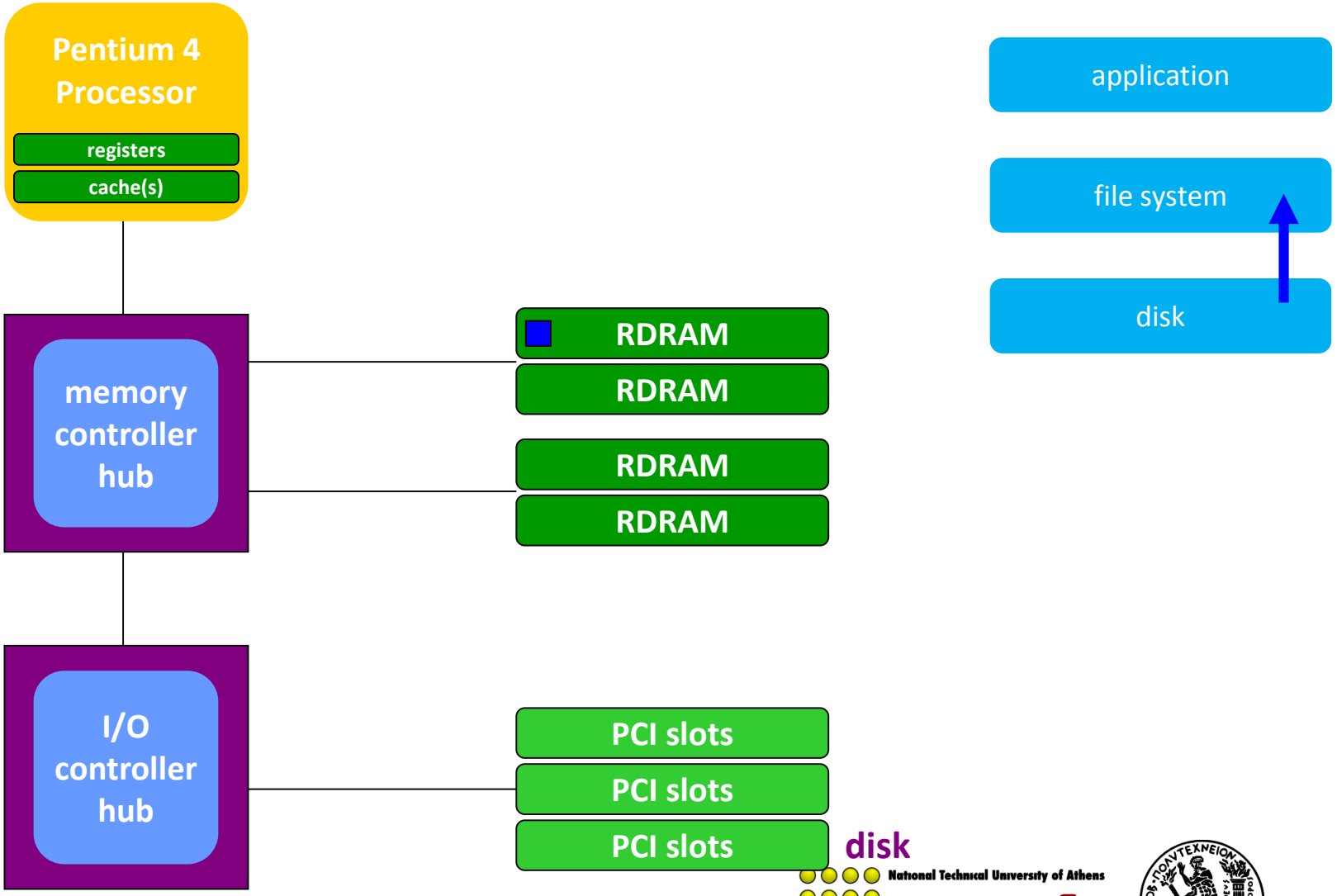
# Example:
# Intel Hub Architecture (850 Chipset)



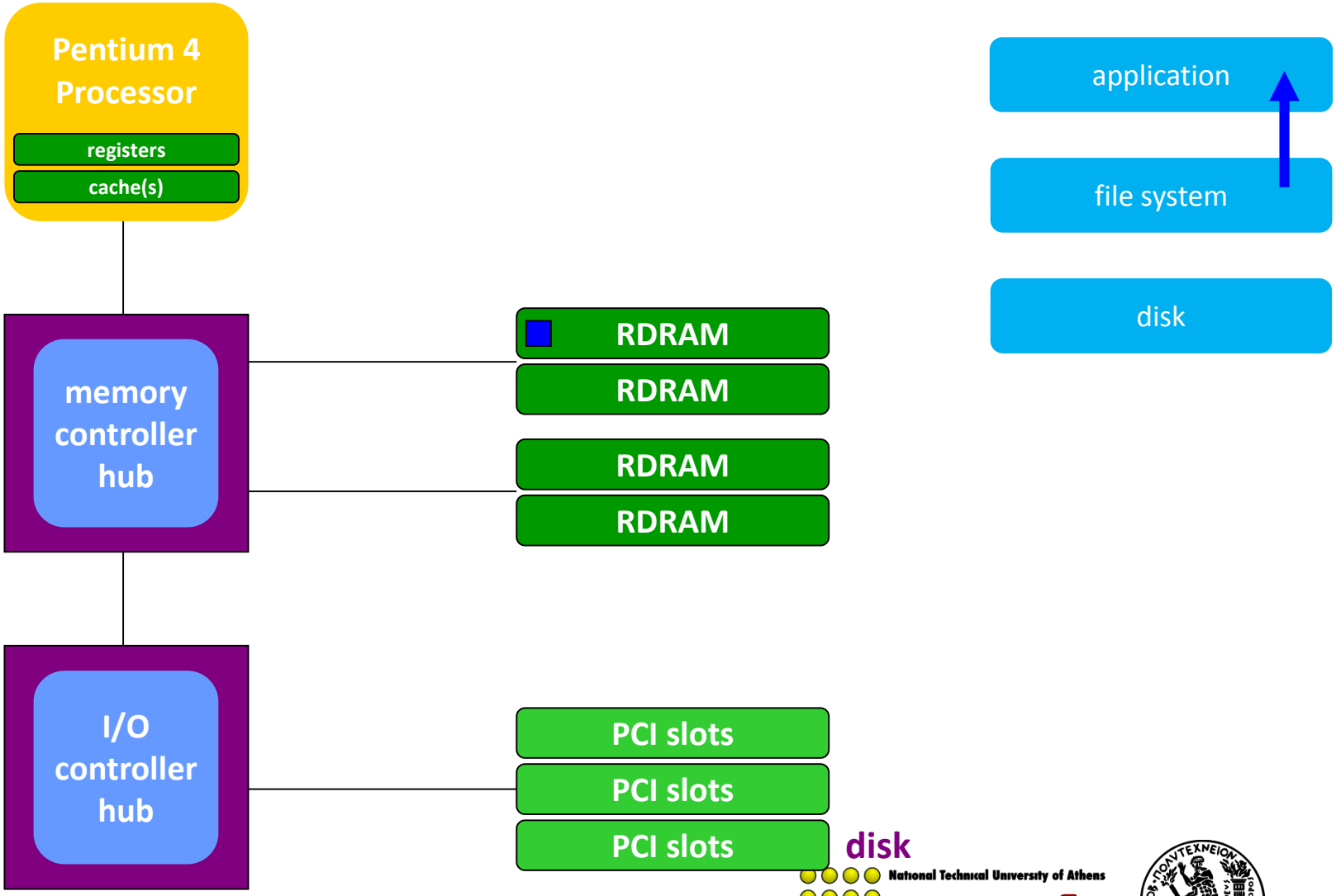
# Example:
# Intel Hub Architecture (850 Chipset)



