ΕΚΔΟΣΕΙΣ ΚΛΕΙΔΑΡΙΘΜΟΣ

ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

ΣΥΣΤΗΜΑΤΑ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΟΣ ΙΙ

Ι. Κ. ΚΑΒΟΥΡΑ
Dip. Comp. Sci., M.Sc. (Comp. Sci.), Ph. D. (Comp. Sci.)

ΚΛΕΙΔΑΡΙΘΜΟΣ

ΕΚΔΟΣΗ 7η
ΑΘΗΝΑ 2009

Περιεχόμενα

ΚΕΦΑΛΑΙΟ 7

ΔΙΑΧΕΙΡΙΣΗ ΜΝΗΜΗΣ (MEMORY MANAGEMENT)

ΜΕΡΟΣ ΠΡΩΤΟ: ΥΛΙΣΜΙΚΟ

7.0. Ιστορική αναδρομή

Οι μνήμες των *πρώτων υπολογιστών* είχαν μικρή χωρητικότητα και στοιχίζανε πολύ ακριβά. Οι προγραμματιστές των υπολογιστών αυτών ξόδευαν πολύ χρόνο προσπαθώντας να *επικαλύψουν (overlay)* τα προγράμματά τους στη μνήμη. Οι προγραμματιστές χώριζαν τα προγράμματά τους σε τμήματα που χωρούσαν στη διαθέσιμη μνήμη και που ήταν γνωστά ως *επικαλύψεις (overlays)*. Για να εκτελεστεί κάποιο πρόγραμμα, το πρώτο "*τμήμα επικάλυψης*" *(overlay segment)* φορτωνόταν στη μνήμη και εκτελούνταν για κάποιο χρονικό διάστημα. Όταν τελείωνε η εκτέλεσή του, διάβαζε και φόρτωνε στη μνήμη το δεύτερο τμήμα επικάλυψης και του μετέφερε τον έλεγχο κ.ο.κ., ώσπου να εκτελεστεί ολόκληρο το πρόγραμμα. Οι προγραμματιστές έπρεπε να αποφασίζουν οι ίδιοι για το πώς θα διαιρούσαν τα προγράμματά τους σε τμήματα, για το πού θα αποθηκεύονταν τα τμήματα αυτά στη βοηθητική μνήμη και ακόμη να φροντίζουν για τη μεταφορά τους από τη βοηθητική στην κύρια μνήμη (βλ. και Κεφ. 0.1).

Παρόλο που οι *πρώτοι σύγχρονοι μικροϋπολογιστές* διέθεταν σχετικά μεγάλη κεντρική μνήμη, μόνο ένα μέρος της μνήμης αυτής –της τάξης των 640 Kbytes– μπορούσε να προσπελαστεί άμεσα, λόγω του περιορισμένου μήκους λέξης των υπολογιστών αυτών. Έτσι τα λειτουργικά συστήματα των υπολογιστών αυτών επέτρεπαν –μέσω μιας ειδικής εντολής (που σε πολλά συστήματα, π.χ. συστήματα BASIC, ήταν γνωστή ως `CHAIN` (αλυσίδα)) ή μέσω των γλωσσών υψηλού επιπέδου– τη διαίρεση των προγραμμάτων σε **τμήματα (segments)**, δηλαδή σε σύνολα υποπρογραμμάτων, που μπορούσαν να φορτωθούν στη μνήμη ανεξάρτητα από το κύριο πρόγραμμα. Για παράδειγμα, τα υποπρογράμματα που δίνουν αρχικές τιμές

στις μεταβλητές ή στις δομές δεδομένων ενός προγράμματος, τα υποπρογράμματα που εμφανίζουν/εκτυπώνουν τα τελικά αποτελέσματα του προγράμματος και τα υποπρογράμματα που καλούνται (χρησιμοποιούνται) πολύ σπάνια κατά την εκτέλεση του προγράμματος μπορούν να δηλωθούν ως (ανεξάρτητα, ελπίζουμε) τμήματα.

Η δήλωση ενός υποπρογράμματος ως "τμήματος"[1] δεν επηρεάζει τη σημασιολογία του προγράμματος. Επηρεάζει μόνο το χρόνο εκτέλεσής του και τις απαιτήσεις του σε μνήμη. Το "τμήμα-υποπρόγραμμα" και όλα τα φωλιασμένα του υποπρογράμματα (εκτός αυτών που είναι επίσης τμήματα) αποτελούν μία ομάδα, που ονομάζεται *τμήμα κώδικα (code segment)*. Ένα πρόγραμμα και τα υποπρογράμματά του μεταγλωττίζονται σε ένα τμήμα κώδικα, εκτός αν μερικά από τα υποπρογράμματά του έχουν δηλωθεί ως τμήματα.

Ένα τμήμα κώδικα *διαμένει στο δίσκο (disc resident),* μέχρις ότου να χρησιμοποιηθεί, και ο χώρος που καταλαμβάνει στη μνήμη μπορεί να επικαλυφθεί όταν τελειώσει η εκτέλεσή του. Έτσι η χρήση της διαθέσιμης μνήμης μπορεί να βελτιωθεί με το να δηλώσουμε τα υποπρογράμματα που χρησιμοποιούνται μόνο μία φορά ή σπάνια ως τμήματα. Η απροσδόκητη όμως συμπεριφορά των προγραμμάτων (για παράδειγμα, οι κλήσεις των υποπρογραμμάτων που εξαρτώνται από τα δεδομένα και η εκτέλεση των αναδρομικών υποπρογραμμάτων) καθιστά την εφαρμογή της τεχνικής αυτής αδύνατη σε πολλές περιπτώσεις.

