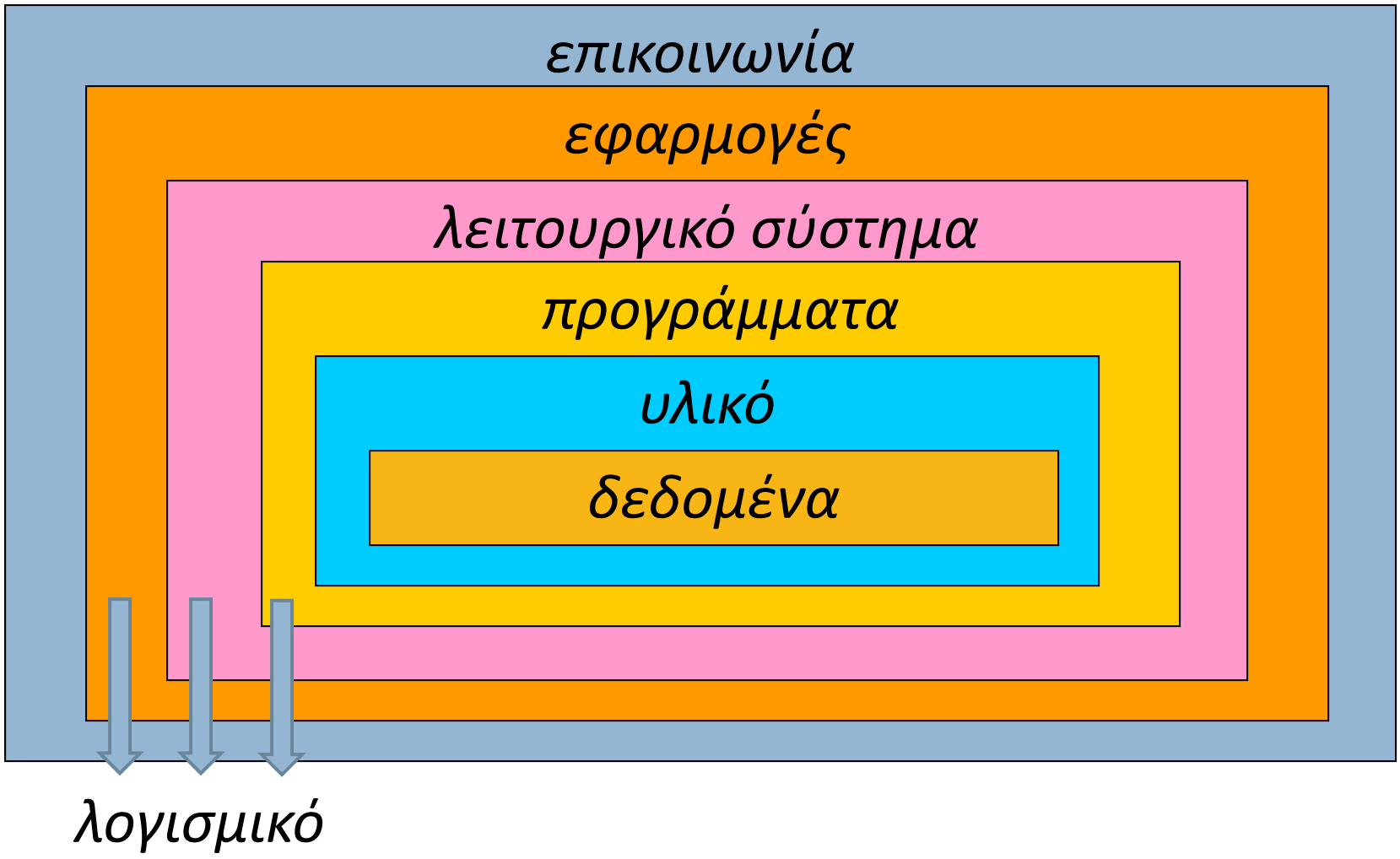
ΕΠΛ 001:
ΕΙΣΑΓΩΓΗ ΣΤΗΝ
ΕΠΙΣΤΗΜΗ ΤΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

Λειτουργικά συστήματα

Στόχοι

- Να εξηγήσουμε τη διάκριση μεταξύ ***λογισμικού συστημάτων*** και ***λογισμικού εφαρμογών***.
- Να περιγράψουμε τι είναι τα ***λειτουργικά συστήματα*** και να δούμε σύντομα την ιστορία της εξέλιξής τους.
- Να περιγράψουμε τις σημαντικότερες έννοιες και τεχνικές γύρω από τη ***διαχείριση μνήμης***.
- Να περιγράψουμε τις βασικές έννοιες σε σχέση με τη ***διαχείριση διεργασιών***.
- Να περιγράψουμε κάποιους απλούς αλγορίθμους για τον ***χρονοπρογραμματισμό της ΚΜΕ***.

Υπολογιστικά συστήματα: Στρώματα



Λογισμικό

Λογισμικό (*software*) είναι το σύνολο των *προγραμμάτων* που μπορεί να εκτελέσει ένα υπολογιστικό σύστημα.

Λογισμικό: Δύο κατηγορίες

Λογισμικό συστημάτων

π.χ.:

πρόγραμμα που στέλνει κείμενα στον εκτυπωτή

πρόγραμμα που φορτώνει δεδομένα από τον σκληρό δίσκο στην κύρια μνήμη

πρόγραμμα που εγκαθιστά την κάμερα

πρόγραμμα που επιλέγει ποια διεργασία θα εκτελέσει η ΚΜΕ

Λογισμικό εφαρμογών

π.χ.:

πρόγραμμα επεξεργασίας κειμένων (word processor)

ιστοπλοηγός (web browser)

πρόγραμμα ηλεκτρονικού ταχυδρομείου

πρόγραμμα ζωγραφικής

πρόγραμμα διαχείρισης βάσης δεδομένων

πρόγραμμα επεξεργασίας φύλλων εργασίας

Λογισμικό: Δύο κατηγορίες

Λογισμικό συστημάτων

Λογισμικό συστημάτων (*system software*) είναι τα προγράμματα που εξυπηρετούν ανάγκες του υλικού του Η/Υ.

Η ύπαρξη και λειτουργία του δεν γίνεται άμεσα αντιληπτή στον χρήστη.

Χρησιμοποιείται κατά την κατασκευή ή εκτέλεση του λογισμικού εφαρμογών.

Λογισμικό εφαρμογών

Λογισμικό εφαρμογών (*applications software*) είναι τα προγράμματα που εξυπηρετούν ανάγκες του γενικού χρήστη.

Ο χρήστης έρχεται σε επαφή μαζί του καθημερινά.

Κατά την εκτέλεσή του χρησιμοποιεί το λογισμικό συστημάτων.

Λογισμικό: Δύο κατηγορίες

Λογισμικό: Δύο κατηγορίες

Λογισμικό συστημάτων

λειτουργικά συστήματα (*operating systems*): Unix, Windows, Ubuntu, κ.ά.

οδηγοί υλικού (*device drivers*): μεσολαβούν μεταξύ ΛΣ και μιας συσκευής.

προγράμματα υπηρεσιών (*utilities*): π.χ. μορφοποίηση δίσκου, παύση διεργασιών.

μεταγλωττιστές, -φραστές (assemblers, compilers): π.χ. μεταφραστής C++.

Λογισμικό εφαρμογών

λογισμικό γενικής χρήσης: π.χ. Emacs, Acrobat Reader, Firefox, Real Player, κ.ά.

εξειδικευμένο λογισμικό: π.χ. διαχείριση εστιατορίου, διαχείριση αποθήκης, λογιστικά πακέτα, κ.ά.

ολοκληρωμένο λογισμικό: συλλογή εφαρμογών με συγγενές αντικείμενο π.χ. MS Office, Lotus Notes.

Λειτουργικά συστήματα

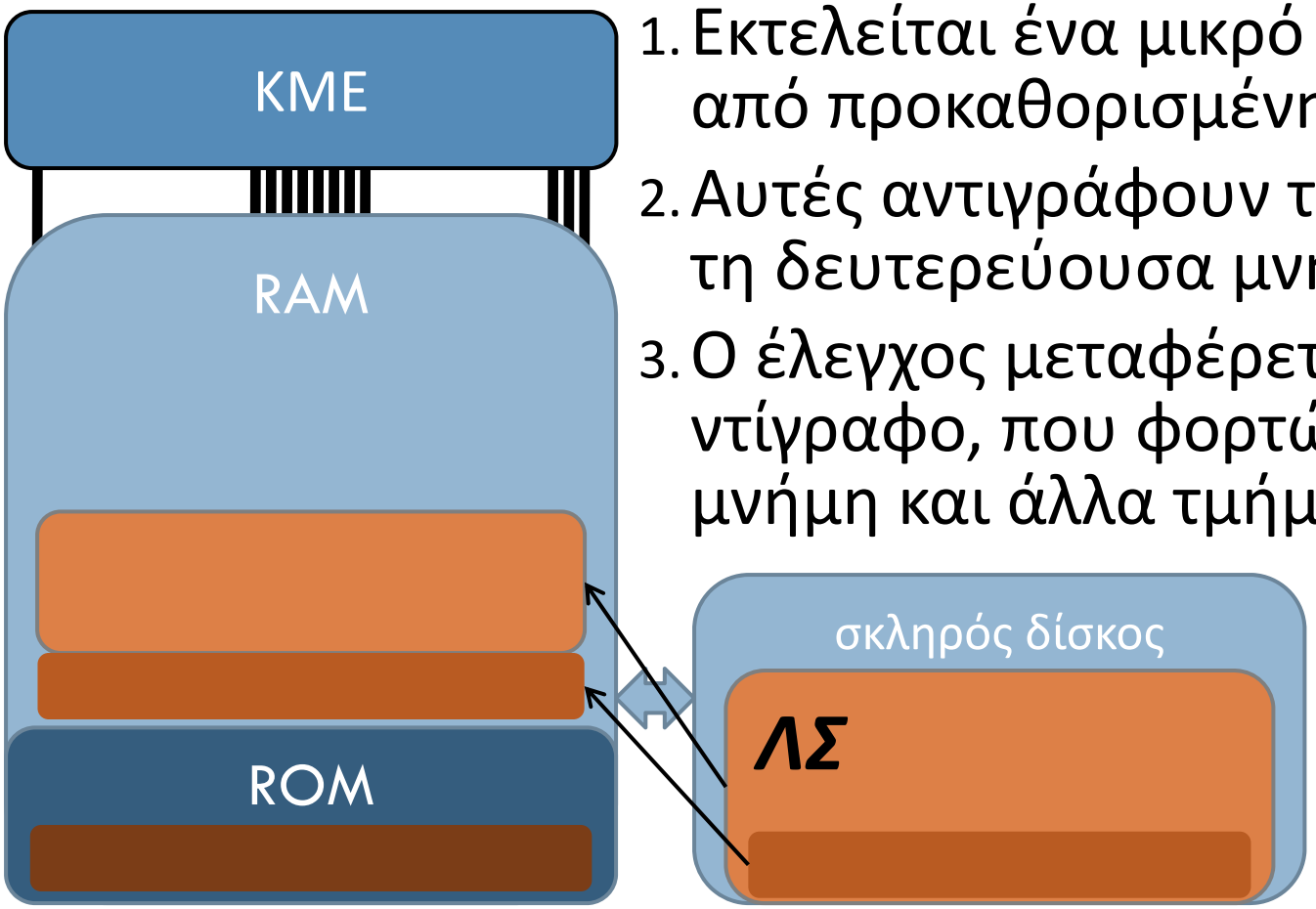
Λειτουργικό σύστημα (*ΛΣ, operating system,* OS) είναι ένα σύνολο προγραμμάτων που:

- προσφέρουν το περιβάλλον μέσω του οποίου ο χρήστης επικοινωνεί με το υλικό του Η/Υ.
- προσφέρουν το περιβάλλον μέσω του οποίου οι εφαρμογές εκτελούνται στο υλικό του Η/Υ.
- κατανέμουν τους πόρους του υπολογιστή στα προγράμματα που ζητούν να τους χρησιμοποιήσουν.
- διευκολύνουν τη ρύθμιση των παραμέτρων του υλικού του Η/Υ ώστε να λειτουργεί αποδοτικά.

Λειτουργικά συστήματα

Εκκίνηση λειτουργικού συστήματος

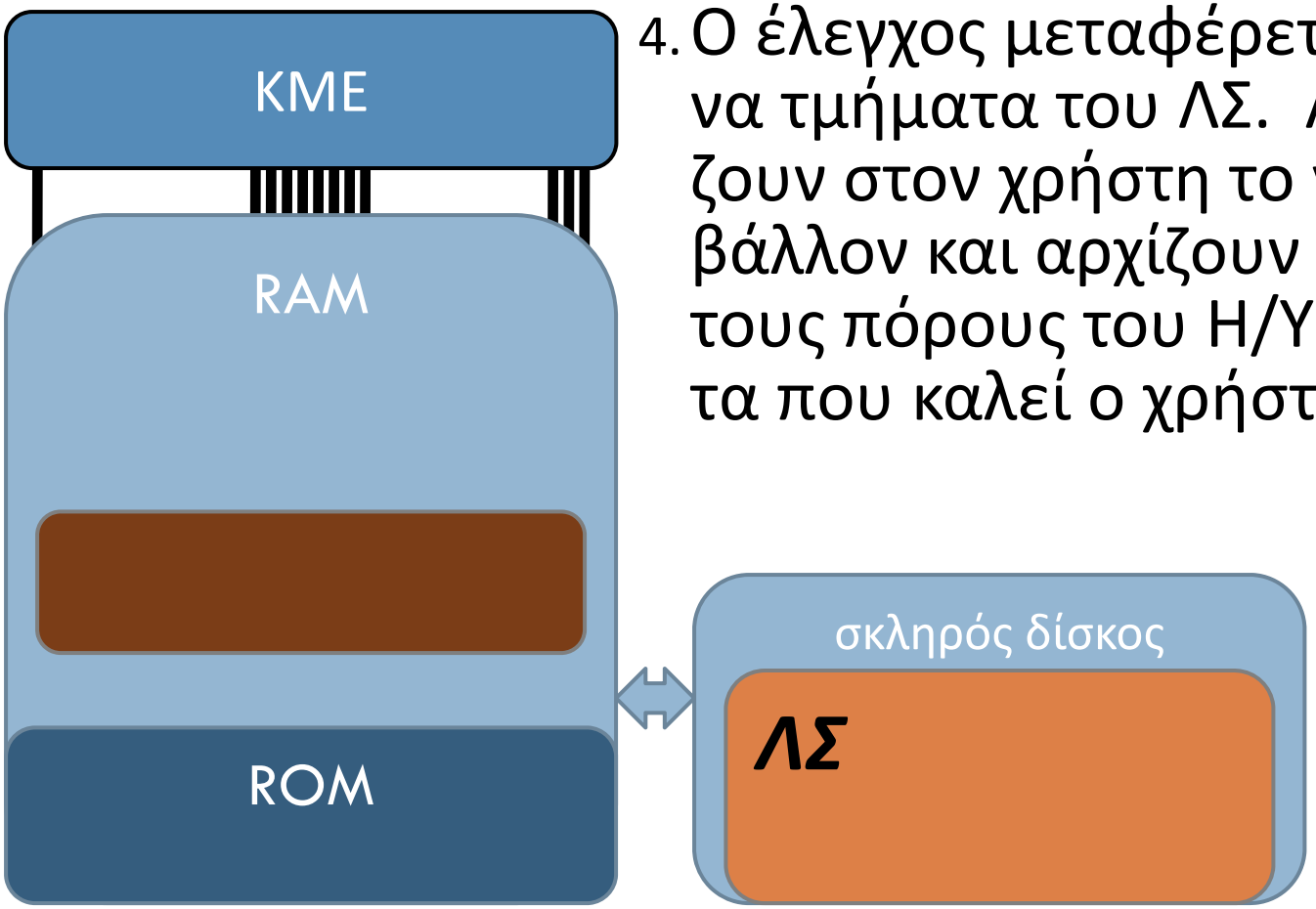
Το ΛΣ αναλαμβάνει τον έλεγχο λίγο μετά την εκκίνηση του Η/Υ. Αναλυτικά, κατά την εκκίνηση (booting process):



1. Εκτελείται ένα μικρό σύνολο εντολών από προκαθορισμένη θέση της ROM.
2. Αυτές αντιγράφουν τμήμα του ΛΣ από τη δευτερεύουσα μνήμη στην κύρια.
3. Ο έλεγχος μεταφέρεται σε αυτό το αντίγραφο, που φορτώνει στην κύρια μνήμη και άλλα τμήματα του ΛΣ.

Εκκίνηση λειτουργικού συστήματος

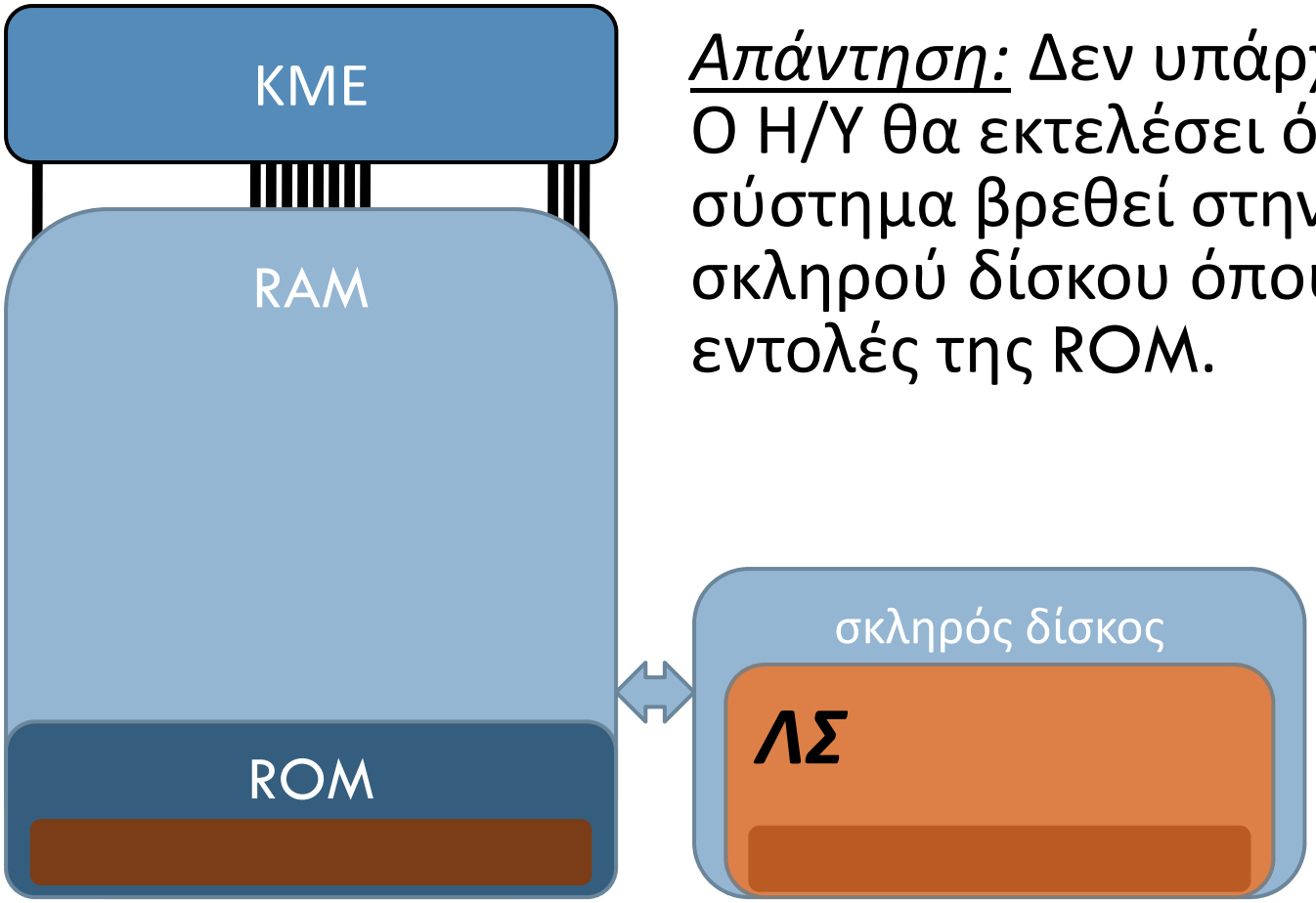
Το ΛΣ αναλαμβάνει τον έλεγχο λίγο μετά την εκκίνηση του Η/Υ. Αναλυτικά, κατά την εκκίνηση (booting process):



4. Ο έλεγχος μεταφέρεται στα φορτωμένα τμήματα του ΛΣ. Αυτά παρουσιάζουν στον χρήστη το γραφικό περιβάλλον και αρχίζουν να κατανέμουν τους πόρους του Η/Υ στα προγράμματα που καλεί ο χρήστης.