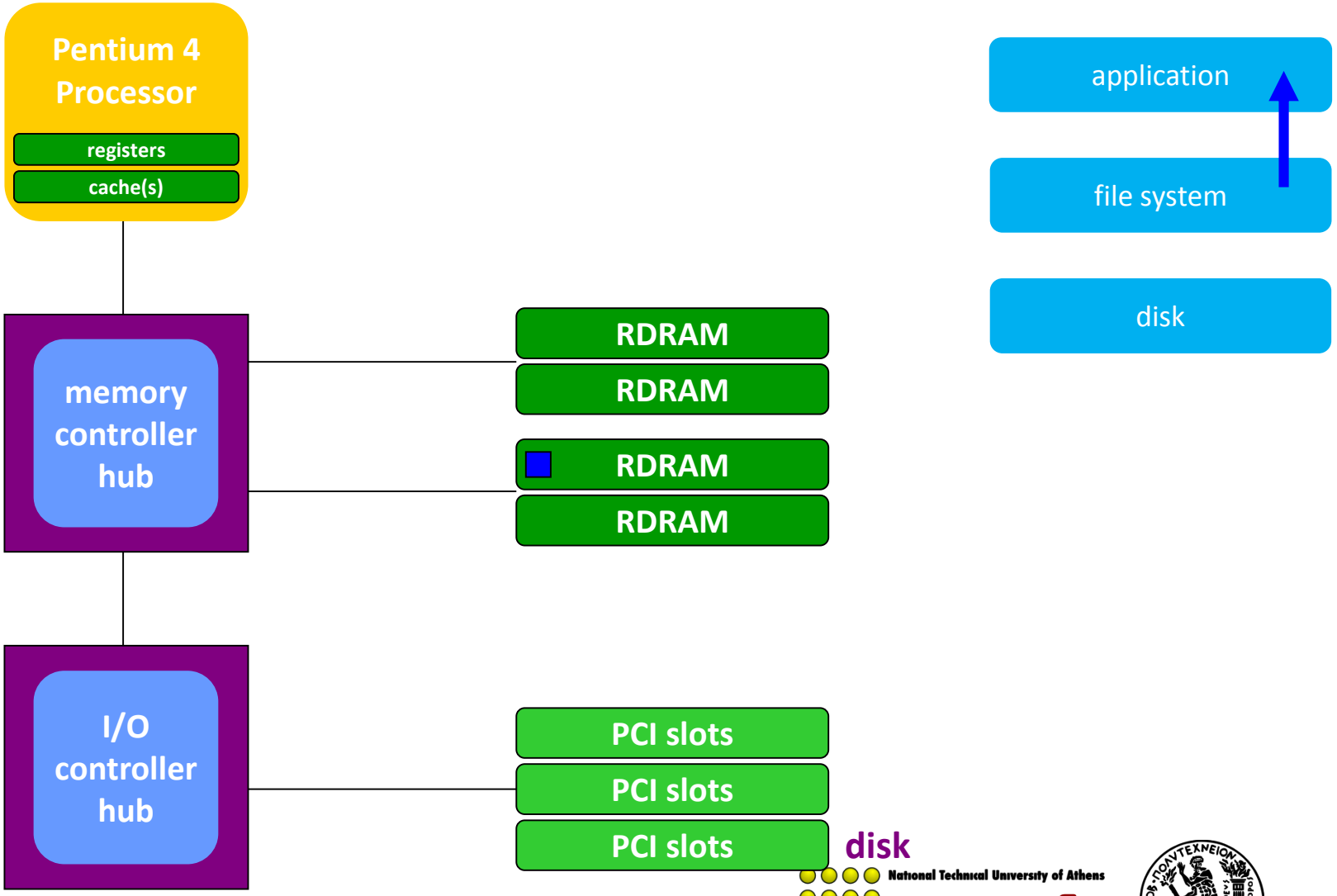
# Example:
# Intel Hub Architecture (850 Chipset)



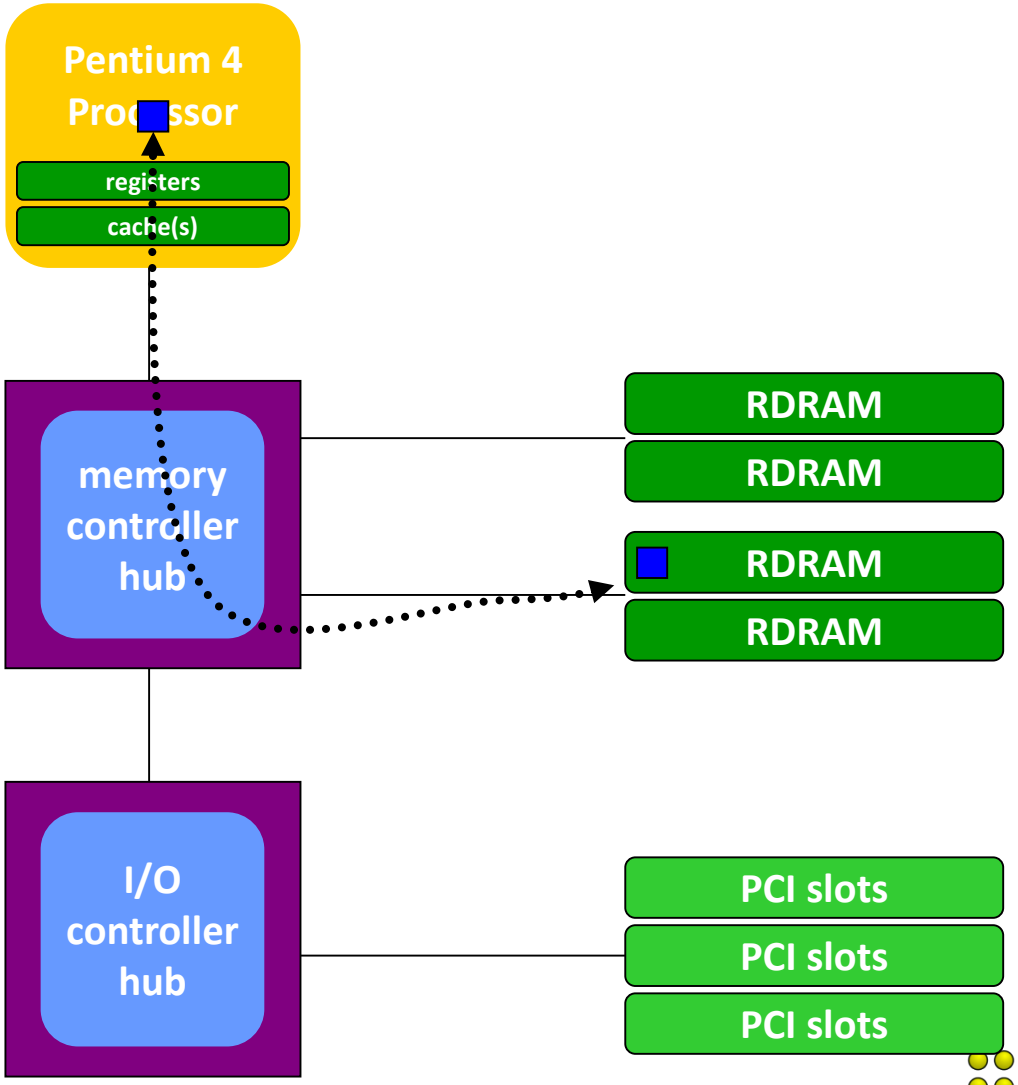
# Example:
# Intel Hub Architecture (850 Chipset)



# Example:
# Intel Hub Architecture (850 Chipset)



# Example:
# Intel Hub Architecture (850 Chipset)




application

file system

disk

# Intel 32-bit Architecture (IA32): Basic Execution Environment

- Address space: $1 – 2^{36}$ (64 GB),
  each program may have a linear address space of 4 GB ($2^{32}$)

  Basic program execution registers:

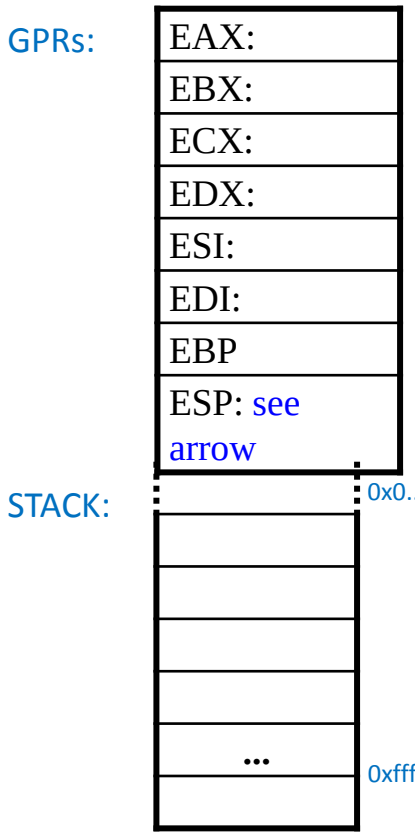
# Intel 32-bit Architecture (IA32): Basic Execution Environment

- Address space: $1 – 2^{36}$ (64 GB),
  each program may have a linear address space of 4 GB ($2^{32}$)

  Basic program execution registers:
  - 8 general purpose registers (EAX, EBX, ECX, EDX, ESI, EDI, EBP and ESP)
  - 6 segment registers (CS, DS, SS, ES, FS and GS)
  - 1 flag register (EFLAGS)
  - 1 instruction pointer register (EIP)

    Stack – a continuous array of memory locations
  - Current stack is referenced by the SS register
  - ESP register – stack pointer
  - EBP register – stack frame base pointer (fixed reference)
  - PUSH – stack grow, add item (ESP decrement)
  - POP – remove item, stack shrinks (ESP increment)

    Several other registers like Control, MMX,
    XMM, FPU, MTRR, MSR and performance monitoring



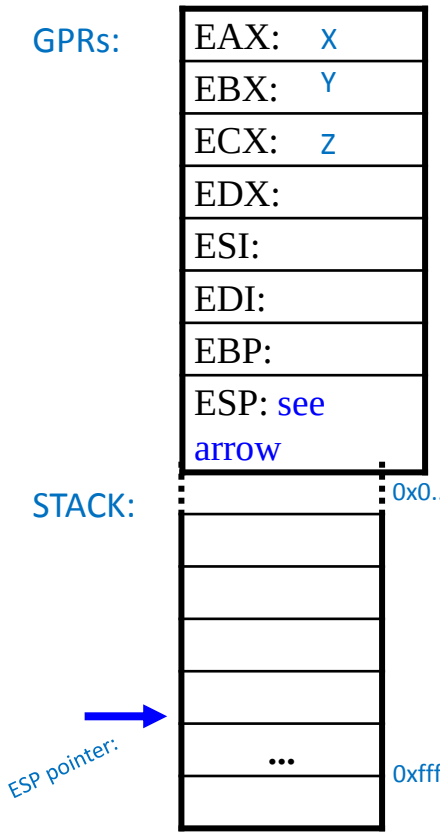
# Intel 32-bit Architecture (IA32): Basic Execution Environment

- Address space: 1 – $2^{36}$ (64 GB),
  each program may have a linear address space of 4 GB ($2^{32}$)

  Basic program execution registers:
  - 8 general purpose registers (EAX, EBX, ECX, EDX, ESI, EDI, EBP and ESP)
  - 6 segment registers (CS, DS, SS, ES, FS and GS)
  - 1 flag register (EFLAGS)
  - 1 instruction pointer register (EIP)