Η εμφάνιση των *συστημάτων πολυπρογραμματισμού* στις αρχές της δεκαετίας του '60 σήμαινε ότι η μνήμη ενός υπολογιστή έπρεπε να μπορούσε να (κατα)μεριστεί συγχρόνως σε περισσότερα από ένα προγράμματα. Αν όλα τα προγράμματα μπορούσαν να φορτωθούν ταυτόχρονα, δε θα υπήρχε κανένα πρόβλημα. Στην πράξη όμως η χωρητικότητα της κύριας μνήμης είναι, ακόμα και στα σύγχρονα μεγάλα συστήματα, πεπερασμένη και τα προγράμματα των χρηστών δεν είναι πάντοτε μικρά.

---

[1] Στις UCSD, TURBO και THINK Pascal αυτό γίνεται με το να γραφεί η ειδική λέξη **`segment`** πριν από τις ειδικές λέξεις **`process`**, **`procedure`** ή **`function`**. Σ' ένα πρόγραμμα της UCSD Pascal μπορούν να δηλωθούν μέχρι `255` τμήματα. Τα σώματα των τμημάτων αυτών πρέπει να δηλωθούν πριν από τα σώματα των κοινών υποπρογραμμάτων που ανήκουν στο ίδιο τμήμα κώδικα. Έτσι, αν ένα τμήμα-υποπρόγραμμα χρειάζεται ένα υποπρόγραμμα που δεν είναι τμήμα, το τελευταίο πρέπει να δηλωθεί ως **`forward`**. Στη συμβολική γλώσσα των υπολογιστών της σειράς 86, τα τμήματα ορίζονται με τις ψευδοεντολές `SEGMENT` και `ENDS` (βλ. Τόμο Ι, Κεφ. 3.12.8.1).

Μία από τις πρώτες τεχνικές που χρησιμοποιούσαν τα συστήματα πολυπρογραμματισμού ήταν η **ανταλλαγή των προγραμμάτων (program swapping)** – γνωστή και ως **ανταλλαγή των εργασιών (job swapping)** ή ως *επαναφορά-εκτόπιση (roll-in-roll-out*)– μεταξύ της βοηθητικής και της κύριας μνήμης. Όταν ερχόταν η σειρά μιας (δι)εργασίας[2] να εκτελεστεί, τότε (το πρόγραμμα ή τα προγράμματά της) φορτωνόταν στη μνήμη και ενεργοποιούνταν. Όταν η (δι)εργασία εξαντλούσε το τεμάχιο (κβάντο) χρόνου που της είχε καταχωρηθεί ή όταν απαιτούσε είσοδο/έξοδο που την ανάγκαζε να περιμένει, τότε το πρόγραμμά της αποθηκευόταν πίσω στη βοηθητική μνήμη και φορτωνόταν το πρόγραμμα της επόμενης (δι)εργασίας κ.ο.κ. Αν υπήρχε αρκετή μνήμη, τότε μια (δι)εργασία μπορούσε να εκτελεστεί ταυτόχρονα με την ανταλλαγή των προγραμμάτων των άλλων (δι)εργασιών μεταξύ βοηθητικής και κύριας μνήμης.

Η τεχνική ανταλλαγής των προγραμμάτων έχει δύο *μειονεκτήματα*. Είναι, πρώτον, *χρονοβόρα*, αν και ο χρόνος επιβάρυνσης μπορεί να ελαττωθεί με τη χρήση επανεισαγόμενων προγραμμάτων και με τον περιορισμό των χρηστών σε μία μόνο γλώσσα υψηλού επιπέδου [όπως στα πρώτα "συστήματα (κατα)μερισμού χρόνου της BASIC" (BASIC time-sharing systems)].

Δεύτερον, τα προγράμματα εκτελούν συνήθως ένα υποσύνολο των υποπρογραμμάτων τους: τα υπόλοιπα υποπρογράμματά τους δεν είναι απαραίτητο να βρίσκονται διαρκώς στη μνήμη. Έτσι η ανταλλαγή ολόκληρων των προγραμμάτων μπορεί να είναι άσκοπη και να καταναλώνει περισσότερο χώρο της διαθέσιμης μνήμης απ' όσο πραγματικά χρειάζονται τα προγράμματα αυτά.

7.1. Σκοποί

Οι κύριοι *σκοποί* διαχείρισης της μνήμης ενός συστήματος πολυπρογραμματισμού είναι οι εξής:

(i) *Δυναμική μεταθεσιμότητα (Dynamic relocatability),* που επιτρέπει τη γρήγορη τοποθέτηση των προγραμμάτων σε διαφορετικές θέσεις της μνήμης έτσι, ώστε να βελτιστοποιείται η χρήση της διαθέσιμης μνήμης.

---

[2] Στα πρώτα συστήματα πολυπρογραμματισμού χρησιμοποιούνταν ο όρος *"εργασία"* ή *"πρόγραμμα"* αντί του όρου *"διεργασία"*.

