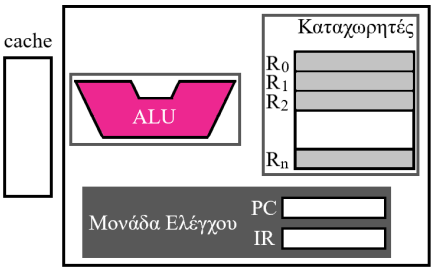

Κεντρική Μονάδα Επεξεργασίας (ΚΜΕ)



Αριθμητική και Λογική Μονάδα (ΑΛΜ - ALU)
Εκτελεί μια πληθώρα πράξεων:
- Μεταφορά περιεχομένων (π.χ. LOAD)
- Αριθμητικές πράξεις (με καταχωρητές)
- Λογικές πράξεις (με πύλες)
- Ολισθήσεις, περιστροφές, κλιμακώσεις

Καταχωρητές
Αποτελούν θέσεις προσωρινής αποθήκευσης και κάθε CPU έχει τουλάχιστον τους εξής:
- PC (program counter)

Περιέχει τη διεύθυνση της επόμενης εντολής που θα εκτελεστεί.

- IR (instruction register)

Αποθηκεύεται η εντολή που εκτελείται.

- ACC (accumulator)

Εκτελεί τις πράξεις.

- MAR (memory address register)

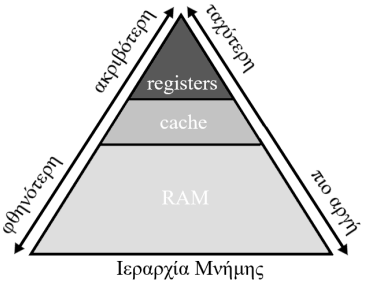
Για να δώσω μια διεύθυνση στη μνήμη, πρέπει να τη τοποθετήσω στη στον MAR.

- MDR (memory data register)

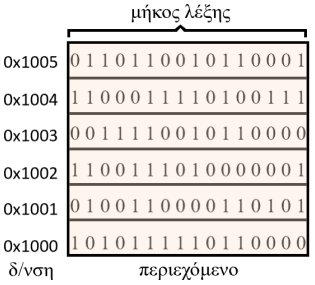
Χρησιμοποιείται και για read αλλά και για write από και πάνω αντίστοιχα στη μνήμη.

Μονάδα ελέγχου
Ελέγχει τη λειτουργία του κάθε υποσυστήματος της ΚΜΕ.

Κρυφή μήνμη (cache)
Η κρυφή μνήμη είναι ενσωματωμένη στην ΚΜΕ και επιτρέπει την αποθήκευση και ταχύτατη πρόσβαση σε ένα μικρό όγκο δεδομένων (π.χ. το 1‰ των περιεχομένων της RAM). Τα δεδομένα αναζητούνται πρώτα στην κρυφή μνήμη και (αν δεν βρεθούν) μετά στην κεντρική μνήμη.

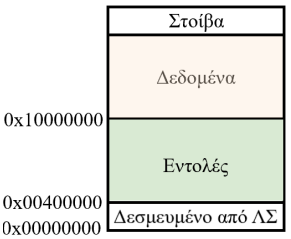


Κεντρική μνήμη (RAM)
Η κεντρική μνήμη είναι προσωρινός αποθηκευτικός χώρος για τις εντολές ενός προγράμματος και τα δεδομένα που αυτό διαχειρίζεται. Αποτελείται από ένα σύνολο θέσεων αποθήκευσης σε καθένα από το οποίο αντιστοιχεί μία μοναδική διεύθυνση.



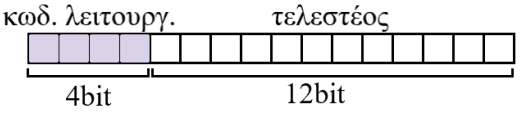
Οργάνωση κεντρικής μνήμης

ⓘ Η στοίβα αποθηκεύει τη διεύθυνση επιστροφής (στο κύριο πρόγραμμα), όταν ολοκληρωθεί η εκτέλεση μίας ρουτίνας του προγράμματος.



Μορφή εντολών

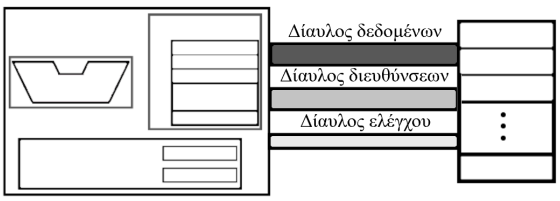
Η μορφή και οι τύποι μορφών διαφέρουν μεταξύ αρχιτεκτονικών (π.χ. x86, MIPS).



Εντολή	***Κωδ.***	***Τελεστέοι***			***Ενέργεια***
	d_1	d_2	d_3	d_4	
HALT	0				Διακόπτει την εκτέλεση του προγράμματος
LOAD	1	R_D	M_S		$R_D \leftarrow M_S$
STORE	2	M_D	R_S		$M_D \leftarrow R_S$
ADDI	3	R_D	R_{S1}	R_{S2}	$R_D \leftarrow R_{S1} + R_{S2}$
ADDF	4	R_D	R_{S1}	R_{S2}	$R_D \leftarrow R_{S1} + R_{S2}$
MOVE	5	R_D	R_S		$R_D \leftarrow R_S$
NOT	6	R_D	R_S		$R_D \leftarrow \overline{R}_S$
AND	7	R_D	R_{S1}	R_{S2}	$R_D \leftarrow R_{S1}$ AND R_{S2}
OR	8	R_D	R_{S1}	R_{S2}	$R_D \leftarrow R_{S1}$ OR R_{S2}
XOR	9	R_D	R_{S1}	R_{S2}	$R_D \leftarrow R_{S1}$ XOR R_{S2}
INC	A	R			$R \leftarrow R + 1$
DEC	B	R			$R \leftarrow R - 1$
ROTATE	C	R	n	0 ή 1	Rot_n R
JUMP	D	R	n		Αν $R_0 \neq R$ τότε PC = *n*, ειδάλλως συνέχισε

Υπόμνημα: R_S, R_{S1}, R_{S2}: Δεκαεξαδική διεύθυνση των καταχωρητών προέλευσης
R_D: Δεκαεξαδική διεύθυνση του καταχωρητή προορισμού
M_S: Δεκαεξαδική διεύθυνση της θέσης μνήμης προέλευσης
M_D: Δεκαεξαδική διεύθυνση της θέσης μνήμης προορισμού
n: δεκαεξαδικός αριθμός
d_1, d_2, d_3, d_4: 1ο, 2ο, 3ο, και 4ο δεκαεξαδικό ψηφίο

Επικοινωνία Μνήμης - ΚΜΕ



Το έυρος του *δίαυλου δεδομένων* καθορίζεται από το μήκος της λέξης (π.χ. 32 bits για 32-bit λέξη).

Το έυρος του *διάυλου διευθύνσεων* καθορίζεται από το σύνολο θέσεων μνήμης (δηλαδή n bit για 2^n θέσεις).

Το έυρος του *διάυλου ελέγχου* καθορίζεται από το σύνολο των εντολών ελέγχου (m bits για 2^m εντολές).

Κύκλος μηχανής

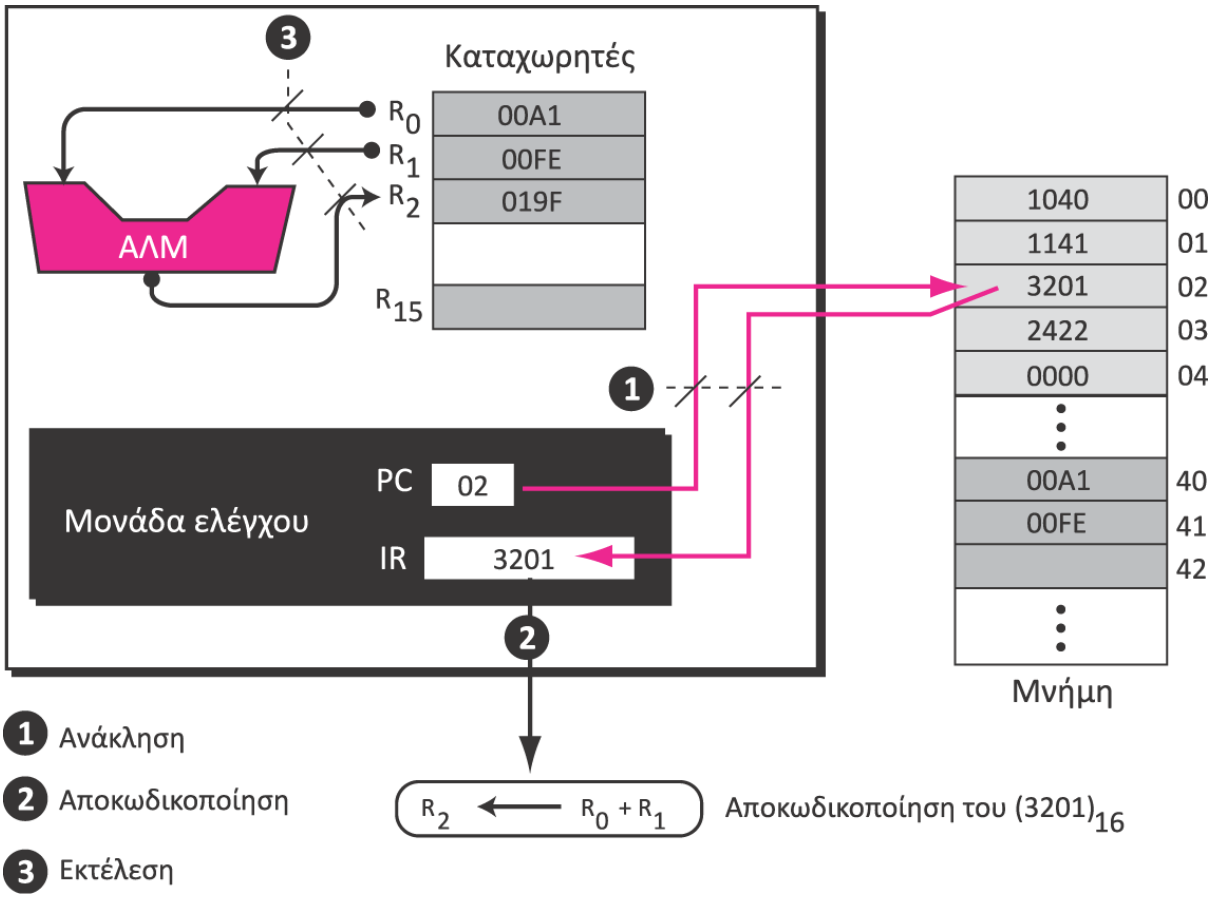
• Ανάκληση

- *Αντιγράφεται η επόμενη εντολή (όπου δείχνει ο PC) στον καταχωρητή εντολών*
- *Αυξάνεται το PC κατά ένα*

• Αποκωδικοποίηση:

- *Αποκωδικοποιείται η εντολή (π.χ. LOAD, ADD) και οι τελεστές της (π.χ. καταχωρητές δεδομένων, διευθύνσεις μνήμης)*

π.χ. εκτέλεση εντολής 3 (ADD)



Λειτουργικό Σύστημα
Ένα ολοκληρωμένο σύνολο προγραμμάτων που λειτουργεί ως ενδιάμεσος μεταξύ των χρηστών και του υλικού του Υπολογιστικού Συστήματος (ΥΣ). Βασικές εργασίες ενός ΛΣ αποτελούν:

- **Διαχείριση διεργασιών**
- **Διαχείριση συσκευών Εισόδου/Εξόδου**
- **Διαχείριση μνήμης**
- **Διαχείριση αρχείων**

Διαχείριση Διεργασιών

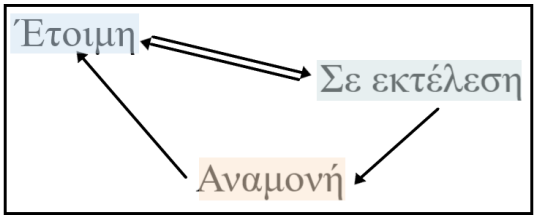
Διεργασία
Ένα πρόγραμμα που τρέχει. (γενικός ορισμός)

Κάθε διεργασία περιλαμβάνει:
• Context (περιβάλλον διεργασίας)
Κάθε διεργασία που τρέχει γράφει στους καταχωρητές της CPU (ACC, PC κτλ.). Η διεργασία κάποτε τελειώνει. Όταν τελειώσει, οι τιμές των καταχωρητών περνάνε στο χώρο μνήμης της διεργασίας.

• Τον κώδικα του προγράμματος
Διαφορά προγράμματος και διεργασίας είναι πως η διεργασία αποτελεί ένα εκτελούμενο στιγμιότυπο του προγράμματος. Για παράδειγμα είναι δυνατόν να εκτελούνται ταυτόχρονα πολλές διεργασίες από το ίδιο πρόγραμμα (π.χ. διορθωτής κειμένου).