    Stack – a continuous array of memory locations
  - Current stack is referenced by the SS register
  - ESP register – stack pointer
  - EBP register – stack frame base pointer (fixed reference)
  - PUSH – stack grow, add item (ESP decrement)
  - POP – remove item, stack shrinks (ESP increment)

    Several other registers like Control, MMX,
    XMM, FPU, MTRR, MSR and performance monitoring



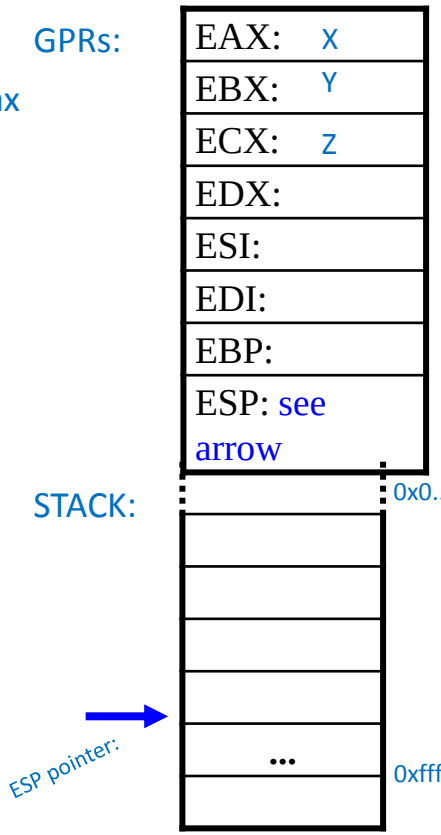
# Intel 32-bit Architecture (IA32): Basic Execution Environment

- Address space: $1 – 2^{36}$ (64 GB),
  each program may have a linear address space of 4 GB ($2^{32}$)

  Basic program execution registers:
  - 8 general purpose registers (EAX, EBX, ECX, EDX, ESI, EDI, EBP and ESP)
  - 6 segment registers (CS, DS, SS, ES, FS and GS)
  - 1 flag register (EFLAGS)
  - 1 instruction pointer register (EIP)

  Stack – a continuous array of memory locations
  - Current stack is referenced by the SS register
  - ESP register – stack pointer
  - EBP register – stack frame base pointer (fixed reference)
  - PUSH – stack grow, add item (ESP decrement)
  - POP – remove item, stack shrinks (ESP increment)

  Several other registers like Control, MMX,
  XMM, FPU, MTRR, MSR and performance monitoring



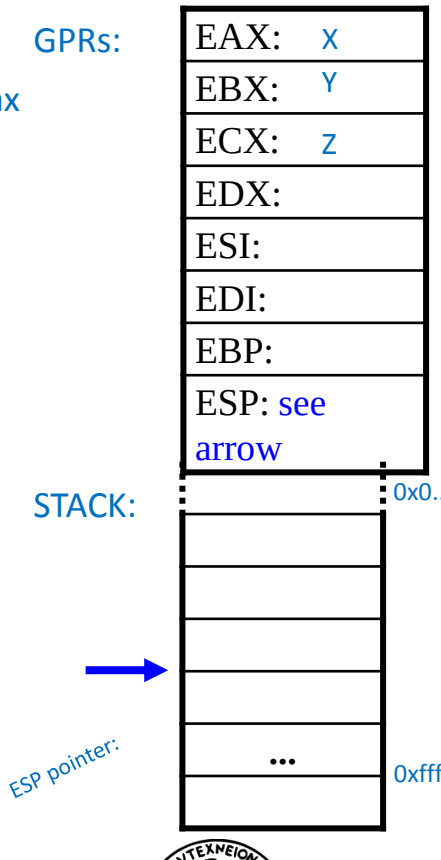
# Intel 32-bit Architecture (IA32): Basic Execution Environment

- Address space: $1 – 2^{36}$ (64 GB),
  each program may have a linear address space of 4 GB ($2^{32}$)

  Basic program execution registers:
  - 8 general purpose registers (EAX, EBX, ECX, EDX, ESI, EDI, EBP and ESP)
  - 6 segment registers (CS, DS, SS, ES, FS and GS)
  - 1 flag register (EFLAGS)
  - 1 instruction pointer register (EIP)

  Stack – a continuous array of memory locations
  - Current stack is referenced by the SS register
  - ESP register – stack pointer
  - EBP register – stack frame base pointer (fixed reference)
  - PUSH – stack grow, add item (ESP decrement)
  - POP – remove item, stack shrinks (ESP increment)

  Several other registers like Control, MMX,
  XMM, FPU, MTRR, MSR and performance monitoring



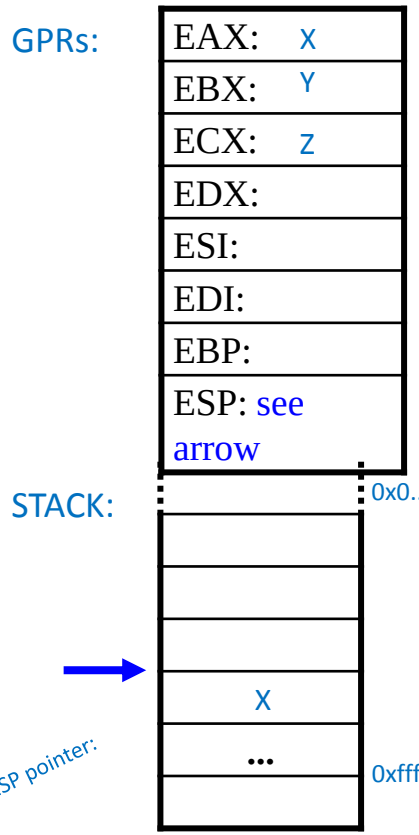
# Intel 32-bit Architecture (IA32): Basic Execution Environment

- Address space: $1 – 2^{36}$ (64 GB),
  each program may have a linear address space of 4 GB ($2^{32}$)

  Basic program execution registers:
  - 8 general purpose registers (EAX, EBX, ECX, EDX, ESI, EDI, EBP and ESP)
  - 6 segment registers (CS, DS, SS, ES, FS and GS)
  - 1 flag register (EFLAGS)
  - 1 instruction pointer register (EIP)

  Stack – a continuous array of memory locations
  - Current stack is referenced by the SS register
  - ESP register – stack pointer
  - EBP register – stack frame base pointer (fixed reference)
  - PUSH – stack grow, add item (ESP decrement)
  - POP – remove item, stack shrinks (ESP increment)

  Several other registers like Control, MMX,
  XMM, FPU, MTRR, MSR and performance monitoring



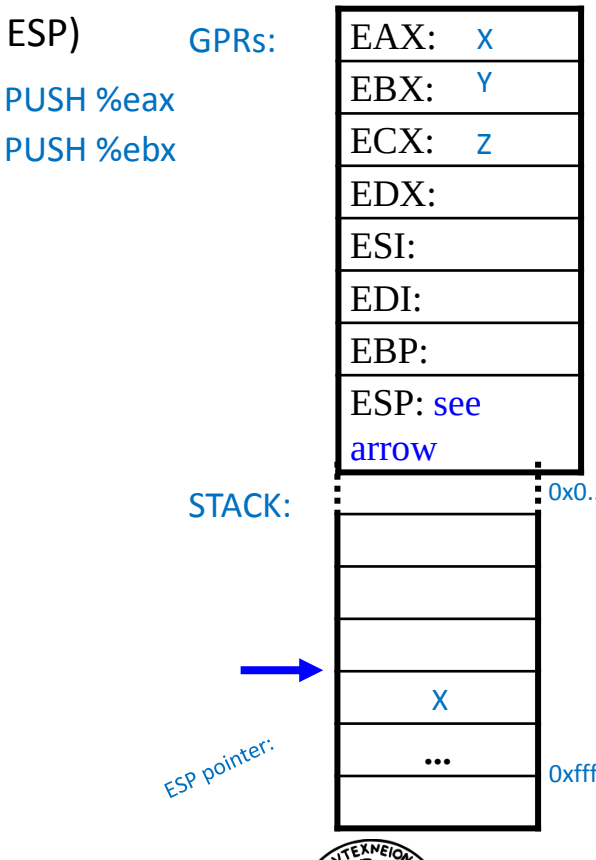
# Intel 32-bit Architecture (IA32): Basic Execution Environment

- Address space: $1 – 2^{36}$ (64 GB),
  each program may have a linear address space of 4 GB ($2^{32}$)

  Basic program execution registers:
  - 8 general purpose registers (EAX, EBX, ECX, EDX, ESI, EDI, EBP and ESP)
  - 6 segment registers (CS, DS, SS, ES, FS and GS)
  - 1 flag register (EFLAGS)
  - 1 instruction pointer register (EIP)

  Stack – a continuous array of memory locations
  - Current stack is referenced by the SS register
  - ESP register – stack pointer
  - EBP register – stack frame base pointer (fixed reference)
  - PUSH – stack grow, add item (ESP decrement)
  - POP – remove item, stack shrinks (ESP increment)

  Several other registers like Control, MMX,
  XMM, FPU, MTRR, MSR and performance monitoring