(ii) *Προστασία (Protection),* που εξασφαλίζει την ακεραιότητα των διεργασιών με το να τους απαγορεύεται η δυνατότητα ν' αλλάξουν τα περιεχόμενα των θέσεων της μνήμης που δεν ανήκουν στο χώρο των διευθύνσεών τους.

(iii) *Εικονική* ή *λογική μνήμη (Virtual* ή *logical memory)* μεγάλης χωρητικότητας, που απαλλάσσει τους χρήστες από την περιορισμένη χωρητικότητα της πραγματικής (φυσικής) μνήμης.

(iv) *Λογική οργάνωση του προγράμματος σε διάφορες δομοενότητες (modules)*, που μπορούν να μεταφραστούν, να προστατευτούν και να (κατα)μεριστούν ανεξάρτητα η μία από την άλλη.

Οι σκοποί αυτοί μπορούν να επιτευχθούν με τη βοήθεια του υλισμικού, όπως θα περιγράψουμε σ' αυτό το κεφάλαιο, και του σχετικού λογισμικού, που θα περιγράψουμε στο επόμενο κεφάλαιο.

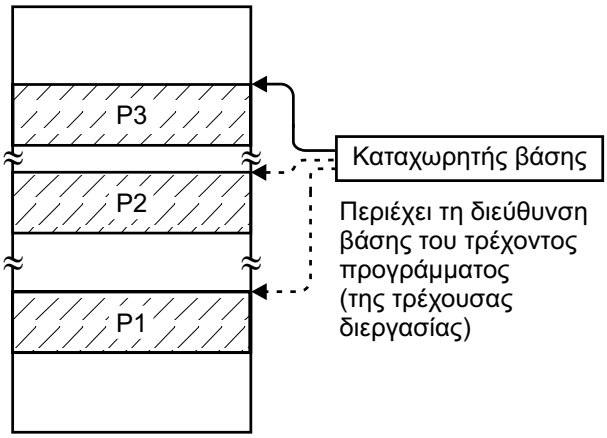
7.2. Αναφορά διεύθυνσης μέσω Καταχωρητή Βάσης (Base Register Addressing)

Η μέθοδος αναφοράς διεύθυνσης μέσω καταχωρητή βάσης είναι παρόμοια με τη σχετική αναφορά και την αναφορά διεύθυνσης μέσω καταχωρητή δείκτη, και έχει ήδη περιγραφεί στο Κεφάλαιο 3.5 του πρώτου τόμου. Η ΚΜΕ των υπολογιστών που χρησιμοποιούν τη μέθοδο αυτή περιέχει έναν καταχωρητή που ονομάζεται **καταχωρητής βάσης (base register)** ή **καταχωρητής μετάθεσης (relocation register)**. Στους υπολογιστές αυτούς η τελική διεύθυνση μιας εντολής αναφοράς στη μνήμη είναι ίση με το άθροισμα των περιεχομένων του καταχωρητή βάσης και της τελικής διεύθυνσης που προσδιορίζεται από την εντολή.

Η *λειτουργία του υλισμικού* των υπολογιστών αυτών διαφέρει από αυτή που περιγράψαμε στα Κεφάλαια 4.3 και 6.4.2 του πρώτου τόμου μόνο στο ότι οι εντολές εκχώρησης περιεχομένων καταχωρητών στον καταχωρητή διευθύνσεων της μνήμης, `MAR`, περιέχουν στο δεξί τους μέλος την πρόσθεση του όρου `[Καταχωρητής_Βάσης]`.

Το *πλεονέκτημα* της μεθόδου αυτής είναι ότι επιτρέπει τη **δυναμική μετάθεση (dynamic relocation)** των προγραμμάτων. Σ' ένα μεγάλο σύστημα υπολογιστών συνυπάρχουν συνήθως ταυτόχρονα διάφορα προγράμματα στην κύρια μνήμη και είναι επιθυμητό να μπορούν να μετακινούνται αυτά εύκολα σ' οποιαδήποτε θέση

της διαθέσιμης μνήμης. Όλα τα προγράμματα μεταφράζονται σχετικά ως προς τη θέση 0 της μνήμης. Ένα ολόκληρο πρόγραμμα μπορεί να **φορτωθεί δυναμικά (dynamic loading)** σ' οποιαδήποτε διεύθυνση της μνήμης, με το να φορτωθούν οι εντολές και τα δεδομένα του στις θέσεις που αρχίζουν από τη διεύθυνση αυτή *(διεύθυνση βάσης)* και με το να αποθηκευτεί η διεύθυνση αυτή στον καταχωρητή βάσης, πριν εκτελεστεί (διανεμηθεί στην ΚΜΕ) το πρόγραμμα αυτό. Ο *(σχετικός)* **δυναμικός φορτωτής** δε χρειάζεται ν' αλλάξει τα πεδία διεύθυνσης των εντολών αναφοράς στη μνήμη. Σε πολλούς υπολογιστές η εντολή που φορτώνει τον καταχωρητή βάσης με μία διεύθυνση είναι προνομιούχα.



Σχήμα 7.1. Αναφορά διεύθυνσης μέσω καταχωρητή βάσης.

