Λειτουργικά Συστήματα

Ενότητα 1: Εισαγωγικές Έννοιες. Ιστορία ΛΣ. Ιεραρχία Εφαρμογών. Ρυθμοί Λειτουργίας kernel/user.

Δρ. Μηνάς Δασυγένης
mdasyg@ieee.org
Εργαστήριο Ψηφιακών Συστημάτων και Αρχιτεκτονικής Υπολογιστών
http://arch.icte.uowm.gr/mdasyg
Τμήμα Μηχανικών Πληροφορικής και Τηλεπικοινωνιών

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.

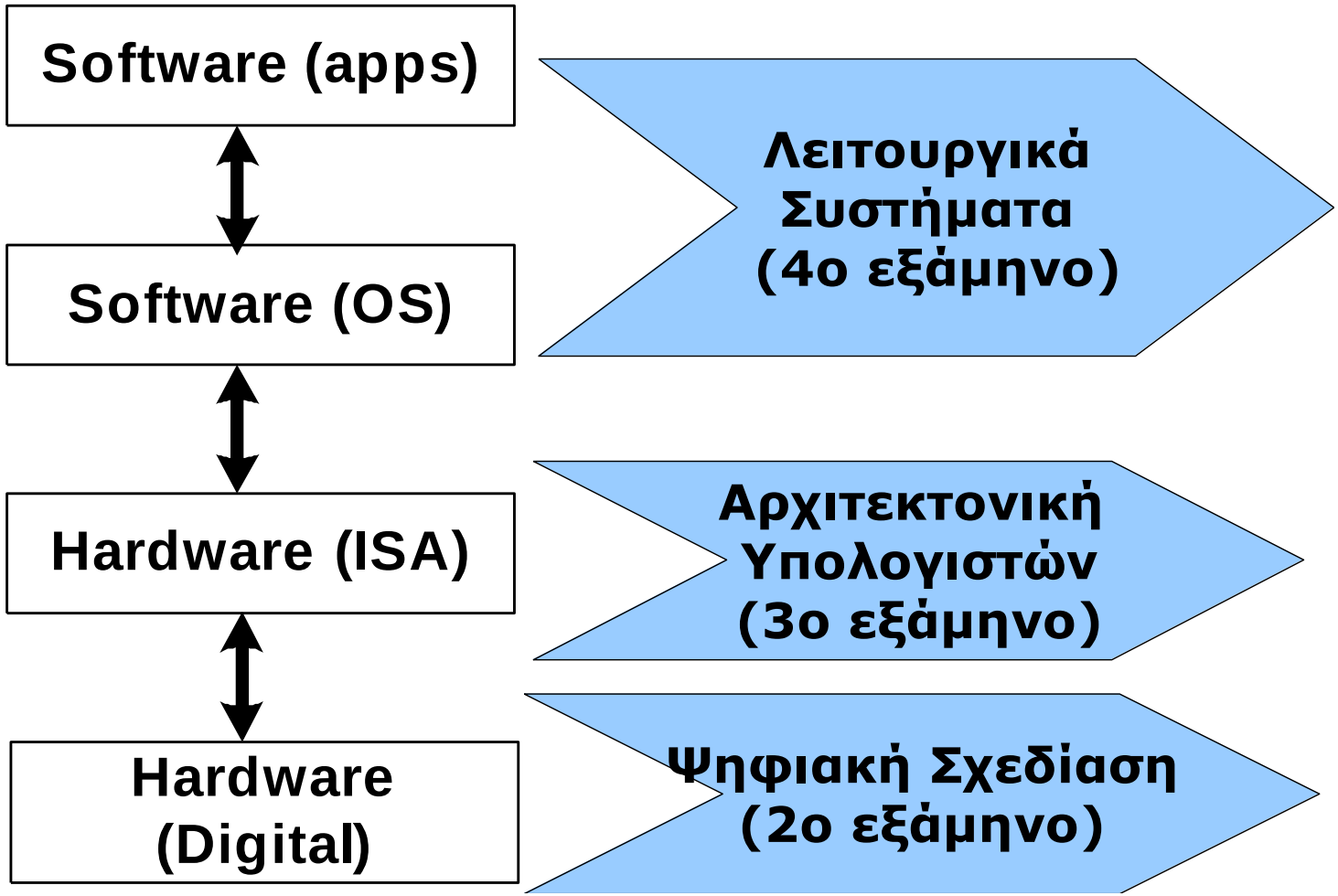
Χρηματοδότηση

- 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 της θέσης των λειτουργικών συστημάτων στην πληροφορική.
- Η κατανόηση των ΛΣ ως μηχανισμός αφαίρεσης.
- Η κατανόηση των ΛΣ ως διαχειριστής πόρων.
- Ιστορική ανασκόπηση των ΛΣ.

Σύνδεση με την Αρχιτεκτονική Η/Υ



Τα ΛΣ ως συνέχεια της Αρχιτεκτονικής Η/Υ

Όπως είδαμε, ένα σύγχρονο υπολογιστικό σύστημα αποτελείται από υλικό (hardware):

- Έναν ή περισσότερους επεξεργαστές.
- Κεντρική Μνήμη.
- Σκληρούς/Οπτικούς Δίσκους.
- Εκτυπωτές.
- Ποικίλες συσκευές εισόδου/εξόδου.

Η διαχείριση και ο συγχρονισμός όλων αυτών των πολύπλοκων συσκευών απαιτεί ένα στρώμα λογισμικού (software) ➔ Αυτό ονομάζεται **λειτουργικό σύστημα**.

Που βρίσκεται το ΛΣ; (1/3)

- Η ISA (assembly) ενός επεξεργαστή είναι το επίπεδο που βρίσκεται ανάμεσα στο λογισμικό και το υλικό.
- Μπορούμε να κατασκευάσουμε ένα πρόγραμμα γραμμένο σε assembly για μια συγκεκριμένη αρχιτεκτονική, το οποίο θα εκτελεί μια συγκεκριμένη λειτουργία.
- Όμως, αυτό απαιτεί, ο προγραμματιστής να προγραμματίσει συναρτήσεις πρόσβασης και χρήσης πλήθος λειτουργιών χαμηλού επιπέδου, όπως πρόσβαση στο δίσκο, συσκευές I/O (οθόνη, πληκτρολόγιο).
- Ακόμη και η εγγραφή ενός χαρακτήρα στην οθόνη απαιτεί δεκάδες εντολές (δεν είναι διαθέσιμο το int που είδαμε).

Που βρίσκεται το ΛΣ; (2/3)

- Αν ο προγραμματιστής θέλει η μηχανή του να εκτελεί εκτός από ένα πρόγραμμα, και επιπρόσθετα προγράμματα, τότε απαιτείται ένα μεγάλο ποσό κώδικα για την **προστασία** των εφαρμογών, την **αλληλεπίδραση** και πολλά άλλα.
- Όλα αυτά τα προβλήματα (και πλήθος άλλων) οδήγησαν στη δημιουργία και ευρεία υιοθέτηση των λειτουργικών συστημάτων.
- Το λειτουργικό σύστημα μπορεί να χαρακτηριστεί **ως ένα πολύ μεγάλο και σύνθετο πρόγραμμα** (λογισμικό) που κατευθύνει τον επεξεργαστή (υλικό).

Που βρίσκεται το ΛΣ; (3/3)

- Η αποστολή του ΛΣ είναι η διαχείριση όλων αυτών των συσκευών και η παροχή προγραμμάτων χρήση με απλούστερη διασύνδεση με το υλικό.
- Το ΛΣ βρίσκεται **αμέσως μετά το υλικό και πριν από τα υπόλοιπα προγράμματα**.
- Το ΛΣ εκτελείται στον επεξεργαστή σε κατάσταση διαχειριστή (supervisor) ή σε κατάσταση πυρήνα (kernel), ενώ τα υπόλοιπα προγράμματα σε κατάσταση χρήστη (user).
- Όλοι οι σύγχρονοι επεξεργαστές υποστηρίζουν τουλάχιστον αυτούς του 2 ρυθμούς.

Ο ρυθμός πυρήνα (kernel)

Ο πυρήνας του Λ.Σ. αναφέρεται στο κύριο τμήμα του Λ.Σ. το οποίο υλοποιεί τις δύο βασικές οντότητες (processes και files) του Λ.Σ.

Αυτός ο κώδικας είναι **προστατευμένος**, με την έννοια ότι δεν ανήκει σε κανένα χρήστη (δεν είναι «user process»).

Διαφορετικά, ο κάθε χρήστης θα μπορούσε να αλλάξει αυτόν τον κώδικα και έτσι να μονοπωλήσει τους πόρους του συστήματος, CPU, RAM, δίσκοι, τερματικά, κ.λπ.

Μόνο το ΛΣ εκτελείται σε κατάσταση πυρήνα

Αυτή η προστασία επιτυγχάνεται χρησιμοποιώντας δύο τρόπους λειτουργίας: **user mode** και **kernel mode**. Το σύστημα βρίσκεται υπό προστασία όταν βρίσκεται σε kernel mode. Δηλαδή, ειδικές εντολές που διαχειρίζονται τους πόρους του συστήματος μπορούν να εκτελεσθούν μόνο όταν το σύστημα λειτουργεί σε kernel mode.

Το Λ.Σ. είναι το μόνο system s/w το οποίο εκτελείται σε kernel mode.

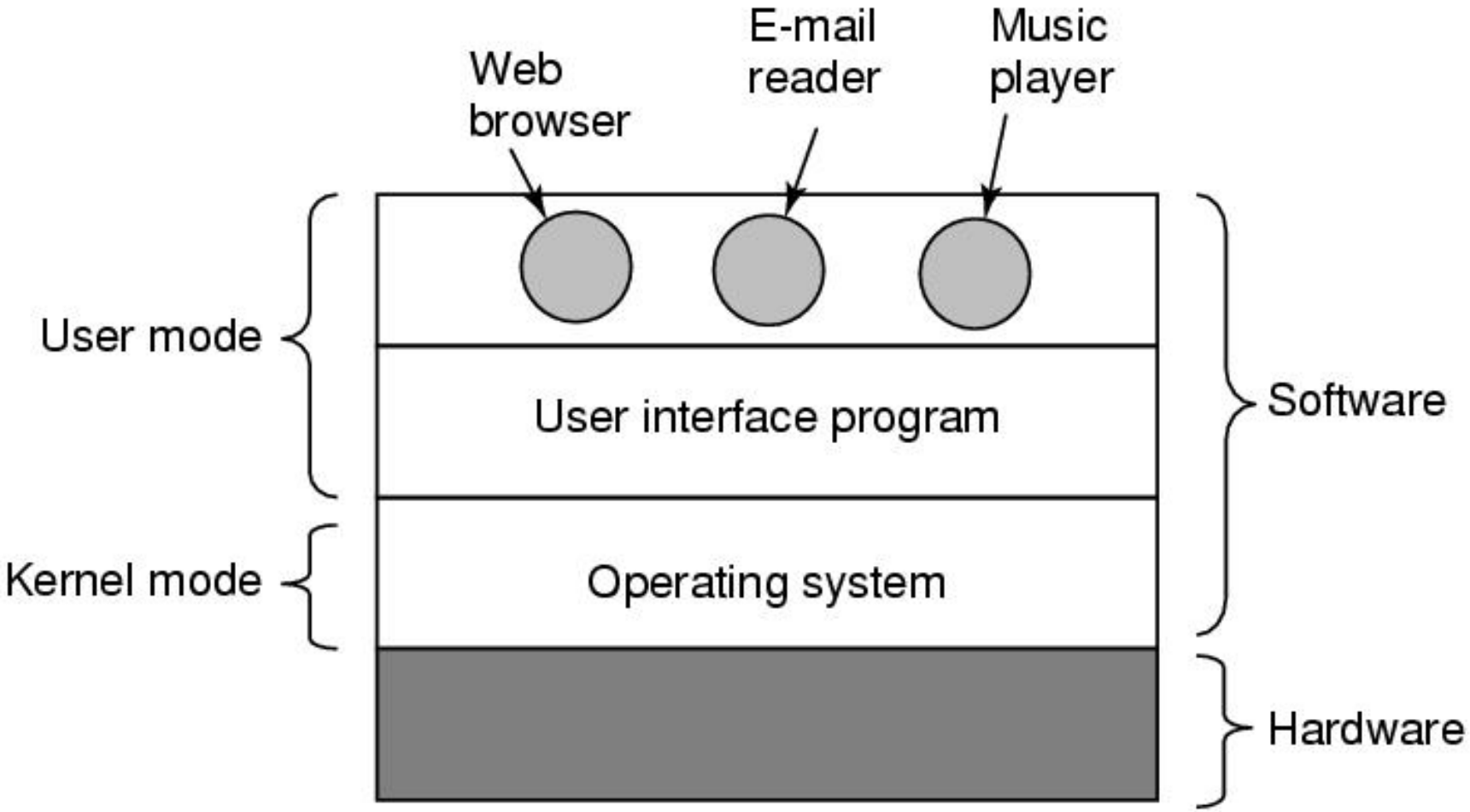
Τι είναι το λειτουργικό σύστημα;

- Λειτουργικό σύστημα είναι ένα πρόγραμμα το οποίο, από την άποψη του προγραμματιστή, προσθέτει μια ποικιλία νέων εντολών και δυνατοτήτων πάνω και πέρα από εκείνες που παρέχει το επίπεδο ISA.
- Οι βασικές του λειτουργίες είναι δύο:
  - Παρουσιάζει **ένα μηχανισμό αφαίρεσης**, ο οποίος αφαιρεί όλες τις λεπτομέρειες που καθιστούν περίπλοκη τη χρήση του υλικού.
  - Παρουσιάζει **μια εικονική μηχανή**, η οποία είναι ευκολότερο να προγραμματιστεί και να διαχειριστεί (διαχειριστής πόρων).