• …

Καταστάσεις διεργασίας



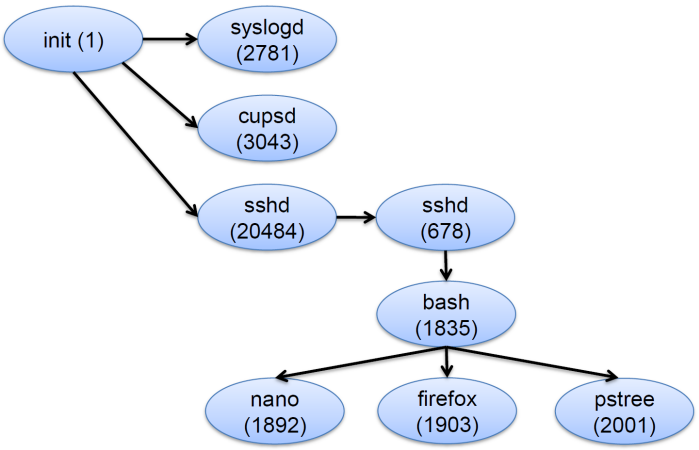
Έτοιμη: Διεργασία που είναι έτοιμη να τρέξει, έχει πάρει κβάντα και περιμένει

Σε εκτέλεση: Διεργασία που χρησιμοποιεί τη CPU μέχρι να τελειώσουν τα κβάντα

ⓘ Μετά την εκτέλεση υπάρχουν δύο περιπτώσεις:
1. Τέλος κβάντων -> Έτοιμη (γίνεται context switch)
2. ΣΕ ΑΝΑΜΟΝΗ: Η διεργασία περιμένει να συμβεί ένα γεγονός (π.χ. να γίνει accessible ένα μέρος μνήμης) για να γίνει ξανά Έτοιμη (γίνεται context switch)

Ιεραρχία διεργασιών

Μια γονική διεργασία δημιουργεί διεργασίες παιδιά. Παράδειγμα:



ⓘ διαφάνειες 14-18 AIC105 L02.pdf

Χρονοδρομολογητής ΚΜΕ

Επιλέγει μεταξύ των διεργασιών που είναι έτοιμες για εκτέλεση και αναθέτει την ΚΜΕ σε μια από αυτές.

- Χρονοδρομολογητής χωρίς διακοπές
  Από τη στιγμή που θα δοθεί η ΚΜΕ σε μια διεργασία, η διεργασία διατηρεί την ΚΜΕ μέχρι να ολοκληρωθεί η εκτέλεσή της.

- Χρονοδρομολογητής με διακοπές
  Κάθε διεργασία λαμβάνει ένα κλάσμα του χρόνου της ΚΜΕ. Με το πέρας αυτού, η εκτέλεση της διακόπτεται και στη συνέχεια επιλέγεται η εκτέλεση άλλης διεργασίας.

Μετρικές

• Χρησιμοποίηση της ΚΜΕ CPU (*utilization*)
Πόσο ποσοστό του χρόνου CPU είναι χρήσιμο για κάθε διεργασία.

• Ρυθμός διεκπεραίωσης (*throughput*)
Πλήθος διεργασιών που ολοκληρώνουν την εκτέλεσή τους στη μονάδα του χρόνου.

• Χρόνος ολοκλήρωσης (*turnaround time*)
Συμβολικός χρόνος παραμονής μιας διεργασίας στο σύστημα.

• Χρόνος αναμονής (*waiting time*)
Ο χρόνος που μια διεργασία περιμένει στην ουρά έτοιμων διεργασιών.

Βελτιστοποίηση των μετρικών

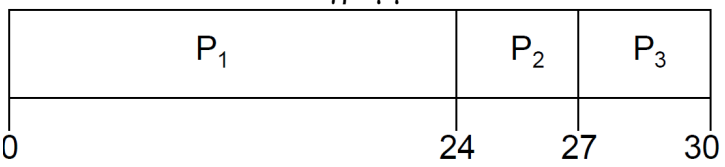
Η βελτιστοποίηση όλων των παραμέτρων απαιτεί συμβιβασμούς.

First Come - First Serve (FCFS)

Διεργασία	Χρόνος εκτέλεσης
P_1	24 ms
P_2	3 ms
P_3	3 ms

π.χ. οι διεργασίες φτάνουν με τη σειρά P_1, P_2, P_3.

Διάγραμμα:



Μέσος Χρόνος Αναμονής: (0+24+27) / 3 = 17ms
Μέσος Χρόνος Ολοκλήρωσης: (24+27+30) / 3 = 27ms
Δημιουργείται φαινόμενο convoy όταν μικρές διεργασίες βρίσκονται πίσω από μεγάλες.

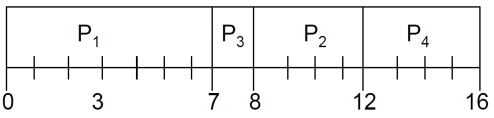
Shortest Job Next (SJN)

Επιλέγεται πάντα η διεργασία με το μικρότερο χρόνο.

Χωρίς διακοπές: Από τη στιγμή που θα δοθεί η ΚΜΕ σε μια διεργασία, η επιλογή της επόμενης διεργασίας γίνεται όταν τελειώσει η τρέχουσα διεργασία.

Διεργασία	Χρόνος άφιξης	Χρόνος εκτέλεσης
P_1	0	7 ms
P_2	2	4 ms
P_3	4	1 ms
P_4	5	4 ms

Διάγραμμα:

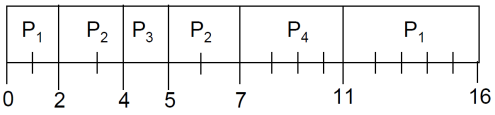


Μέσος Χρόνος Αναμονής:(0 + (8-2) + (7-4) + (12-5)) / 4 = 4ms
Μέσος Χρόνος Ολοκλήρωσης: (7 + (12-2) + (8-4) + (16-5)) / 4 = 8ms

Με διακοπές: Αν μια διεργασία εισέλθει στο σύστημα με χρόνο εκτέλεσης μικρότερο από αυτό που απομένει στην τρέχουσα διεργασία, τότε γίνεται διακοπή. Το σχήμα αυτό είναι γνωστό ως Shortest Remaining Time Next (SRTN).

Διεργασία	Χρόνος άφιξης	Χρόνος εκτέλεσης
P_1	0	7 ms
P_2	2	4 ms
P_3	4	1 ms
P_4	5	4 ms

Διάγραμμα:



MXA: ((11-2) + (5-4) + (4-4) + (7-5)) / 4 = 3ms
MXO: (16 + (7-2) + (5-4) + (11-5)) / 4 = 7ms

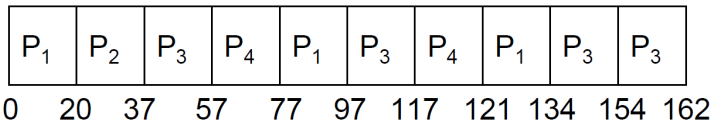
Round Robin (εκ περιτροπής)

Κάθε διεργασία λαμβάνει ένα μικρό κλάσμα του χρόνου της ΚΜΕ (κβάντο), συνήθως 10-100 ms. Μόλις παρέλθει αυτό το χρονικό διάστημα, η διεργασία διακόπτεται και προστίθεται στο τέλος της ουράς έτοιμων διεργασιών

ⓘ πολύ μεγάλο q ο αλγόριθμος μετατρέπεται σε FCFS
ⓘ πολύ μικρό q το σύστημα αναλώνεται σε εναλλαγές διεργασιών

Διεργασία	Χρόνος εκτέλεσης
P_1	53 ms
P_2	17 ms
P_3	68 ms
P_4	24 ms

Διάγραμμα (με q = 20):



MXA: ((77-20+121-97) + 20 + (37+97-57+134-117) + (57+117-77)) / 4 = 73ms
MXO: (134+37+162+121) / 4 = 113.5 ms

Διαχείριση συσκευών Εισόδου/Εξόδου

Μπορούν να χωριστούν σε δύο κατηγορίες. Σε αυτή των **συσκευών block** (π.χ. σκληρός δίσκος) όπου αποθηκεύουν πληροφορία σε μονάδες σταθερού μεγέθους που λέγονται blocks και ένα block αποτελεί τη μικρότερη μονάδα πληροφορίας που μεταφέρεται από και προς τη συσκευή, με το κάθε ένα έχει τη δική του διεύθυνση και μπορεί να εγγραφεί ή ανακτηθεί ανεξάρτητα από τα άλλα blocks.

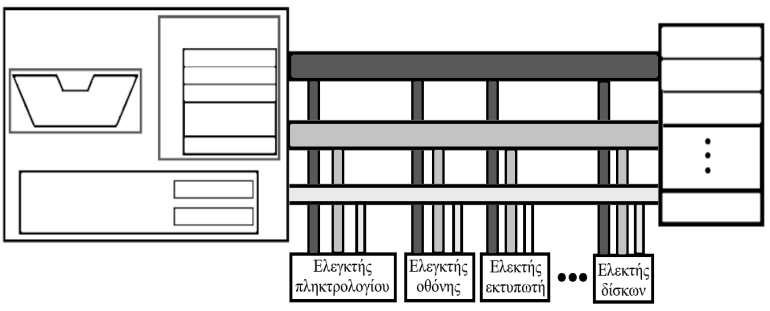
Και σε αυτή των **συσκευών χαρακτήρων** (π.χ. πληκτρολόγιο, εκτυπωτής) όπου διαχειρίζονται μια σειρά χαρακτήρων και δεν υπάρχει η δυνατότητα διευθυνσιοποίησης πληροφορίας και ανεξάρτητης πρόσβασης.

Driver συσκευής

Ονομάζεται ο κώδικας των *συναρτήσεων πρόσβασης* & *ρουτίνας εξυπηρέτησης* μιας συσκευής, τα οποία υλοποιεί το ΛΣ.

Ε/Ε και ΚΜΕ

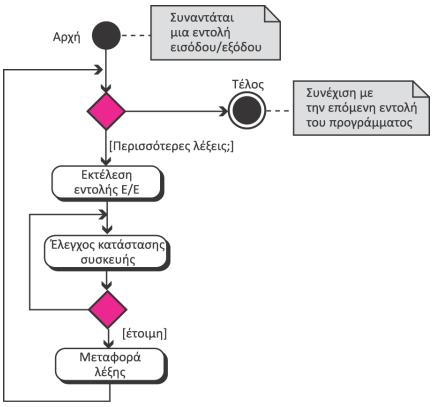
Οι συσκευές Ε/Ε και η ΚΜΕ μπορούν να λειτουργούν ταυτόχρονα. Κάθε συσκευή Ε/Ε περιλαμβάνει ένα controller - ελεγκτή κ' κάθε ελεγκτής έχει ένα buffer - μνήμη προσωρινής αποθήκευσης. Η ΚΜΕ μετακινεί δεδομένα μεταξύ κύριας μνήμης και buffer.

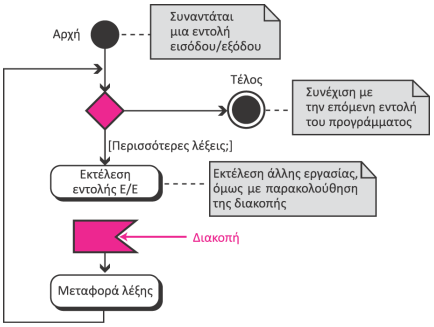


Τρόποι χειρισμού Ε/Ε

1. Μέσω προγράμματος (ΚΜΕ)

Η μεταφορά δεδομένων μεταξύ της ΚΜΕ και της συσκευής γίνεται από εντολή του προγράμματος (αποτελεί την απλούστερη μέθοδο αλλά σπαταλά το χρόνο της ΚΜΕ).





2. Μέσω διακοπής (interrupt)

Ο controller της συσκευής ενεργοποιεί το κύκλωμα διακοπής της ΚΜΕ, η οποία διακόπτει την εκτέλεση της τρέχουσας διεργασίας.

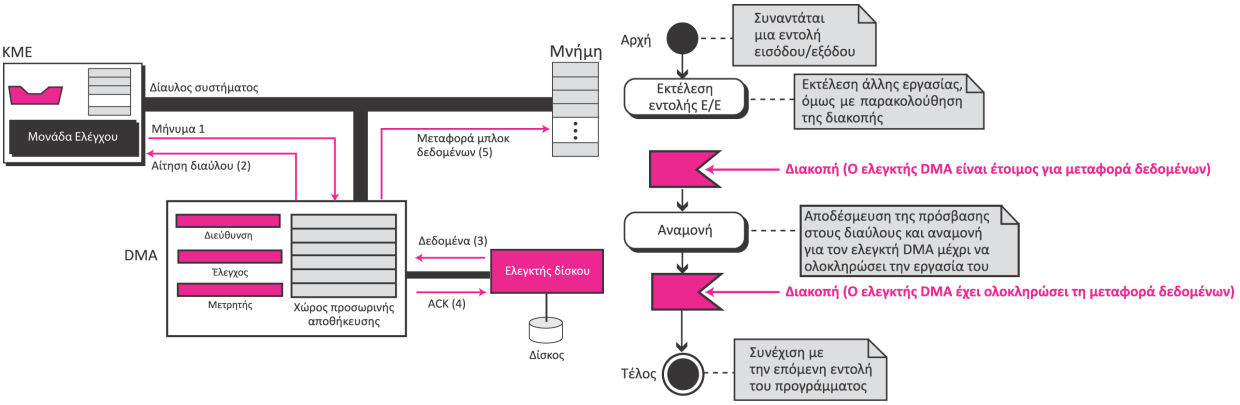
Το Λ/Σ προσδιορίζει τον τύπο της διακοπής που συνέβη ρωτώντας τις συσκευές με τη σειρά - polling, με βάση τον αριθμό τις διακοπής, ή συνδυάζοντας τις δύο μεθόδους, και μεταφέρει τον έλεγχο στον κώδικα της ρουτίνας εξυπηρέτησης διακοπών (interrupt handler).