Οι προσωπικοί μικροϋπολογιστές της IBM, που βασίζονταν στους επεξεργαστές 8086 ή 80286, διέθεταν τέσσερις καταχωρητές βάσης που χρησιμοποιούνταν για τα τμήματα κώδικα, στοίβας, δεδομένων και "επιπλέον" δεδομένων όπως αναφέραμε στα Κεφάλαια 2.6.1 και 3.12.1 του πρώτου τόμου. Το *μειονέκτημα* αναφοράς διεύθυνσης μέσω καταχωρητών βάσης είναι η αδυναμία προστασίας των προγραμμάτων. Ένα πρόγραμμα μπορεί να προσπελάσει διαθέσιμη μνήμη που δεν του ανήκει και να καταστρέψει τα άλλα προγράμματα τα οποία διαμένουν ταυτόχρονα μ' αυτό στη μνήμη.

Ένας τρόπος που χρησιμοποιήθηκε στους υπολογιστές της σειράς 360 της IBM για να επιτευχθεί η προστασία της μνήμης ήταν ο εξής: Η μνήμη των υπολογιστών αυτών θεωρούνταν διαιρεμένη σε περιοχές μήκους 2Κ δυφιοσυλλαβών, καθεμία

από τις οποίες είχε έναν *κώδικα προστασίας (protection code)* τεσσάρων δυφίων. Η λέξη κατάστασης του προγράμματος (βλ. Τόμο Ι, Κεφ. 6.1) περιείχε ένα τετραψήφιο *κλειδί*. Το υλισμικό των υπολογιστών 360 παγίδευε κάθε προσπάθεια προσπέλασης μιας περιοχής της μνήμης, της οποίας ο κώδικας προστασίας διέφερε από το κλειδί που περιείχε η λέξη κατάστασης του προγράμματος της τρέχουσας διεργασίας. Εφόσον οι εντολές αλλαγής του κώδικα προστασίας και του κλειδιού ήταν προνομιούχες, μόνο το λειτουργικό σύστημα μπορούσε να τις χρησιμοποιήσει, κι έτσι οι διεργασίες των χρηστών και το λειτουργικό σύστημα μπορούσαν να προστατευτούν.

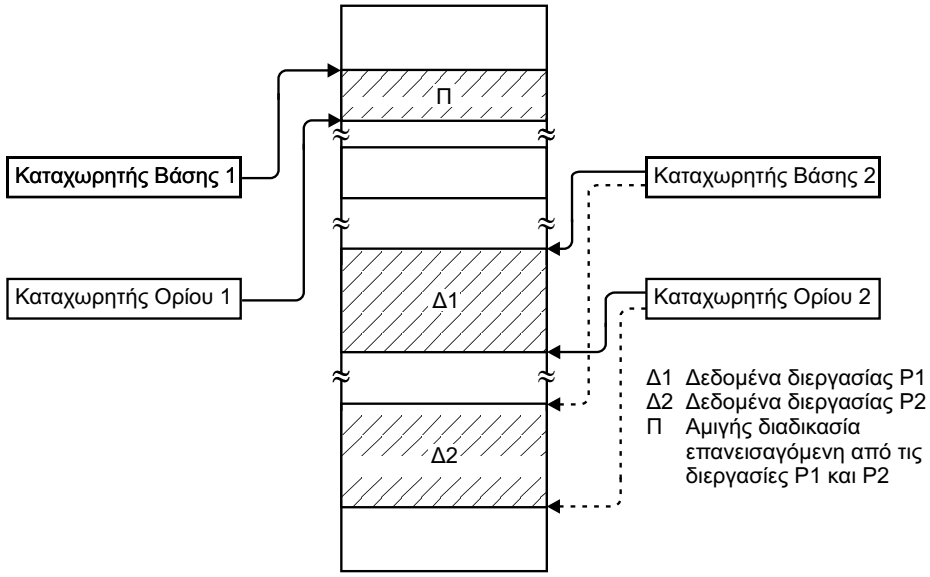
7.3. Καταχωρητές Βάσης (Base Registers) και Καταχωρητές Ορίου (Limit Registers)

Η ΚΜΕ κάποιων υπολογιστών περιέχει δύο καταχωρητές: έναν **καταχωρητή βάσης (base register)**, όπως περιγράψαμε παραπάνω, κι έναν **καταχωρητή ορίου (limit register)**, που περιέχει τη μέγιστη διεύθυνση του τμήματος στη μνήμη ή έναν **καταχωρητή μήκους (length register)**, που περιέχει το μήκος του τμήματος. Στους υπολογιστές αυτούς η τελική διεύθυνση των εντολών αναφοράς στη μνήμη είναι επίσης ίση με το άθροισμα των περιεχομένων του καταχωρητή βάσης και της τελικής αναφοράς που προσδιορίζουν οι εντολές.