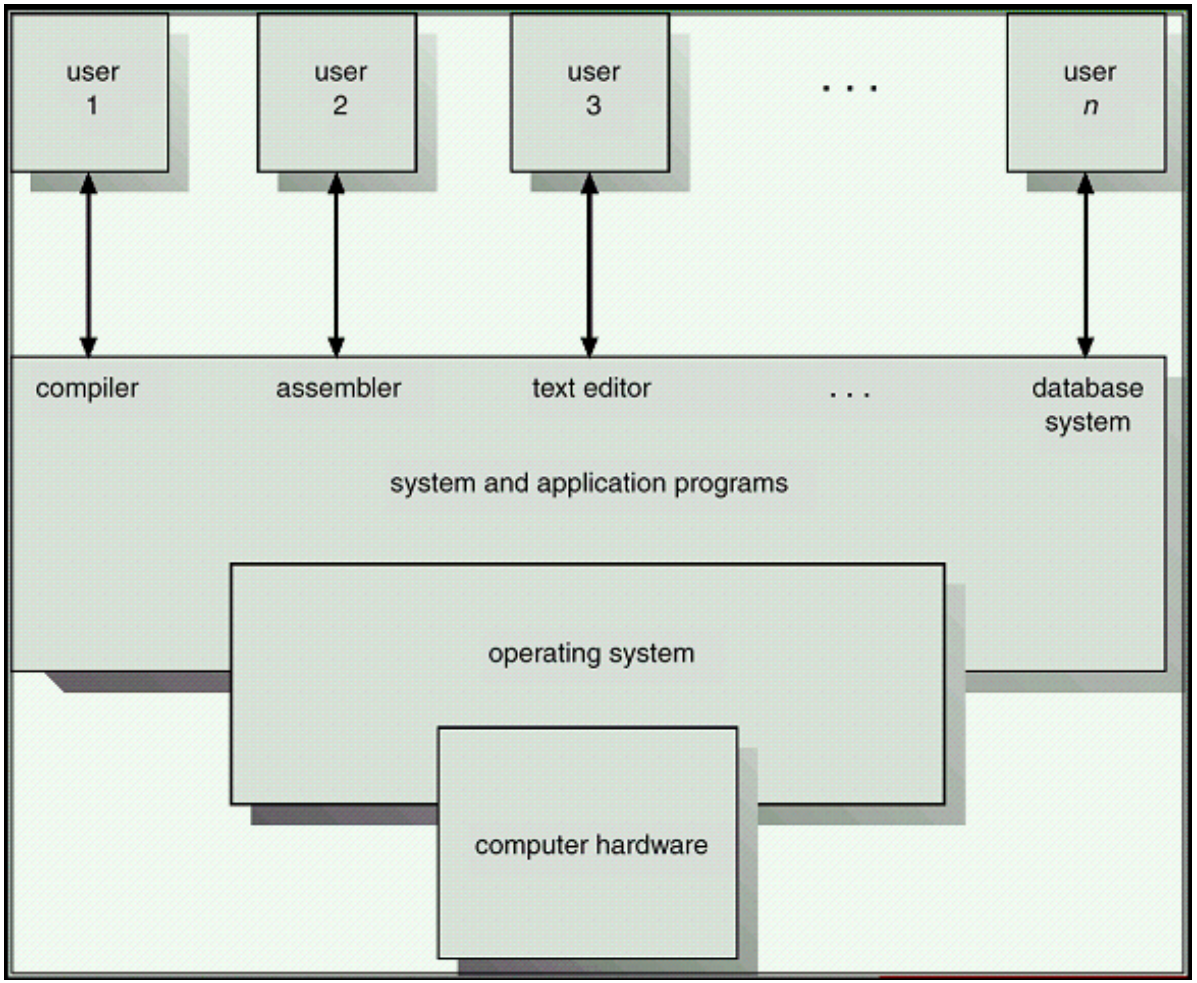
Το ΛΣ ως διαχειριστής πόρων

- Μία εφαρμογή χρησιμοποιεί τον επεξεργαστή κειμένου (editor) για να φτιάξει ένα αρχείο που περιέχει ένα πρόγραμμα (πηγαίο κώδικα). Μετά χρησιμοποιεί τον μεταφραστή (compiler) για να δημιουργηθεί το «object code», δηλαδή ο κώδικας που μπορεί να τρέξει στη μηχανή.
- Το Λ.Σ. καλείται να αποθηκεύσει τον πηγαίο κώδικα και το object code (στον δίσκο). Όταν ο κώδικας «τρέχει», το Λ.Σ. καλείται να χρησιμοποιήσει τους απαραίτητους πόρους (CPU – Μνήμη,κλπ) για να προσφερθούν οι κατάλληλες υπηρεσίες.

Που βρίσκεται το ΛΣ (1/3)



Που βρίσκεται το ΛΣ (2/3)



Που βρίσκεται το ΛΣ (3/3)

- Το ΛΣ εκτελείται σε κατάσταση πυρήνα.
- Επιτρέπει την εκτέλεση λογισμικού ανωτέρου επιπέδου όπως:
  - Ερμηνευτής εντολών (φλοιός).
  - Παραθυρικά συστήματα (window manager).
  - Μεταγλωττιστής (compiler).
  - Διορθωτής (editor).
- Αυτά δεν ανήκουν στο λειτουργικό σύστημα. Εκτελούνται σε κατάσταση χρήστη. Ο καθένας λοιπόν μπορεί να γράψει το δικό του compiler ή το δικό του editor, όχι όμως το χειριστή διακοπών ΛΣ.

Εκτέλεση ενός προγράμματος (1/2)

- Το πρόγραμμα δημιουργείται, μεταφράζεται, παράγεται ο αντικειμενικός κώδικας, αποθηκεύεται σε ένα αρχείο (δευτερεύουσα μνήμη).
- Όταν το πρόγραμμα ζητείται να εκτελεστεί, τότε μέρος του μεταφέρεται στην κύρια μνήμη.
- Κάθε εντολή του προγράμματος μεταφέρεται από την κύρια μνήμη στον επεξεργαστή (fetch/decode/execute/store).

➔Απαιτείται λοιπόν διαχείριση κύριας και δευτερεύουσας μνήμης, CPU, περιφερειακών που γίνεται από το Λειτουργικό Σύστημα.

Το λειτουργικό σύστημα ως μια εικονική μηχανή

- Το λειτουργικό σύστημα είναι δομημένο με τέτοιο τρόπο, ώστε **να δημιουργεί την ψευδαίσθηση στις εφαρμογές** ότι αυτές είναι οι μόνες που υπάρχουν και εκτελούνται στο υλικό.
- Δε μπορεί να παρεμβάλλει η μια διεργασία την άλλη και να προκαλέσει πρόβλημα.
- Η κάθε διεργασία δε μπορεί να καταλάβει την ύπαρξη άλλων διεργασιών.
- Η κάθε διεργασία μπορεί αν θελήσει να χρησιμοποιήσει όλους τους πόρους (π.χ. μνήμη) του συστήματος (ή τουλάχιστον να έχει την ψευδαίσθηση) . Έτσι π.χ. είναι δυνατό να εκτελούμε 10 εργασίες που η κάθε μια απαιτεί 2GB RAM σε ένα μηχάνημα που έχει συνολικά φυσική μνήμη 2GB RAM.
- Μπορούμε να εκτελούμε πολλές περισσότερες διεργασίες από ότι αν δεν είχαμε λειτουργικό σύστημα.

Εκτέλεση ενός προγράμματος (2/2)

Καταλαβαίνουμε από τα προηγούμενα:

- Το ΛΣ είναι το κεντρικό κομμάτι λογισμικού το οποίο διαχειρίζεται τους πόρους του υλικού.
- Επιτρέπει πολλαπλά προγράμματα να εκτελούνται την ίδια στιγμή.
- Επιτρέπει την χωρική και χρονική πολυπλεξία των συσκευών.

Αυτή η θεώρηση όμως δεν αρκεί !

Το λειτουργικό σύστημα ως μηχανισμός αφαίρεσης

- Κρύβει από τον προγραμματιστή τις λεπτομέρειες του υλικού.
- Ο χρήστης μπορεί να χρησιμοποιήσει έναν υπολογιστή χωρίς να ενδιαφέρεται για το πως λειτουργεί το υλικό.
- Ο προγραμματιστής δε χρειάζεται να δημιουργήσει πολύπλοκο κώδικα χαμηλού επιπέδου. Προγραμματίζει σε υψηλό επίπεδο.

ΠΑΡΑΔΕΙΓΜΑ: Αν δεν υπήρχε λειτουργικό σύστημα για να γράψουμε ένα αρχείο θα έπρεπε να ρυθμιστεί ο ελεγκτής του δίσκου, να ρυθμιστεί ο bus master, να βρεθεί η ακριβής τοποθεσία στο δίσκο, να μετακινηθεί ο βραχίονας των κεφαλών κατάλληλα, να δοθεί η εντολή εγγραφής..... (και πολλά άλλα).

- ✓ Με το λειτουργικό σύστημα, ο χρήστης κάνει “SAVE” .
- ✓ Με το λειτουργικό σύστημα, ο προγραμματιστής κάνει fwrite(...).

Πως μια διεργασία χρησιμοποιεί λειτουργίες που δε μπορεί να κάνει σε κατάσταση χρήστη

Για να μπορέσει μια διεργασία χρήστη (user process) να χρησιμοποιήσει τους πόρους πρέπει **να καλέσει τις κατάλληλες ρουτίνες** του kernel.

Αυτό επιτυγχάνεται μέσω **system calls**.

➔ υπάρχουν system calls όπως **fork(), exec(), malloc(), read(), printf()** τα οποία αφορούν τη δημιουργία ενός process, την εκτέλεση από κάποιο process ενός προγράμματος, την παροχή μνήμης, την ανάκτηση τμήματος αρχείου, την εκτύπωση στην οθόνη, κ.λπ.

Σύνοψη ρόλων

- Ως μια εκτεταμένη (extended) ή ιδεατή (virtual) μηχανή που παρέχει μία **διασύνδεση μεταξύ χρήστη και Η/Υ**.
- Ως **διαχειριστής των πόρων** του συστήματος (resource manager).

Το λειτουργικό σύστημα είναι λογισμικό

- Το Λ.Σ. λειτουργεί όπως ένα οποιοδήποτε άλλο είδος λογισμικού, με την έννοια ότι χρειάζεται τον επεξεργαστή για να εκτελεσθεί.
- Ως μέρος των ενεργειών του, το Λ.Σ. Πολύ συχνά εκχωρεί τον έλεγχο της ΚΜΕ στα υπόλοιπα προγράμματα που εκτελούνται και αναμένει από την ΚΜΕ να του επιτρέψει να την επαναχρησιμοποιήσει.

Σχέση του ΛΣ με το υλικό

- Σχέση Εξάρτησης μεταξύ ΛΣ και αρχιτεκτονικής.
- Πολλές **δυνατότητες των ΛΣ χρειάζονται υποστήριξη από το Υλικό**.
- Νέες **δυνατότητες του Υλικού χρειάζονται υποστήριξη από το ΛΣ**, ώστε να είναι διαθέσιμες στις εφαρμογές.

Ποιοι είναι οι δυο θεμελιώδεις στόχοι;

- **Αποδοτικότητα**: Όσο περισσότερο καλύτερη εκμετάλλευση του υλικού.
- **Άνεση και εξυπηρέτηση**: Φιλική και απλή χρήση του υλικού.

Πόσα ΛΣ υπάρχουν;

- Τα πρώτα ολοκληρωμένα λειτουργικά συστήματα παρουσιάστηκαν το 1960.
- Από τότε εκατοντάδες λειτουργικά συστήματα έχουν παρουσιασθεί, κάποια από τα οποία εγκαταλείφθηκαν τελείως, κάποια που ύστερα από μερικά χρόνια εγκατάλειψης τα ανέλαβαν άλλοι, και κάποια που συνεχίζουν να υπάρχουν για δεκαετίες.
- Τα λειτουργικά συστήματα έχουν διαφορετικούς στόχους.

Ιστορία των ΛΣ (1/3)

Τα Λ.Σ. εξελίσσονται με τον χρόνο.

- Αναβάθμιση συσκευών και χρήση νέων συσκευών.
- Νέες υπηρεσίες.
- Διορθώσεις.

Ιστορία των ΛΣ (2/3)

Γενιές Υπολογιστών:

1η (1945–55) Vacuum Tubes.

2η (1955–65) Transistors and Batch Systems.

3η (1965–1980) ICs and Multiprogramming.

4η (1980–Present) Personal Computers.

Ιστορία των ΛΣ (3/3)

Γενιές Λειτουργικών συστημάτων:

- Σειριακή επεξεργασία (serial processing).
- Συστήματα σειριακής επεξεργασίας δέσμης (simple batch systems).
- Συστήματα πολυπρογραμματισμού σειριακής επεξεργασίας δέσμης (multiprogrammed batch systems).
- Συστήματα καταμερισμού χρόνου (time sharing systems).