Εκκίνηση λειτουργικού συστήματος

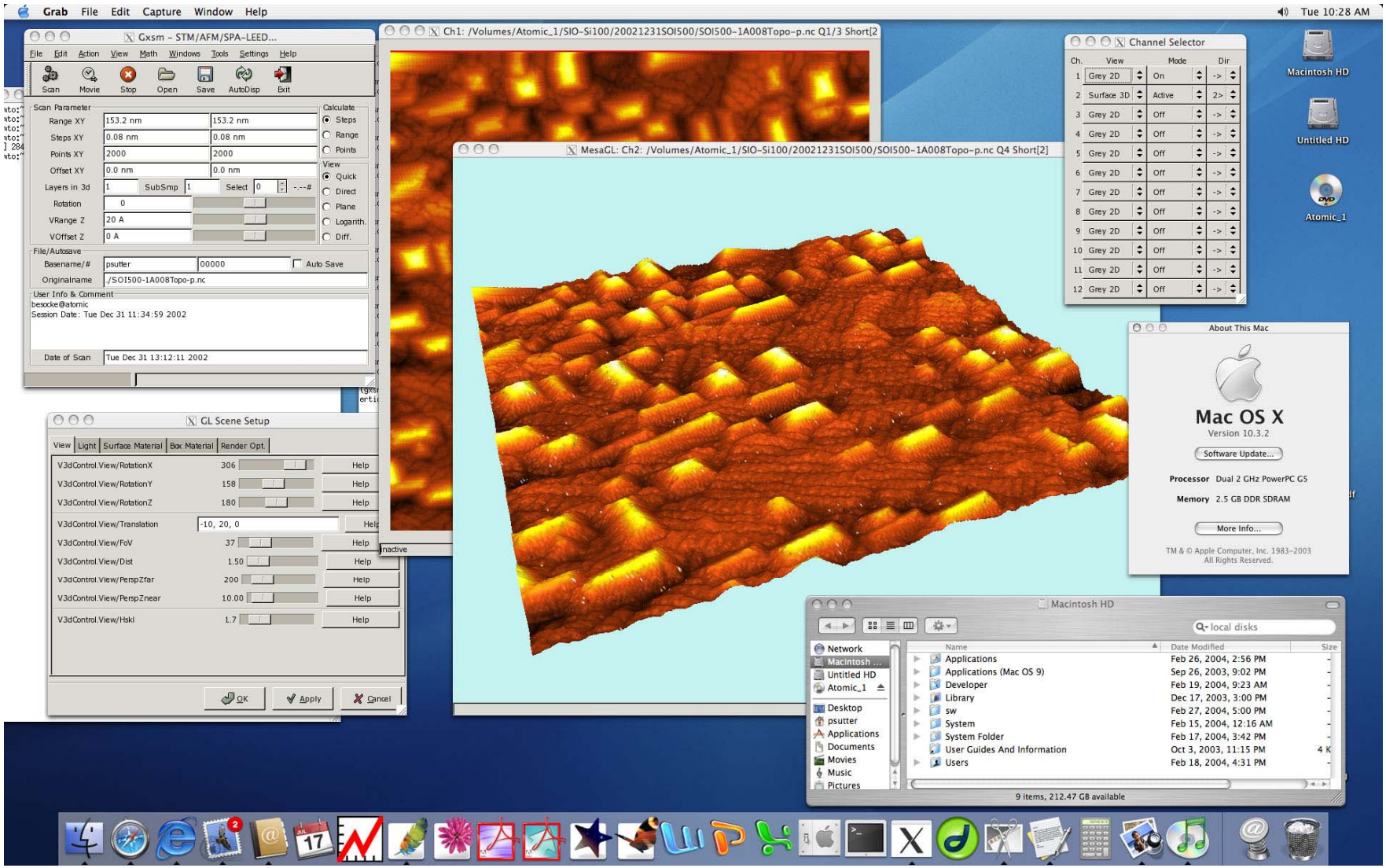
Ερώτηση: Ένας Η/Υ μπορεί να εκτελεί μόνο κάποιο συγκεκριμένο λειτουργικό σύστημα;



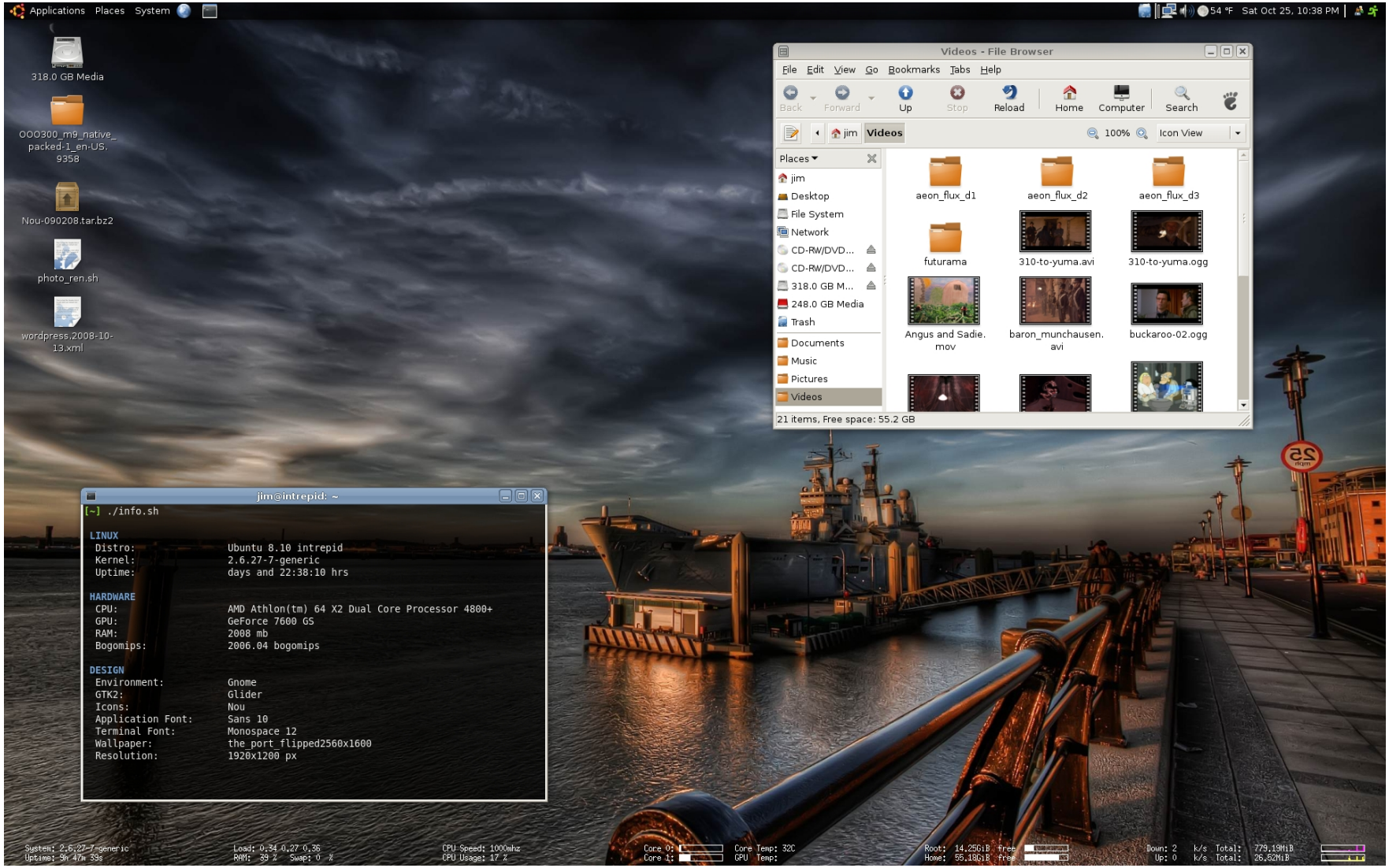
Απάντηση: Δεν υπάρχει περιορισμός. Ο Η/Υ θα εκτελέσει όποιο λειτουργικό σύστημα βρεθεί στην περιοχή του σκληρού δίσκου όπου ψάχνουν οι εντολές της ROM.