Όταν ολοκληρωθεί η ρουτίνα εξυπηρέτησης διακοπών, ο έλεγχος επιστρέφει στο ΛΣ που αναστέλει την εκτέλεση της λειτουργίας που διακόπηκε. Λόγω των παραπάνω, ο αποδοτικός και άμεσος χειρισμός διακοπών είανι από τα πιο κρύσιμα σημεία του ΛΣ.

Διακοπή	Περιγραφή
IRQ0	Ρολόι συστήματος
IRQ1	Πληκτρολόγιο
IRQ2	Ελεγκτής διακοπών
IRQ3	Σειριακή θύρα (COM2)
IRQ4	Σειριακή θύρα (COM1)
IRQ5	Κάρτα ήχου
IRQ6	Ελεγκτής μονάδας δισκέτας
IRQ7	Παράλληλη θύρα
IRQ8	Ρολόι πραγματικού χρόνου
IRQ9	Διαθέσιμο για περιφερειακή συσκευή
IRQ10	Διαθέσιμο για περιφερειακή συσκευή
IRQ11	Διαθέσιμο για περιφερειακή συσκευή
IRQ12	Ποντίκι
IRQ13	Μαθηματικός επεξεργαστής
IRQ14	Πρωτεύων ελεγκτής IDE
IRQ15	Δευτερεύων ελεγκτής IDE

3. Μέσω άμεσης προσπέλασης μνήμης (DMA)

Χρησιμοποιείται για συσκευές Ε/Ε υψηλής ταχύτητας που μπορούν να μεταδώσουν πληροφορία σε ταχύτητες συγκρίσιμες με αυτές της μνήμης (π.χ. σκληρός δίσκος, κάρτα δικτύου). Ο ελεγκτής συσκευής μεταφέρει μεγάλα τμήματα δεδομένων από την τοπική του μνήμη κατευθείαν στην κύρια μνήμη χωρίς την παρέμβαση της ΚΜΕ.



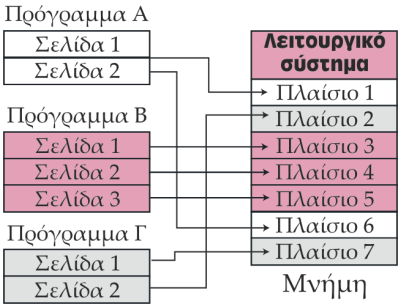
Διαχείριση μνήμης

- Μονοπρογραμματισμός

Η μνήμη χωρίζεται σε δύο τμήματα, ένα για το ΛΣ και ένα για *μία* διεργασία. Έχει χαμηλή απόδοση, ειδικά με διεργασίες που αναλώνονται σε Ε/Ε.

- Πολυπρογραμματισμός

Τεμαχισμός της μνήμης σε διάφορα τμήματα, ώστε κάθε διαφορετική εργασία να καταλαμβάνει κάθε διαφορετικό τμήμα. Έτσι κάθε χρονική στιγμή υπάρχουν πολλές διεργασίες που είναι φορτωμένες και εκτελούνται από το Υ/Σ.



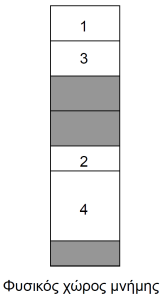
• Σελιδοποίηση

Η μνήμη χωρίζεται σε ισομεγέθη τμήματα που ονομάζονται πλαίσια (frames).

Το πρόγραμμα διαιρείται σε ισομεγέθη τμήματα που ονομάζονται σελίδες (pages).

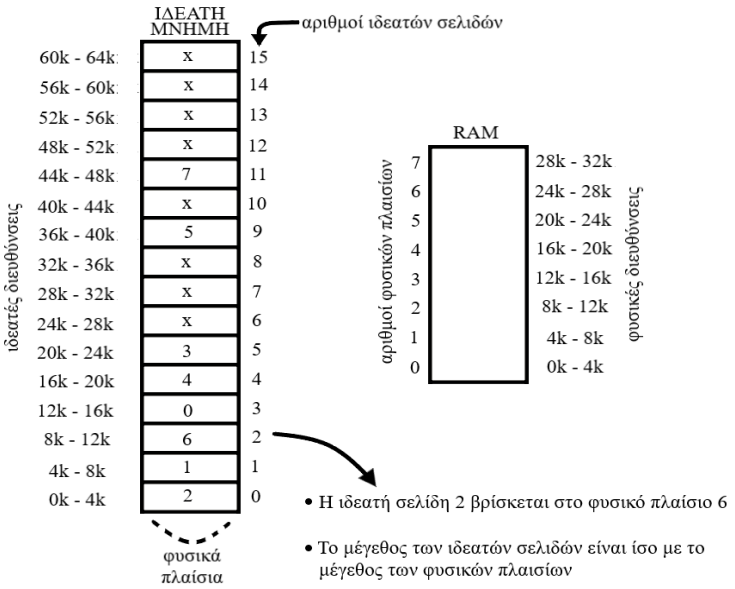
• Κατάτμηση

Το πρόγραμμα χωρίζεται σε τμήματα διαφορετικού μεγέθους (σε αντίθεση με τη σελιδοποίηση όπου τα τμήματα έχουν το ίδιο μέγεθος)



Εικονική μνήμη

Αποτελεί ένα εικονικό χώρο διευθύνσεων που καλύπτει τόσο την κύρια μνήμη όσο και μέρος κάποιας αποθηκευτικής μονάδας (συνήθως του σκληρού δίσκου). Απλουστεύει την εκτέλεση ενός προγράμματος χωρίς να έχει φορτωθεί όλο στη φυσική μνήμη του ΗΥ. Με τις εικονικές δ/νσεις υποστηρίζονται χώροι δ/νσεων που ξεπερνούν τη διαθέσιμη φυσική μνήμη, άρα & το μέγεθος ενός προγράμματος μπορεί να ξεπερνά κατά πολύ τη διαθέσιμη φυσική μνήμη.

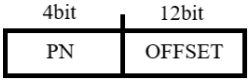


Εύρεση φυσικής διεύθυνσης

Η CPU διαβάζει και ζητάει ιδεατές διευθύνσεις από το Memory Management Unit (Μονάδα Διαχείρισης Μνήμης). Το MMU επικοινωνεί με το Page Map Table (Πίνακας Σελιδών) και διαβάζει εάν υπάρχει η ιδεατή σελίδα και που βρίσκεται στη φυσική μνήμη. Η CPU μαθαίνοντας αυτή τη πληροφορία, ζητάει πλέον τη φυσική διεύθυνση.

π.χ. Η CPU ζητάει την διεύθυνση 4090.

Ιδεατή διεύθυνση:

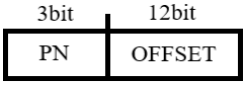


0000 111111111010 [4090 με βάση 10]

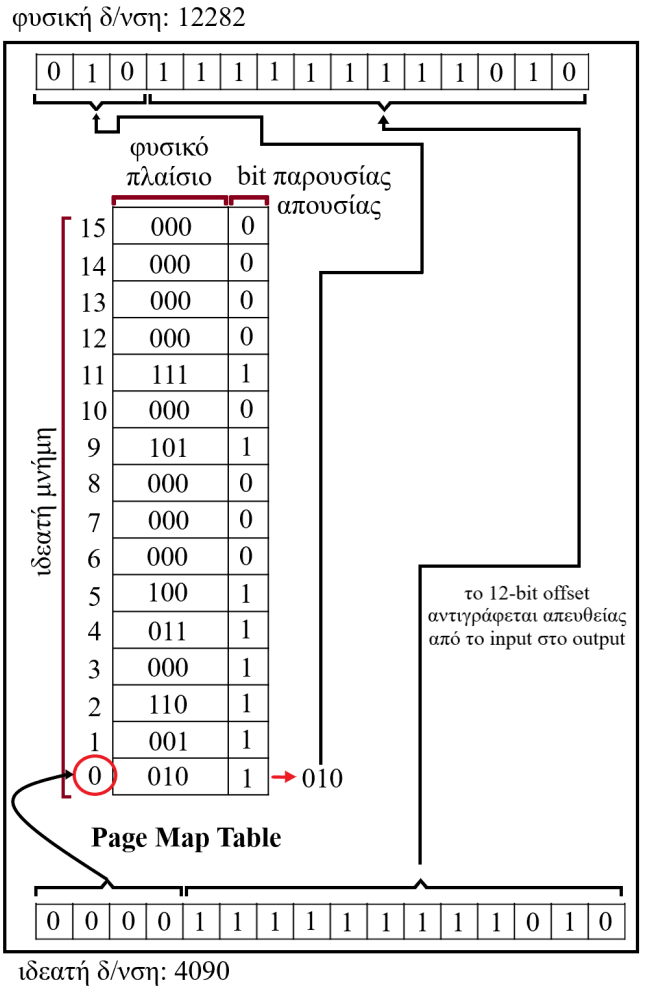
Η CPU παίρνει το Page Number κομμάτι και το δίνει στο MMU το οποίο με τη σειρά του το αναζητά στο PMT.

Η ιδεατή σελίδα 0 (0000) βρίσκεται στο φυσικό πλασίο 2 (010). Αυτή η πληροφορία δίνεται από το MMU στη CPU.

Φυσική διεύθυνση:



010 111111111010 [12282 με βάση 10]



Bit εγκυρότητας

Με κάθε καταχώρηση του PMT συσχετίζεται ένα bit. Αν αυτό είναι 1 → στη μνήμη, αν είναι 0 → στο μέσο αποθήκευσης. Αν το bit εγκυρότητας είναι 0 τότε προκαλείται σφάλμα.

Σφάλμα σελίδας

1. Εντοπισμός της θέσης της σελίδας στο μέσο αποθήκευσης
2. Εύρεση ελεύθερου πλαισίου
    a. Αν υπάρχει ελεύθερο πλαίσιο, χρησιμοποίησε το
    b. Αν δεν υπάρχει ελεύθερο πλαίσιο, χρησιμοποίησε αλγόριθμο εντοπισμού του πλαισίου που θα αντικατασταθεί
3. Μεταφορά της σελίδας στο ελεύθερο πλαίσιο
4. Ενημέρωση του πίνακα σελιδών

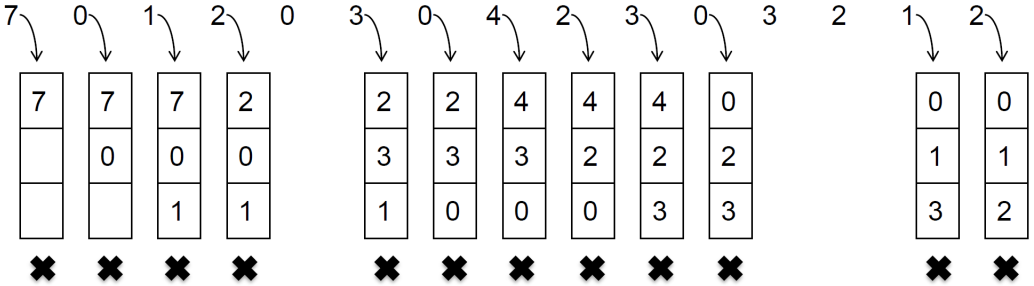
a. Τροποποίηση της αντίστοιχης καταχώρησης του πίνακα σελιδών ώστε να δείχνει στο πλαίσιο όπου είναι αποθηκευμένη η σελίδα.
b. Όριση bit εγκυρότητας σε 1, ώστε να δείχνει ότι η σελίδα είναι στη μνήμη

5. Επανεκκίνηση της διεργασίας

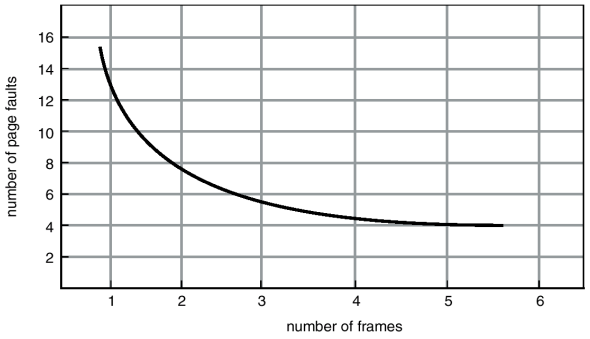
Για τη περίπτωση **2. b.** χρειάζεται ένας αλγόριθμος ο οποίος οδηγεί σε ελάχιστο αριθμό σφαλμάτων σελίδας, και επομένως σε ελάχιστες μεταφορές σελίδων από & προς το μέσο αποθήκευσης.

Αλγόριθμος First-In-First-Out (FIFO)

Ως θύμα επιλέγεται κάθε φορά η πιο παλιά σελίδα, δηλαδή αυτή που έχει παραμείνει στη μνήμη το μεγαλύτερο χρονικό διάστημα.