1η γενιά

Vacuum Tubes (1η γενιά)

- Ο Η/Υ αποτελείται από λυχνίες κενού και κάρτες καλωδιακών συνδέσεων.
- Ουσιαστικά δεν υπάρχει Λ.Σ.
- Το «πρόγραμμα» αποτελείται από κάρτες καλωδιακών συνδέσεων (plugboards) που «έτρεχαν» στον Η/Υ (αν και όταν κάποιες από τις περίπου 20.000 λυχνίες κενού δεν ήταν καμένες).
- Προς τις αρχές του ’50 γίνεται «αναβάθμιση» με τη χρήση διάτρητων καρτών (punched cards) αντί για κάρτες καλωδιακών συνδέσεων .
- Ευθύνη του χρήστη να αναλάβει ο ίδιος όλες τις λειτουργίες που σχετίζονται με την εκτέλεση κάποιου προγράμματος: φόρτωμα του μεταγλωττιστή, βιβλιοθηκών, κλπ., παραγωγή συμβολικής γλώσσας, φόρτωμα του συμβολομεταφραστή και παραγωγή κώδικα.

Vacuum Tubes (1η γενιά)- Προβλήματα

Χρονοδρομολόγηση

- Συνήθως ο Η/Υ δεσμεύεται από τους χρήστες σε χρονικά διαστήματα της 0.5 ώρας.
- Ένας χρήστης δεσμεύει το σύστημα για μισή ώρα αλλά το χρησιμοποιεί μόνο 20 λεπτά. Κατ' επέκταση το σύστημα είναι ανενεργό για 10 λεπτά.
- Ο χρήστης δεν καταφέρνει να τελειώσει σε μισή ώρα και αναγκάζεται να φύγει, χωρίς να έχει ολοκληρώσει τη δουλειά του.

Χρόνος προετοιμασίας του συστήματος

- Ο χρήστης είναι υποχρεωμένος να αναλάβει ο ίδιος όλες τις διαδικασίες εκτέλεσης του προγράμματος που συμπεριλαμβάνουν τη μεταφορά ταινιών και τη φόρτωσή τους, διάβασμα διάτρητων καρτελών, κλπ.
- Αν συμβεί κάποιο σφάλμα, ο άτυχος χρήστης πρέπει να ξεκινήσει πάλι από την αρχή.

❖ Συμπερασματικά, γίνεται μεγάλη σπατάλη χρόνου που οδηγεί σε **σημαντική υποχρησιμοποίηση** του Η/Υ, που εκείνη την εποχή ήταν πολύ ακριβός.

2η γενιά

Transistors and Batch Systems (2η γενιά)

- Ο Η/Υ αποτελείται από κρυσταλλοτριόδους (transistors) που είναι πιο αξιόπιστοι από τις λυχνίες. Αυτό έχει θετικό αντίκτυπο και στην ανάπτυξη του λογισμικού.
- Αναπτύσσεται η έννοια της εργασίας (job), αποτελούμενης από ένα ή περισσότερα προγράμματα γραμμένα σε κάποια γλώσσα προγραμματισμού μαζί με τα δεδομένα τους. Κάθε εργασία διαβάζεται από τον Η/Υ με χρήση διάτρητων καρτών.
- Δημιουργείται η έννοια της σειριακής επεξεργασίας (serial processing): μόλις τελειώνει η εκτέλεση μίας εργασίας αρχίζει αμέσως να εκτελείται η επόμενη.
- Για την πιο αποδοτική χρήση του Η/Υ, ο χρήστης παύει να έχει άμεσο έλεγχο του συστήματος αλλά παραδίδει τις κάρτες με το πρόγραμμά του σε κάποιον χειριστή ο οποίος τις φορτώνει στον Η/Υ.

Transistors and Batch Systems ($2^{η}$ γενιά): Παρακολουθητής

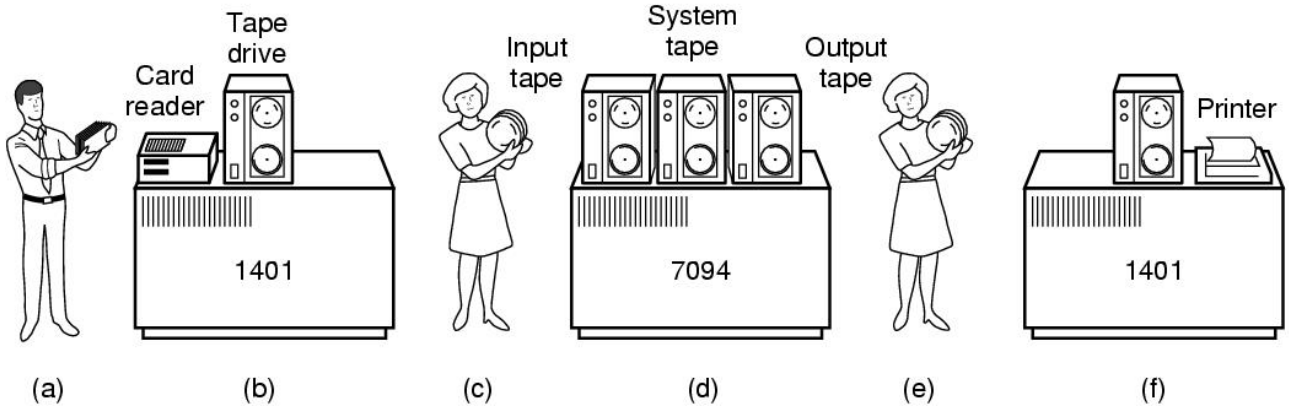
- Η σειριακή επεξεργασία και η εκτέλεση ενός προγράμματος μετά το άλλο, γίνεται δυνατή με τη χρήση ενός λογισμικού που λέγεται παρακολουθητής (monitor), μέρος του οποίου βρίσκεται πάντα στη μνήμη.
- Ο παρακολουθητής διαβάζει μία εργασία η οποία φορτώνεται στην περιοχή χρηστών της μνήμης και της δίνει τον έλεγχο της ΚΜΕ για να εκτελεσθεί. Η κάθε εργασία διαμορφώνεται με τέτοιο τρόπο έτσι ώστε μόλις ολοκληρώσει την εκτέλεσή της, ο έλεγχος επιστρέφει στον παρακολουθητή ο οποίος φορτώνει την επόμενη εργασία, κοκ.
- Τα αποτελέσματα της εκτέλεσης της κάθε εργασίας στέλνονται σε κάποια συσκευή εξόδου για να τα παραλάβει ο χρήστης.

Transistors and Batch Systems (2η γενιά)

(a) ο προγραμματιστής φέρνει διάτρητες κάρτες σε ένα μηχάνημα το οποίο τις διαβάζει και γράφει σε μια μαγνητική κασέτα.

(c) η μαγνητική κασέτα μεταφέρεται στον κεντρικό υπολογιστή και τα δεδομένα γράφονται στην μαγνητική κασέτα εξόδου.

(e) η μαγνητική κασέτα εξόδου μεταφέρεται σε ένα άλλο σύστημα που έχει έναν εκτυπωτή.



Transistors and Batch Systems (2η γενιά): Γλώσσες ελέγχου

- Οι γλώσσες ελέγχου εργασιών (job control languages) παρέχουν στον παρακολουθητή σημαντικές οδηγίες για την εκτέλεση της κάθε εργασίας, όπως:
  - Ποιον μεταφραστή να χρησιμοποιήσει.
  - Ποια δεδομένα να χρησιμοποιήσει.
- Πριν την εκτέλεση ενός προγράμματος, ο παρακολουθητής διαβάζει τις κάρτες που αντιστοιχούν στις οδηγίες αυτές και φορτώνει αμέσως τα βοηθητικά προγράμματα που χρειάζονται (μεταφραστές, συμβολομεταφραστές, κλπ.), ελαττώνοντας έτσι σημαντικά τη σπατάλη χρόνου μεταξύ της εκτέλεσης δύο εργασιών.

Transistors and Batch Systems (2η γενιά): υποστήριξη παρακολουθητή

- Το προαναφερθέν μοντέλο εκτέλεσης εργασιών, για να λειτουργήσει σωστά, πρέπει να υποστηρίζει τα ακόλουθα:
  - **Προστασία μνήμης**: Ο χώρος της μνήμης που χρησιμοποιείται από τον παρακολουθητή δεν πρέπει να είναι προσπελάσιμος στο πρόγραμμα του χρήστη που εκτελείται.
  - **Χρονικό μετρητή**: Ένα πρόγραμμα χρήστη δεν πρέπει να εκτελείται επ' αόριστον.
  - **Προνομιούχες εντολές** (privileged instructions): Όταν ένα πρόγραμμα χρήστη ζητήσει την εκτέλεση μίας τέτοιας εντολής, τότε ο έλεγχος πρέπει να επιστρέψει (πιθανόν προσωρινά) στον παρακολουθητή (π.χ. εντολές εισόδου/εξόδου).
  - **Διακοπές** (interrupts). Επιτρέπει την πιο εύκολη και αποδοτική μετάπτωση ελέγχου μεταξύ του παρακολουθητή και του εκτελούμενου προγράμματος.
- Τα παραπάνω συνδέονται με τις καταστάσεις kernel/user των CPU.

Transistors and Batch Systems (2η γενιά): Προβλήματα

- Ακόμα και με την εισαγωγή του παρακολουθητή και της σειριακής επεξεργασίας, η ΚΜΕ, η πιο σημαντική μονάδα ενός Η/Υ, **υποχρησιμοποιείται** σε απαράδεκτα χαμηλό ρυθμό.
- Το πρόβλημα βρίσκεται στο ότι οι συσκευές Ε/Ε είναι κατά κανόνα πολύ πιο αργές από την ΚΜΕ.

Π.χ.

- Διάβασμα μίας εγγραφής από κάποιο αρχείο: 15 μs.
- Εκτέλεση 100 εντολών:1 μs.
- Γράψιμο μίας εγγραφής πίσω στο αρχείο: 15 μs.
- Σύνολο χρόνου: 31 μs.
- Ποσοστιαία χρήση ΚΜΕ: 1/31 = 0,032 = 3,2%.

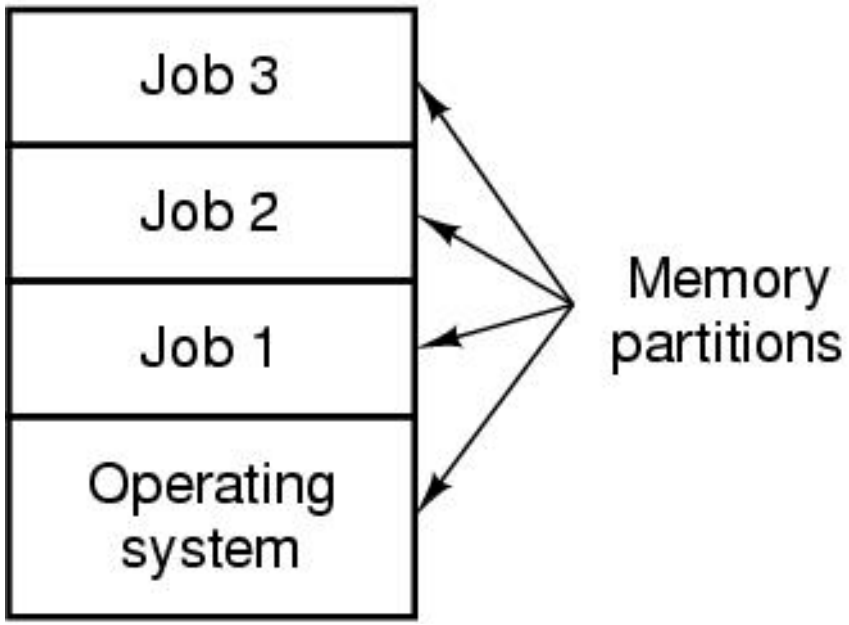
Τι ονομάζεται batching (ομαδοποίηση)

- Μια έννοια που αναπτύχθηκε περίπου στις αρχές της δεκαετίας του ‘60. Αφορά τη δημιουργία ομάδων προγραμμάτων τις οποίες ένας χειριστής «φόρτωνε» μαζί στον υπολογιστή των ημερών. Τα προγράμματα αυτά εκτελούνταν σειριακά (το ένα μετά το άλλο).
- Αυτό ήταν μία βελτίωση σε σχέση με το προηγούμενο τρόπο λειτουργίας, όπου ο χειριστής «φόρτωνε» κάθε εργασία (job) (το πρόγραμμα του χρήστη, τον compiler, κλπ) ξεχωριστά -- μία χρονοβόρα διαδικασία.

3η γενιά

ICs and Multiprogramming

- Στον πολυπρογραμματισμό επιτρέπονται πολλαπλές εργασίες ταυτόχρονα στη μνήμη που εκτελούνται ψευδοπαράλληλα.



Τι ονομάζεται πολυπρογραμματισμός (1/2);

- Η υποστήριξη πολλών ταυτόχρονα εκτελούμενων διεργασιών από το σύστημα (που μπορεί να έχει έναν επεξεργαστή).
- Πολλά είδη προγραμμάτων περιέχουν αρκετή Ε/Ε (I/O -- ονομάζονται I/O bound – σε αντίθεση με τα CPU-bound).
- Όταν καλούν κάποια I/O διαδικασία (π.χ. πρόσβαση σε δίσκο) δεν χρειάζονται τον κεντρικό επεξεργαστή .

➔ Αν υπάρχει μόνο ένα πρόγραμμα που τρέχει στο σύστημα, τότε σπαταλάται χωρίς λόγο ένας σημαντικός πόρος (CPU cycles).