Παραδείγματα ΛΣ: Windows

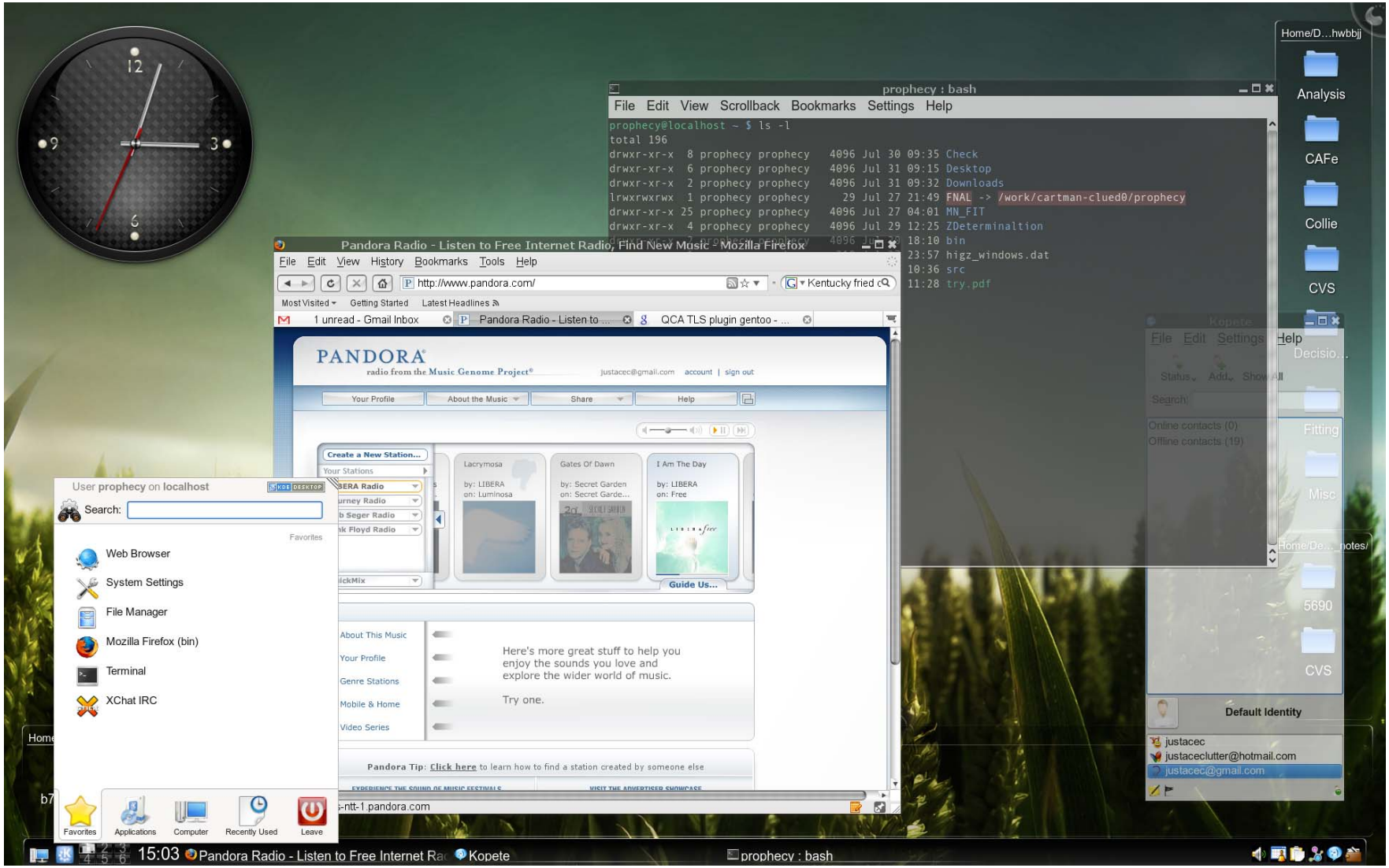
Παραδείγματα ΛΣ: Mac OS



Παραδείγματα ΛΣ: *Ubuntu*



Παραδείγματα ΛΣ: *Linux*



Παραδείγματα ΛΣ: *Palm* OS

Παραδείγματα ΛΣ: *Symbian OS*

Παραδείγματα ΛΣ: *Apple iOS*

Παραδείγματα ΛΣ: *Android OS*

Κατηγορίες ΛΣ

Τα λειτουργικά συστήματα μπορούν να διακριθούν:

Με βάση το ***πλήθος των χρηστών*** που μπορούν να χρησιμοποιήσουν τον Η/Υ, σε:

1. ενός χρήστη (*single-user*): PalmOS, SymbianOS.
2. πολλών χρηστών (*multi-user*): Windows, Linux, MacOS.

Με βάση το ***πλήθος των εργασιών*** που μπορούν να εκτελούνται ταυτόχρονα:

1. μίας εργασίας (*single-tasking*): PalmOS, SymbianOS.
2. πολλών εργασιών (*multi-tasking*): Windows, Linux, MacOS.

Εξέλιξη ΛΣ

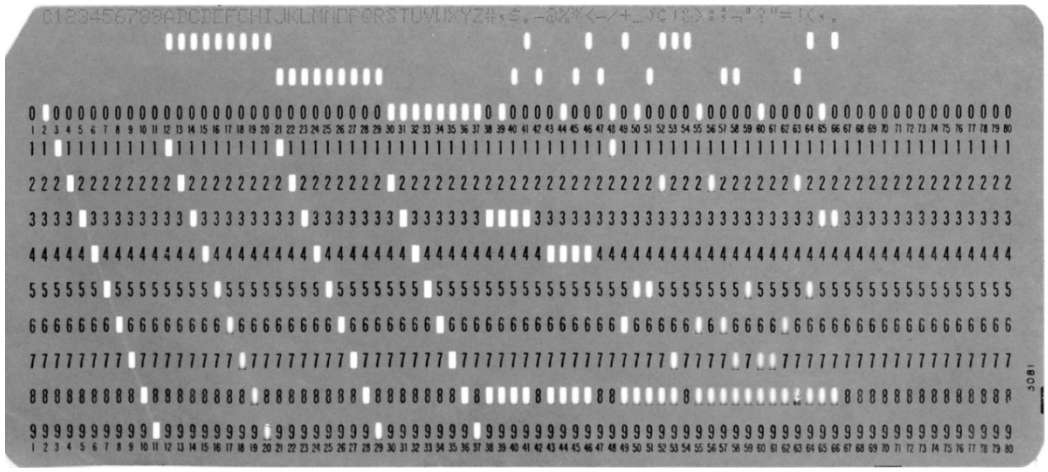
Τα λειτουργικά συστήματα έφθασαν να έχουν την σημερινή τους πολυπλοκότητα και ποικιλία μετά από δεκαετίες εξέλιξης.

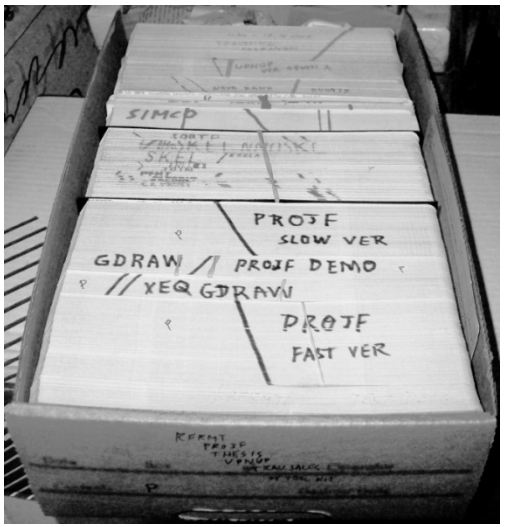
Ας δούμε κάποια σημαντικά στάδια της εξέλιξης αυτής.

Εξέλιξη ΛΣ: Επεξεργασία κατά δεσμίδες

Στις δεκαετίες ’60-70, οι Η/Υ ήταν ογκώδεις, δύσχρηστοι, ακριβοί, και δυσεύρετοι. Μπορούσαν να τους χρησιμοποιούν μόνο οι εργαζόμενοι σε πανεπιστήμια/εταιρίες.

Για να χρησιμοποιήσει τον Η/Υ, ο χρήστης τύπωνε σε ***διάτρητες κάρτες*** (*punch cards*) το πρόγραμμα που ήθελε να εκτελέσει (με οδηγίες για τους πόρους που απαιτεί). Το αποτέλεσμα ήταν μια στοίβα καρτών (***εργασία***, *job*).





Εξέλιξη ΛΣ: Επεξεργασία κατά δεσμίδες

Στη συνέχεια, ο χρήστης παρέδιδε την εργασία στον ***χειριστή*** (*operator*) του Η/Υ, έναν εξειδικευμένο υπάλληλο.

Αυτός προετοίμαζε τον Η/Υ για να διαβάσει τη στοίβα των καρτών και να εκτελέσει το πρόγραμμα.

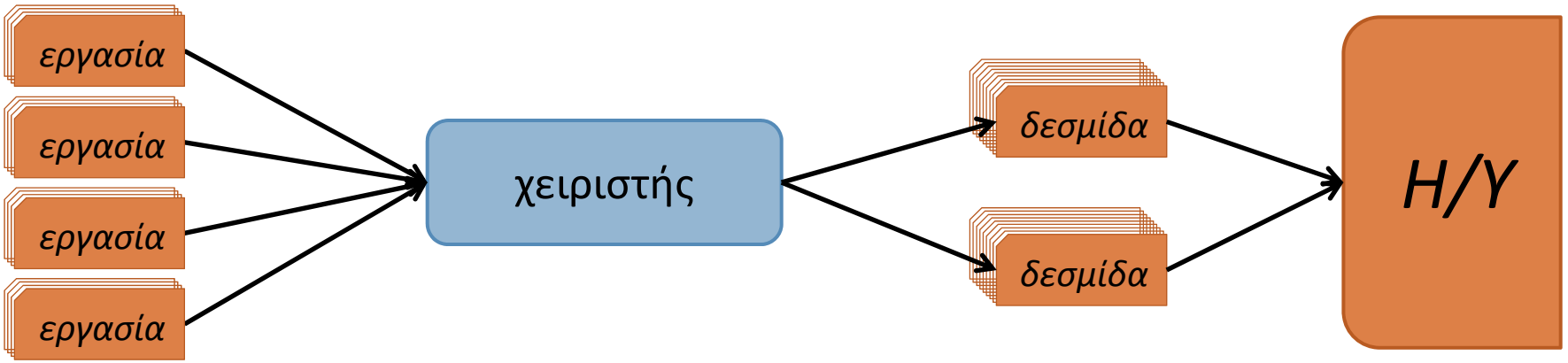
Μετά το πέρας της εκτέλεσης, ο χρήστης ειδοποιούνταν να παραλάβει μια εκτύπωση των αποτελεσμάτων.

Εξέλιξη ΛΣ: Επεξεργασία κατά δεσμίδες

Ο χειριστής δεν εκτελούσε κάθε εργασία χωριστά, αλλά τις ομαδοποιούσε σε ***δεσμίδες*** (*batches*) ανάλογα με τους πόρους που απαιτούσαν, και εκτελούσε τις δεσμίδες.

Έτσι, αντί να ετοιμάζει τον Η/Υ πριν την εκτέλεση κάθε εργασίας, τον ετοίμαζε πριν την εκτέλεση κάθε δεσμίδας.

Η διαδικασία είναι γνωστή ως ***επεξεργασία κατά δεσμίδες*** (*batch processing*) και επιβιώνει με κάποιες μορφές και στα σύγχρονα ΛΣ.