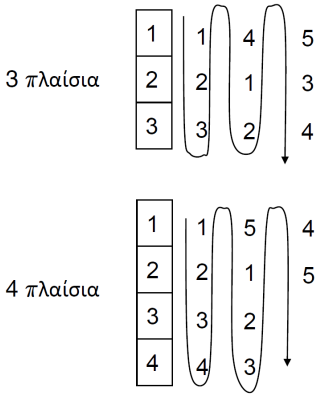
Αναμενόμενη συμπεριφορά αλγορίθμου

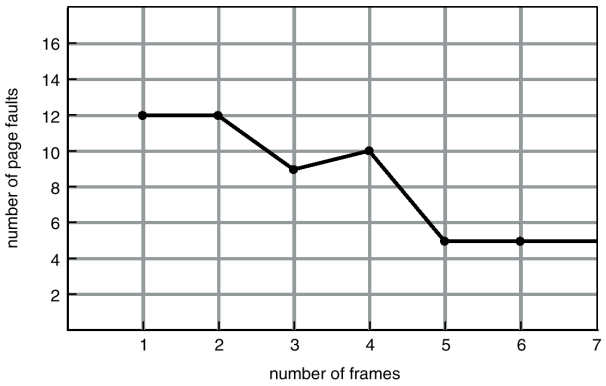


Μεγαλύτερος αριθμός πλαισίων αναμένεται να οδηγεί σε λιγότερα σφάλματα σελίδας.

Παράδοξο Belady

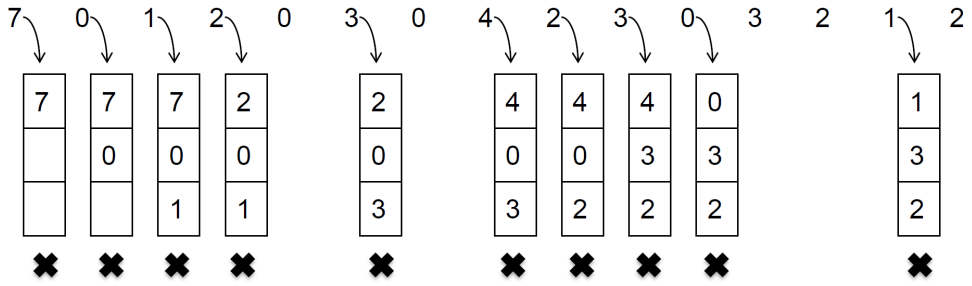
Σελίδες αναφοράς: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5





Αλγόριθμος Least Recently Used (LRU)

Ως θύμα επιλέγεται κάθε φορά η σελίδα στην οποία έγινε αναφορά λιγότερο πρόσφατα, με βάση το timestamp που έχει κάθε καταχώρηση στο πίνακα, δηλαδή γίνεται επιλογή του μικρότερου timestamp.



10 σφάλματα σελίδας

Αλγόριθμος Least Frequently Used (LFU)

Ως θύμα επιλέγεται κάθε φορά η σελίδα της οποίας ο μετρητής του πλήθους των αναφορών που έχουν γίνει σε αυτή, είναι ο μικρότερος. Αυτό γίνεται με τη λογική πως σελίδες που έχουν χρησιμοποιηθεί συχνά, έχουν μεγάλη πιθανότητα να χρησιμοποιηθούν και στο μέλλον.

Πολιτικές αντικατάστασης

Γενική αντικατάσταση (global replacement): *Μια διεργασία επιλέγει ένα πλαίσιο για αντικατάσταση από τη λίστα όλων των πλαισίων (μπορεί να πάρει ένα πλαίσιο από μια άλλη διεργασία)*

Τοπική αντικατάσταση (local replacement): *Κάθε διεργασία επιλέγει από το σύνολο των πλαισίων που έχουν ανατεθεί σε αυτή (και επομένως δεν επηρεάζει την κατάσταση των σελίδων/πλαισίων των υπολοίπων)*

Πτώση της απόδοσης (thrasing)

Αν σε μια διεργασία δεν έχουν εκχωρηθεί αρκετά πλαίσια, αυτό μπορεί να οδηγήσει σε μεγάλο αριθμό σφαλμάτων σελίδας. Έτσι η ΚΜΕ σπαταλάει το χρόνο της διεργασίας μεταφέροντας σελίδες από/προς το μέσο αποθήκευσης (thrashing).

Στο σημείο που ξεκινάει η εμφάνιση του φαινομένου (το διαπιστώνουμε με την παρακολούθηση του ρυθμού σφαλμάτων σελίδας των διεργασιών), θα πρέπει να μειωθεί ο βαθμός πολυπρογραμματισμού.

Το φαινόμενο του thrashing μπορεί να περιοριστεί με χρήση τοπικών πολιτικών τοποθέτησης & για να αποτραπεί πρέπει η κάθε διεργασία να καταλαμβάνει όσα πλαίσια χρειάζεται.

Συστήματα αρίθμησης

Ένα σύστημα αρίθμησης με βάση r, είναι ένα σύστημα που χρησιμοποιεί διακριτά σύμβολα για r ψηφία. Συχνά, ο αριθμός γράφεται μέσα σε παρένθεση και η βάση r τοποθετείται από δίπλα. π.χ. $(4)_{10} = (100)_2$

Δυαδικό σύστημα

Αφού έχει βάση r = 2, χρησιμοποιεί δύο ψηφία, το 0 και το 1. Κάθε συντελεστής a ενός δυαδικού συστήματος πολλαπλασιάζεται με 2^i.

π.χ. $(100)_2 = 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 4 + 0 + 0 = (4)_{10}$

Σε περίπτωση μη-ακέραιων αριθμών, οι δυνάμεις δεξιά της υποδιαστολής μειώνονται.

π.χ. $(100.011)_2 = 4 + 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} = 4 + 0 \times ½ + 1 \times ¼ + 1 \times ⅛ = (4.375)_{10}$

Οι πρώτες δυνάμεις του 2

$2^0 = 1$	$2^8 = 256$	ⓘ Επειδή στο δυαδικό σύστημα δεν υπάρχει ακέραια δύναμη του 2, στην οποία αν το υψώσουμε παίρνουμε 1000, στους υπολιστές **Κ εννούμε 1024** επειδή το 2^{10} είναι η πλησιέστερη δύναμη του 2 στο 1000
$2^1 = 2$	$2^9 = 512$	
$2^2 = 4$	$2^{10} = 1024 = 1K$	
$2^3 = 8$	$2^{11} = 2048 = 2K$	
$2^4 = 16$	$2^{12} = 4096 = 4K$	
$2^5 = 32$	$2^{13} = 8192 = 8K$	
$2^6 = 64$	$2^{14} = 16384 = 16K$	
$2^7 = 128$	$2^{15} = 32768 = 32K$	

Μετατροπή ακέραιου δεκαδικού σε δυαδικό αριθμό

1. Μέθοδος διαιρέσεων

Ξεκινάω από έναν δεκαδικό αριθμό και διαιρούμε διαδοχικά με το 2 λαμβάνοντας ένα ακέραιο πηλίκο και ένα υπόλοιπο. Το πηλίκο διαιρείται ξανά με το 2 και παράγει νέο πηλίκο και υπόλοιπο. Η διαδικασία σταματάει όταν το πηλίκο γίνει 0. Στη συνέχεια γράφουμε τα υπόλοιπα διαίρεσης με σειρά: πιο πρόσφατο → λιγότερο πρόσφατο. *π.χ.*

Δεκαδικός αριθμός	Ακέραιο πηλίκο διαίρεσης με το 2	Υπόλοιπο διαίρεσης με το 2
12	6	0 ↑
6	3	0 ↑
3	1	1 ↑
1	0	1 ↑

Επομένως το $(12)_{10} = (00001100)_2$

ⓘ Όλοι οι αριθμοί γράφονται σε μονάδες πολλαπλάσιες του byte = 8bit.

ⓘ Ο signed int ξοδεύει ένα bit (το αριστερότερο) για πρόσημο (1 - αρνητικό, 0 - θετικό), σε σχέση με το unsigned int όπου το αριστερότερο bit αντιστοιχεί σε ποσότητα).

2. Αριθμός λίγο μεγαλύτερος από δύναμη του 2

Βρίσκουμε τη μεγαλύτερη δύναμη του 2 που "χωράει" στο ζητούμενο δεκαδικό αριθμό. Επαναλαμβάνουμε μια σειρά προσθέσεων δίνοντας τιμές στους άλλους συντελεστές. *π.χ.*

Η μεγαλύτερη ακέραια δύναμη του 2 στην οποία αν αυτό πολλαπλασιαστεί θα "χωράει" το $(25)_{10}$ είναι η 4η όπου $2^4 = 16$. Από αυτό περισσεύουν 25-16 = 9

Ο επόμενος συντελεστής αντιστοιχεί σε *8αδα*. Χωράει *8αδα* στο 9; ΝΑΙ και από αυτό περισσεύει 9-8 = 1

Ο επόμενος συντελεστής αντιστοιχεί σε *4αδα*. Χωράει *4αδα* στο 1; ΟΧΙ

Ο επόμενος συντελεστής αντιστοιχεί σε *2αδα*. Χωράει *2αδα* στο 1; ΟΧΙ

Ο επόμενος συντελεστής αντιστοιχεί σε *1αδα*. Χωράει *1αδα* στο 1; ΝΑΙ και από αυτό περισσεύει 1-1 = 0

Επομένως το $(25)_{10} = (00011001)_2$

ⓘ Οι άρτιοι αριθμοί έχουν δεξιότερο bit = 0, ενώ οι περιττοί = 1

ⓘ Αν βρούμε ότι η μεγαλύτερη δύναμη του 2 που χωράει στον ζητούμενο αριθμό είναι η x, τότε απαιτούνται x+1 bit. *π.χ.* στο 75 η δύναμη αυτή είναι η x = 6 ($2^6 = 64$). Άρα ο αριθμός απαιτεί 6 + 1 = 7 bit

3. Αριθμός λίγο μικρότερος από δύναμη του 2

Όταν προσπαθώ να μετατρέψω δεκαδικό ο οποίος είναι λίγο μικρότερος από ακριβής δύναμη του 2 μπορώ να βρω την δύναμη του 2 που είναι λίγο μεγαλύτερη από τον αριθμό που ψάχνουμε - ας πούμε 2^n, να γράψω το 2^n-1, το οποίο είναι n μονάδες και να αφαιρέσω κατάλληλα. *π.χ.*

Ο αριθμός $(4090)_{10}$ είναι λίγο μικρότερος από τον $(4096)_{10}$ ο οποίος είναι = 2^{12}. Μπορούμε να συμβολίσουμε τον αριθμό $(4095)_{10}$ ο οποίος είναι 2^{12}-1 ως μια ακολουθία 12 άσσων. δηλ. $(4095)_{10} = 2^{12}-1 = 111111111111$. Μπορώ να χρησιμοποιώ αυτό και να

αφαιρέσω κατάλληλα. $(4095)_{10}$ - $(4090)_{10}$ = $(5)_{10}$ αφαιρόντας από τους άσσους από τις θέσεις 2^2 και 2^0 (δηλ. 4 και 1).

Επομένως το $(4090)_{10}$ = $(0000111111111010)_2$

Μετατροπή μη-ακέραιου δεκαδικού σε δυαδικό αριθμό

Το ακέραιο μέρος μετατρέπεται με έναν από τους παραπάνω τρόπους ενώ το κλασματικό μέρος απαιτεί μια σειρά πολλαπλασιασμών με το 2 έως ότου το κλασματικό μέρος που θα πάρουμε να είναι 0 και επειδή αυτό δεν είναι βέβαιο, μέχρι να φτάσουμε στην επιθυμητή ακρίβεια. Στη συνέχεια γράφουμε τους συντελεστές με σειρά: λιγότερο πρόσφατο → πιο πρόσφατο. *π.χ.*

Το ακέραιο μέρος του $(20.625)_{10}$ θα γίνει $(10100)_2$ με βάση τα προηγούμενα.

Το κλασματικό μέρος:

Αριθμός	Γινόμενο με το 2	Κλασματικό μέρος	Συντελεστής (ακέραιο μέρος πολ/σμού)
0.625	1.25	0.25	1 ↓
0.25	0.5	0.5	0 ↓
0.5	1.0	0	1 ↓

Επομένως το $(20.625)_{10}$ = $(10100.101)_2$

Πρόσθεση στο δυαδικό σύστημα

Υπάρχουν μόνο 4 πιθανά αποτελέσματα:

1. 0 + 0 = 0
2. 0 + 1 = 1
3. 1 + 0 = 1
4. 1 + 1 = 0 και ένα κρατούμενο

.

π.χ. 0 1 1 0 Πρώτη στήλη: 0 + 0 = 0.

0 1 1 0 + Δεύτερη στήλη: 1 + 1 = 0 και ένα κρατούμενο.

1 1 Τρίτη στήλη: 1 + 1 + 1 = 0 + 1 = 1 και ένα κρατούμενο.