Η *δυναμική μετάθεση (dynamic relocation)* των προγραμμάτων επιτυγχάνεται επίσης με το να φορτωθούν οι εντολές και τα δεδομένα τους στις διευθύνσεις, που αρχίζουν από τη διεύθυνση βάσης, και με το να γίνει αποταμίευση της διεύθυνσης αυτής στον καταχωρητή βάσης του υπολογιστή. Η *προστασία* της μνήμης των διεργασιών επιτυγχάνεται, είτε αν συγκρίνουμε την τελική διεύθυνση των εντολών αναφοράς στη μνήμη με τα περιεχόμενα του καταχωρητή ορίου είτε αν συγκρίνουμε τα περιεχόμενα του πεδίου διεύθυνσης των εντολών αυτών με τα περιεχόμενα του καταχωρητή μήκους του υπολογιστή. Έτσι η *λειτουργία του υλισμικού* των υπολογιστών αυτών διαφέρει από αυτήν που περιγράψαμε στα Κεφάλαια 6.4.2 και 4.3 του πρώτου τόμου στο ότι περιλαμβάνει (μικρο)εντολές της μορφής:

```
if (τελική_διεύθυνση < 0 || [Καταχωρητής_Βάσης] +
                    τελική_διεύθυνση) > [Καταχωρητής_Ορίου])
```

```
                // ή (πεδίο_διεύθυνσης > [Καταχωρητής_Μήκους])
    στείλε στο υλισμικό ένα σήμα
                        // το αντίστοιχο της εντολής signal
    "παγίδας_παραβίασης_της_μνήμης";
                                    // memory_violation_trap
else MAR = [Καταχωρητής_Βάσης] + τελική_διεύθυνση;
```



Σχήμα 7.2. Αναφορά διεύθυνσης μέσω καταχωρητών Βάσης και Ορίου. [Περιεχόμενα των καταχωρητών Βάσης και Ορίου κατά την εκτέλεση της διεργασίας P1 (διεργασίας P2, διακεκομμένα)].

Το *μειονέκτημα* της μεθόδου αυτής είναι ότι ο **χώρος** των **εικονικών** ή **λογικών** ή **σχετικών διευθύνσεων (virtual** ή **logical** ή **relative address space)** του προγράμματος είναι αναγκαστικά μικρότερος από ή ίσος με τη χωρητικότητα της διαθέσιμης μνήμης, δηλαδή του **χώρου** των **πραγματικών** ή **φυσικών** ή **απόλυτων διευθύνσεων (real** ή **physical** ή **absolute address space)**. Σε αντίθεση, αν ένας υπολογιστής διαθέτει δύο ζεύγη καταχωρητών βάσης και ορίου (ή μήκους), τότε μπορούν εύκολα να υλοποιηθούν επανεισαγόμενα προγράμματα (αμιγείς διαδικασίες), όπως δείχνει το Σχήμα 7.2.

Ο μεταγλωττιστής ή ο συμβολομεταφραστής ξεχωρίζει τις (αμετάβλητες) εντολές του προγράμματος από τα (μεταβλητά) δεδομένα του. Το τμήμα εντολών ή *τμήμα κώδικα (code segment)* του προγράμματος μπορεί έτσι να χρησιμοποιηθεί από διάφορες διεργασίες, εφόσον κάθε διεργασία έχει το δικό της *τμήμα δεδομένων (data segment)*. Οι προσωπικοί υπολογιστές που βασίζονταν στους επεξεργαστές 80286 διέθεταν τέσσερις καταχωρητές ορίου, ενώ αυτοί που βασίζονταν στους επεξεργαστές 80386 και πάνω είχαν έξι καταχωρητές (επιλογέων) τμημάτων και επιπλέον παρείχαν έναν ιεραρχικό μηχανισμό προστασίας όπως θα περιγράψουμε στα Κεφάλαια 7.9.1 και 11.3.3.

7.4. Σελιδοποίηση (Paging)

Η κύρια μνήμη (ο χώρος των φυσικών διευθύνσεων) των υπολογιστών που χρησιμοποιούν σελιδοποίηση αποτελείται από ισομεγέθη τμήματα ορισμένου μήκους –γνωστού ως *μεγέθους σελίδας (page size)*– που ονομάζονται **πλαίσια σελίδων (page frames)** και κάθε πρόγραμμα (ο χώρος των εικονικών διευθύνσεων) θεωρείται ότι είναι διαιρεμένο(ς) σε **σελίδες (pages)** του ιδίου μεγέθους. Τα πλαίσια σελίδων μπορούν να κατανεμηθούν (καταχωρηθούν) στις διεργασίες που βρίσκονται στο σύστημα έτσι, ώστε σε κάποια χρονική στιγμή μια διεργασία να έχει μερικές από τις σελίδες της –τις *ενεργές* της *σελίδες (active pages)*– στην κύρια μνήμη και τις υπόλοιπες –τις *μη ενεργές* της *σελίδες (inactive pages)*– στη βοηθητική μνήμη.

Για να υλοποιηθεί η σελιδοποίηση, χρειάζεται ένας *πίνακας σελίδων (page table)* για κάθε διεργασία. Ο πίνακας αυτός (ή η βάση του) αποθηκεύεται στο περιβάλλον της διεργασίας όπως περιγράψαμε στο Κεφάλαιο 5.5. Κάθε στοιχείο του πίνακα αυτού ονομάζεται **περιγραφητής** της αντίστοιχης **σελίδας (page descriptor)** του προγράμματος και περιέχει τα εξής:

(i) ένα **δυφίο παρουσίας (present bit),** που καθορίζει αν η σελίδα βρίσκεται στην κύρια μνήμη ή όχι

(ii) τη διεύθυνση (ή τον αριθμό του πλαισίου) της σελίδας στη βοηθητική ή/και στην κύρια μνήμη αντίστοιχα, και πιθανώς

(iii) **δυφία** που καθορίζουν τον τρόπο **προστασίας (protection bits)** της σελίδας[3] και ίσως κι άλλες λεπτομέρειες.