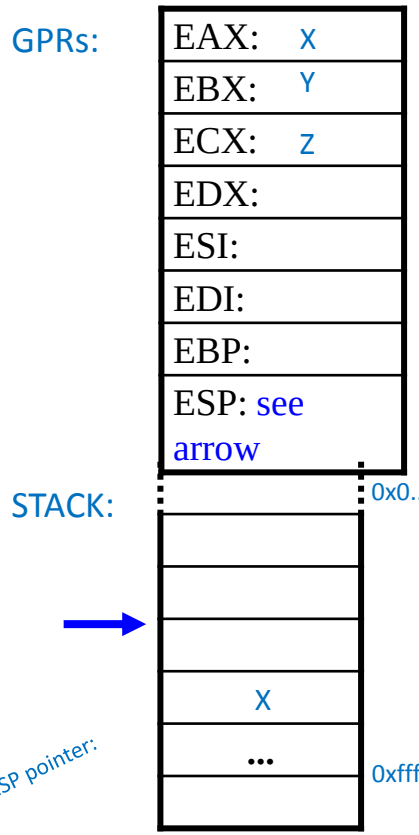
# Intel 32-bit Architecture (IA32): Basic Execution Environment

- Address space: $1 – 2^{36}$ (64 GB),
  each program may have a linear address space of 4 GB ($2^{32}$)

  Basic program execution registers:
  - 8 general purpose registers (EAX, EBX, ECX, EDX, ESI, EDI, EBP and ESP)
  - 6 segment registers (CS, DS, SS, ES, FS and GS)
  - 1 flag register (EFLAGS)
  - 1 instruction pointer register (EIP)

  Stack – a continuous array of memory locations
  - Current stack is referenced by the SS register
  - ESP register – stack pointer
  - EBP register – stack frame base pointer (fixed reference)
  - PUSH – stack grow, add item (ESP decrement)
  - POP – remove item, stack shrinks (ESP increment)

  Several other registers like Control, MMX,
  XMM, FPU, MTRR, MSR and performance monitoring



# Intel 32-bit Architecture (IA32): Basic Execution Environment

- Address space: $1 – 2^{36}$ (64 GB),
  each program may have a linear address space of 4 GB ($2^{32}$)

Basic program execution registers:

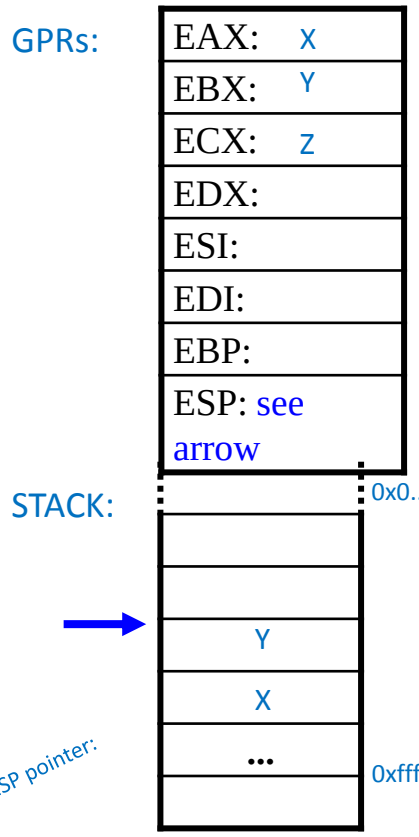
- 8 general purpose registers (EAX, EBX, ECX, EDX, ESI, EDI, EBP and ESP)
- 6 segment registers (CS, DS, SS, ES, FS and GS)
- 1 flag register (EFLAGS)
- 1 instruction pointer register (EIP)

Stack – a continuous array of memory locations

- Current stack is referenced by the SS register
- ESP register – stack pointer
- EBP register – stack frame base pointer (fixed reference)
- PUSH – stack grow, add item (ESP decrement)
- POP – remove item, stack shrinks (ESP increment)

Several other registers like Control, MMX,
XMM, FPU, MTRR, MSR and performance monitoring

PUSH %eax
PUSH %ebx



# Intel 32-bit Architecture (IA32): Basic Execution Environment

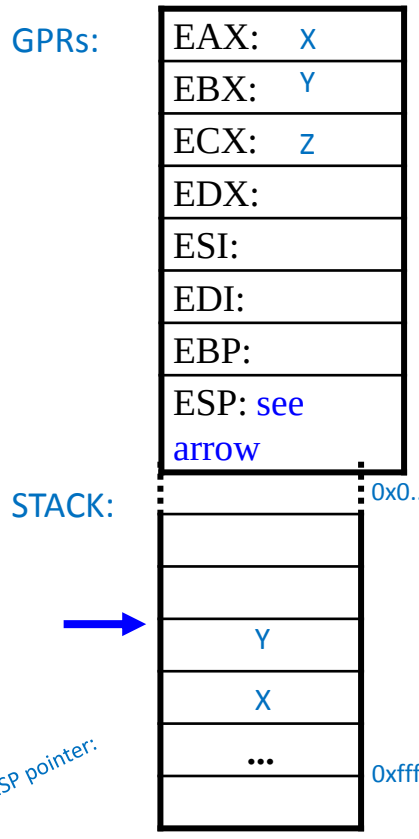
- Address space: $1 – 2^{36}$ (64 GB),
  each program may have a linear address space of 4 GB ($2^{32}$)

  Basic program execution registers:
  - 8 general purpose registers (EAX, EBX, ECX, EDX, ESI, EDI, EBP and ESP)
  - 6 segment registers (CS, DS, SS, ES, FS and GS)
  - 1 flag register (EFLAGS)
  - 1 instruction pointer register (EIP)

  Stack – a continuous array of memory locations
  - Current stack is referenced by the SS register
  - ESP register – stack pointer
  - EBP register – stack frame base pointer (fixed reference)
  - PUSH – stack grow, add item (ESP decrement)
  - POP – remove item, stack shrinks (ESP increment)

  Several other registers like Control, MMX,
  XMM, FPU, MTRR, MSR and performance monitoring

PUSH %eax
PUSH %ebx
PUSH %ecx



# Intel 32-bit Architecture (IA32): Basic Execution Environment

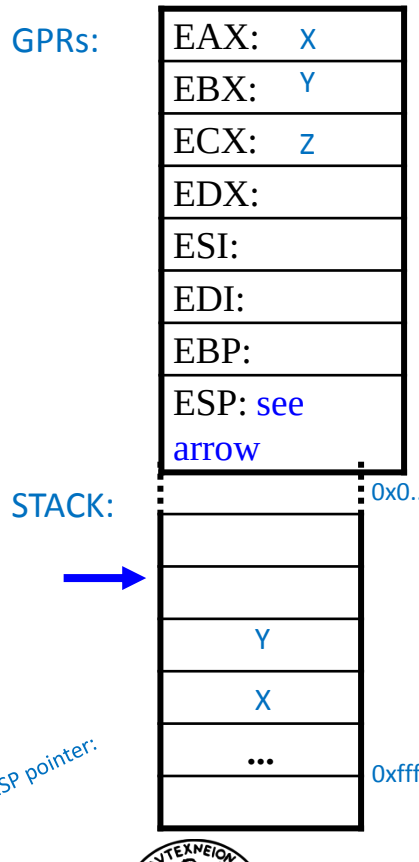
- Address space: $1 – 2^{36}$ (64 GB),
  each program may have a linear address space of 4 GB ($2^{32}$)

  Basic program execution registers:
  - 8 general purpose registers (EAX, EBX, ECX, EDX, ESI, EDI, EBP and ESP)
  - 6 segment registers (CS, DS, SS, ES, FS and GS)
  - 1 flag register (EFLAGS)
  - 1 instruction pointer register (EIP)

  Stack – a continuous array of memory locations
  - Current stack is referenced by the SS register
  - ESP register – stack pointer
  - EBP register – stack frame base pointer (fixed reference)
  - PUSH – stack grow, add item (ESP decrement)
  - POP – remove item, stack shrinks (ESP increment)

  Several other registers like Control, MMX,
  XMM, FPU, MTRR, MSR and performance monitoring

PUSH %eax
PUSH %ebx
PUSH %ecx



# Intel 32-bit Architecture (IA32): Basic Execution Environment

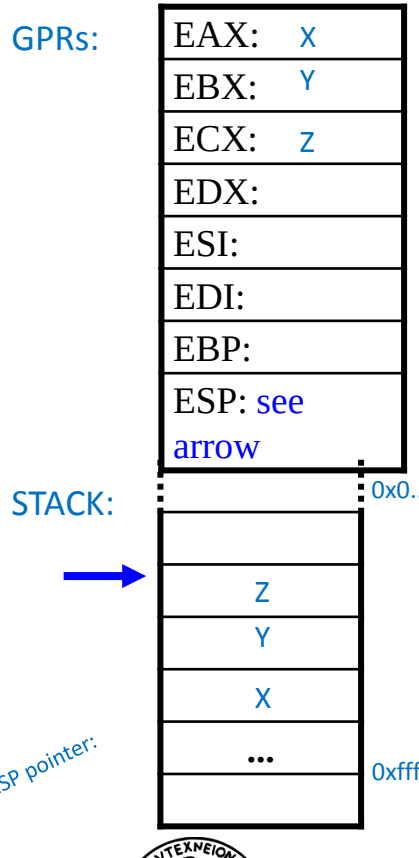
- Address space: $1 – 2^{36}$ (64 GB),
  each program may have a linear address space of 4 GB ($2^{32}$)

  Basic program execution registers:
  - 8 general purpose registers (EAX, EBX, ECX, EDX, ESI, EDI, EBP and ESP)
  - 6 segment registers (CS, DS, SS, ES, FS and GS)
  - 1 flag register (EFLAGS)
  - 1 instruction pointer register (EIP)

    Stack – a continuous array of memory locations
  - Current stack is referenced by the SS register
  - ESP register – stack pointer
  - EBP register – stack frame base pointer (fixed reference)
  - PUSH – stack grow, add item (ESP decrement)
  - POP – remove item, stack shrinks (ESP increment)