——————— Τέταρτη στήλη: 0 + 0 + 1 = 0 + 1 = 1

1 1 0 0 =

Οι πρώτοι 16 ακέραιοι σε 4 συστήματα αρίθμησης

Δεκαδικό	Δυαδικό	Οκταδικό	Δεκαεξαδικό
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Συμπληρώματα αριθμών

Συμπλήρωμα (complement) ενός αριθμού N είναι ένας άλλος αριθμός, ο οποίος συμπληρώνει τον N ως προς έναν αριθμό αναφοράς

• Ως προς βάση μείον ένα: $r^n - 1 - N$

• Ως προς βάση: $r^n - N$

Δεκαδικό σύστημα

π.χ. αριθμός: N = 975, βάση: r = 10, πλήθος ψηφίων αριθμού N: n = 3

- Συμπλήρωμα ως προς βάση μείον ένα (ως προς 9)

Όταν αφαιρούμε $r^n - 1$ προκύπτουν n 9 (στο παράδειγμα με r = 10 και n = 3, $r^n - 1 = 10^3 - 1 = 1000 - 1 = 999$). Επομένως για να βρω τη βάση αρκεί να αφαιρέσω όλα τα ψηφία του αριθμού από 9, δηλαδή 999 - 975 = 024

- Συμπλήρωμα ως προς βάση (ως προς 10)

Αφαιρούμε r^n-N, όπου r^n είναι μία μονάδα ακολουθούμενη από n μηδενικά. Όσο αφαιρείται το N από r^n, αυτό που συμβαίνει είναι ότι το πρώτο ψηφίο του N από δεξιά αφαιρείται από 10 (δηλαδή από την βάση), ενώ τα άλλα από το 9 (βάση μείον 1).

```
  0 9 9
  1 0 1 0
-   9 7 5
      2 5
```

Δυαδικό σύστημα

0 - 0 : 0 και δεν έχουμε δανειζόμενο

0 - 1 : 1 και ένα δανειζόμενο

1 - 0 : 1 και δεν έχουμε δανειζόμενο

1 - 1 : 0 και δεν έχουμε δανειζόμενο

- Συμπλήρωμα ως προς βάση μείον ένα (ως προς 1)

π.χ. αριθμός: N = 10101, βάση: r = 2, πλήθος ψηφίων αριθμού N: n = 5

Γνωρίζουμε πως $r^n = 2^5 = 32 = 100000$. Αν αφαιρέσουμε 32 - 1 (δηλ. $100000 - 000001)_2$ έχουμε:

Δηλαδή ουσιαστικά η ποσότητα $r^n - 1$ είναι n μονάδες άρα το συμπλήρωμα ως προς βάση μείον ένα προκύπτει αν αφαιρέσω όλα τα ψηφία του N από 1. Λόγω του γεγονότος ότι $r^n - 1$ είναι n μονάδες, τα bit του αριθμού N ουσιαστικά αντιστρέφονται. Δηλαδή $\Sigma_1(10101) = 01010$

Κανόνας εύρεσης $\Sigma_1(N)$: Αντιστρέφουμαι τα bit του N

- Συμπλήρωμα ως προς βάση (ως προς 2)

π.χ. αριθμός: N = 1001000, βάση r = 2, πλήθος ψηφίων αριθμού N: n = 8

Έχουμε να αφαιρέσουμε το N από το r^n, μια ποσότητα που γράφεται ως μια μονάδα και n μηδενικά. Αυτό σημαίνει τα εξής:

1. Αν το N έχει στο δεξί μέρος μηδενικά, τότε αυτά (όσα είναι) θα παραμείνουν.
2. Όταν συναντήσουμε τον πρώτο άσσο από δεξία, αυτός αφαιρείται από 0 και δίνει αποτέλεσμα 1 και ένα δανειζόμενο άρα ο πρώτος άσσος από δεξιά θα παραμείνει.
3. Το δανειζόμενο που κατέβηκε λόγω του πρώτου άσσου από δεξιά, θα οδηγήσει στο να κατεβαίνουν δανειζόμενα σε κάθε αφαίρεση από εκείνο το σημείο. Άρα όλα τα bit που ακολουθούν μετά τον πρώτο άσσο στα δεξιά αντιστρέφονται.

```
  1 1 1 1
  1 0 0 0 0 0 0 0
- 1 0 0 1 0 0 0
-----------------
  0 1 1 1 0 0 0
```

Κανόνας εύρεσης $\Sigma_2(N)$: Τυχόν μηδενικά στο δεξί μέρος παραμένουν, ο πρώτος άσσος από δεξιά παραμένει και όλα τα άλλα αντιστρέφονται.

Αναπαράσταση αρνητικών

Χρησιμοποιείται το συμπλήρωμα Σ_2 γιατί έχει την ιδιότητα πως αν προσθέσουμε 2 συμπλήρωματικούς και αγνοήσουμε το αριστερότερο bit που θα είναι 1, τότε το αποτέλεσμα που απομένει είναι 0.

π.χ. Ο αριθμός 18 σε 1 byte είναι: 00010010
Το -18 είναι το $\Sigma_2(18)$ = 11101110

Αν κάνουμε τη πρόσθεση των 2 έχουμε:

```
    1 1 1 1 1 1 1
      0 0 0 1 0 0 1 0
+     1 1 1 0 1 1 1 0
-----------------------
  1 0 0 0 0 0 0 0 0
```

Ο άσσος είναι έξω από τα όρια του byte και αγνοείται. Το υπόλοιπο είναι 0.

Σημαντικά ερωτήματα

• Ποιος είναι ο μεγαλύτερος μη-προσημασμένος αριθμός που χωράει σε 1 byte;
11111111 = 255

• Ποιος είναι ο μεγαλύτερος μη-προσημασμένος αριθμός που χωράει σε 2 bytes;
11111111 11111111 = 65535

• Ποιος είναι ο μικρότερος μη-προσημασμένος αριθμός που χωράει σε 1 byte;
00000000 = 0

• Ποιος είναι ο μικρότερος μη-προσημασμένος αριθμός που χωράει σε 2 bytes;
00000000 00000000 = 0

• Ποιος είναι ο μεγαλύτερος προσημασμένος αριθμός που χωράει σε 1 byte;
01111111 = +127 *[7 μονάδες και το αριστερότερο bit είναι το + πρόσημο]*

• Ποιος είναι ο μεγαλύτερος προσημασμένος αριθμός που χωράει σε 2 bytes;
01111111 11111111 = +32767 *[15 μονάδες και το αριστερότερο bit είναι το + πρόσημο]*

• Ποιος είναι ο μικρότερος προσημασμένος αριθμός που χωράει σε 1 byte;
10000000 ⇒ αν πάρουμε το Σ_2 αυτού έχουμε 10000000 = 128 άρα πρόκειται για το -128

• Ποιος είναι ο μικρότερος προσημασμένος αριθμός που χωράει σε 2 bytes;
10000000 00000000 ⇒ αν πάρουμε το Σ_2 έχουμε 10000000 00000000 = 32768 άρα πρόκειται για το -32768

Επομένως στη περίπτωση όπου ένας αριθμός είναι μη-προσημασμένος, για να βρούμε ποιος είναι ακολουθούμε τον τρόπο μετατροπής, ενώ αν είναι αρνητικός προσημασμένος παίρνουμε το συμπλήρωμα ως προς 2 για να βρούμε τον αντίστοιχο θετικό.

Αφαίρεση προσημασμένων
Για την αφαίρεση M-N, προσθέτουμε στο M το Σ_2 του N
1. Αν M>=N τότε αγνοούμε το τελικό κρατούμενο

Για M = 45 και N = 33 θα έχουμε:

M = $(45)_{10}$ = $(00101101)_2$ και N = $(33)_{10}$ = $(00100001)_2$ με το $\Sigma_2(33)$ = 11011111

```
  1 1 1 1 1 1 1 1
    0 0 1 0 1 1 0 1
+   1 1 0 1 1 1 1 1
-----------------------
  1 0 0 0 0 1 1 0 0
```

Αγνοούμε το τελικό κρατούμενο και έχουμε $(00001100)_2 = (12)_{10}$

2. Αν M<N τότε βρίσκουμε το Σ_2 του αποτελέσματος για να δούμε ποιος είναι ο αριθμός

Για M = 45 και N = 56 θα έχουμε:

M = $(45)_{10}$ = $(00101101)_2$ και N = $(56)_{10}$ = $(00111000)_2$ με το $\Sigma_2(56)$ = 11001000

```
      1
    0 0 1 0 1 1 0 1
+   1 1 0 0 1 0 0 0
-----------------------
    1 1 1 1 0 1 0 1
```

Βρίσκουμε το Σ_2 του αποτελέσματος το οποίο είναι $(00001011)_2$

Αυτό είναι ίσο με $(11)_{10}$ άρα το αποτέλεσμα είναι $(-11)_{10}$

Πράξεις με προσημασμένους αριθμούς

Με δεδομένο ότι με 1 byte μπορούμε να χωρέσουμε τους αριθμούς από -128 ως 127, υπερχείλιση έχουμε όταν το αποτέλεσμα μιας πράξης είναι εκτός του διαστήματος αυτού.

Η υπερχείλιση ελέγχεται με βάση τα 2 τελευταία κρατούμενα των πράξεων. Αν αυτά είναι ίσα (είτε 0 είτε 1) δεν έχουμε υπερχείλιση προσήμου, αλλά αν δεν είναι ίσα τότε έχουμε υπερχείλιση.

- A = 96 και B = 50
- Ai και Bi είναι αντίστοιχα τα bit του αριθμού A και B
- Ci είναι τα κρατούμενα που προκύπτουν από πράξη στην αμέσως προηγούμενη στήλη με το C0 να είναι το αρχικό κρατούμενο & το C1 το κρατούμενο που προκύπτει από την πρόσθεση των bit A0 και B0 κ.ο.κ.
- Για τους ελέγχους υπερχείλισης ελέγχουμε τα bit C7 και C8

π.χ. όπου δεν υπάρχει υπερχείλιση
A - B = 96 - 50

A = $(96)_{10}$ = $(01100000)_2$ και B = $(50)_{10}$ = $(00110010)_2$ με το $\Sigma_2(50)$ = $(11001110)_2$

8	7	6	5	4	3	2	1	0	i	
1	**1**	0	0	0	0	0	0	0	Ci	Για υπερχείλιση ελέγχουμε το C7 και C8 και αφού είναι ίσα, δεν έχουμε. Αγνούμε το τελικό κρατούμενο και το αποτέλεσμα είναι $(00101110)_2$ = $(46)_{10}$
	0	1	1	0	0	0	0	0	Ai	
	1	1	0	0	1	1	1	0	Bi	
	0	0	1	0	1	1	1	0	Si	

ⓘ Όταν το αποτέλεσμα είναι θετικό, τα δύο τελευταία κρατούμενα είναι 1, και όταν είναι αρνητικό, αυτά είναι 0.

π.χ. όπου υπάρχει υπερχείλιση θετικού προσήμου
A + B = 96 + 50

A = $(96)_{10}$ = $(01100000)_2$ και B = $(50)_{10}$ = $(00110010)_2$

8	7	6	5	4	3	2	1	0	i	
0	**1**	1	0	0	0	0	0	0	Ci	Από το αποτέλεσμα προκύπτει πως 96+50 = κάποιος αρνητικός αφού το πρόσημο έχει υπερχειλίσει εκτός των ορίων του αριθμού και είναι το κρατούμενο C8. Αν γράψουμε τον αριθμό 0 10010010 = 146 αλλά το πρόσημο 0 είναι εκτός των ορίων και έτσι δεν χωράει σε 1 byte. Ο αθροιστής στέλνει μήνυμα στη CPU ότι υπάρχει σφάλμα.
	0	1	1	0	0	0	0	0	Ai	
	0	0	1	1	0	0	1	0	Bi	
	1	0	0	1	0	0	1	0	Si	

π.χ. όπου υπάρχει υπερχείλιση αρνητικού προσήμου
- A - B = - 96 - 50

$A = (96)_{10} = (01100000)_2$ με το $\Sigma_2(96) = (10100000)_2$
$B = (50)_{10} = (00110010)_2$ με το $\Sigma_2(50) = (11001110)_2$

8	7	6	5	4	3	2	1	0	i
1	**0**	0	0	0	0	0	0	0	Ci
	1	0	1	0	0	0	0	0	Ai
	1	1	0	0	1	1	1	0	Bi
	0	1	1	0	1	1	1	0	Si

Από το αποτέλεσμα προκύπτει πως -96-50 = κάποιος θετικός αφού το πρόσημο έχει υπερχειλίσει εκτός των ορίων του αριθμού και είναι το κρατούμενο C8. Αν γράψουμε τον αριθμό 1 01101110 = -146 αλλά το πρόσημο 1 είναι εκτός των ορίων και έτσι δεν χωράει σε 1 byte. Ο αθροιστής στέλνει μήνυμα στη CPU ότι υπάρχει σφάλμα

Συμπερασματικά έχουμε 4 περιπτώσεις *[2 με υπερχείλιση και 2 χωρίς]*:
• C7 = C8 = 0 τότε το αποτέλεσμα είναι αρνητικό και το κρατούμενο αγνοείται.
• C7 = C8 = 1 τότε το αποτέλεσμα είναι θετικό και το κρατούμενο αγνοείται.
• C7 = 0 και C8 = 1, τότε έχουμε υπερχείλιση αρνητικού προσήμου αφού προστέθηκαν αρνητικοί αριθμοί και το αποτέλεσμα είναι θετικό.
• C7 = 1 και C8 = 0, τότε έχουμε υπερχείλιση θετικού προσήμου αφού προστέθηκαν θετικοί αριθμοί και το αποτέλεσμα είναι αρνητικό.

Αριθμοί κινητής υποδιαστολής

Με το να φέρουμε έναν αριθμό σε μορφή 1.xxxxx, μπορούμε να μην αποθηκεύσουμε το 1, αλλά να υποννοείται το οποίο μας επιτρέπει να διπλασιάσουμε την ακρίβεια έχοντας 1 επιπλέον bit.