Επιπλέον, το υλισμικό ενός συστήματος που χρησιμοποιεί σελιδοποίηση πρέπει να διαθέτει έναν ειδικό καταχωρητή, που ονομάζεται **καταχωρητής πίνακα σελίδων (page table register)** και περιέχει τη διεύθυνση βάσης του πίνακα των σελίδων της *τρέχουσας διεργασίας* πριν αυτή διανεμηθεί. Για να προσδιοριστεί η τελική (φυσική) διεύθυνση της μνήμης στην οποία αναφέρεται μία εντολή (αναφοράς στη μνήμη), το *υλισμικό* διερμηνεύει τις (εικονικές) διευθύνσεις των εντολών του προγράμματος με τέτοιο τρόπο, ώστε τα πλέον σημαντικά δυφία του πεδίου διεύθυνσης της εντολής να αποτελούν τον *αριθμό της σελίδας (page number)* του προγράμματος, ενώ τα λιγότερο σημαντικά δυφία να αποτελούν τον *αριθμό της λέξης (word number)* μέσα στη σελίδα αυτή.

Έτσι η τελική διεύθυνση μιας εντολής αναφοράς στη μνήμη

A

διερμηνεύεται από το υλισμικό ως

αριθμός_σελίδας	αριθμός_λέξης

όπου ο αριθμός_σελίδας αποτελεί τα πλέον σημαντικά δυφία της τελικής διεύθυνσης και ο αριθμός_λέξης τα λιγότερο σημαντικά δυφία αυτής[4].

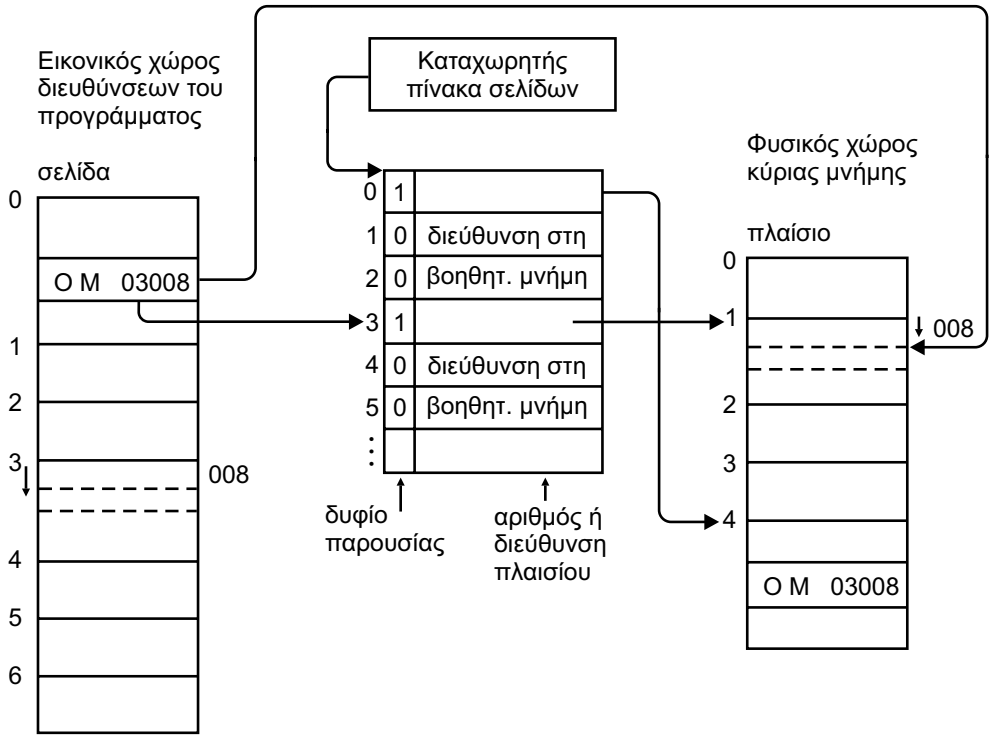
Εννοείται ότι το μήκος του πεδίου διεύθυνσης των εντολών αναφοράς στη μνήμη είναι αρκετό για να μπορούν οι εντολές αυτές να αναφερθούν σε *ολόκληρη την εικονική μνήμη* (που είναι σχεδόν πάντα πολύ μεγαλύτερη από τη φυσική μνήμη). Ο αριθμός της σελίδας χρησιμοποιείται από το υλισμικό, για να επιλέξει το αντίστοιχο στοιχείο του πίνακα των σελίδων της διεργασίας. Αν το δυφίο παρουσίας υποδεικνύει ότι η σελίδα βρίσκεται στην κύρια μνήμη (είναι

---

[3] Σε μερικά συστήματα τα δυφία αυτά μπορεί απλώς να καθορίζουν αν η *σελίδα* αποτελείται από εντολές *(code page)* ή από *δεδομένα (data page)*, ενώ σ' άλλα συστήματα αν η *σελίδα* επιτρέπεται *να προσπελαστεί για ανάγνωση μόνο (Read-Only access)*, *για εγγραφή (Write access)*, *για τροποποίηση* (*R*ead-*Write access)* ή *για εκτέλεση (Xecute access).*

[4] Οι αριθμοί αυτοί είναι ίσοι με A/μέγεθος_σελίδας και A%μέγεθος_σελίδας, αντίστοιχα, διότι το μέγεθος_σελίδας είναι μία δύναμη του δύο, για παράδειγμα 64, 128, 256 ή 512 λέξεις κ.λπ. έως 4 Kbytes ή και παραπάνω.