Τι ονομάζεται πολυπρογραμματισμός (2/2);

- Η έννοια του **multiprogramming** (που αναπτύχθηκε στα τέλη του ‘60) επιτρέπει τη «συμβίωση» πολλών προγραμμάτων που τρέχουν, που φορτώνονταν σε διαφορετικά τμήματα της μνήμης. Π.χ. όταν το ένα πρόγραμμα κάνει I/O, τότε το ΛΣ «δίνει» τη CPU σε κάποιο άλλο πρόγραμμα που τη χρειάζεται, κ.ο.κ. Έτσι αυξάνεται σημαντικά η απόδοση του συστήματος.

Τι ονομάζεται ετεροχρονισμός (spooling)

- Η τεχνική του multiprogramming συνδυάστηκε επιτυχώς με την τεχνική του spooling.
- Με αυτή τη τεχνική, προγράμματα διαβάζονταν κατ' ευθείαν στο δίσκο του συστήματος, παράλληλα με το τρέξιμο των προγραμμάτων που βρίσκονταν στη μνήμη του υπολογιστή.
- Όταν κάποιο απ' αυτά τα προγράμματα τελείωνε, τότε κάποιο άλλο διαβαζόταν από τον δίσκο στο τμήμα της μνήμης που έμεινε κενό, και άρχιζε να τρέχει.
- Το πρόβλημα με τα παραπάνω έγκειται στο μεγάλο χρόνο απόκρισης (από τη στιγμή που ο χρήστης έδινε το πρόγραμμα του στον χειριστή, μέχρι να πάρει τα αποτελέσματα). Δηλαδή το πρόβλημα είναι η σειριακή εκτέλεση των προγραμμάτων.
- Έτσι προέκυψε η έννοια του χρονομερισμού (timesharing).

Τι ονομάζεται χρονομερισμός;

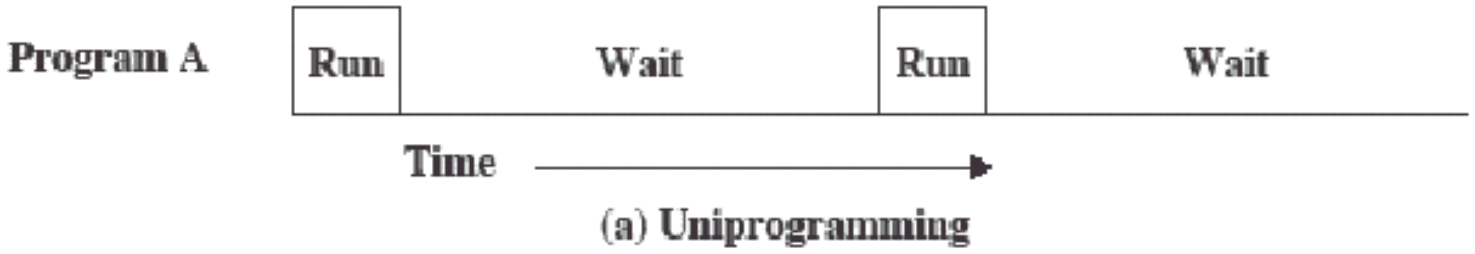
- Πρόκειται για μία μορφή multiprogramming.
- Παρατηρήθηκε ότι πολλοί χρήστες συνήθως «σκεφτόντουσαν» τις επόμενες κινήσεις τους και έτσι δεν χρειάζονταν τη CPU.
- Η CPU μοιραζόταν στους χρήστες περιοδικά.
- Σε κάθε περίοδο ένας χρήστης είχε τη CPU μέχρι να τελειώσει η περίοδος, ή μέχρι να προκύψει I/O, ή να αρχίσει να «σκέφτεται».

ICs and Multiprogramming 3η γενιά

- Ο Η/Υ αποτελείται από ολοκληρωμένα κυκλώματα (large scale integration), που βελτιώνουν την ταχύτητα και αυξάνουν τη χωρητικότητα μνήμης σε έναν Η/Υ.
- Αναπτύσσονται νέες τεχνικές, όπως:
  - **Πολυπρογραμματισμός** (multiprogramming): Η ταυτόχρονη ύπαρξη στην μνήμη περισσοτέρων της μίας εργασίας και η εναλλακτική εκτέλεσή τους στην ΚΜΕ.
  - **Χρονοδρομολόγηση** (scheduling): Πολιτικές με βάση τις οποίες καθορίζεται με ποια σειρά και για πόσο χρονικό διάστημα θα εκτελείται η κάθε εργασία.
  - **Διαχείριση μνήμης** (memory management): Πολιτικές με βάση τις οποίες καθορίζεται πως θα μοιρασθεί η υπάρχουσα μνήμη στις εργασίες που είναι φορτωμένες στο σύστημα.
  - **Καταμερισμός χρόνου** (time sharing): Η υποστήριξη ταυτόχρονης σύνδεσης σε ένα Η/Υ από πολλούς χρήστες μέσω τερματικών σταθμών.

Μονοπρογραμματισμός

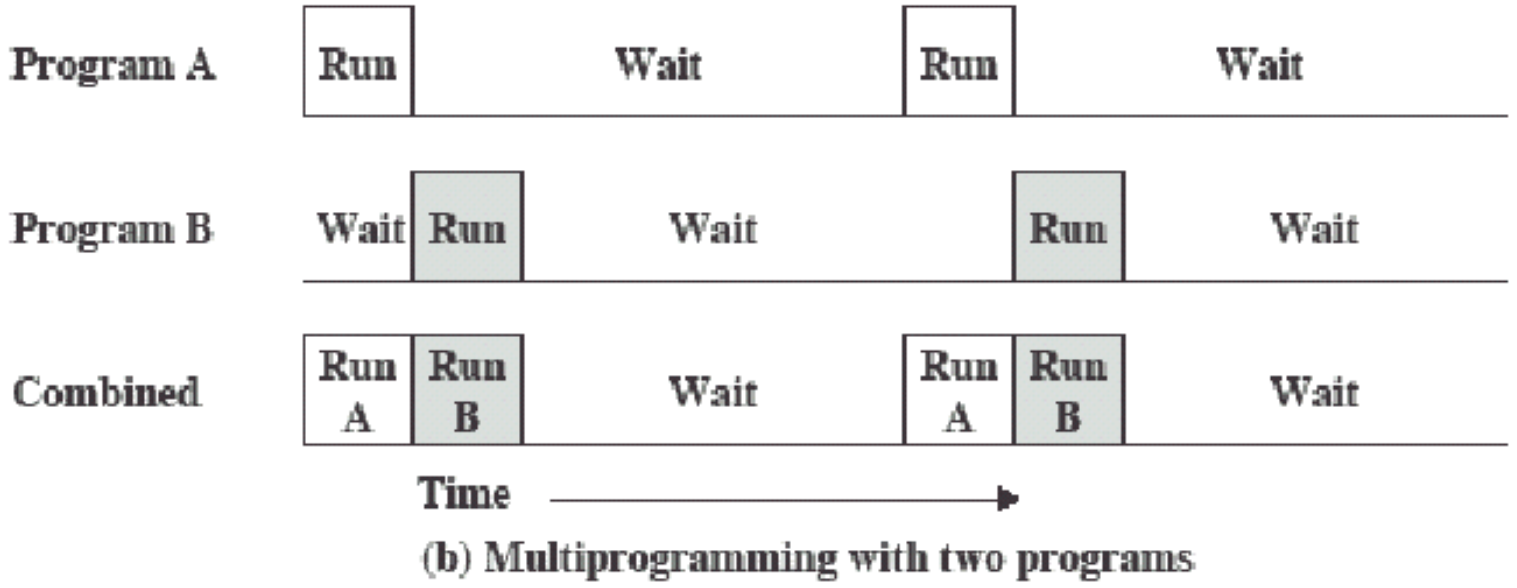
- Στον μονοπρογραμματισμό (uniprogramming), η ΚΜΕ μόλις συναντήσει μία εντολή Ε/Ε είναι υποχρεωμένη να περιμένει σε ανενεργή κατάσταση μέχρις ότου ολοκληρώσει την εκτέλεσή της αυτή η εντολή, για να συνεχίσει την εκτέλεση των επόμενων (υπολογιστικών) εντολών του προγράμματος.



(a) Uniprogramming

Πολυπρογραμματισμός (1/2)

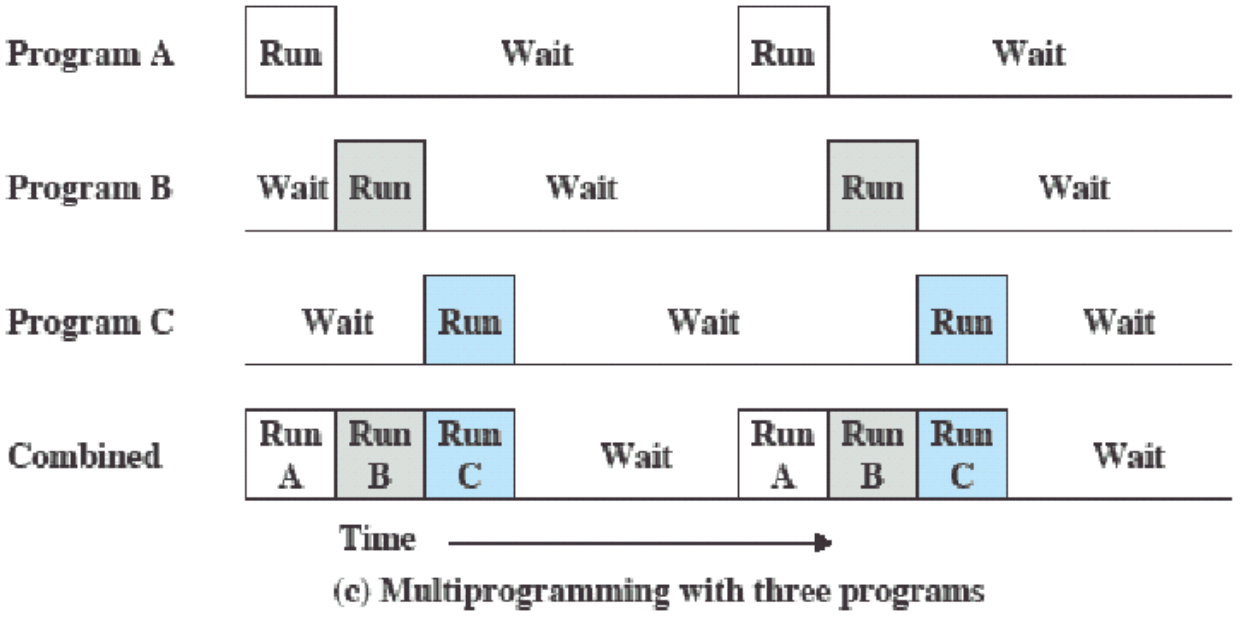
- Αν όμως στη μνήμη υπήρχαν 2 προγράμματα, η ΚΜΕ θα ήταν ανενεργή για μικρότερο χρονικό διάστημα.



(b) Multiprogramming with two programs

Πολυπρογραμματισμός (2/2)

- Στη γενική περίπτωση, όσα περισσότερα προγράμματα υπάρχουν ταυτόχρονα στο μνήμη, τόσο μικρότερο αναμένεται να είναι το χρονικό διάστημα στο οποίο η ΚΜΕ θα βρίσκεται σε ανενεργή κατάσταση.



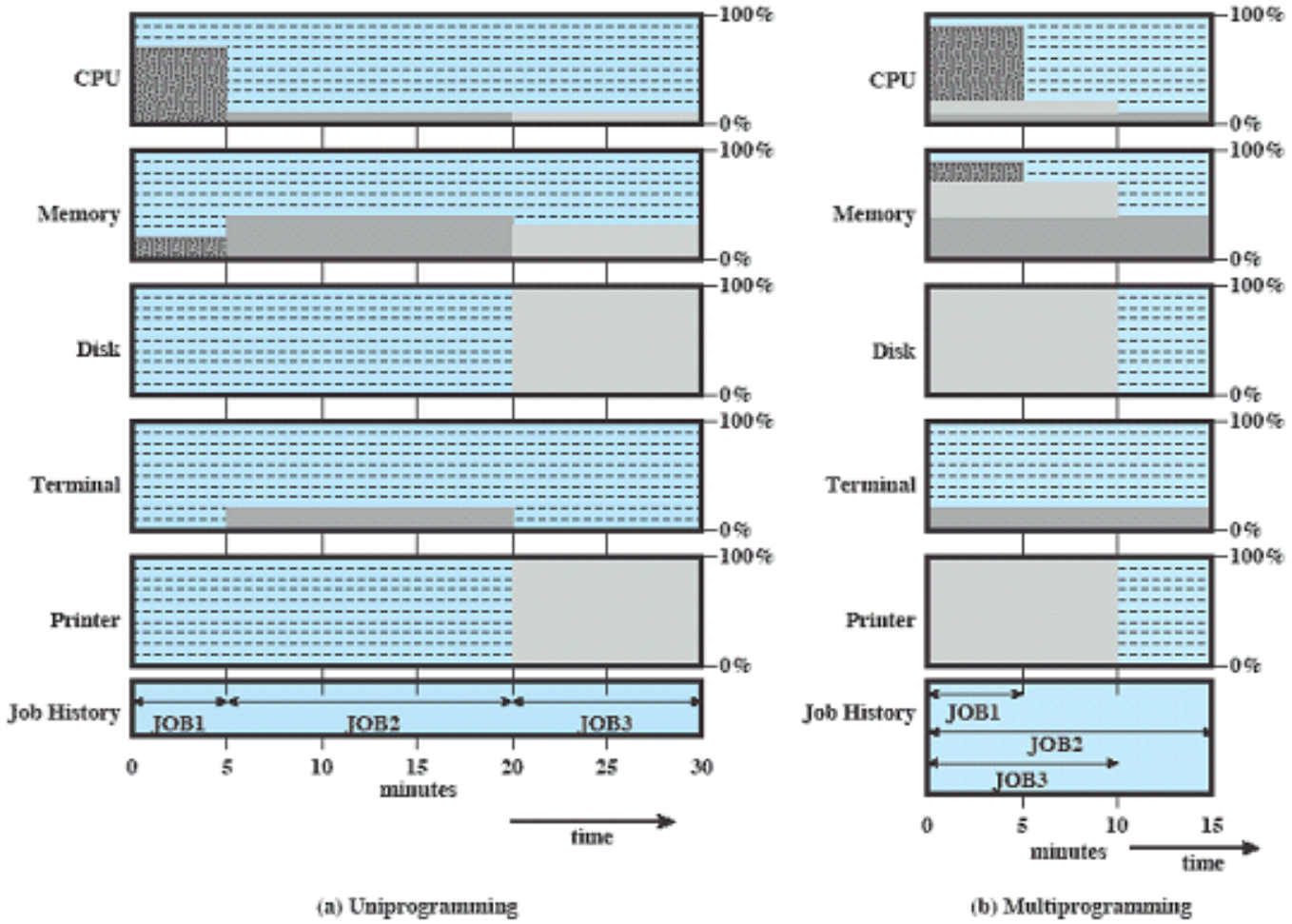
(c) Multiprogramming with three programs