Μετατροπή αριθμού σε μορφή 1.xxxxx

Έστω πως έχουμε τον αριθμό 100101.11 = 100101.11 x 1 = 100101.11 x 2^0, και πως θέλουμε να τον φέρουμε σε μορφή 1.xxxxx, δηλαδή να μετακινήσουμε την υποδιαστολή κατά 5 θέσεις προς τα αριστερά. Αυτό θα έχει ως συνέπεια να αυξηθεί ο εκθέτης κατά 5, δηλαδή: 100101.11 x 2^0 = 1.0010111 x 2^5.

Η υποδιαστολή μπορεί αντίστοιχα να μετακινηθεί και προς τα δεξιά. Έστω πως έχουμε τον αριθμό 0.011 και θέλουμε να τον φέρουμε σε μορφή 1.xxxxx. Θα πρέπει να μετακινήσουμε την υποδιαστολή κατά 2 θέσεις δεξιά και επομένως θα γίνει: 1.1 x 2^{-2}

Πρότυπο IEEE 754

Έχει 3 τμήματα, αυτά του προσήμου αριθμού (1 για αρνητικό και 0 για θετικό), αυτό του εκθέτη (ο οποίος ορίζει το τμήμα της υποδιαστολής και έχει μέγεθος 8 bit - άρα μη προσημασμένος) και το κλασματικό τμήμα (mantissa).

Ανεξάρτητα της τιμής του εκθέτη, αυτό που αποθηκεύεται στην πραγματικότητα είναι η λεγόμενη υπέρβαση-127. Δηλαδή αν ο εκθέτης είναι 2 θα γραφτεί E' = 127 + 2 = 129

Απλή ακρίβεια (32 bits)

Ας πούμε πως θέλουμε τον αριθμό 28.625 σε αναπαράσταση απλής ακρίβειας:

1. Γράφουμε τον αριθμό σε δυαδική μορφή: $(28.625)_{10} = (11100.101)_2$

2. Φέρνουμε τον αριθμό σε μορφή 1.xxxxx: για να έρθει ο παραπάνω αριθμός σε αυτή τη μορφή πρέπει να μετακινηθεί η υποδιαστολή 4 θέσεις προς τα αριστερά, και δηλαδή έχουμε: 11100.101 x 2^0 = 1.1100101 x 2^4

3. Γράφω το E': αφού E = 4 τότε E' = 127 + 4 = 131 το οποίο είναι $(10000011)_2$

4. Αποθηκεύω το κλασματικό μέρος χωρίς το **1**.xxxxx

Επομένως θα έχουμε: 0 10000011 1100101 (& 16 μηδενικά)

Τιμές εκθέτη
Για κανονικούς αριθμούς το E' είναι από 1 (δηλ. E = -126) ως 254 (δηλ. E = 127).

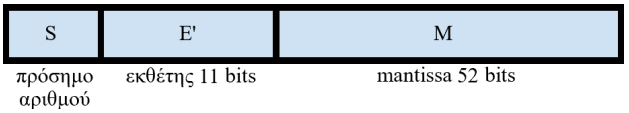
Η τιμή E' = 255 (δηλ. E = 128) με Mantissa <> 0 αναπαριστά μη αριθμό (π.χ. αρνητική ρίζα).
Η τιμή E' = 255 (δηλ. E = 128) με Mantissa = 0 αναπαριστά διαίρεση με το 0.

Η τιμή E' = 0 (δηλ. E = -127) αναπαριστά το 0. Αυτό συμβαίνει καθώς:

ⓘ Το 1 αναπαρίσταται ως εξής:
$1 = 1.0 \times 2^0$, άρα E = 0 ⇒ E' = 127 + 0 ⇒ E' = 127 και έχουμε:
0 01111111 00000000000000000000000

ⓘ Το 0 συμβαίνει να έχει την ίδια αναπάρασταση, οπότε για να αποθηκευτεί έχουμε:
0 00000000 00000000000000000000000
[το 0 αναπαρίσταται από τη τιμή E = -127 άρα E' = 127-127 = 0]

Διπλή ακρίβεια (64 bits)
Σε αυτή τη περίπτωση έχουμε υπέρβαση - 1023 και το E' έχει τιμές από 0-2047 με τις ειδικές περιπτώσεις να είναι E' = 0 (για το 0) και E' = 2047 (για μη αριθμούς και διαιρέσεις με το 0).



Συγκρίσεις αριθμών
Όταν συγκρίνουμε θετικούς ή αρνητικούς αριθμούς ή συνδυασμό των 2 πρέπει να χρησιμοποιήσουμε συγκριτή μεγέθους ο οποίος είναι πολύπλοκος διότι πρέπει να λάβει υπόψη πολλές διαφορετικές περιπτώσεις.

σύγκριση θετικών αριθμών
π.χ. έστω σύγκριση του 86 $(01010110)_2$ με το 54 $(00110110)_2$
Ξεκινάμε από τα αριστερά και βλέπουμε πως τα πρώτα bit είναι ίσα (0 = 0) που σημαίνει πως οι αριθμοί είναι και οι δύο θετικοί. Συνεχίζουμε και στο επόμενο bit βλέπουμε πως ο ένας αριθμός έχει 1 και ο άλλος 0, άρα ο αριθμός με το 1 είναι μεγαλύτερος 01010110 > 00110110 ή 86 > 54.

σύγκριση αρνητικών αριθμών
π.χ. έστω σύγκριση του -54 $(11001010)_2$ με το -86 $(10101010)_2$
Ξεκινάμε από τα αριστερά και βλέπουμε πως τα πρώτα bit είναι ίσα (1 = 1) που σημαίνει πως οι αριθμοί είναι και οι δύο αρνητικοί. Συνεχίζουμε και στο επόμενο bit βλέπουμε πως ο ένας αριθμός έχει 1 και ο άλλος 0, άρα ο αριθμός με το 1 είναι μεγαλύτερος 11001010 > 10101010 ή -54 > -86.

σύγκριση συνδυασμού θετικών και αρνητικών αριθμών
π.χ. έστω σύγκριση του 86 $(01010110)_2$ με το -86 $(10101010)_2$

Ξεκινάμε από τα αριστερά και βλέπουμε πως το πρώτο bit του 86 είναι 0 και του -86 είναι 1 άρα 01010110 > 10101010 ή 86 > -86.

ⓘ Αν είχα έναν τρόπο να συγκρίνω αποκλειστικά θετικούς αριθμούς, η σύγκριση θα ήταν ταχύτερη αφού θα είχα μόνο μια περίπτωση, και γι' αυτό χρησιμοποιείται η υπέρβαση, ώστε οι εκθέτες να είναι θετικοί αριθμοί.

Πρόσθεση αριθμών κινητής υποδιαστολής
π.χ. έστω πως θέλουμε να προσθέσουμε τους αριθμούς 28.625 και 118.5

28.625 = 11100.101 = 1.1100101 x 2^4 άρα E' = 127 + 4 = 131 = 10000011

118.5 = 1110110.1 = 1.1101101 x 2^6 άρα E' = 127 + 6 = 133 = 10000101

Συγκρίνουμε τους εκθέτες των 2 αριθμών και φέρνουμε τον αριθμό με τον μικρότερο εκθέτη στη κατάλληλη μορφή, ώστε οι εκθέτες να εξισωθούν. Το 28.625 έχει εκθέτη 4 και πρέπει να γίνει 6 άρα πρέπει να προστεθούν αριστερά 2 μηδενικά. Δηλαδή 1.1100101 x 2^4 = 0.011100101 x 2^6

Στη συνέχεια προσθέτουμε τα κλασματικά μέρη:

```
  1 1   1 1 1       1
    1 . 1 1 0 1 1 0 1 0 0
+   0 . 0 1 1 1 0 0 1 0 1
-------------------------
  1 0 . 0 1 0 0 1 1 0 0 1
```

Το αποτέλεσμα 0.010011001 δεν είναι κανονικοποιημένο και για να κανονικοποιηθεί πρέπει να βάλουμε το κρατούμενο που παρήχθη από την πρόσθεση, στο αποτέλεσμα. Άρα η υποδιαστολή θα πάει μια θέση αριστερά, που σημαίνει πως ο εκθέτης θα γίνει 6 + 1 = 7. Τελικά θα αποθηκευτεί:

S	E'=127+7	M
0	10000110	0010011001(+13 μηδενικά)

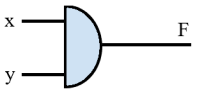
Αφαίρεση αριθμών κινητής υποδιαστολής
Ομοίως με τα παραπάνω αλλά αφαιρούνται τα κλασματικά μέρη.

Λογικές πύλες

Λογική πύλη είναι ένα μικρό κύκλωμα το οποίο δίνει συγκεκριμένη τιμή εξόδου για κάθε συνδυασμό τιμών στην είσοδο. Αν οι είσοδοι είναι N σε πλήθος δημιουργούν συνολικά 2^N συνδυασμούς. Δηλαδή αν έχω π.χ. N = 2 εισόδους, σχηματίζουν $2^2 = 4$ συνδυασμούς (ελαχιστόροι).

Λογική πύλη ΚΑΙ (AND)

Ο πίνακας αυτός, ο οποίος για κάθε συνδυασμό δείχνει τη τιμή εξόδου λέγεται πίνακας αληθείας. Μέσα από αυτόν, μπροούμε να μελετήσουμε τη συμπεριφορά μιας πύλης.



Είσοδος		Έξοδος
X	Y	F
0	0	0
0	1	0
1	0	0
1	1	1

Κάθε έξοδος σε όλα τα κυκλώματα μπορεί να **εκφραστεί αλγεβρικά**. Για να γίνει αυτό:

1. Κοιτάζουμε τους συνδυασμούς εισόδων στους οποίους η F = 1
2. Γι' αυτούς τους συνδυασμούς, εξετάζουμε τις εισόδους. Αν είσοδος = 1 γράφεται σε κανονική μορφή, ενώ αν είσοδος = 0 γράφεται σε συμπληρωματική μορφή (οι συμπληρωματικές μορφές γράφονται με τόνο π.χ. Α').
3. Γράφουμε κάθε συνδυασμό ως γινόμενο (λογικό ΚΑΙ).

π.χ. Ακολουθώντας τα παραπάνω βήματα και αναφορικά με τη λογική πύλη ΚΑΙ, μπορεί να εκφραστεί ως F = XY

π.χ. Βάση του παρακάτω πίνακα αληθείας για 2 εισόδους, F;

X	Y	δεκαδική τιμή συνδυασμού (ελαχιστόρος)	F
0	0	0	0
0	1	1	1
1	0	2	0
1	1	3	1

Η F = 1 για 2 συνδυασμούς, οι οποίοι δεν μπορούν να ισχύουν ταυτόχρονα (ή ο ένας ή ο άλλος). Η F = 1 δηλαδή, όταν (X = 0 ΚΑΙ Y = 1) Ή (X = 1 ΚΑΙ Y = 1)

Επομένως θα έχουμε: F = X'Y + XY *(επειδή το ή συμβολίζεται με +)*

π.χ. Βάση του παρακάτω πίνακα αληθείας για 3 εισόδους, F;

A	B	C	δεκαδική τιμή συνδυασμού (ελαχιστόρος)	F
0	0	0	0	1
0	0	1	1	0
0	1	0	2	1
0	1	1	3	0

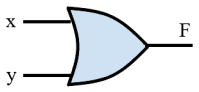
1	0	0	4	0
1	0	1	5	0
1	1	0	6	0
1	1	1	7	1

Η F = 1 για 3 συνδυασμούς οι οποίοι δεν μπορούν να ισχύουν ταυτόχρονα. Η F = 1 δηλαδή, όταν (Α = 0 ΚΑΙ Β = 0 ΚΑΙ Γ = 0) Ή (Α = 0 ΚΑΙ Β = 1 ΚΑΙ Γ = 0) Ή (Α = 1 ΚΑΙ Β = 1 ΚΑΙ Γ = 1).