Εξέλιξη ΛΣ: Επεξεργασία κατά δεσμίδες

Ο ρόλος του ΛΣ ήταν απλώς να φορτώνει τις εργασίες της δεσμίδας στη μνήμη και να μεταφέρει τον έλεγχο:

1. Τον έλεγχο έχει το ΛΣ.
2. Το ΛΣ φορτώνει στη μνήμη την εργ#1 και της δίνει τον έλεγχο.
3. Η εργ#1 τερματίζει και ο έλεγχος επιστρέφει στο ΛΣ.
4. Το ΛΣ φορτώνει στη μνήμη την εργ#2 και τις δίνει τον έλεγχο.
5. Η εργ#2 τερματίζει και ο έλεγχος επιστρέφει στο ΛΣ.
6. …

Εξέλιξη ΛΣ: Πολυπρογραμματισμός

Πρόβλημα: Έστω ότι η εργ#2 ζητάει από τον εκτυπωτή να τυπώσει έναν μεγάλο όγκο δεδομένων.

Όσο τα δεδομένα μεταφέρονται στον εκτυπωτή, η ΚΜΕ μένει ανενεργή --- παρόλο που η εργ#3 περιμένει...

Λύση: ***Πολυπρογραμματισμός*** (*multi-programming*).

Οι εργασίες φορτώνονται ταυτόχρονα στη μνήμη. Όποτε αυτή που έχει τον έλεγχο δεν χρησιμοποιεί την ΚΜΕ (επειδή περιμένει κάποια Ε/Ε), το ΛΣ μεταφέρει τον έλεγχο στην επόμενη.

Εξέλιξη ΛΣ: Πολυπρογραμματισμός

Π.χ.:

1. Τον έλεγχο έχει το ΛΣ και φορτώνει τις εργασίες στη μνήμη.
2. Το ΛΣ δίνει τον έλεγχο στη διεργ#1.
3. Η διεργ#1 αρχίζει να τυπώνει και ο έλεγχος επιστρέφει στο ΛΣ.
4. Το ΛΣ δίνει τον έλεγχο στη διεργ#2.
5. Η διεργ#2 τερματίζει και ο έλεγχος επιστρέφει στο ΛΣ.
6. …

Ένα πρόγραμμα που έχει φορτωθεί στην μνήμη για να εκτελεστεί λέγεται ***διεργασία*** (process).

Εξέλιξη ΛΣ: Χρονομερισμός

Πρόβλημα: Έστω ότι η διεργ#2 απασχολεί την ΚΜΕ διαρκώς για ώρες, ενώ η διεργ#3 την χρειάζεται μόνο για λίγα λεπτά.

Η διαχείριση αδικεί την διεργ#3…

Λύση: ***Χρονομερισμός*** (*time sharing*).

Το ΛΣ μεταφέρει τον έλεγχο στην κάθε διεργασία μόνο για κάποιο μικρό χρόνο. Όταν αυτός εξαντληθεί, το ΛΣ ανακτά τον έλεγχο και τον μεταφέρει στην επόμενη διεργασία, πάλι για μικρό χρόνο, κ.ο.κ.

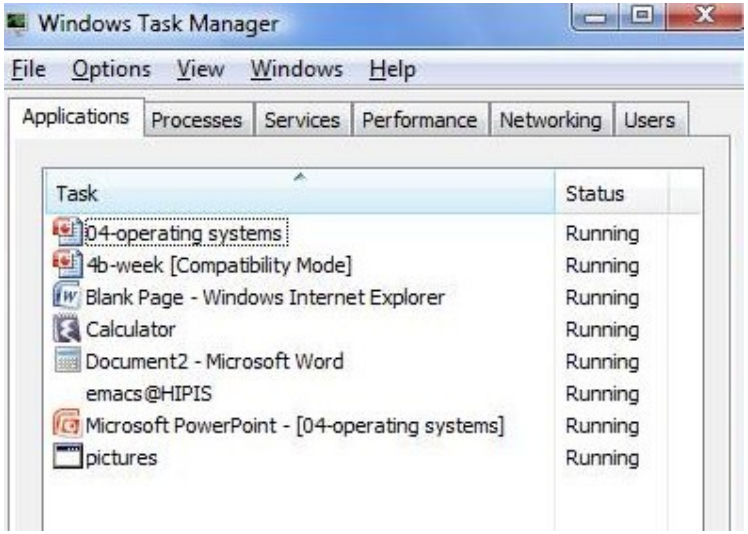
Εξέλιξη ΛΣ: Χρονομερισμός

Η διαχείριση είναι πολύ πιο δίκαιη.

Αν η ΚΜΕ είναι αρκετά ισχυρή και διεργασίες σχετικά λίγες, η εναλλαγή μεταξύ τους δεν προκαλεί πρόβλημα. Ακόμη κι αν εκτελούν διαδραστικά προγράμματα, ο χρήστης βλέπει (στο τερματικό) σχεδόν ό,τι θα έβλεπε και αν η διεργασία του χρησιμοποιούσε την ΚΜΕ αποκλειστικά.

Αυτός ο νοητός αποκλειστικός Η/Υ που μοιάζει να έχει μπροστά του ο χρήστης λέγεται ***νοητή μηχανή*** (*virtual machine*).

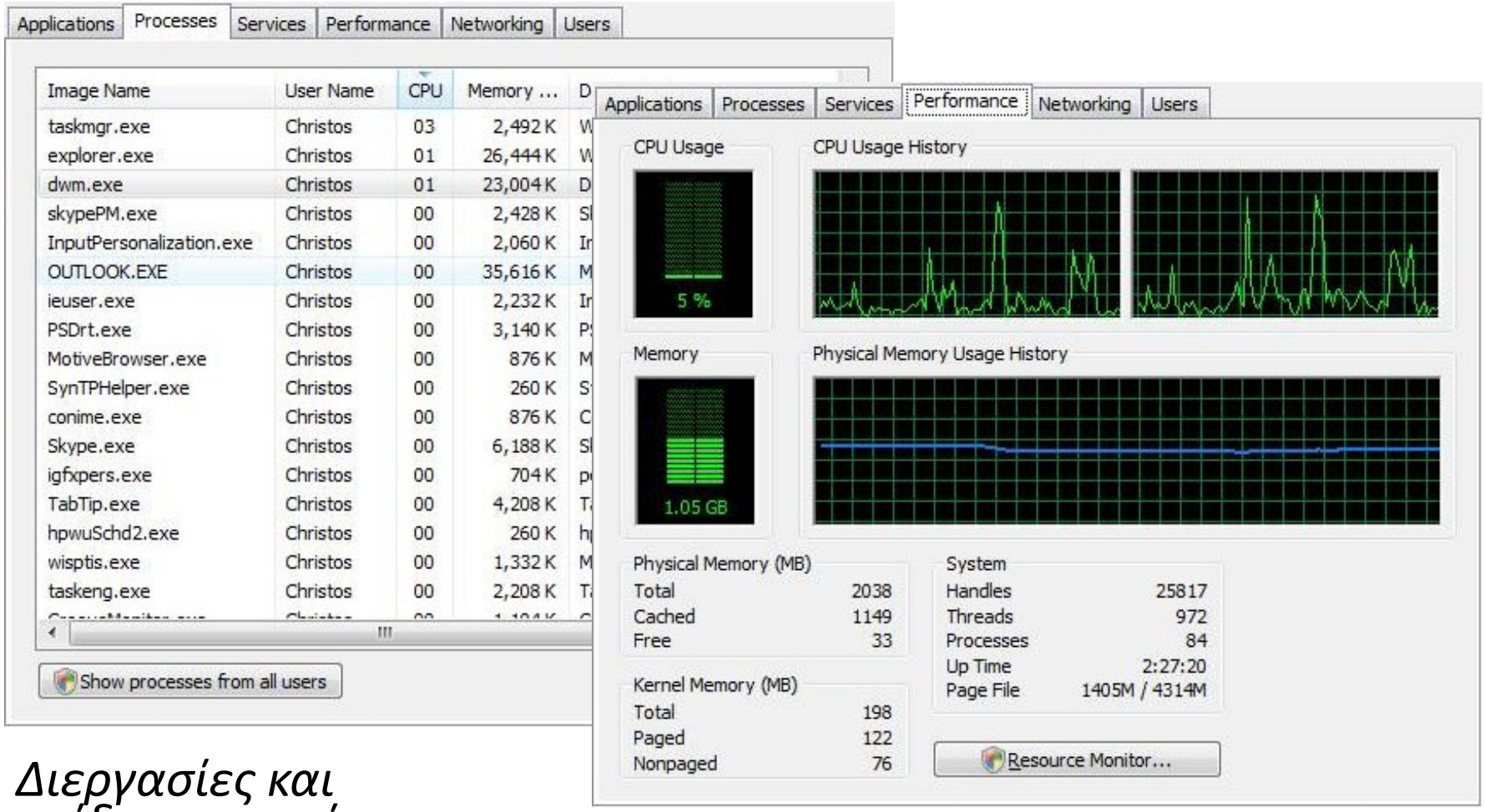
Εξέλιξη ΛΣ: Διαχείριση πόρων



Ως αποτέλεσμα της εξέλιξής τους, τα σύγχρονα ΛΣ έχουν φθάσει να επιτελούν ένα πολύ σύνθετο έργο:

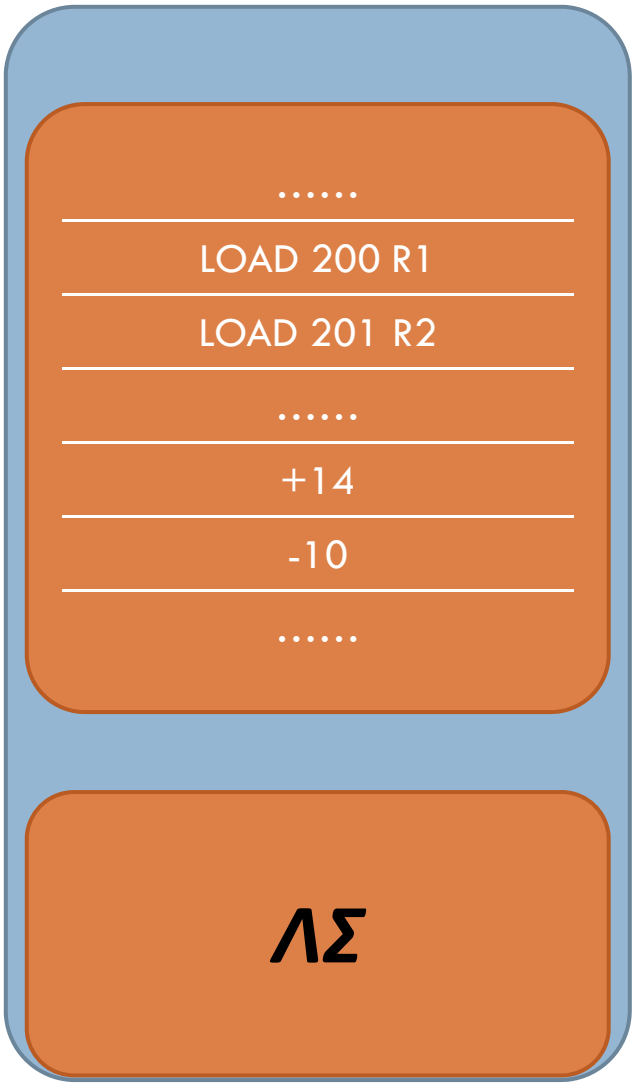
1. παρακολούθησης όλων των διεργασιών που είναι φορτωμένες στην κύρια μνήμη, και
2. κατανομής των πόρων του συστήματος (ΚΜΕ, κύρια/δευτερεύουσα μνήμη, μονάδες Ε/Ε) στις διεργασίες αυτές.