Παράδειγμα πολυπρογραμματισμού

- Έστω έχουμε 3 εργασίες με τα παρακάτω χαρακτηριστικά:

	JOB1	JOB2	JOB3
Type of job	Heavy compute	Heavy I/O	Heavy I/O
Duration	5 min	15 min	10 min
Memory required	50 M	100 M	75 M
Need disk?	No	No	Yes
Need terminal?	No	Yes	No
Need printer?	No	No	Yes

Σύγκριση μονοπρογραμματισμού και πολυπρογραμματισμού (1/2)



Σύγκριση μονοπρογραμματισμού και πολυπρογραμματισμού (2/2)

	Uniprogramming	Multiprogramming
Processor use	22%	43%
Memory use	30%	67%
Disk use	33%	67%
Printer use	33%	67%
Elapsed time	30 min	15 min
Throughput rate	6 jobs/hr	12 jobs/hr
Mean response time	18 min	10 min

Συστήματα καταμερισμού χρόνου

- Σε πολλές περιπτώσεις, ο χρήστης πρέπει να έχει άμεση και διαλογική (interactive) πρόσβαση στο σύστημα.
- Σε αυτές τις περιπτώσεις οι χρήστες χρησιμοποιούν το σύστημα μέσω της χρήσης τερματικών σταθμών.
- Η ΚΜΕ πρέπει να διαμοιράζει το χρόνο της, εξυπηρετώντας ταυτόχρονα τις ανάγκες όλων των χρηστών.

Οι συγκρουόμενες απαιτήσεις των συστημάτων δέσμης και των διαλογικών συστημάτων

	Πολυπρογραμματισμός δέσμης	**Χρονομερισμός**
Κύριος στόχος	Μεγιστοποίηση χρήσης επεξεργαστή	Ελαχιστοποίηση χρόνου απόκρισης
Πηγή των οδηγιών για το λειτουργικό σύστημα	Εντολές γλώσσας εργασίας ελέγχου που παρέχονται με την εργασία	Εντολές δοσμένες στο τερματικό

Το πρώτο ΛΣ καταμερισμού χρόνου CTSS

- Compatible Time-Sharing System (CTSS):
  – Αναπτύχθηκε στο MIT στα πλαίσια του προγράμματος MAC.
- Το σύστημα είχε 32.000 λέξεις μνήμης, από τις οποίες ο παρακολουθητής χρησιμοποιούσε τις 5.000.
- Ένα πρόγραμμα φορτωνόταν στη μνήμη ξεκινώντας πάντα από τη θέση 5,000.
- Κάθε 0,2 δευτερόλεπτα ένας διακόπτης μετέφερε τον έλεγχο του συστήματος στον παρακολουθητή ο οποίος είχε τη δυνατότητα να δώσει την ΚΜΕ σε ένα άλλο πρόγραμμα για να εκτελεσθεί.
- Αν υπήρχε ανάγκη, το πρόγραμμα από το οποίο έφευγε ο έλεγχος χρήσης της ΚΜΕ αποθηκευόταν στο δίσκο, για να φορτωθεί ξανά σε κάποιο μελλοντικό στάδιο.

ICs and Multiprogramming
3η γενιά: Προβλήματα

- Τα βασικά χαρακτηριστικά της 3ης γενιάς αποτελούν ουσιαστικά και προβλήματα τα οποία πρέπει να αντιμετωπισθούν.
- Η ταυτόχρονη ύπαρξη πολλών προγραμμάτων στη μνήμη, δημιουργεί την **ανάγκη διαφύλαξης** των δεδομένων και πληροφοριών ενός προγράμματος από τα υπόλοιπα (π.χ. ένα πρόγραμμα να τροποποιεί τα δεδομένα κάποιου άλλου προγράμματος).
- Με την ταυτόχρονη πρόσβαση στο σύστημα από πολλούς χρήστες, τα αρχεία ενός χρήστη πρέπει να προστατευθούν από **μη εξουσιοδοτημένη πρόσβαση** σε αυτά από άλλους χρήστες.
- Ο ανταγωνισμός από πολλαπλά προγράμματα και χρήστες για **πρόσβαση σε κοινούς πόρους** του συστήματος πρέπει να επιλυθεί με τρόπο δίκαιο και ικανοποιητικό για όλους τους εμπλεκόμενους.

4η γενιά

Personal Computers 4η γενιά (1/2)

- Ο Η/Υ αποτελείται από κυκλώματα μεγάλης και πολύ μεγάλης ολοκλήρωσης (LSI, VLSI) τα οποία μείωσαν δραστικά το κόστος του με αποτέλεσμα την ανάπτυξη προσωπικών Η/Υ.
- Ανάπτυξη Λ.Σ. ενός χρήστη με έμφαση στη φιλικότητα προς τον χρήστη (π.χ. DOS, OS/2, Windows, OS X, android).
- Ανάπτυξη ισχυρών σταθμών εργασίας (workstations) και δικτύων που οδήγησαν στην ανάπτυξη λειτουργικών συστημάτων για δίκτυα και κατανεμημένων (distributed) λειτουργικών συστημάτων.

Personal Computers 4η γενιά (2/2)

- Αλματώδης αύξηση της απόδοσης του υλικού (MIPS).
- Η δημιουργία του WWW αύξησε την κατανεμημένη επεξεργασία και οδήγησε στην ανάγκη ενσωμάτωσης διαδικτυακών διεργασιών.
- Καθιέρωση της αντικειμενοστραφούς τεχνολογίας.
- Διάδοση και ανάπτυξη της τεχνολογίας ανοικτού κώδικα.
- Εμφάνιση του LINUX.

Personal Computers
4η γενιά (2000 και μετά)

- Εμφάνιση του middleware (λογισμικό που συνδέει δύο ξεχωριστές εφαρμογές – συχνά σε δίκτυο).
- Εφαρμογές (web services) που δημοσιεύονται στο Internet και χρησιμοποιούνται από χρήστες μέσω συνδέσεων υψηλών ταχυτήτων (DSL κλπ).
- Εμφάνιση βελτιωμένων αρχιτεκτονικών δικτύων και αύξηση της παράλληλης επεξεργασίας.
- Χρήση Λ.Σ. τύπου POSIX (Portable Operating System Interface).
- Υπολογιστική δυνατότητα σε φορητές συσκευές (PDA's, cell phones κλπ).

Ο ζωολογικός κήπος των λειτουργικών συστημάτων

- Mainframe operating systems.
- Server operating systems.
- Multiprocessor operating systems.
- Personal computer operating systems.
- Handheld operating systems.
- Embedded operating systems.
- Sensor node operating systems.
- Real-time operating systems.
- Smart card operating systems.
- .....

Κάποιες Κατηγορίες ΛΣ (1/2)

Λ.Σ. Δικτύων (Network operating systems):

Σε αυτά τα συστήματα το Λ.Σ. δίνει την δυνατότητα επικοινωνίας με άλλες μηχανές συνδεδεμένες με το ίδιο δίκτυο. Για παράδειγμα, διεργασίες σε ένα υπολογιστή μπορούν να ζητήσουν αρχεία να μεταφερθούν από ένα άλλο υπολογιστή (ftp). Ακόμα μπορούν να κάνουν “remote login” σε άλλους υπολογιστές και να χρησιμοποιήσουν τους πόρους τους.

Κατανεμημένα Λ.Σ. (Distributed operating systems):

Τα κατανεμημένα Λ.Σ. παρέχουν επίσης τις παραπάνω δυνατότητες. Αλλά με τρόπο διαφανή. Ο χρήστης δεν χρειάζεται να γνωρίζει ποιο αρχείο έχει αποθηκευτεί σε ποιον υπολογιστή κ.λπ. ή γενικά να γνωρίζει τίποτα περί κατανομής. Έτσι φαίνεται ότι το Λ.Σ. είναι ένα κεντρικό και όχι κατανεμημένο Λ.Σ.

Κάποιες Κατηγορίες ΛΣ (2/2)

Λ.Σ. Πολυμέσων:
Βασική έννοια η υποστήριξη βίντεο και ήχου – ροών (continuous media). Ιδιαιτερότητες στον χρονοπρογραμματισμό / χρονομερισμό για την ευαισθησία στον χρονισμό παρουσίασης.

Λ.Σ. Πραγματικού χρόνου:
Ιδιαιτερότητες στον χρονοπρογραμματισμό / χρονομερισμό «επιεικείς ή αυστηρές προθεσμίες» (soft ή hard deadlines) – διαφοροποίηση σχετικά με τις συνέπειες μη ικανοποίησης προθεσμιών.

Βαθμός δυσκολίας ανάπτυξης ενός Λ.Σ

- Το Λ.Σ. είναι ένα από τα πιο πολύπλοκα είδη λογισμικού που αναπτύσσονται.
- Μερικά σχετικά στατιστικά στοιχεία:
  - Το Multics ανακοινώθηκε το 1963 και λειτούργησε το 1969.
  - Το OS 360 όταν πρωτοβγήκε στην αγορά είχε 1000 λάθη.
  - Το Windows NT υπέφερε επί 7 χρόνια με προβλήματα.
  - Το Windows Vista ουσιαστικά ποτέ δεν καθιερώθηκε και αντικαταστάθηκε πολύ γρήγορα από το Windows 7.
  - Ένα μικρό Λ.Σ. έχει μέγεθος 100K και ένα μεγάλο 10M γραμμών κώδικα.
  - Η δημιουργία ενός Λ.Σ. χρειάζεται μεταξύ 100-1000 ανθρωποχρόνων (man years).

Γιατί υπάρχουν τόσα πολλά ΛΣ;

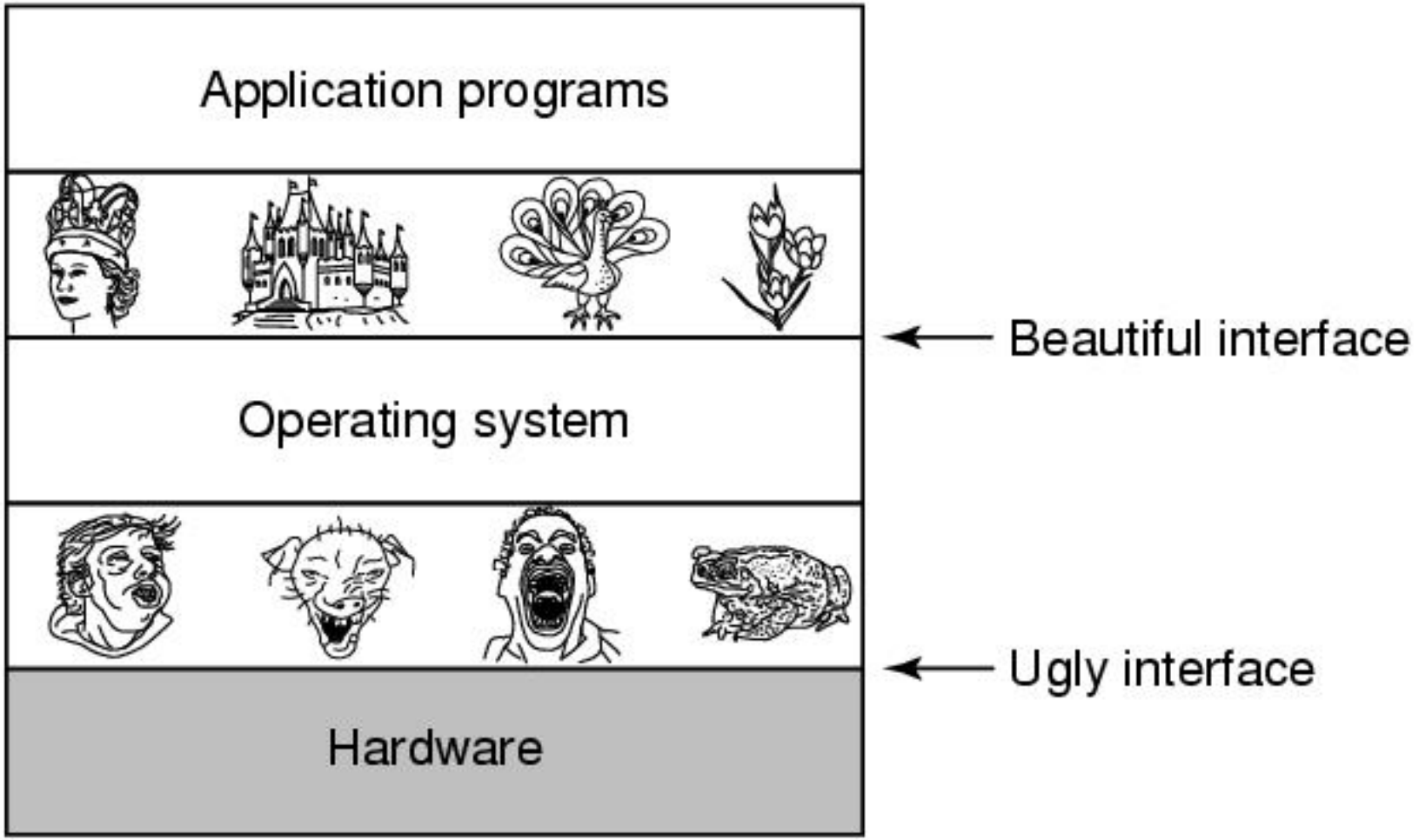
- Υπάρχουν διαφορετικές ανάγκες.
  - γενικού σκοπού, διακομιστές, ενσωματωμένα συστήματα, παιχνιδομηχανές, βιομηχανικά συστήματα, πραγματικού χρόνου.....
- Υπάρχουν (υπήρχαν) διαφορετικές εταιρίες.
  - Microsoft, IBM, SCO, Nokia, Google...
- Υπάρχουν εθελοντές.
  - Linux, FreeBSD,..
- Υπάρχει διαφορετικό hardware (διαφορετικό ISA).
  - Intel x86, IBM Cell, ARM, AVR, TI,...