    Several other registers like Control, MMX,
    XMM, FPU, MTRR, MSR and performance monitoring

PUSH %eax
PUSH %ebx
PUSH %ecx



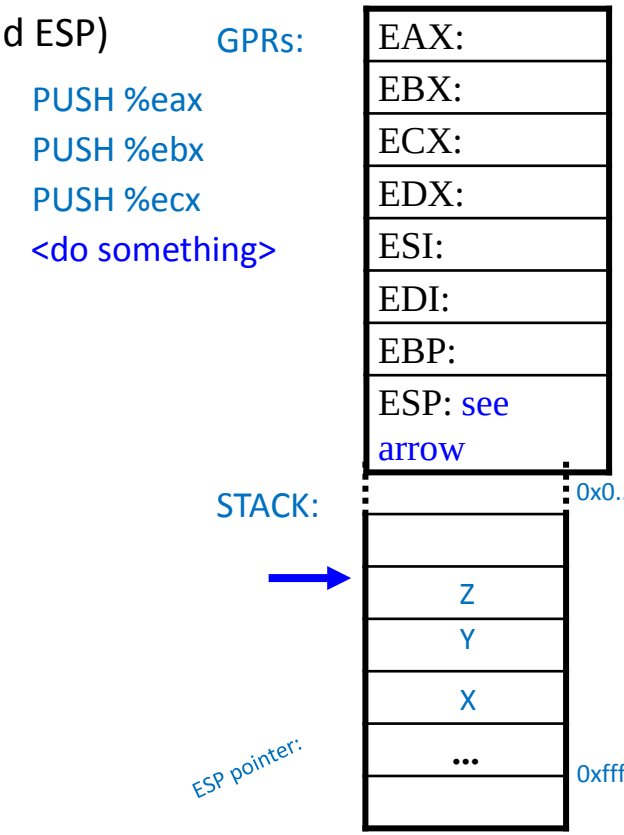
# Intel 32-bit Architecture (IA32): Basic Execution Environment

- Address space: $1 – 2^{36}$ (64 GB),
  each program may have a linear address space of 4 GB ($2^{32}$)

  Basic program execution registers:
  - 8 general purpose registers (EAX, EBX, ECX, EDX, ESI, EDI, EBP and ESP)
  - 6 segment registers (CS, DS, SS, ES, FS and GS)
  - 1 flag register (EFLAGS)
  - 1 instruction pointer register (EIP)

    Stack – a continuous array of memory locations
  - Current stack is referenced by the SS register
  - ESP register – stack pointer
  - EBP register – stack frame base pointer (fixed reference)
  - PUSH – stack grow, add item (ESP decrement)
  - POP – remove item, stack shrinks (ESP increment)

    Several other registers like Control, MMX,
    XMM, FPU, MTRR, MSR and performance monitoring



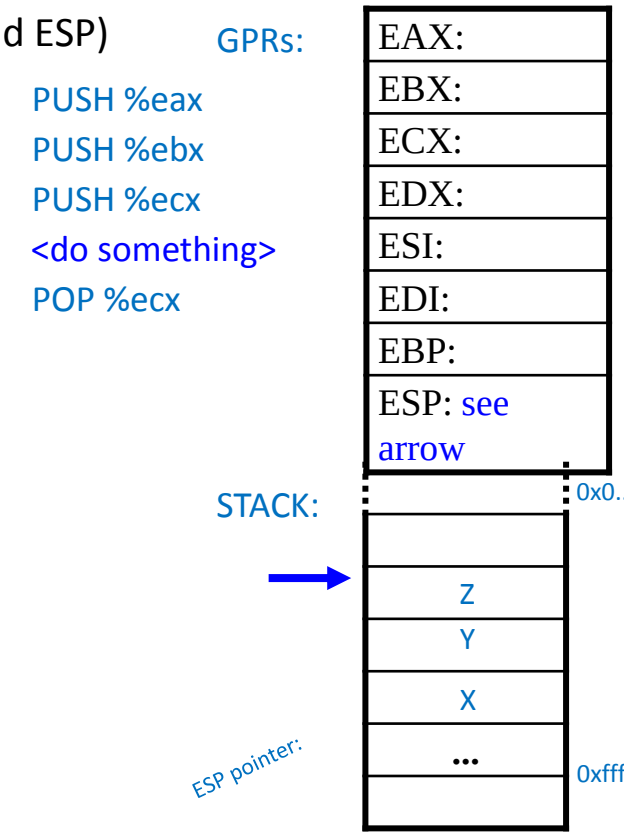
# Intel 32-bit Architecture (IA32): Basic Execution Environment

- Address space: 1 – $2^{36}$ (64 GB),
  each program may have a linear address space of 4 GB ($2^{32}$)

  Basic program execution registers:
  - 8 general purpose registers (EAX, EBX, ECX, EDX, ESI, EDI, EBP and ESP)
  - 6 segment registers (CS, DS, SS, ES, FS and GS)
  - 1 flag register (EFLAGS)
  - 1 instruction pointer register (EIP)

    Stack – a continuous array of memory locations
  - Current stack is referenced by the SS register
  - ESP register – stack pointer
  - EBP register – stack frame base pointer (fixed reference)
  - PUSH – stack grow, add item (ESP decrement)
  - POP – remove item, stack shrinks (ESP increment)

    Several other registers like Control, MMX,
    XMM, FPU, MTRR, MSR and performance monitoring



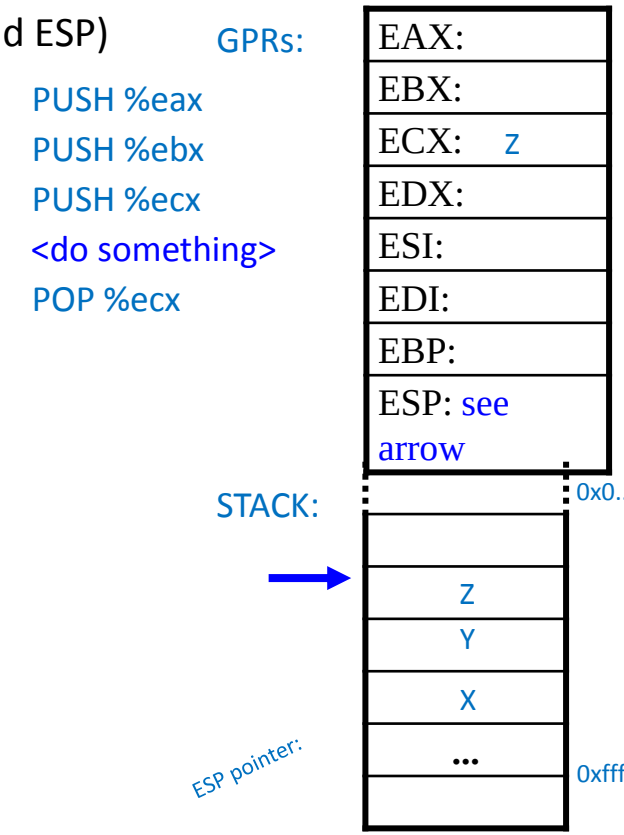
# Intel 32-bit Architecture (IA32): Basic Execution Environment

- Address space: $1 – 2^{36}$ (64 GB),
  each program may have a linear address space of 4 GB ($2^{32}$)

  Basic program execution registers:
  - 8 general purpose registers (EAX, EBX, ECX, EDX, ESI, EDI, EBP and ESP)
  - 6 segment registers (CS, DS, SS, ES, FS and GS)
  - 1 flag register (EFLAGS)
  - 1 instruction pointer register (EIP)

    Stack – a continuous array of memory locations
  - Current stack is referenced by the SS register
  - ESP register – stack pointer
  - EBP register – stack frame base pointer (fixed reference)
  - PUSH – stack grow, add item (ESP decrement)
  - POP – remove item, stack shrinks (ESP increment)

    Several other registers like Control, MMX,
    XMM, FPU, MTRR, MSR and performance monitoring



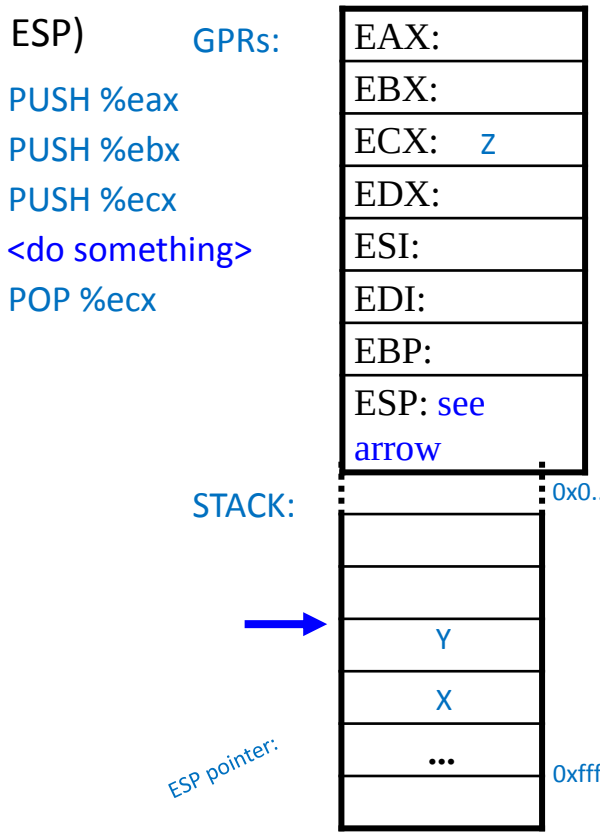
# Intel 32-bit Architecture (IA32): Basic Execution Environment