Εξέλιξη ΛΣ: Διαχείριση πόρων



Διεργασίες και επίδοση συστήματος, όπως φαίνονται στον Task Manager του ΛΣ Windows.

Διαχείριση μνήμης: Φόρτωση



Κάθε στιγμή, στη RAM συνυπάρχουν το ΛΣ και ≥0 διεργασίες.

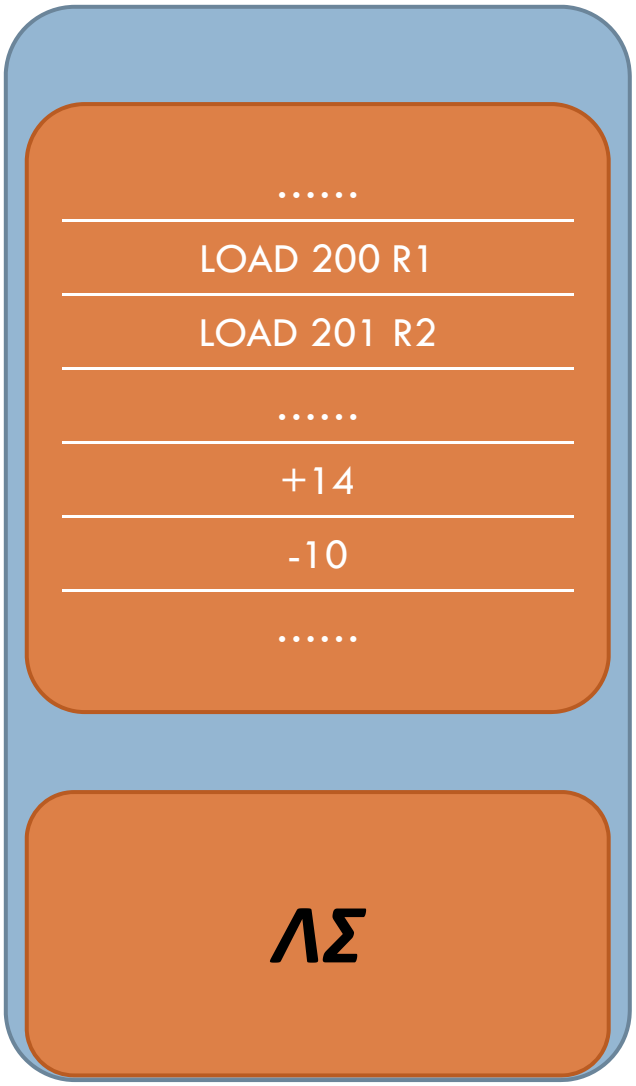
Για να φορτώσει ένα πρόγραμμα από τη δευτερεύουσα μνήμη στη RAM, το ΛΣ αντιγράφει τις λέξεις μία προς μία.

Η αντιγραφή γίνεται σε όποια συνεχή περιοχή της μνήμης είναι βολική.

Ερώτηση: Πώς ξέρει το πρόγραμμα πού στη μνήμη θα βρεθούν τα δεδομένα του;

Π.χ., πώς ήξερε η εντολή «LOAD 200 R1» ότι ο αριθμός «+14» θα κατέληγε πράγματι στη θέση 200 της μνήμης;

Διαχείριση μνήμης: Φόρτωση



Ερώτηση: Πώς ξέρει το πρόγραμμα πού στη μνήμη θα βρεθούν τα δεδομένα του;

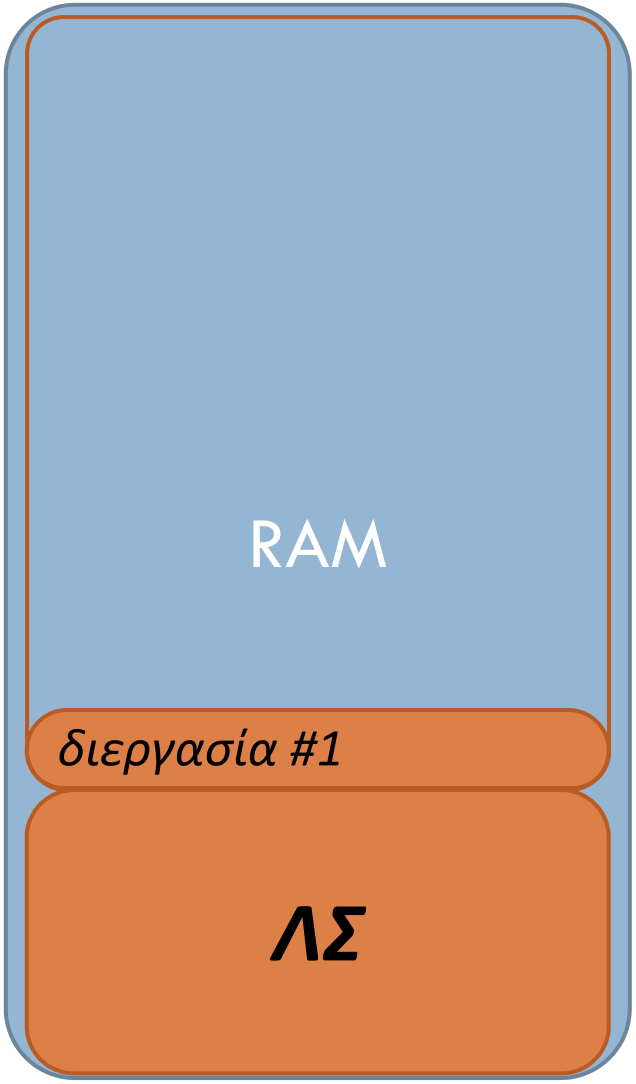
Απάντηση: Δεν ξέρει.

Στη δευτερεύουσα μνήμη, το πρόγραμμα αναφέρεται στα δεδομένα μέσω ***λογικών διευθύνσεων*** (*logical addresses*): π.χ. μέσω των θέσεών τους εντός του προγράμματος.

Κατά τη φόρτωση, το ΛΣ μετατρέπει τις λογικές διευθύνσεις σε ***φυσικές διευθύνσεις*** (*physical addresses*).

Η μετατροπή αυτή λέγεται ***συσχέτιση διευθύνσεων*** (*address binding*).

Διαχείριση μνήμης: Συνεχής



Σε ένα μονοπρογραμματιστικό ΛΣ, κάθε στιγμή η μνήμη περιέχει μόνο το ΛΣ και μία διεργασία, η οποία ελέγχει όλη τη RAM που δεν χρησιμοποιεί το ΛΣ.

Αυτό λέγεται ***διαχείριση συνεχούς μνήμης*** (*single contiguous memory management*).

Έχει όλα τα μειονεκτήματα του μονο-προγραμματισμού. Επιπλέον:

Ερώτηση: Τι γίνεται αν το πρόγραμμα δεν χωράει στη μνήμη;

Απάντηση: Δεν εκτελείται...

Διαχείριση μνήμης: Διαμέριση

Σε ένα πολυπρογραμματιστικό ΛΣ, κάθε στιγμή η μνήμη περιέχει το ΛΣ και πολλές διεργασίες.

Μια μέθοδος να διαχειριστούμε την μνήμη είναι να την χωρίσουμε σε διαμερίσματα, καθένα από τα οποία να φιλοξενεί μία διεργασία.

Τα διαμερίσματα μπορεί να είναι *ισο-ή ανισομεγέθη, σταθερά ή μεταβλητά*.

Αυτό λέγεται ***διαμέριση*** (partitioning).

Διαχείριση μνήμης: Διαμέριση

Μειονεκτήματα:

Με σταθερά διαμερίσματα, είναι δύσκολο να επιλέξουμε σωστό μέγεθος (μικρά⇨κάποια προγράμματα ίσως να μη χωρέσουν· μεγάλα⇨πολλά κενά).

Με μεταβλητά διαμερίσματα, αυξάνει η πολυπλοκότητα του ΛΣ και η χρονική επιβάρυνση που επιφέρει.

Επιπλέον:

Ερώτηση: Τι γίνεται αν ένα πρόγραμμα δεν χωράει στη μνήμη;

Απάντηση: Όπως πριν, δεν εκτελείται...

Διαχείριση μνήμης: Σελιδοποίηση

Μια άλλη μέθοδος είναι να χωρίσουμε και τα προγράμματα σε τμήματα:

Κάθε πρόγραμμα χωρίζεται σε ***σελίδες*** (pages). Η μνήμη χωρίζεται σε ***πλαίσια*** (frames), με μέγεθος αυτό των σελίδων.

Το ΛΣ φορτώνει κάθε σελίδα ενός προγράμματος σε ένα πλαίσιο. Τα πλαίσια που φιλοξενούν ένα πρόγραμμα δεν είναι απαραιτήτως γειτονικά.

Αυτό λέγεται ***σελιδοποίηση*** (paging).