Ποια είναι τα βασικά υποσυστήματα των ΛΣ;

- **Διαχείριση μνήμης.**
  - Παρέχει το ποσό της μνήμης που απαιτεί η κάθε διεργασία. Σελιδοποιεί τη μνήμη. Προστατεύει τη μνήμη.
- **Διαχείριση Αρχείων.**
  - Δημιουργία, Μετονομασία, εγγραφή, διαγραφή και άλλες λειτουργίες. Προστασία.
- **Χρονοπρογραμματισμός Διεργασιών.**
  - Πότε για πόσο και σε ποιον επεξεργαστή θα εκτελεστεί μια διεργασία.
- **Διαχείριση και προστασία πόρων.**
  - Πότε θα δοθεί πρόσβαση και με τι δικαιώματα;
- **Διαχείριση εισόδου εξόδου.**

Παράρτημα

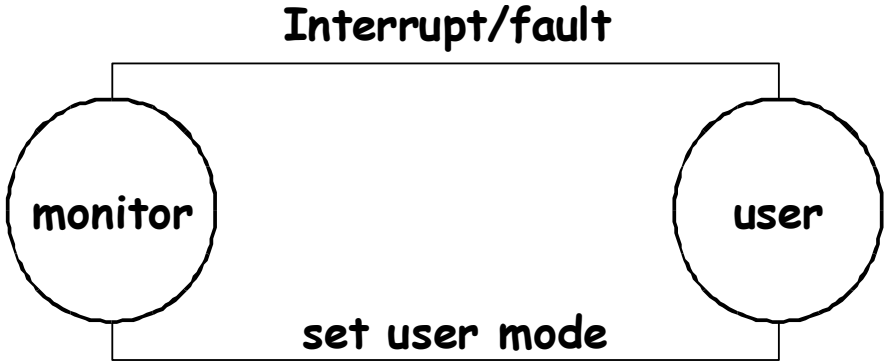
Το ΛΣ ως μηχανή αφαίρεσης κρύβει την “άσχημη” πλευρά του hardware



Εναλλαγή kernel/user mode (1/2)

Ένα mode bit προστίθεται στο υλικό για να δείχνει την τρέχουσα κατάσταση: monitor (0) ή user (1).

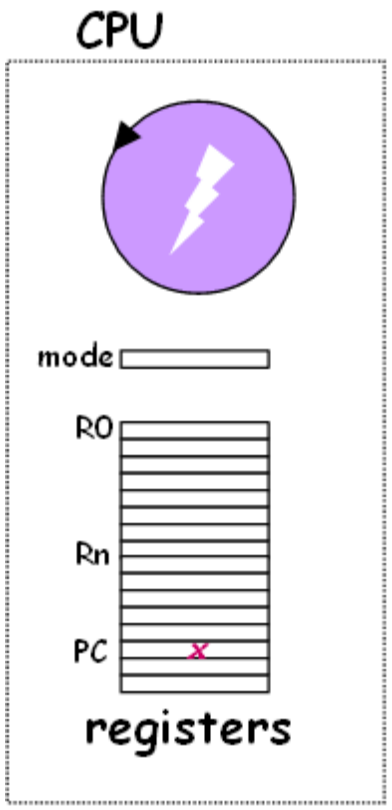
Όταν συμβεί μια διακοπή ή ένα λάθος το υλικό εναλλάσσεται σε monitor mode.

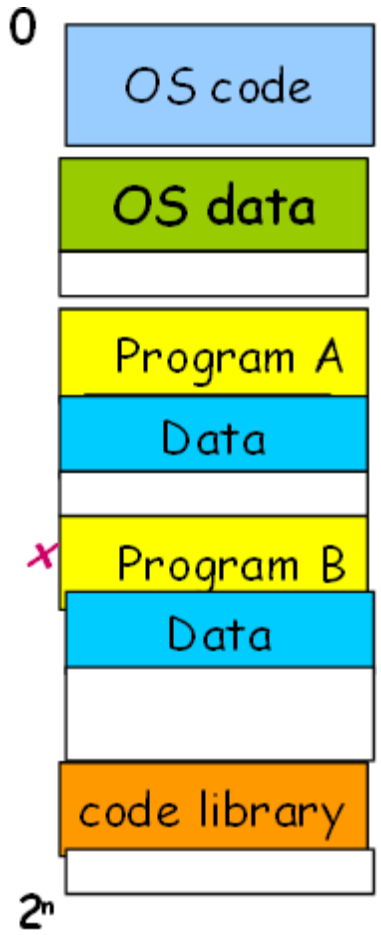


Οι προνομιούχες εντολές (Privileged instructions) μπορούν να προκύψουν μόνον *σε* monitor mode.

Εναλλαγή kernel/user mode (2/2)

- Το mode register bit δείχνει αν η CPU εκτελεί ένα πρόγραμμα χρήστη ή βρίσκεται σε κατάσταση προστασίας του πυρήνα.
- Ορισμένες εντολές ή προσπελάσεις σε δεδομένα είναι δυνατές μόνον όταν η CPU λειτουργεί σε kernel mode.





Εναλλαγή kernel/user mode: Προστασία

- Όλες οι εντολές I/O είναι προνομιούχες εντολές.
- Εξασφαλίζεται έτσι ότι ένα πρόγραμμα χρήστη δεν θα μπορεί ποτέ να αποκτήσει τον έλεγχο του υπολογιστή σε monitor mode (π.χ. ένα πρόγραμμα χρήστη, κατά την εκτέλεσή του να αποθηκεύσει μια νέα διεύθυνση στον πίνακα διανυσμάτων διακοπών).

Παράδειγμα (1/2)

Θεωρήστε ένα πρόγραμμα που θέλει να δημιουργήσει ένα αρχείο με δεδομένα. Τα αρχεία (files) αποθηκεύονται σε δίσκους (π.χ. μαγνητικούς «σκληρούς»), οι οποίοι, εν ολίγοις, αποτελούνται από ένα ηλεκτρονικό τμήμα (ένα controller - επεξεργαστή) και ένα μαγνητικό τμήμα (επιφάνειες δίσκων πάνω στις οποίες γράφουν κεφαλές).

Ο επεξεργαστής δέχεται εντολές για να:

- γράψει (write) ή να,
- ανακτήσει δεδομένα (read) σε (από) συγκεκριμένες διευθύνσεις,
- να μετακινήσει τις κεφαλές σε καινούριες διευθύνσεις, κ.λπ.

Παράδειγμα (2/2)

Η επικοινωνία με τον controller απαιτεί :

- το γράψιμο ειδικών εντολών καθώς και το,
- γράψιμο των παραμέτρων σε συγκεκριμένες διευθύνσεις στη RAM.

Η δομή και η μορφή αυτών των εντολών διαφέρει από controller σε controller. Επιπλέον, ο controller επιστρέφει ειδικούς κώδικες σε ειδική μορφή οι οποίοι πρέπει να αναλυθούν για να επαληθευθεί ότι όλα πήγαν καλά, κ.λπ. Αυτοί οι κώδικες επικοινωνίας είναι πολύπλοκοι, και τα προγράμματα επικοινωνίας με τέτοιες συσκευές περιφέρειας είναι επίσης πολύπλοκα (και μεγάλα)!

Το Λ.Σ. **αναλαμβάνει να απαλλάξει τον προγραμματιστή** από όλες αυτές τις δυσκολίες.

Από αυτή τη δεύτερη θεώρηση, προκύπτει ότι το Λ.Σ. ενεργεί σαν μία μηχανή αφαίρεσης (abstraction machine).

Πως υλοποιείται το system call;

- Το κάθε system call υλοποιείται μέσω μίας ρουτίνας η οποία βρίσκεται σε μία βιβλιοθήκη που συνδέεται (linked) με τον κώδικα της user process. Όλες οι ρουτίνες αυτής της βιβλιοθήκης εκτελούν μία ειδική εντολή, που ονομάζεται TRAP. Η εντολή TRAP είναι αυτή η οποία αλλάζει το σύστημα από user mode σε kernel mode (αλλάζοντας ένα bit σε ένα CPU register το οποίο ορίζει τον τρόπο λειτουργίας του συστήματος).

Επιστροφή από system call

- Επιπλέον, η ρουτίνα που καλεί TRAP είναι υπεύθυνη να τοποθετήσει τις παραμέτρους του system call σε **μία προσυμφωνημένη διεύθυνση**, (συνήθως, CPU registers ή ακόμα και στη στοίβα (stack)) όπου ο πυρήνας θα τις βρει.
- Όταν ο πυρήνας τελειώσει, τότε τοποθετεί επιστρεφόμενες πληροφορίες σε registers και εκτελεί ένα RETURN FROM TRAP ενεργοποιώντας πάλι τη ρουτίνα της βιβλιοθήκης.
- Αυτή η ρουτίνα, επιστρέφει την πληροφορία από τους registers στη user process.

Εναλλακτικοί ορισμοί ΛΣ

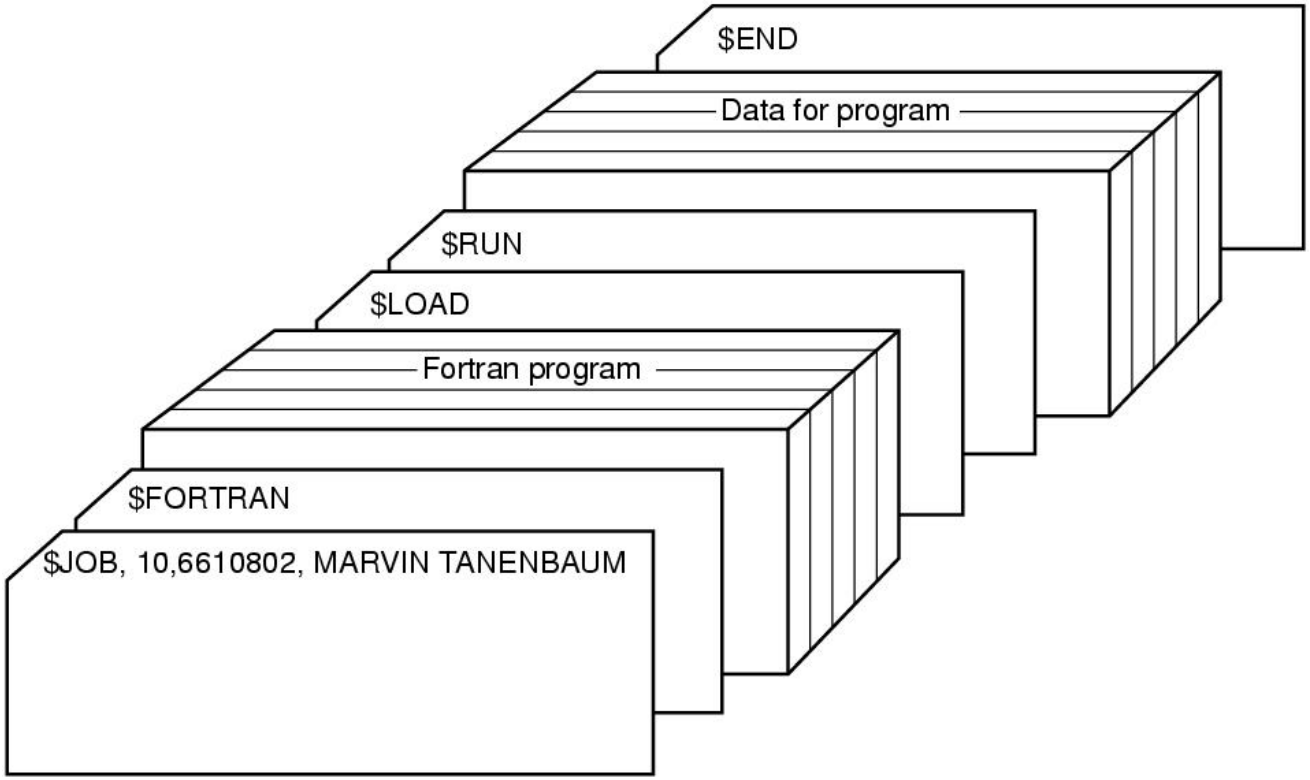
- **top-down**

Παρέχει στα προγράμματα εύκολη και αποδοτική πρόσβαση στους πόρους του συστήματος.