- Address space: $1 – 2^{36}$ (64 GB),
  each program may have a linear address space of 4 GB ($2^{32}$)

  Basic program execution registers:
  - 8 general purpose registers (EAX, EBX, ECX, EDX, ESI, EDI, EBP and ESP)
  - 6 segment registers (CS, DS, SS, ES, FS and GS)
  - 1 flag register (EFLAGS)
  - 1 instruction pointer register (EIP)

    Stack – a continuous array of memory locations
  - Current stack is referenced by the SS register
  - ESP register – stack pointer
  - EBP register – stack frame base pointer (fixed reference)
  - PUSH – stack grow, add item (ESP decrement)
  - POP – remove item, stack shrinks (ESP increment)

    Several other registers like Control, MMX,
    XMM, FPU, MTRR, MSR and performance monitoring



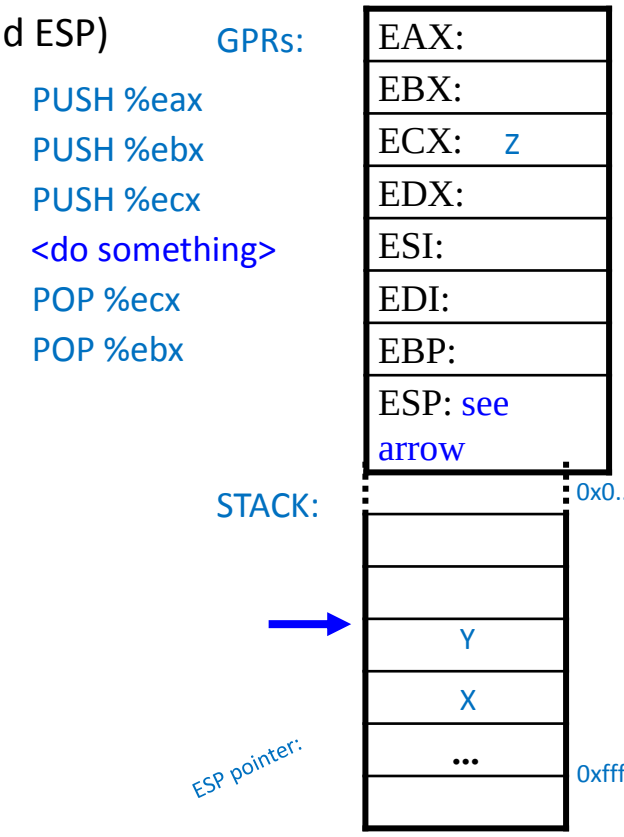
# Intel 32-bit Architecture (IA32): Basic Execution Environment

- Address space: 1 – $2^{36}$ (64 GB),
  each program may have a linear address space of 4 GB ($2^{32}$)

  Basic program execution registers:
  - 8 general purpose registers (EAX, EBX, ECX, EDX, ESI, EDI, EBP and ESP)
  - 6 segment registers (CS, DS, SS, ES, FS and GS)
  - 1 flag register (EFLAGS)
  - 1 instruction pointer register (EIP)

    Stack – a continuous array of memory locations
  - Current stack is referenced by the SS register
  - ESP register – stack pointer
  - EBP register – stack frame base pointer (fixed reference)
  - PUSH – stack grow, add item (ESP decrement)
  - POP – remove item, stack shrinks (ESP increment)

    Several other registers like Control, MMX,
    XMM, FPU, MTRR, MSR and performance monitoring



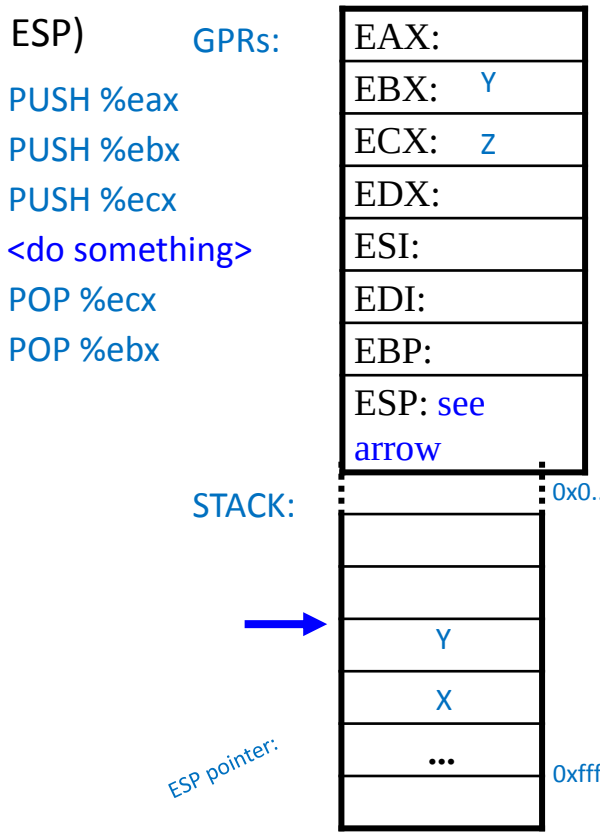
# Intel 32-bit Architecture (IA32): Basic Execution Environment

- Address space: $1 – 2^{36}$ (64 GB),
  each program may have a linear address space of 4 GB ($2^{32}$)

  Basic program execution registers:
  - 8 general purpose registers (EAX, EBX, ECX, EDX, ESI, EDI, EBP and ESP)
  - 6 segment registers (CS, DS, SS, ES, FS and GS)
  - 1 flag register (EFLAGS)
  - 1 instruction pointer register (EIP)

    Stack – a continuous array of memory locations
  - Current stack is referenced by the SS register
  - ESP register – stack pointer
  - EBP register – stack frame base pointer (fixed reference)
  - PUSH – stack grow, add item (ESP decrement)
  - POP – remove item, stack shrinks (ESP increment)

    Several other registers like Control, MMX,
    XMM, FPU, MTRR, MSR and performance monitoring



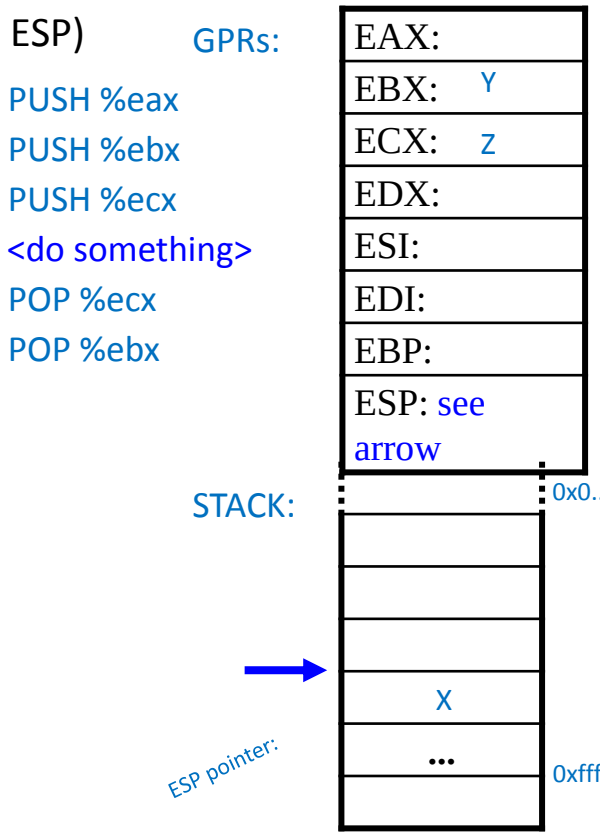
# Intel 32-bit Architecture (IA32): Basic Execution Environment

- Address space: $1 – 2^{36}$ (64 GB),
  each program may have a linear address space of 4 GB ($2^{32}$)

  Basic program execution registers:
  - 8 general purpose registers (EAX, EBX, ECX, EDX, ESI, EDI, EBP and ESP)
  - 6 segment registers (CS, DS, SS, ES, FS and GS)
  - 1 flag register (EFLAGS)
  - 1 instruction pointer register (EIP)

    Stack – a continuous array of memory locations
  - Current stack is referenced by the SS register
  - ESP register – stack pointer
  - EBP register – stack frame base pointer (fixed reference)
  - PUSH – stack grow, add item (ESP decrement)
  - POP – remove item, stack shrinks (ESP increment)

    Several other registers like Control, MMX,
    XMM, FPU, MTRR, MSR and performance monitoring



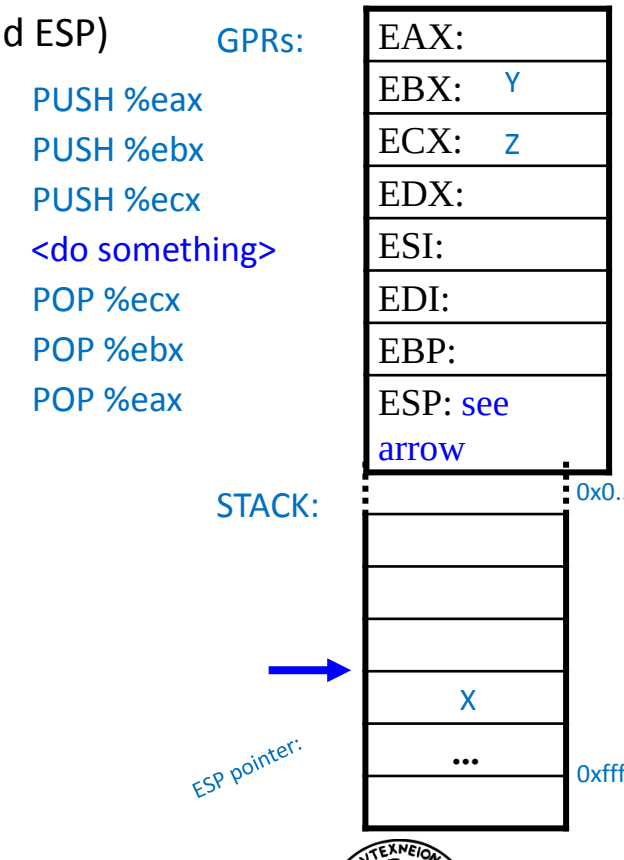
# Intel 32-bit Architecture (IA32): Basic Execution Environment