Διαχείριση μνήμης: Σελιδοποίηση

Ερώτηση: Τι γίνεται αν ένα πρόγραμμα δεν χωράει στη μνήμη;

Απάντηση: Μπορεί να εκτελεστεί! Πώς;

Το ΛΣ εκτελεί το πρόγραμμα ακόμη και χωρίς να είναι φορτωμένες όλες του οι σελίδες. Όποτε η εκτέλεση απαιτήσει σελίδα που δεν είναι στη μνήμη, η σελίδα φορτώνεται (σε πλαίσιο άλλης, που δεν χρησιμοποιείται) και η εκτέλεση συνεχίζει.

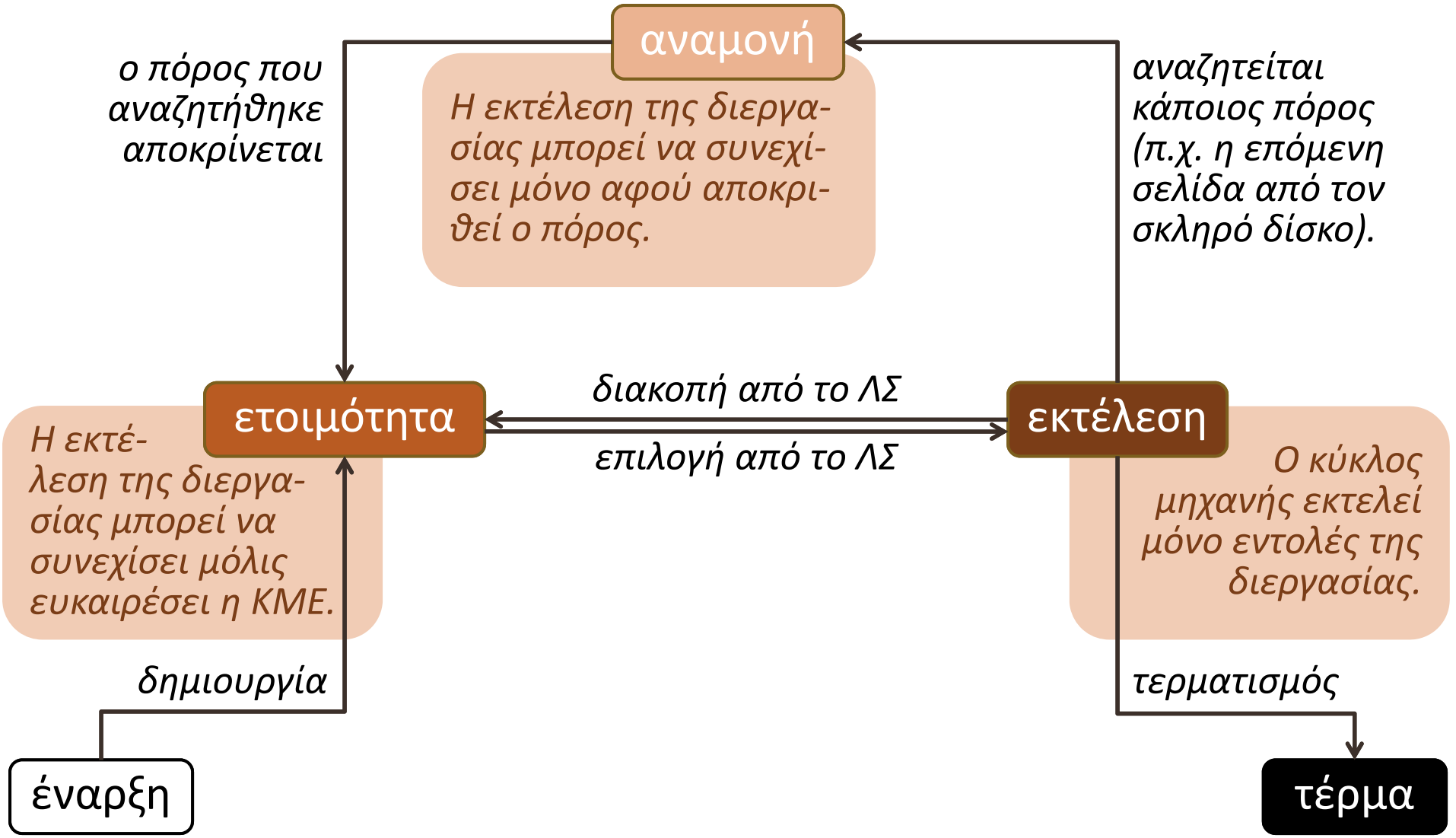
Το αποτέλεσμα λέγεται ***εικονική μνήμη*** (*virtual memory*): η κύρια μνήμη «μοιάζει» τόσο μεγάλη όσο η δευτερεύουσα.

Διαχείριση διεργασιών

Ερώτηση: Πώς κατορθώνει το ΛΣ να παρακολουθεί την εκτέλεση πολλών διεργασιών στην ίδια ΚΜΕ;

Πριν απαντήσουμε, πρέπει να περιγράψουμε όλες τις διαφορετικές καταστάσεις στις οποίες μπορεί να βρεθεί μια διεργασία.

Διαχείριση διεργασιών



Διαχείριση διεργασιών

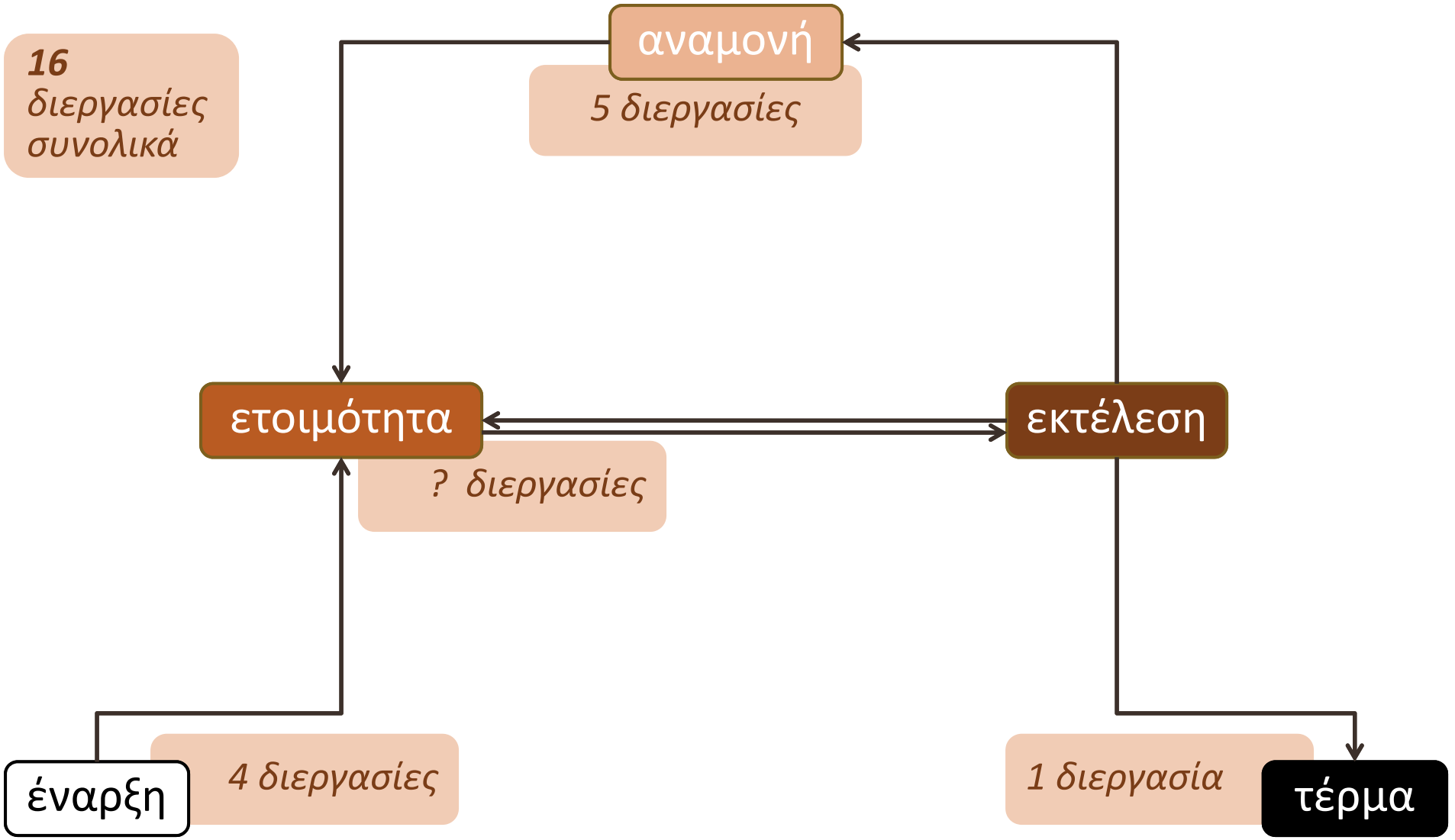
Για κάθε κατάσταση (*έναρξη, ετοιμότητα, αναμονή, εκτέλεση, τέρμα*), το ΛΣ διατηρεί και παρακολουθεί έναν κατάλογο. Ο κατάλογος της κατάστασης s περιέχει 1 εγγραφή για κάθε διεργασία που είναι στην κατάσταση s.

Η εγγραφή περιέχει πληροφορίες για την διεργασία, π.χ.:

- πού βρίσκεται στη μνήμη,
- ποια είναι η επόμενη προς εκτέλεση εντολή της,
- τι περιείχαν οι καταχωρητές της ΚΜΕ την τελευταία φορά που διακόπηκε η εκτέλεση της διεργασίας.