ίσο με `1`, `true`), τότε ο περιγραφητής σελίδας προσδιορίζει το πλαίσιο της μνήμης που περιέχει τη σελίδα.



Σχήμα 7.3. Αναφορά διεύθυνσης μέσω περιγραφητών σελίδων.

Ο αριθμός της λέξης προστίθεται στη διεύθυνση του πλαισίου αυτού για να δώσει την πραγματική (φυσική) τελική διεύθυνση. Η **απεικόνιση της διεύθυνσης (address mapping)** από εικονική σε φυσική επιτυγχάνεται αυτόματα από το υλισμικό για κάθε αναφορά κάθε διεργασίας στην (εικονική και τελικά φυσική) μνήμη όπως δείχνει το Σχήμα 7.3. Αν η απεικόνιση της διεύθυνσης δεν είναι δυνατή, διότι το δυφίο παρουσίας υποδεικνύει ότι η σελίδα δε βρίσκεται στην κύρια μνήμη, τότε το υλισμικό προξενεί μία διακοπή, που είναι γνωστή ως **σφάλμα σελίδας (page fault)**.

Η *λειτουργία του υλισμικού* ενός συστήματος που χρησιμοποιεί σελιδοποίηση διαφέρει από αυτήν που περιγράψαμε στα Κεφάλαια 6.4.2 και 4.3 του πρώτου τόμου στο ότι περιλαμβάνει (μικρο)εντολές της μορφής:

```
αριθμός_σελίδας = τελική_διεύθυνση.αριθμός_σελίδας και,
παράλληλα, αριθμός_λέξης =
                              τελική_διεύθυνση.αριθμός_λέξης;
/* εφόσον το μέγεθος της σελίδας είναι κάποια δύναμη
     του 2, απομονώνονται από το υλισμικό απλώς τα πεδία
     αριθμός_σελίδας και αριθμός_λέξης */
if (πίνακας_σελίδων [αριθμός_σελίδας].δυφίο_παρουσίας)
     MAR = πίνακας_σελίδων[αριθμός_σελίδας].
                          διεύθυνση_πλαισίου[5] + αριθμός_λέξης;
else στείλε σήμα "διακοπής_σφάλματος_σελίδας";
```

Η διακοπή σφάλματος σελίδας (που εξυπηρετείται από την αντίστοιχη διαδικασία του πυρήνα) αφυπνίζει το διαχειριστή της μνήμης (βλ. Κεφ. 4.4.1 και 8.1), που *προσκομίζει* τη ζητούμενη σελίδα από τη βοηθητική μνήμη και την *τοποθετεί* στην κύρια μνήμη ενημερώνοντας τον πίνακα σελίδων της διεργασίας έτσι, ώστε να μπορέσει να προχωρήσει η εκτέλεσή της. Ο διαχειριστής της μνήμης (λογισμικό) πρέπει επίσης να διαλέξει, όταν συμβεί η διακοπή, ποια σελίδα θα πρέπει να *αντικαταστήσει* με τη ζητούμενη, αν δεν υπάρχει κενό πλαίσιο στη μνήμη όπως θα περιγράψουμε στο επόμενο κεφάλαιο.

Παρατηρούμε ότι με τον παραπάνω μηχανισμό χρειάζονται δύο προσπελάσεις για κάθε αναφορά στη μνήμη: μία προσπέλαση στον πίνακα των σελίδων και μία στην τελική διεύθυνση. Για να αποφευχθεί η πρώτη προσπέλαση, χρησιμοποιούνται συνήθως (8 έως 16)[6] γρήγοροι **συνειρμικοί καταχωρητές** ή **καταχωρητές συσχέτισης (associative registers)**, που περιέχουν αντίγραφα των περιγραφητών των σελίδων στις οποίες έγιναν οι πιο πρόσφατες αναφορές.