- Address space: $1 – 2^{36}$ (64 GB),
  each program may have a linear address space of 4 GB ($2^{32}$)

  Basic program execution registers:
  - 8 general purpose registers (EAX, EBX, ECX, EDX, ESI, EDI, EBP and ESP)
  - 6 segment registers (CS, DS, SS, ES, FS and GS)
  - 1 flag register (EFLAGS)
  - 1 instruction pointer register (EIP)

    Stack – a continuous array of memory locations
  - Current stack is referenced by the SS register
  - ESP register – stack pointer
  - EBP register – stack frame base pointer (fixed reference)
  - PUSH – stack grow, add item (ESP decrement)
  - POP – remove item, stack shrinks (ESP increment)

    Several other registers like Control, MMX,
    XMM, FPU, MTRR, MSR and performance monitoring



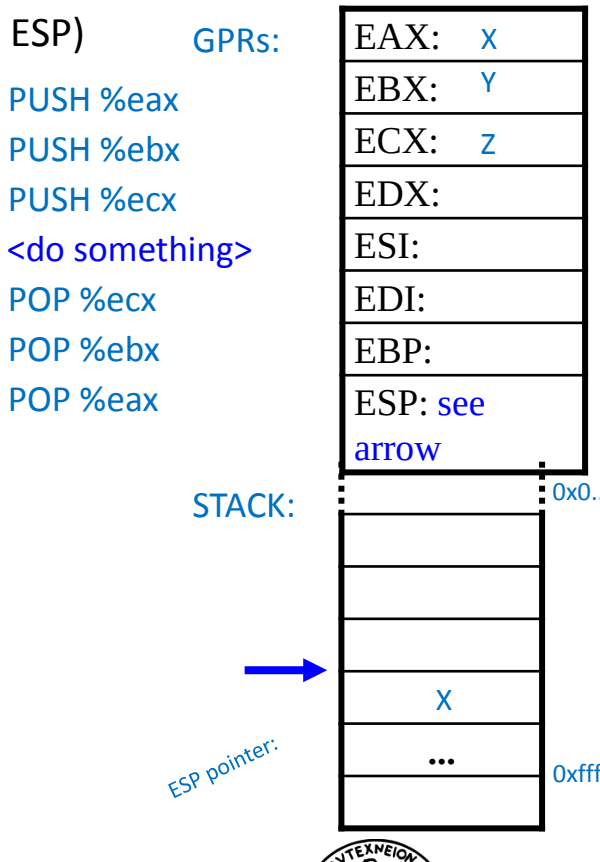
# Intel 32-bit Architecture (IA32): Basic Execution Environment

- Address space: $1 – 2^{36}$ (64 GB),
  each program may have a linear address space of 4 GB ($2^{32}$)

  Basic program execution registers:
  - 8 general purpose registers (EAX, EBX, ECX, EDX, ESI, EDI, EBP and ESP)
  - 6 segment registers (CS, DS, SS, ES, FS and GS)
  - 1 flag register (EFLAGS)
  - 1 instruction pointer register (EIP)

    Stack – a continuous array of memory locations
  - Current stack is referenced by the SS register
  - ESP register – stack pointer
  - EBP register – stack frame base pointer (fixed reference)
  - PUSH – stack grow, add item (ESP decrement)
  - POP – remove item, stack shrinks (ESP increment)

    Several other registers like Control, MMX,
    XMM, FPU, MTRR, MSR and performance monitoring



# Intel 32-bit Architecture (IA32): Basic Execution Environment

- Address space: $1 – 2^{36}$ (64 GB),
  each program may have a linear address space of 4 GB ($2^{32}$)

Basic program execution registers:

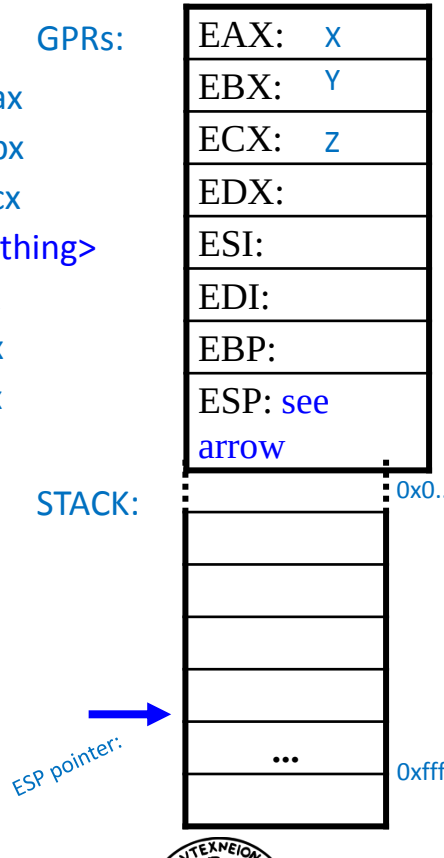
- 8 general purpose registers (EAX, EBX, ECX, EDX, ESI, EDI, EBP and ESP)
- 6 segment registers (CS, DS, SS, ES, FS and GS)
- 1 flag register (EFLAGS)
- 1 instruction pointer register (EIP)

Stack – a continuous array of memory locations

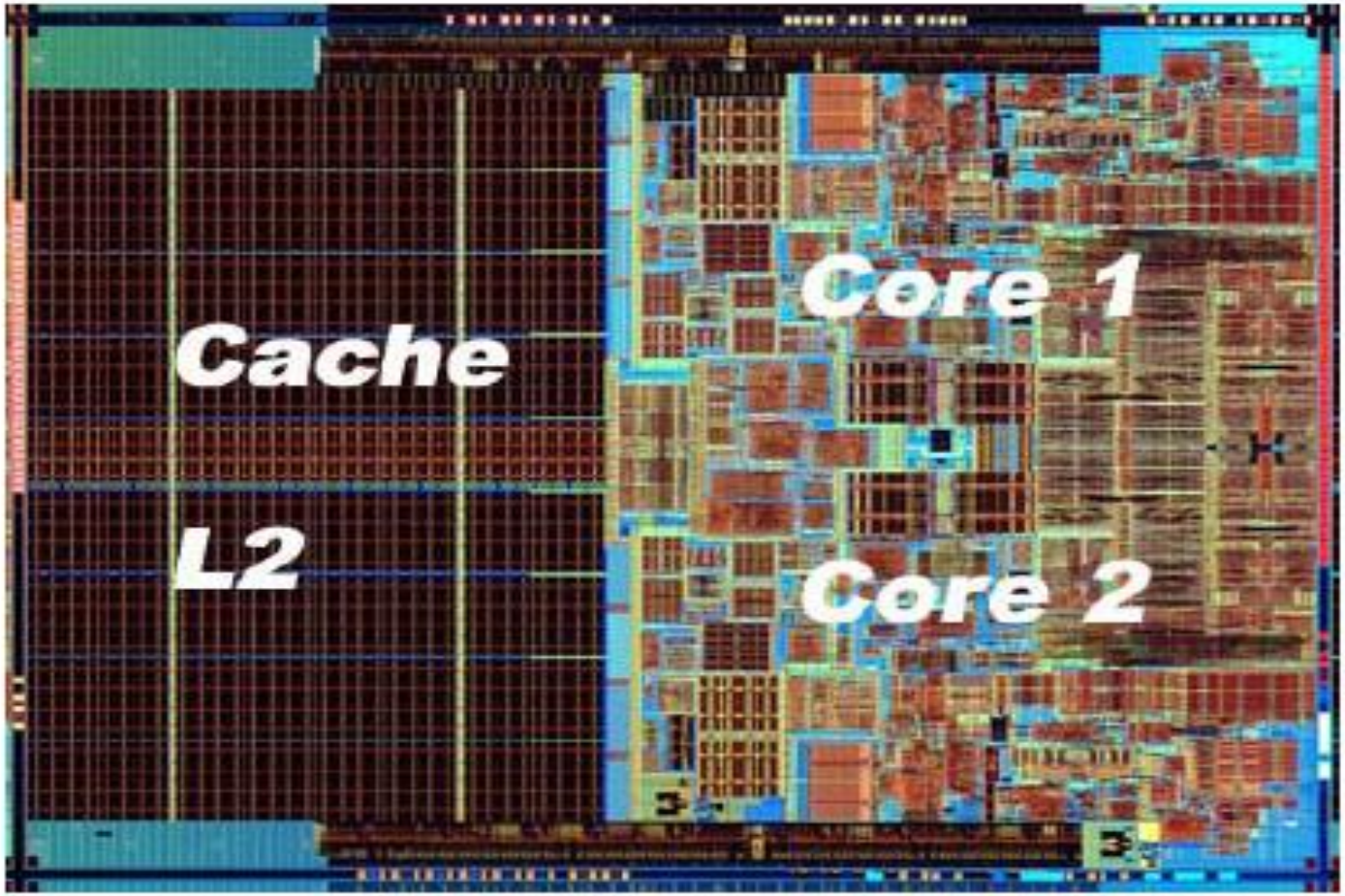
- Current stack is referenced by the SS register
- ESP register – stack pointer
- EBP register – stack frame base pointer (fixed reference)
- PUSH – stack grow, add item (ESP decrement)
- POP – remove item, stack shrinks (ESP increment)