Ερώτηση: Αν η κύρια μνήμη περιέχει ***16*** διεργασίες και οι κατάλογοι για τις καταστάσεις *έναρξη*, *αναμονή*, *τερματισμός* περιέχουν αντίστοιχα ***4***, ***5***, και ***1*** διεργασίες, πόσες διεργασίες περιέχει ο κατάλογος για την *ετοιμότητα*;

Διαχείριση διεργασιών



Διαχείριση διεργασιών

Ερώτηση: Αν η κύρια μνήμη περιέχει ***16*** διεργασίες και οι κατάλογοι για τις καταστάσεις *έναρξη*, *αναμονή*, *τερματισμός* περιέχουν αντίστοιχα ***4***, ***5***, και ***1*** διεργασίες, πόσες διεργασίες περιέχει ο κατάλογος για την *ετοιμότητα*;

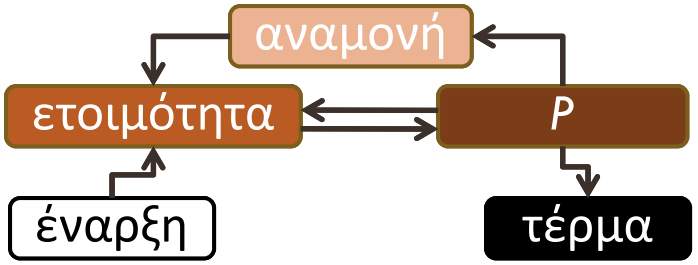
Απάντηση:
4 διεργασίες σε *έναρξη*
5 διεργασίες σε *αναμονή*
1 διεργασίες σε *τερματισμό*
1 διεργασία σε *εκτέλεση* (προφανώς)

11 διεργασίες συνολικά, εκτός *ετοιμότητας*

Άρα σε *ετοιμότητα* βρίσκονται 16 - 11 = 5 διεργασίες.

Χρονοπρογραμματισμός της ΚΜΕ

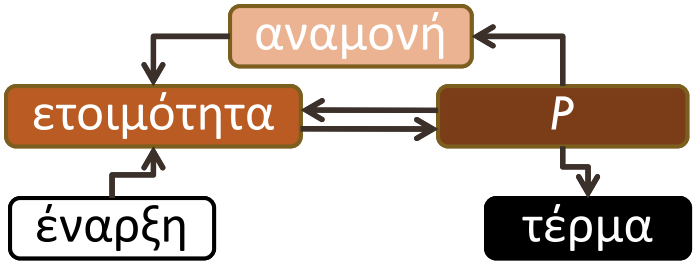
Ερώτηση: Όποτε πρέπει να αντικατασταθεί η υπό *εκτέλεση* διεργασία, πώς το ΛΣ επιλέγει τον αντικαταστάτη της από αυτές που βρίσκονται σε *ετοιμότητα*;



Κάθε ΛΣ απαντά σε αυτό το ερώτημα με ένα σύνολο κανόνων για να προγραμματίζει ποια διεργασία θα εκτελεί κάθε φορά η ΚΜΕ. Λέμε ότι οι κανόνες αυτοί συνιστούν έναν ***αλγόριθμο χρονοπρογραμματισμού της ΚΜΕ*** (*CPU scheduling algorithm*).

Χρονοπρογραμματισμός της ΚΜΕ

Ο αλγόριθμος χρονοπρογραμματισμού καλείται όποτε ενδέχεται να αλλάξει η υπό εκτέλεση διεργασία *P*, δηλαδή:



1. όποτε η *P τερματίσει*,
2. όποτε η *P τεθεί σε αναμονή*,
3. όποτε η *P* εξαντλήσει τον χρόνο που της ανατέθηκε,
4. όποτε κάποια διεργασία σε αναμονή τεθεί σε ετοιμότητα (ίσως να έχει προτεραιότητα έναντι της *P*).

Στις περιπτώσεις 3 και 4, ο χρονοπρογραμματισμός που συντελείται λέγεται ***προεκτοπιστικός*** (*preemptive*). Στις 1 και 2, λέγεται ***μη προεκτοπιστικός*** (*non-preemptive*).

Χρονοπρογραμματισμός της ΚΜΕ

Τρεις απλοί αλγόριθμοι χρονοπρογραμματισμού:

1. ***Κατά σειρά άφιξης*** (*first-come first-served, FCFS*): εφαρμόζεται *μη προεκτοπιστικά* και προτάσσει τη διεργασία *που τέθηκε νωρίτερα σε ετοιμότητα*.
2. ***Κατά μικρότερο χρόνο εκτέλεσης*** (*shortest job next, SJN*): εφαρμόζεται *μη προεκτοπιστικά* και προτάσσει τη διεργασία *που θα απασχολήσει λιγότερο την ΚΜΕ*.
3. ***Κυκλικά*** (*round robin, RR*): εφαρμόζεται *προεκτοπιστικά* και διατρέχει τις διεργασίες *κυκλικά*. Η ΚΜΕ εκτελεί κάθε διεργασία για το πολύ 1 ***κβάντο χρόνου*** (*time slice*) τη φορά. Αν αυτό εξαντληθεί, το ΛΣ επιστρέφει τη διεργασία σε κατάσταση ετοιμότητας, και αρχίζει να εκτελεί την επόμενη από τις διεργασίες σε ετοιμότητα.

Χρονοπρογραμματισμός της ΚΜΕ

Ερώτηση: Έστω ότι οι διεργασίες p_1, p_2, p_3, p_4, p_5 τίθενται σε ετοιμότητα σχεδόν ταυτόχρονα (αλλά με την παραπάνω σειρά) και ότι θα απασχολήσουν την ΚΜΕ για όσες χρονικές μονάδες αναφέρει ο πίνακας:

p_1	p_2	p_3	p_4	p_5
140	*75*	*320*	*280*	*125*

Για καθέναν από τους αλγορίθμους *FCFS*, *SJN*, *RR*, ποιο είναι το ***διάγραμμα Gantt*** (*Gantt chart*) που θα προκύψει αν εφαρμόσουμε τον αλγόριθμο σε αυτές τις διεργασίες;

(Το διάγραμμα Gantt απεικονίζει ποια διεργασία εκτελεί η ΚΜΕ κάθε στιγμή ---βλ. επόμενες σελίδες.)

Χρονοπρογραμματισμός της ΚΜΕ

Απάντηση: Με τον *FCFS*, οι διεργασίες «μπαίνουν» στην ΚΜΕ με την σειρά:

p_1, p_2, p_3, p_4, p_5

και την απασχολούν κατά το παρακάτω διάγραμμα Gantt:

p_1	p_2	p_3	p_4	p_5
140	75	320	280	125

Χρονοπρογραμματισμός της ΚΜΕ

Απάντηση: Με τον *SJN*, οι διεργασίες «μπαίνουν» στην ΚΜΕ με την σειρά:

p_2, p_5, p_1, p_4, p_3

και την απασχολούν κατά το παρακάτω διάγραμμα Gantt:

p_1	p_2	p_3	p_4	p_5
140	75	320	280	125

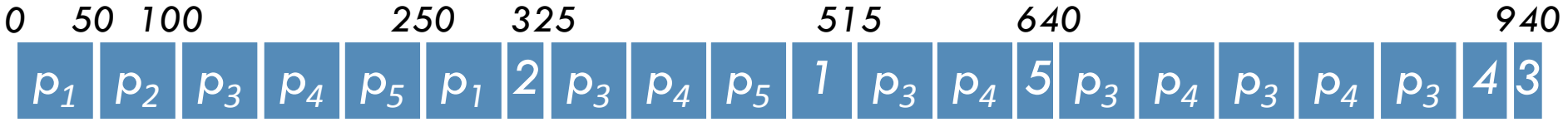
Χρονοπρογραμματισμός της ΚΜΕ

Απάντηση: Με τον *RR*, οι διεργασίες «μπαίνουν» στην ΚΜΕ με την σειρά:

$p_1, p_2, p_3, p_4, p_5, p_1, p_2, p_3, \ldots$

και την απασχολούν κατά το παρακάτω διάγραμμα Gantt (έστω ***κβάντο*** 50 μονάδων):

p_1	p_2	p_3	p_4	p_5
140	75	320	280	125



Χρονοπρογραμματισμός της ΚΜΕ

Ερώτηση: Ποιος από τους αλγορίθμους είναι καλύτερος;

Απάντηση: Δεν υπάρχει μοναδικό κριτήριο.

Ένα από τα κριτήρια που χρησιμοποιούνται είναι ο ***μέσος χρόνος ολοκλήρωσης*** (*average turnaround time*) επί όλων των διεργασιών.

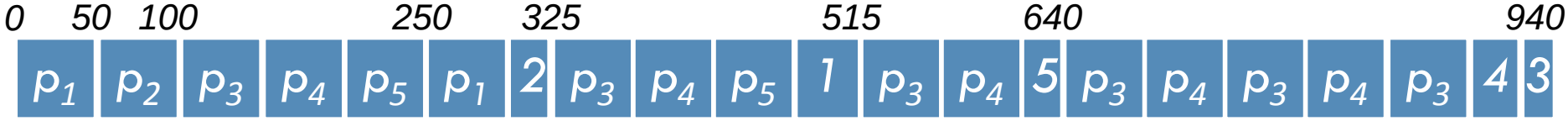
Εδώ, ***χρόνος ολοκλήρωσης*** (*turnaround time*) μιας διεργασίας είναι ο χρόνος που μεσολαβεί από τη στιγμή που η διεργασία τίθεται σε ετοιμότητα για πρώτη φορά μέχρι τη στιγμή που παύει να εκτελείται για τελευταία φορά.

Χρονοπρογραμματισμός της ΚΜΕ

Ερώτηση: Αν όλες οι διεργασίες τέθηκαν ταυτόχρονα σε ετοιμότητα, ποιος είναι ο *μέσος χρόνος ολοκλήρωσης* στα παρακάτω διαγράμματα Gantt;

Απάντηση: (140+215+535+815+940)/5 = **529**

Απάντηση: (75+200+340+620+940)/5 = **435**



Απάντηση: (515+325+940+920+640)/5 = **668**

Χρονοπρογραμματισμός της ΚΜΕ

Επομένως, από την άποψη του μέσου χρόνου ολοκλήρωσης, η αλγόριθμος *SJN* ήταν ο καλύτερος.

Όμως ο *SJN* έχει και μειονεκτήματα:

1. Δεν είναι πάντα δυνατόν να γνωρίζουμε την διάρκεια των διεργασιών.
2. Δεν αποκλείει το φαινόμενο της ***στέρησης*** (*starvation*). (Πρόκειται για το φαινόμενο κατά το οποίο μια διεργασία δεν εκτελείται ποτέ διότι διαρκώς καταφθάνουν και τίθενται σε ετοιμότητα άλλες διεργασίες, με μικρότερη διάρκεια.)