Επομένως θα έχουμε: F = Α'Β'Γ' + Α'ΒΓ' + ΑΒΓ

Λογική πύλη Ή (OR)

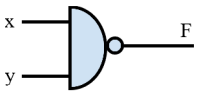
Η πύλη Ή δίνει 1 όταν έστω μια είσοδος της είναι ίση με 1. Αλγεβρικά γράφεται F = X + Y, που προκύπτει από την εφαρμογή ταυτότητας στο: F = X'Y + XY' + XY



Είσοδος		Έξοδος
X	Y	F
0	0	0
0	1	1
1	0	1
1	1	1

Λογική πύλη ΟΧΙ-ΚΑΙ (NAND)

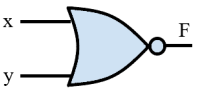
Η πύλη ΟΧΙ-ΚΑΙ είναι ουσιαστικά η AND αλλά με άρνηση, η οποία υποδηλώνεται από το κύκλο. Δίνει F = 1 όταν υπάρχει τουλάχιστον ένα 0 στις εισόδους (δηλ. αντίθετα της AND). Αλγεβρικά γράφεται F = (XY)' *(δεδομένου πως AND: F= XY)*



Είσοδος		Έξοδος
X	Y	F
0	0	1
0	1	1
1	0	1
1	1	0

Λογική πύλη ΟΥΤΕ (NOR)

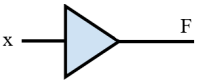
Η πύλη ΟΥΤΕ είναι ουσιαστικά η OR αλλά με άρνηση, η οποία υποδηλώνεται από το κύκλο. Δίνει F = 1 όταν όλες οι είσοδοι είναι 0 (δηλ. αντίθετα της OR). Αλγεβρικά γράφεται F = (X + Y)' *(δεδομένου πως OR: F = X + Y)*



Είσοδος		Έξοδος
X	Y	F
0	0	1
0	1	0
1	0	0
1	1	0

Λογική πύλη ΑΠΟΜΟΝΩΤΗΣ (buffer)

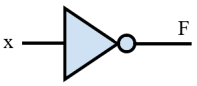
Έχει μια είσοδο και δίνει ως έξοδο ότι είναι αυτή. Επομένως F = X



Είσοδος	Έξοδος
X	F
0	0
1	1

Λογική πύλη ΑΝΤΙΣΤΡΟΦΕΑΣ (NOT)

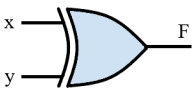
Έχει μια είσοδο και δίνει ως έξοδο το αντίστροφο αυτής. Επομένως F = X'



Είσοδος	Έξοδος
X	F
0	1
1	0

Λογική πύλη ΑΠΟΚΛΕΙΣΤΙΚΟ Ή (XOR)

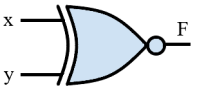
Η πύλη XOR, δίνει 1 όταν: για δύο εισόδους, αυτοί διαφέρουν μεταξύ τους ή διαφορετικά υπάρχει περιττός αριθμός μονάδων στις εισόδους, και για πάνω από δύο εισόδους, όταν το πλήθος των άσσων δεν είναι άρτιο (δηλ. αν είναι άρτιο τότε F = 0). Αλγεβρικά γράφεται: F = X'Y + XY'



Είσοδος		Έξοδος
X	Y	F
0	0	0
0	1	1
1	0	1
1	1	0

Λογική πύλη ΑΠΟΚΛΕΙΣΤΙΚΟ ΟΥΤΕ (XNOR)

Η πύλη XNOR, δίνει 1 όταν οι είσοδοι είναι ίσες μεταξύ τους. Αλγεβρικά γράφεται: X'Y' + XY



Είσοδος		Έξοδος
X	Y	F
0	0	1
0	1	0
1	0	0
1	1	1

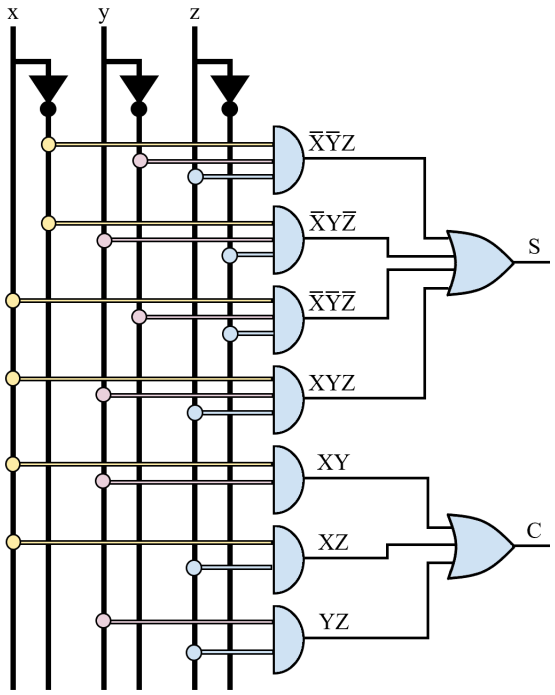
Συνδυαστικά κυκλώματα

- Πολυπλέκτες
- Αποκωδικοποιητές
- Συγκριτές
- Αθροιστές-Αφαιρέτες

Πλήρης αθροιστής

Ένα κύκλωμα που αναλαμβάνει να προσθέσει μια στήλη bit. Γι' αυτό, δέχεται 3 εισόδους A, B και C_{in} (κρατούμενο εισόδου) και παράγει 2 εξόδους (S_i και C_{out} (κρατούμενο εξόδου). Για να υλοποιήσουμε αθροιστή που προσθέτει αριθμούς n bit πρέπει να ενώσουμε n τέτοια κυκλώματα.

Είσοδοι			ελαχιστόρος	Έξοδοι	
A	B	C_{in}		S	C_{out}
0	0	0	0	0	0
0	0	1	1	1	0
0	1	0	2	1	0
0	1	1	3	0	1
1	0	0	4	1	0
1	0	1	5	0	1
1	1	0	6	0	1
1	1	1	7	1	1



$S = X'Y'C_{in} + X'YC_{in}' + XY'C'_{in} + XYC_{in}$ το οποίο γράφεται επίσης $S = \Sigma(1, 2, 4, 7)$ δηλαδή το άθροισμα των ελαχιστόρων 1, 2, 4, 7 όπου κάθε ελαχιστόρος είναι ένα γινόμενο.

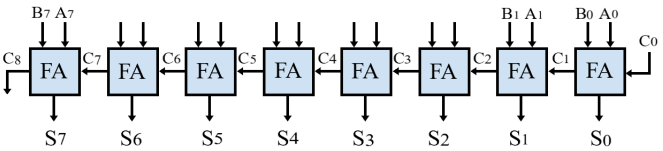
$C_{out} = X'YC_{in} + XY'C_{in} + XYC_{in}' + XYC_{in}$ το οποίο γράφεται επίσης $C_{out} = \Sigma(3, 5, 6, 7)$

ⓘ Ενναλακτικά επειδή S = 1 όταν το πλήθος των μονάδων είναι περριτό, θα μπορούσαμε να συνδέσουμε τα σήματα X, Y, Z σε μία XOR τριών εισόδων της οποίας έξοδος θα ήταν S, δηλαδή $S = X \text{ XOR } Y \text{ XOR } C_{in}$

ⓘ Με βάση το $C_{out} = X'YC_{in} + XY'C_{in} + XYC_{in}' + XYC_{in} \Rightarrow C_{out} = XY + XC_{in} + YC_{in}$

Αθροιστής n bit

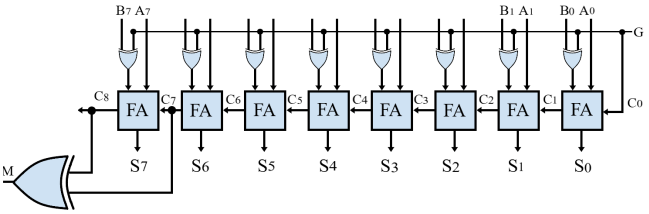
FA είναι το κύκλωμα πλήρους αθροιστή bit που υπάρχει παραπάνω.



π.χ. έστω 2 μη-προσημασμένοι αριθμοί A = 24 $(00011000)_2$ και B = 100 $(01100100)_2$
Η πρόσθεση με βάση το παραπάνω κύκλωμα αθροιστή θα γίνει ως εξής:
Στήλη 0: A0 = 0, B0 = 0, C0 = 0, άρα S0 = 0 και το C1 = 0.
Στήλη 1: A1 = 0, B1 = 0, C1 = 0, άρα S1 = 0 και το C2 = 0.
Στήλη 2: A2 = 0, B2 = 1, C2 = 0, άρα S2 = 1 και το C3 = 0.
Στήλη 3: A3 = 1, B3 = 0, C3 = 0, άρα S3 = 1 και το C4 = 0.
Στήλη 4: A4 = 1, B4 = 0, C4 = 0, άρα S4 = 1 και το C5 = 0.
Στήλη 5: A5 = 0, B5 = 1, C5 = 0, άρα S5 = 1 και το C6 = 0.
Στήλη 6: A6 = 0, B6 = 1, C6 = 0, άρα S6 = 1 και το C7 = 0.
Στήλη 7: A7 = 0, B7 = 0, C7 = 0, άρα S7 = 0 και το C8 = 0.
Άρα το άθροισμα είναι S7S6S5S4S3S2S1S0 = 01111100 = 124

Αθροιστής-αφαιρέτης με έλεγχο προσήμου

Τα σήματα A τροφοδοτούν τον αθροιστή στην κανονική τους μορφή. Τα B περνούν από μία XOR και ειδικότερα κάθε bit B_i λαμβάνει μέρος σε μία πράξη XOR μαζί με ένα σήμα G. Δηλαδή έχουμε B_0 XOR G, B_1 XOR G, ..., B_7 XOR G



Το σήμα G έχει 2 περιπτώσεις:

- G = 0 όπου θα εκτελείται η πράξη B_i XOR 0
    - Bi = 0, τότε η πράξη XOR θα δώσει 0 αφού 0 XOR 0 = 0
    - Bi = 1, η πράξη XOR θα δώσει 1 αφού 1 XOR 0 = 1

Παρατηρούμε πως αποτέλεσμα των XOR είναι ότι και το B_i, που σημαίνει πως οι αθροιστές FA θα εκτελούν την πράξη $A_i + B_i$ + C0 δηλαδή πρόσθεση και τελικά το κύκλωμα λειτουργεί ως αθροιστής.

- G = 1 όπου θα εκτελείται η πράξη B_i XOR 1
    - B_i = 0, τότε η πράξη XOR θα δώσει 1 αφού 0 XOR 1 = 1
    - B_i = 1, τότε η πράξη XOR θα δώσει 0 αφού 1 XOR 1 = 0

Παρατηρούμε πως αποτέλεσμα των XOR είναι πάντα το αντίστροφο του B_i δηλαδή B_i XOR 1 = (B_i)', που σημαίνει πως οι αθροιστές FA θα λάβουν ως είσοδο A + B' + C0 (δηλ. 1)

Έχουμε επομένως: A + B' + 1 = A + Σ_1(B) + 1 = A + Σ_2(B) = A-B δηλαδή πρόσθεση, τότε η ALU στέλνει ένα σήμα ίσο με 1 στο G και αυτό μετατρέπει τον αθροιστή σε αφαιρέτη.

ⓘ Για να ελέγξουμε την υπερχείλιση ελέγχουμε αν τα bit C_8 και C_7 είναι ίδια. Αν $C_8 = C_7$ τότε M = 0, ενώ αν $C_7 <> C_8$ τότε M=1 άρα αν το M = 1 τότε αυτό το σήμα δίνεται πίσω σε έναν καταχωρητή της CPU. Αυτό είναι ένα σήμα ειδοποίησης για Overflow, το οποίο εφόσον είναι 1 σημαίνει ότι θα δοθεί μήνυμα λάθους.

Λειτουργικό Σύστημα
Λογισμικό του υπολογιστή που είναι υπεύθυνο για τη διαχείριση και τον συντονισμό των εργασιών και την κατανομή των διαθέσιμων πόρων. Μεσολαβεί μεταξύ λογισμικού και υλικού, επιτρέποντας τις εφαρμογές ή τους χρήστες να αντιλαμβάνονται εμμέσως τον υπολογιστή.

Γραμμή εντολών
Διαβάζει γραμμές κειμένου από τον χρήστη και τα διερμηνεύει προς τον αντίστοιχο λειτουργικό σύστημα. Επιτρέπει περιήγηση στο σύστημα αρχείων του λειτουργικού συστήματος και διάβασμα και εκτέλεση εντολών/προγραμμάτων.

Εντολές ls, pwd, cd, touch

ls	εμφάνιση των αρχείων στον τρέχοντα κατάλογο
pwd	εμφάνιση του τρέχοντα καταλόγου
cd	αλλαγή καταλόγου • **cd .. :** αλλαγή στον πατρικό κατάλογο • **cd / :** αλλαγή στον κεντρικό κατάλογο • **cd ~ :** αλλαγή στον κατάλογο του χρήστη *(ισοδύναμη της "cd")* • **cd ../.. :** αλλαγή στον κατάλογο δύο επιπέδων επάνω
touch	δημιουργία κενού αρχείου

Εντολές mkdir, rmdir, cp, mv, rm, cat, echo και τελεστές >, >>

mkdir directory	δημιουργία καταλόγου
rmdir directory	διαγραφή καταλόγου
cp file1 file2	αντιγραφή αρχείου
mv file1 directory ή file2	μετακίνηση αρχείου (**mv** file1 directory) ή μετονομασία αρχείου (**mv** file1 file2)
rm file	διαγραφή αρχείου
cat file	εμφάνιση του περιεχομένου ενός αρχείου
echo "word"	εμφάνιση ενός μυνήματος
command **>** file	ανακατεύθυνση της εξόδου μια εντολής σε ένα αρχείο με τη δημιουργία νέου αρχείου, οπότε αν το αρχείο υπάρχει ήδη το προηγούμενο περιεχόμενο του σβήνεται
command **>>** file	ανακατεύθυνση της εξόδου μια εντολής σε ένα αρχείο, όπου αν αυτό ήδη υπήρχε το προηγούμενο περιεχόμενο του διατηρείται

Εντολές pico, wc, man, who και τελεστής |

pico file	διορθωτής κειμένου
wc file	μέτρηση λέξεων, γραμμών και χαρακτήρων ενός αρχείου
man command	εμφάνιση οδηγιών για άλλες εντολές
who	εμφάνιση των χρηστών συνδεδεμένων στο σύστημα
command **|** command	δυνατότητα μεταφοράς εξόδου μιας εντολής στην είσοδο μιας άλλης (*π.χ.* **cat** file **| wc**)

Εντολές less, head, tail, grep

less file	εμφάνιση του περιεχομένου ενός αρχείου ανά σελίδα
head file	εμφάνιση των 10 πρώτων γραμμών του αρχείου
tail command	εμφάνιση των 10 τελευταίων γραμμών του αρχείου
grep "word" file	αναζητά κάποια λέξη εντός ενός αρχείου

Εντολή chmod

Με την εντολή chmod μπορούμε να τροποποιήσουμε τα δικαιώματα πρόσβασης σε αρχεία και καταλόγους. Τα δικαιώματα πρόσβασης σε *αρχεία* αποτελούν τα:

1. *ανάγνωση (read)*
2. *εγγραφή (write)*
3. *εκτέλεση (execute)*

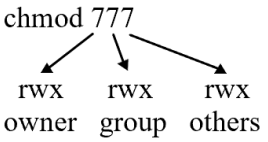
Το κάθε ένα αναφέρεται στα αντίστοιχα όσον αφορά τους *καταλόγους*:

1. *επιτρέπεται η εμφάνιση των αρχείων του καταλόγου*
2. *επιτρέπεται η δημιουργία, μετονομασία ή διαγραφή των αρχείων του καταλόγου, καθώς και η τροποποίηση των δικαιωμάτων στον κατάλογο*
3. *επιτρέπεται η πρόσβαση στον κατάλογο και στα περιεχόμενα του (αρχεία και άλλοι κατάλογοι)*

Η εντολή chmod μπορεί να χρησιμοποιηθεί με *δύο τρόπους*:

numeric mode: **chmod** xxx file1 ή directory

Κάθε x αποτελεί έναν αριθμό από το 0 έως το 7 με το πρώτο να αναφέρεται στον ιδιοκτήτη, το δεύτερο στο group και το τρίτο σε τρίτους.