Several other registers like Control, MMX,
XMM, FPU, MTRR, MSR and performance monitoring

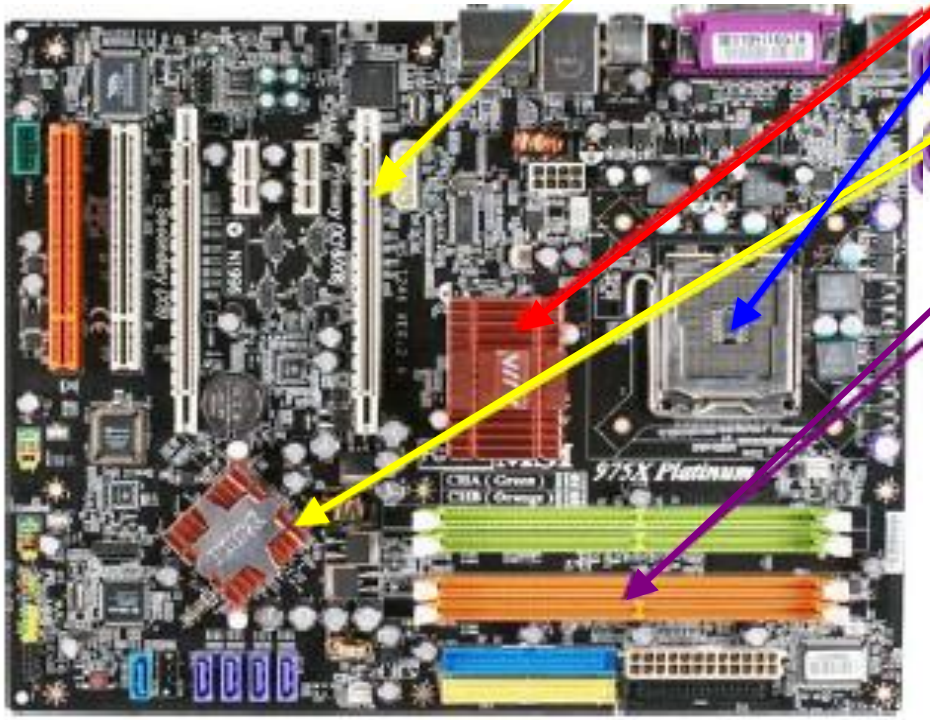
```
PUSH %eax
PUSH %ebx
PUSH %ecx
<do something>
POP %ecx
POP %ebx
POP %eax
```

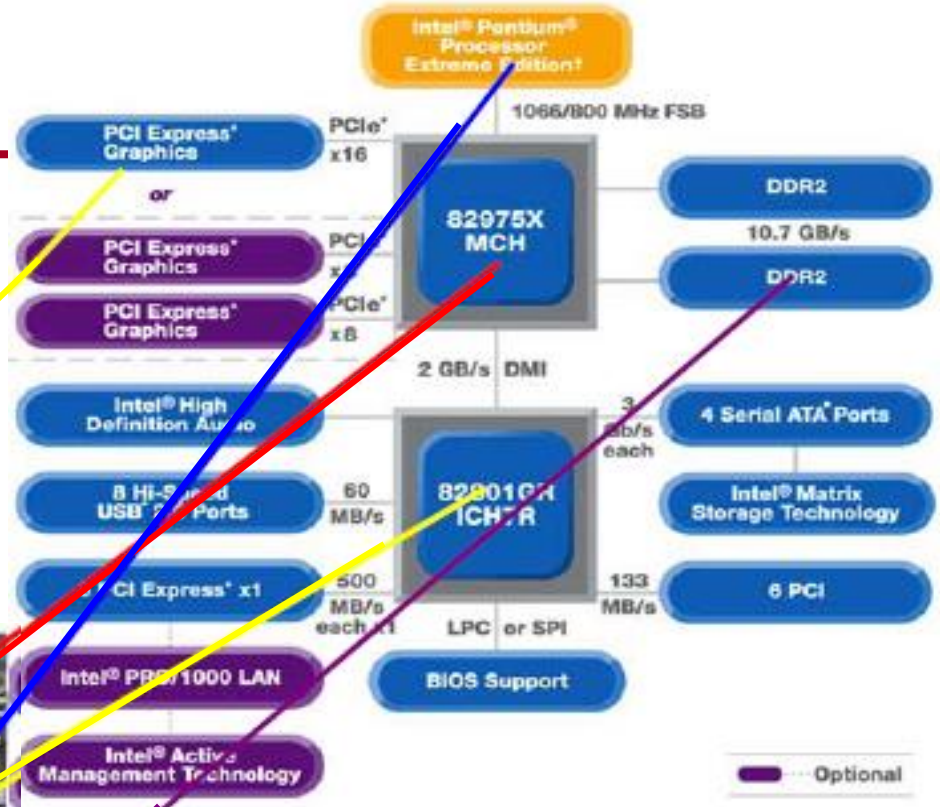


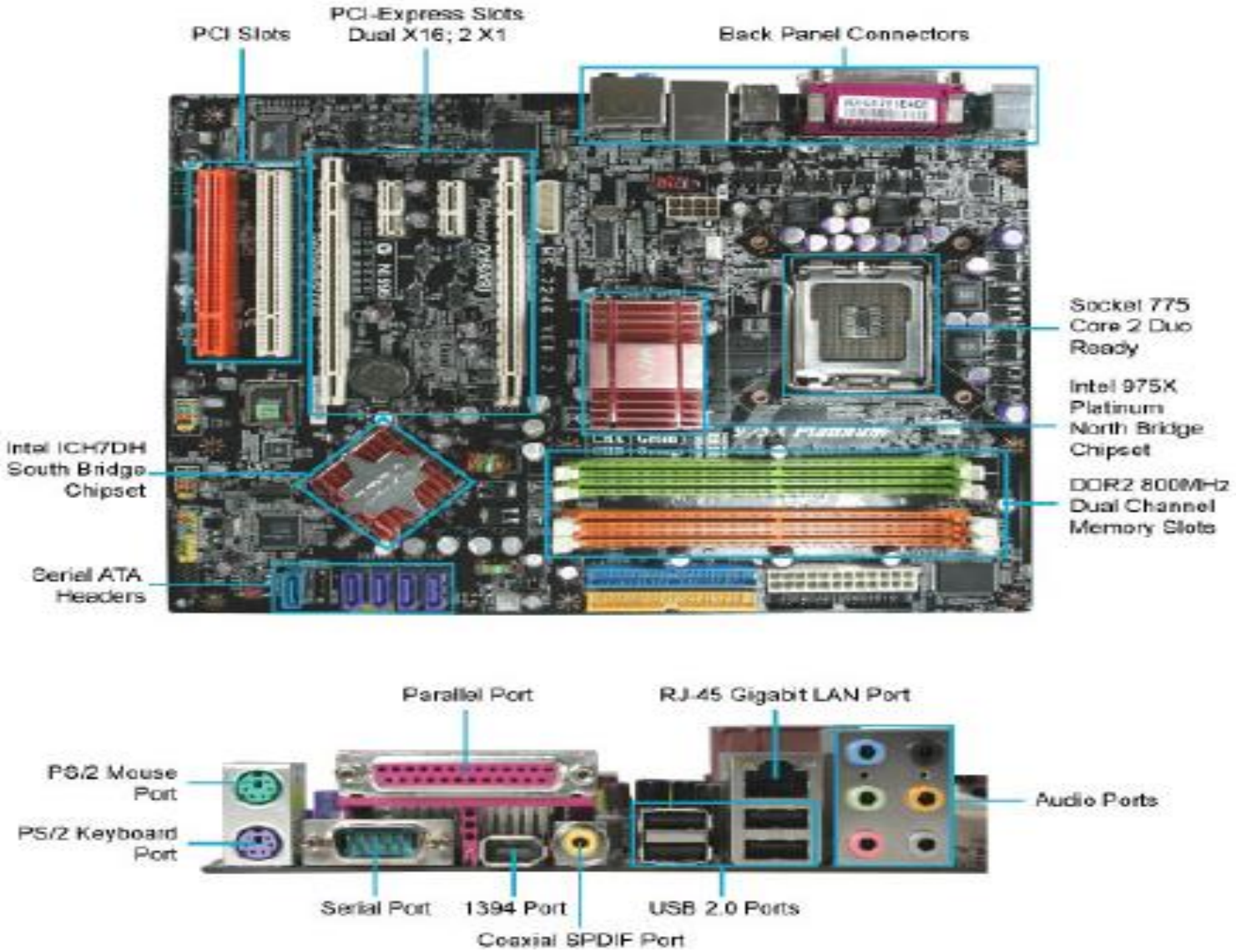
# Core 2 CPU die (Conroe µarch)



# Intel i975X chipset






PCI Slots
PCI-Express Slots
Dual X16; 2 X1
Back Panel Connectors
Socket 775
Core 2 Duo
Ready
Intel 975X
Platinum
North Bridge
Chipset
DDR2 800MHz
Dual Channel
Memory Slots
Intel ICH7DH
South Bridge
Chipset
Serial ATA
Headers
Parallel Port
RJ-45 Gigabit LAN Port
PS/2 Mouse
Port
PS/2 Keyboard
Port
Audio Ports
Serial Port
1394 Port
USB 2.0 Ports
Coaxial SPDIF Port