- **bottom-up**

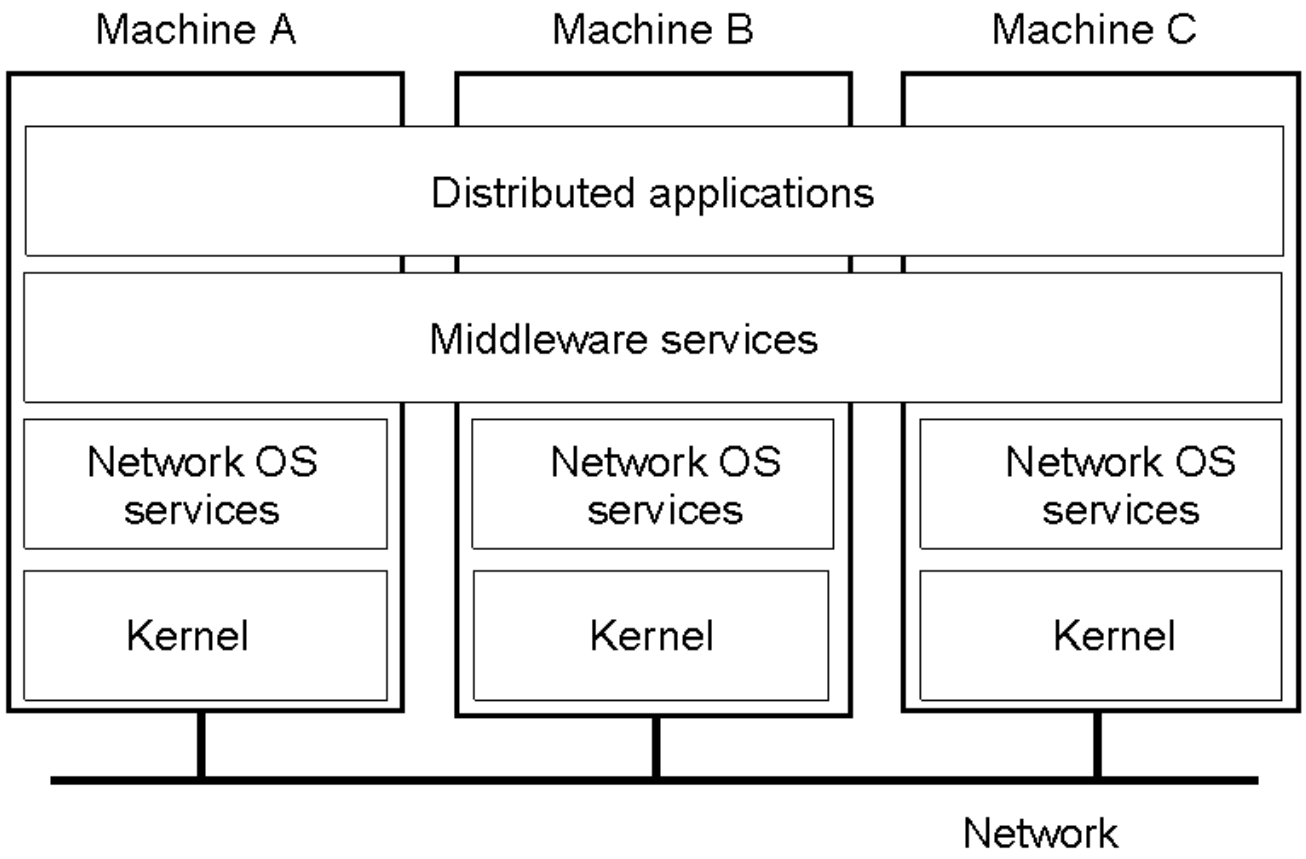
"Παρέχει μια συστηματοποιημένη και ελεγχόμενη κατανομή των επεξεργαστών, των μνημών και των άλλων συσκευών Ε/Ε, ανάμεσα στα διάφορα προγράμματα-πελάτες που ανταγωνίζονται μεταξύ τους για να τα χρησιμοποιήσουν" (Tanenbaum, 2001).

Transistors and Batch Systems

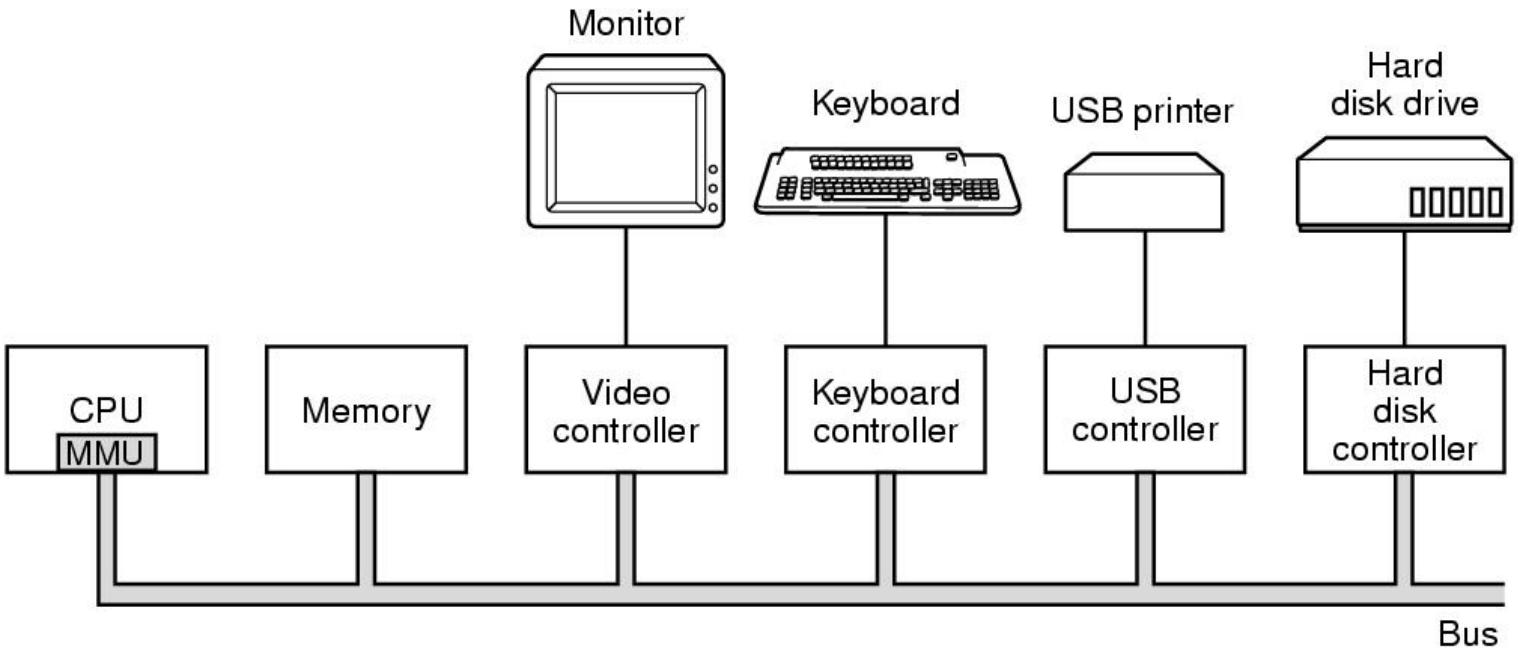


Δομή ενός προγράμματος με κάρτες.

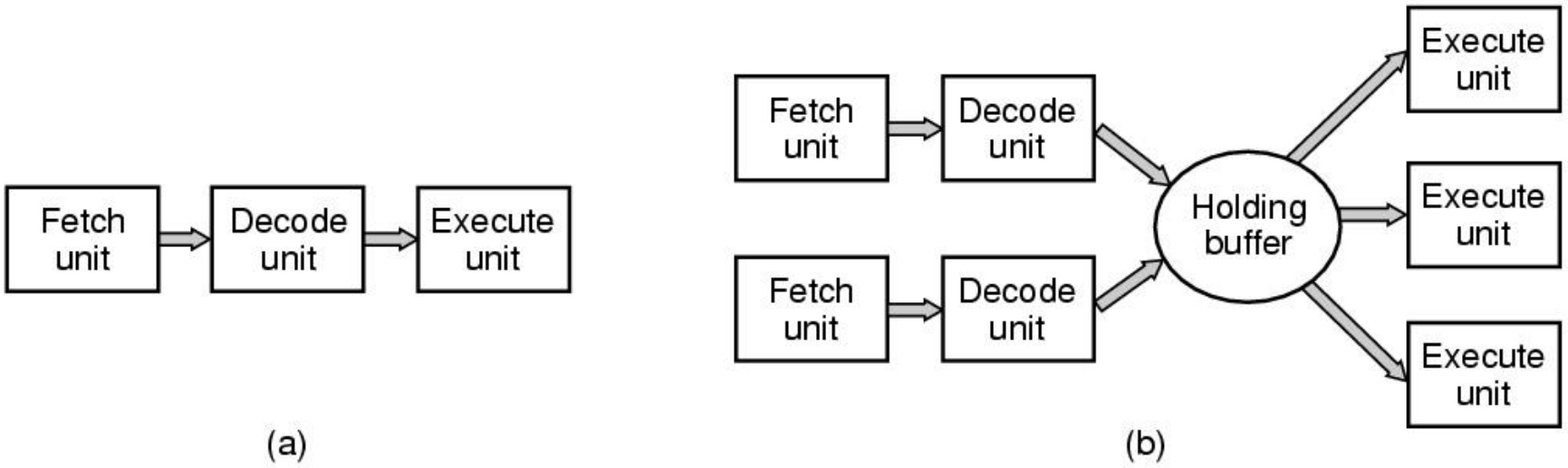
Middleware-based Systems



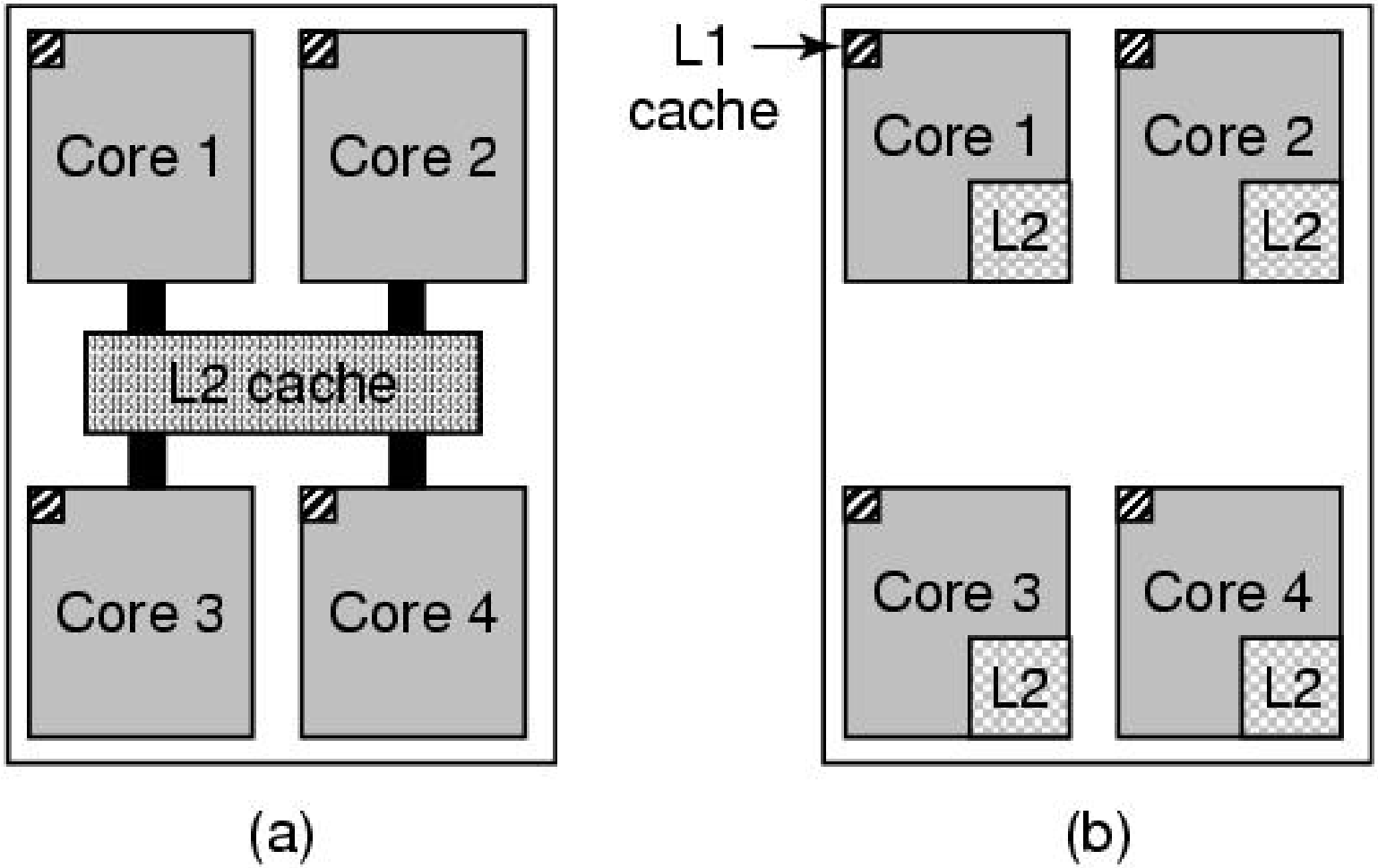
Κάποια συστήματα του προσωπικού υπολογιστή