7	rwx	111
6	rw-	110
5	r-x	101
4	r--	100
3	-wx	011
2	-w-	010
1	--x	001
0	---	000

Ο κάθε αριθμός μπορεί να "μεταφραστεί" με βάση την εξής λίστα:

π.χ. **chmod** 732 θα σήμαινε **rwx -wx -w-**

ugo mode: **chmod** user(s) + ή - permission(s)

Στους user(s) μπορεί να γραφτεί u (user), g (group), o (other) ή κάποιος συνδυασμός αυτών. Στα permission(s) μπορεί να γραφτεί r (read), w (write), x (execute) ή κάποιος συνδυασμός αυτών, ενώ το + χρησιμοποιείται για την προσθήκη permission(s) και το - για την αφαίρεση τους.

π.χ. έχουμε ένα file με permissions **-r--r-xr--** και εκτελέσουμε: **chmod ug+w** το file θα έχει τα εξής permissions **-rw-rwxr--**

ⓘ Το πρώτο σύμβολο αναφέρεται στο αν είναι αρχείο ή κατάλογος. Για παράδειγμα για permissions rw-rwxr-- ενός αρχείου θα είχαμε rw-rwxr-- ενώ ενός κατάλογο drw-rwxr--

Filesystem Hierarchy Standard

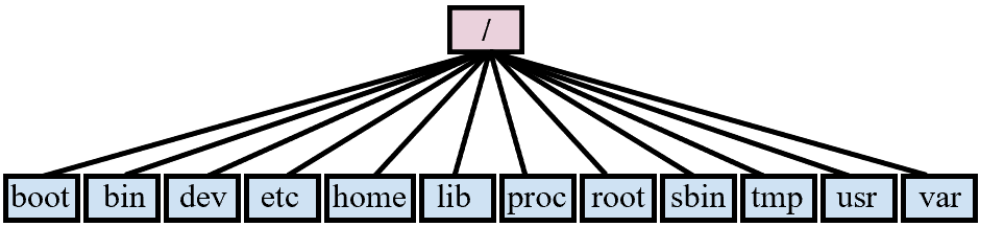
Το Πρότυπο Ιεραρχίας Συστήματος Αρχείων ορίζει τους βασικούς καταλόγους και τα περιεχόμενα τους στα περισσότερα συστήματα Linux.

	shareable	**unshareable**
static	`/usr` `/opt`	`/etc` `/boot`
variable	`/var/mail` `/var/spool/news`	`/var/run` `/var/lock`

Στο FHS δεν υπάρχει οδηγός C:, D:, … και όλοι οι κατάλογοι βρίσκονται κάτω από τον / ο οποίος είναι ο κατάλογος "ρίζα". Είναι επίσης δυνατό να έχουμε πολλαπλά partitions του σκληρού δίσκου και να χρησιμοποιούμε πολλά συστήματα αρχείων.

Κατάλογος /

Υπάρχει μόνον ένας κατάλογος / και αποτελεί την ρίζα του δέντρου ενός συστήματος αρχείων, που σημαίνει πως όλα τα μονοπάτια ξεκινούν από εδώ.

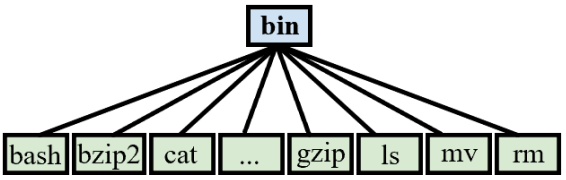


boot

Περιέχει τον πηρήνα του Linux (kernel) και τις ρυθμίσεις του bootloader. Χωρίς το boot δεν μπορεί να ξεκινήσει το ΛΣ.

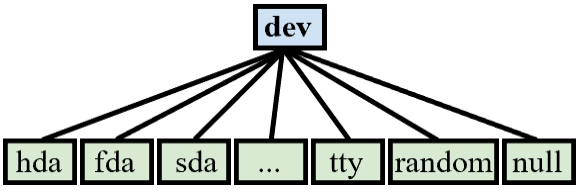
bin

Περιέχει βασικές εντολές για την πλοήγηση στο σύστημα αρχείων και τη διαχείριση τους και απαιτείται για την εκκίνηση του συστήματος.



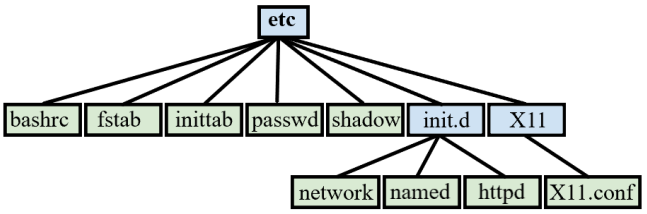
dev

Οι συσκευές (σκληρός δίσκος, πληκτρολόγιο) είναι επίσης αρχεία όπως τα πάντα στο Linux. Το dev directory περιέχει όλα τα αρχεία των συσκευών & έτσι υπάρχει άμεση επικοινώνια με τον οδηγό των συσκευών.



etc

Είναι ο κατάλογος με τις ρυθμίσεις του συστήματος (όπως το Windows registry), όπου όλα τα αρχεία των ρυθμίσεων είναι αρχεία κειμένου και έτσι μπορούν να ανοιχθούν και επεξεργασθούν manually.

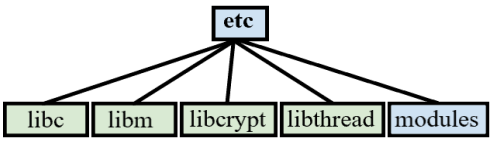


home

Ο προσωπικός κατάλογος του χρήστη συμβολίζεται με τον χαρακτήρα ~. Κάθε χρήστης έχει ένα κατάλογο στο home (π.χ. /home/andreas, /home/despoina). Όλα τα αρχεία των χρηστών αποθηκεύονται εδώ.

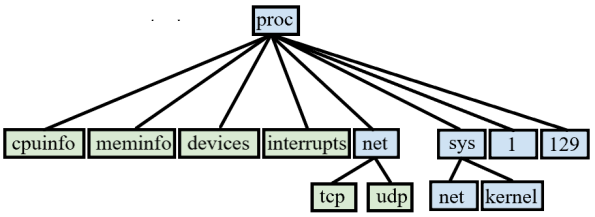
lib

Όλες οι βασικές βιβλιοθήκες βρίσκονται εδώ και συνδέονται δυναμικά (dynamically linked libraries). Ο κατάλογος lib απαιτείται για την εκκίνηση του συστήματος.



proc

Αποτελεί ειδικό, εικονικό κατάλογο που περιέχει πληροφορίες της κατάστασης του συστήματος, δηλαδή προσφέρει διεπαφή με τον πυρήνα (kernel's interface) και γι' αυτό μπορεί να θεωρηθεί ως κέντρο ελέγχου και πληροφοριών του πυρήνα.

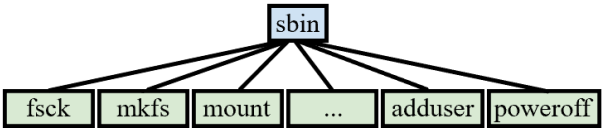


root

Αποτελεί τον προσωπικό κατάλογο του διαχειριστή (ο οποίος ονομάζεται root). Είναι επίσης διαφορετικός από τον κατάλογο / (που επίσης λέγεται ρίζα).

sbin

Περιέχει εντολές που ρυθμίζουν το σύστημα όπως μορφοποίηση σκληρού δίσκου και διαχείριση υλικού. Μόνο ο "root" μπορεί να εκτελέσει πολλά από αυτά τα προγράμματα.

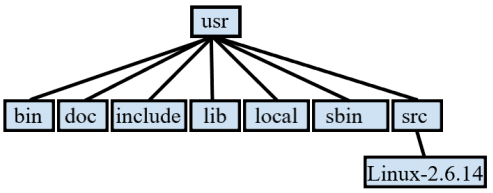


tmp

Αποτελεί προσωρινό κατάλογο, και περιέχει π.χ. προσωρινά δεδομένα που δημιουργούν οι εφαρμογές, αρχεία που μπορούν να αποθηκεύσουν προσωρινά οι χρήστες. Τα περιεχόμενα του διαγράφονται συχνά.

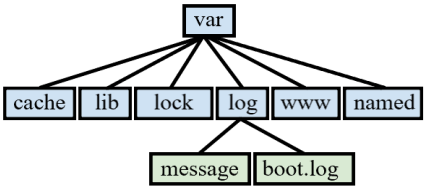
usr

Έχει δευτερεύουσα ιεραρχία και περιέχει χρήσιμα προγράμματα όπως μεταφραστές, εργαλεία κτλ. Δεν είναι απαραίτητος για την εκκίνηση του συστήματος.



var

Ο κατάλογος με τα διάφορα. Είναι δυναμικά αρχεία και ο χρήστης δεν μπορεί να τα αλλάξει.



Διεργασίες

Μια διεργασία είναι μία εκτέλεση ενός προγράμματος και αποτελείται από οδηγίες εκτέλεσης, δεδομένα που διαβάζονται από αρχεία, άλλα προγράμματα, ή αποτελούν εισόδους από έναν χρήστη του συστήματος.

Τύποι διεργασιών

Διαλογικές

- αρχικοποιούνται και ελέγχονται από μια σύνοδο τερματικού
- προσκήνιο (bg), παρασκήνιο (fg)
- *έλεγχος διεργασιών*

Αυτόματες

- δεν συνδέονται με ένα τερματικό
- εκτελούνται συγκεκριμένη μέρα και ώρα (cron, at)
- όταν ο φόρτος του συστήματος είναι αρκετά χαμηλός (batch)

Δαίμονες

- διεργασίες διακομιστή που εκτελούνται συνεχώς
- δαίμονας της σύνδεσης ssh: sshd, δαίμονας Web server: httpd

Έλεγχος διεργασιών

command	εκτέλεση εντολής στο προσκήνιο
command &	εκτέλεση εντολής στο παρασκήνιο
jobs	εμφάνιση εργασιών που εκτελούνται στο παρασκήνιο
ctrl + c	τερματισμός διεργασίας που εκτελείται στο προσκήνιο
ctrl + z	τερματισμός διεργασίας που εκτελείται στο παρασκήνιο
%n	αριθμός εργασίας (job ID) στο παρασκήνιο
bg	ενεργοποίηση προγράμματος που έχει ανασταλεί στο παρασκήνιο
fg	επαναφορά εργασίας στο προσκήνιο
kill	τερματισμός διεργασίας ή εργασίας

Υποδοχές και θύρες

Υποδοχή (socket) είναι η διεπαφή (interface) ανάμεσα σε μία εφαρμογή και το TCP/IP. Παρέχεται από το λειτουργικό σύστημα και χαρακτηρίζεται μοναδικά από:

- Πρωτόκολλο
- Τοπικό IP
- Τοπική Θύρα (port)
- Απομακρυσμένο IP
- Απομακρυσμένη Θύρα (port)

Δικτυακές εντολές στο Linux

ifconfig	Εμφανίζει τις τρέχουσες δικτυακές ρυθμίσεις για κάθε κάρτα δικτύου στο σύστημα.
netstat	Εμφανίζει τις ενεργές συνδέσεις στον Η/Υ και σχετικά στατιστικά.
nslookup	Κάνει «εύρεση στον εξυπηρετητή ονομάτων» (name server lookup) - υπηρεσία DNS (Domain Name Server)
ping	Με την ping ελέγχουμε τη σωστή λειτουργία μιας δικτυακής συσκευής, χρησιμοποιώντας το Internet Control Message Protocol (ICMP) για να στείλει πακέτα ECHO_REQUEST με στόχο απάντηση ECHO_REPLY.
traceroute	Η εντολή traceroute καταγράφει τη διαδρομή που διανύει ένα πακέτο στο Internet. Αναφέρει τους ενδιάμεσους δρομολογητές καθώς και το χρόνο που ξοδεύει το πακέτο για να τους διασχίσει.