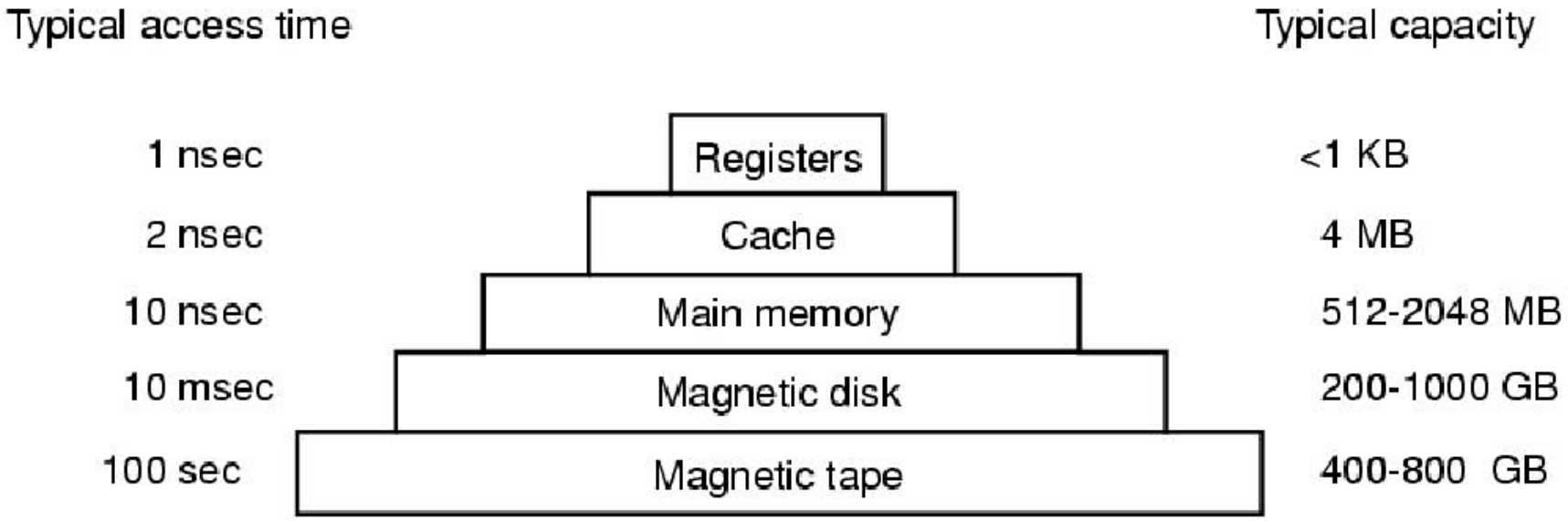
Η έννοια της διασωλήνωσης



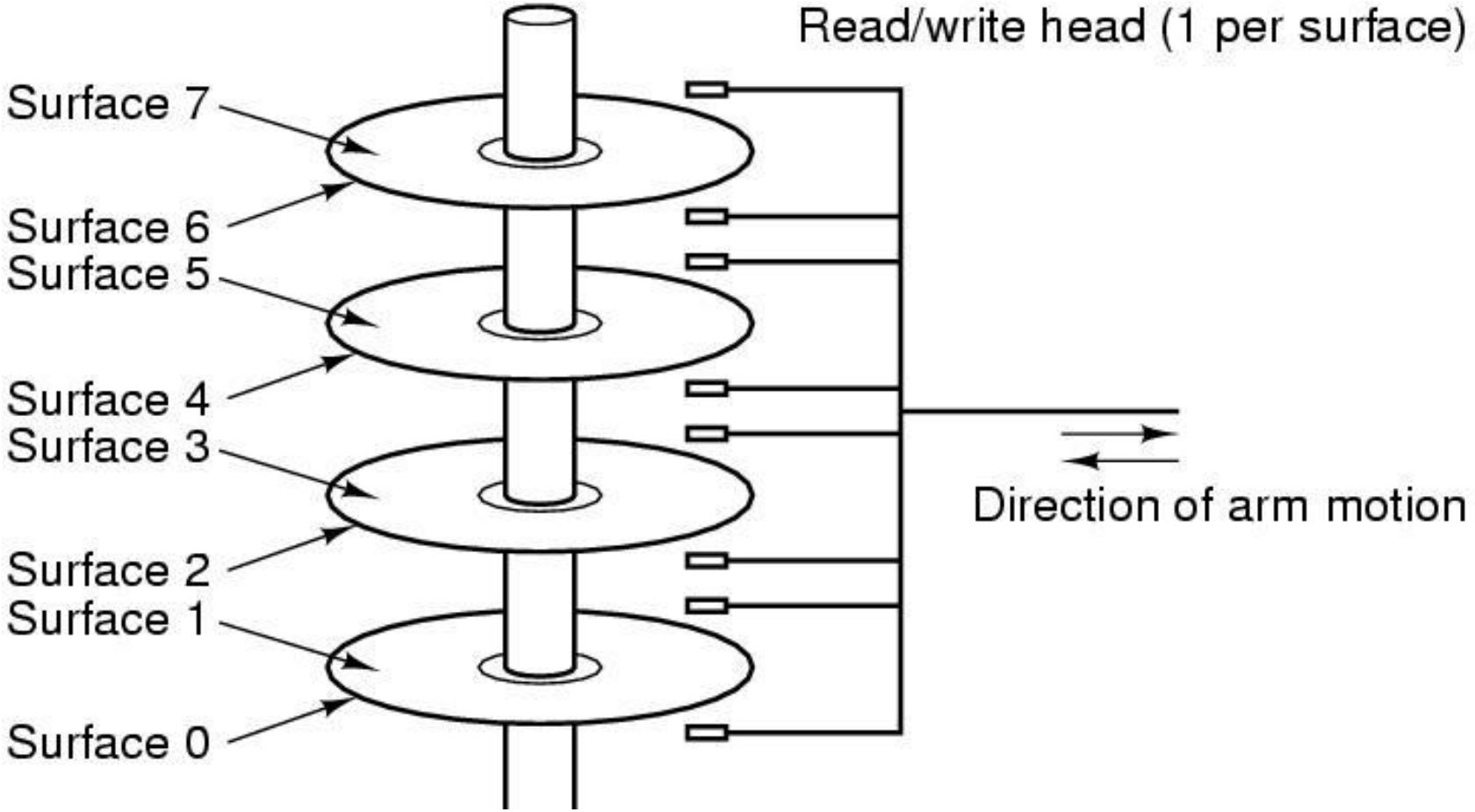
Πολυπύρηνα συστήματα



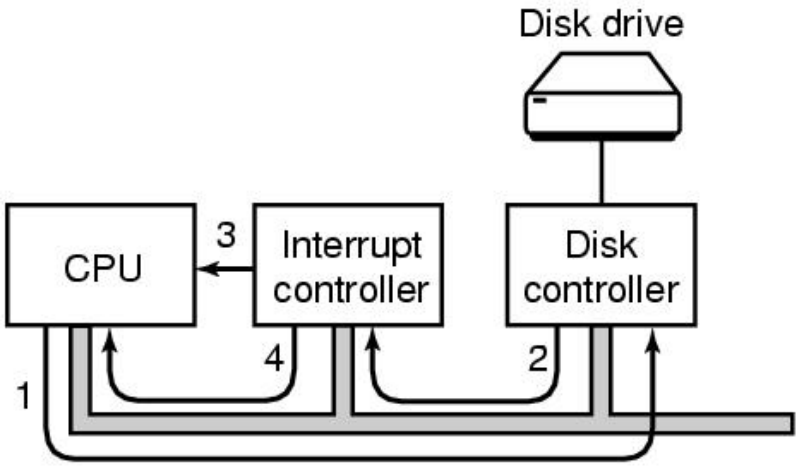
Η ιεραρχία της μνήμης

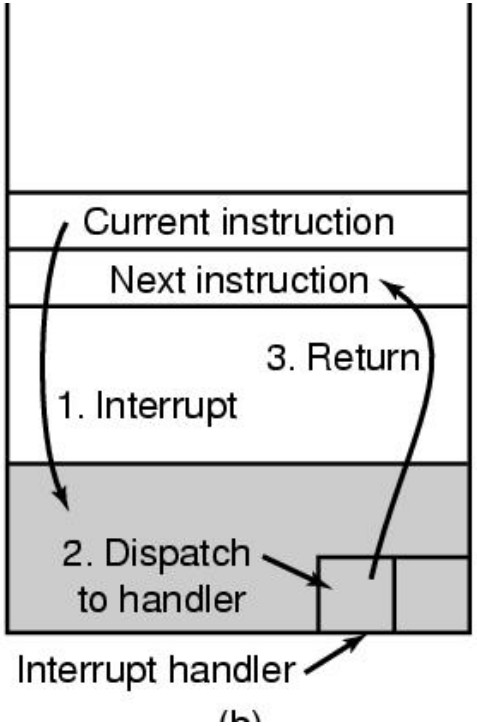


Δομή των μαγνητικών δίσκων

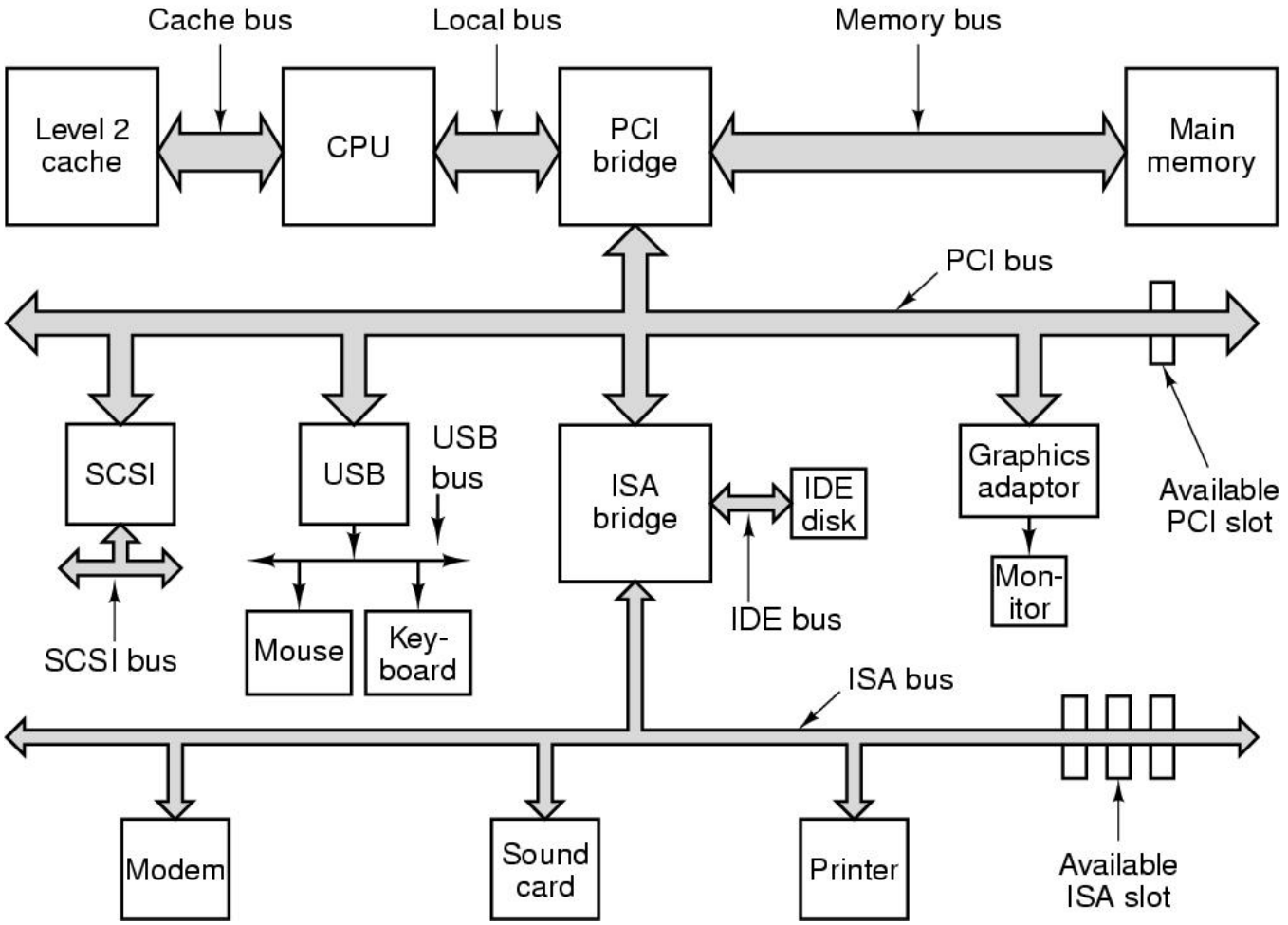


Είσοδος-Έξοδος μέσω IO





Δίαυλοι σε Pentium



Τέλος Ενότητας