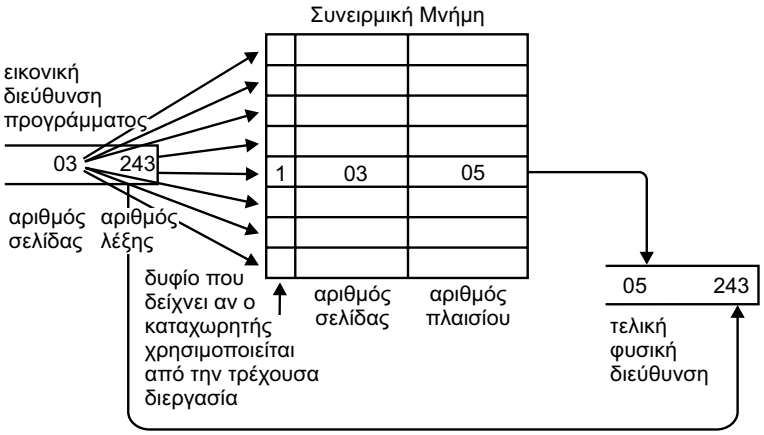
Όταν ανακληθεί μια εντολή του προγράμματος που αναφέρεται στη μνήμη, ο αριθμός της σελίδας της *συγκρίνεται αυτόματα από το υλισμικό ταυτόχρονα με τους αριθμούς των σελίδων που περιέχονται σ' όλους τους συνειρμικούς καταχωρητές* (απ' όπου προήλθε και ο όρος *συνειρμική μνήμη* ή *αποθήκη (associative memory* ή *store*, βλ. Τόμο Ι, Κεφ. 1.1.10). Επειδή οι περισσότερες αναφορές γίνονται στις σελίδες

---

[5] Ή `αριθμός_πλαισίου*μέγεθος_σελίδας` (ο πολλαπλασιασμός υλοποιείται με προς τ' αριστερά μετατόπιση).

[6] Ο Denning έδειξε ότι 8 ως 16 συνειρμικοί καταχωρητές είναι αρκετοί για να εκτελείται το πρόγραμμα χωρίς αισθητή επιβράδυνση (βλ. Άσκ. και Απάντ. 7.1.8).

που έχουν πιο πρόσφατα προσπελαστεί, υπάρχει μεγάλη πιθανότητα να βρεθεί ο αριθμός της σελίδας και να απομονωθεί από ένα συνειρμικό καταχωρητή ο αριθμός ή η διεύθυνση του πλαισίου που την περιέχει όπως δείχνει το Σχήμα 7.4.



Σχήμα 7.4. Αναφορά διεύθυνσης μέσω συνειρμικού καταχωρητή.

Αν ο αριθμός της σελίδας δε βρεθεί, τότε το υλισμικό αναφέρεται στον πίνακα των σελίδων της διεργασίας για να εντοπίσει αν η σελίδα βρίσκεται στη μνήμη ή όχι. Αν η σελίδα είναι παρούσα στη μνήμη, τότε απλώς φορτώνεται ο περιγραφητής της σ' έναν από τους συνειρμικούς καταχωρητές (αφού προηγουμένως αποταμιευτούν τα περιεχόμενα αυτού του καταχωρητή στον αντίστοιχο περιγραφητή)[7]. Οι καταχωρητές φορτώνονται συνήθως κυκλικά (cyclic).

Αν, όμως, η σελίδα δε βρίσκεται στην κύρια μνήμη, τότε το υλισμικό προκαλεί μία διακοπή και το λογισμικό ακολουθεί τη διαδικασία που αναφέραμε παραπάνω, για να προσκομίσει και να τοποθετήσει τη σελίδα. Το λογισμικό χρειάζεται επίσης να ενημερώσει τη συνειρμική μνήμη, αποθηκεύοντας τον περιγραφητή της σελίδας

[7] Εννοείται ότι υπάρχουν περισσότερα πλαίσια σελίδων από καταχωρητές συσχέτισης. Στον πρώτο υπολογιστή που χρησιμοποίησε σελιδοποίηση, στον ATLAS (βλ. Τόμο Ι, Κεφ. 1.1.12), ο οποίος είχε (φυσική) μνήμη 16Κ λέξεων, υπήρχαν 32 συνειρμικοί καταχωρητές, δηλαδή τόσοι όσα τα πλαίσια σελίδων που είχε η μνήμη του. Κάθε σελίδα περιείχε 512 (2^9) λέξεις. Το πεδίο διεύθυνσης των εντολών αναφοράς στη μνήμη είχε μήκος 20 δυφίων, δίνοντας έτσι εικονική μνήμη χωρητικότητας 2^{20} λέξεων. Τα 11 (πλέον σημαντικά) δυφία παρίσταναν τον αριθμό της σελίδας, ενώ τα υπόλοιπα 9 (λιγότερο σημαντικά) τον αριθμό της λέξης.

που αντικαθιστά (στον πίνακα των σελίδων της) και φορτώνοντας τον περιγραφητή της ζητούμενης σελίδας (από τον πίνακα των σελίδων της) στον αντίστοιχο συνειρμικό καταχωρητή.

Η *λειτουργία του υλισμικού* των υπολογιστών που χρησιμοποιούν συνειρμική μνήμη περιέχει (μικρο)εντολές της μορφής:

```
αριθμός_σελίδας = τελική_διεύθυνση.αριθμός_σελίδας και,
παράλληλα, αριθμός_λέξης =
                              τελική_διεύθυνση.αριθμός_λέξης;
                                 // όπως περιγράψαμε παραπάνω
εντόπισε_παράλληλα (συνειρμική μνήμη, αριθμός_σελίδας,
                                          διεύθυνση[8], βρέθηκε);
if (!βρέθηκε)
if (!πίνακας_σελίδων[αριθμός_σελίδας].δυφίο_παρουσίας)
                             σήμανε διακοπή_σφάλματος_σελίδας;
else
{
     αποταμίευσε τα περιεχόμενα του (επόμενου)
                                       συνειρμικού καταχωρητή;
     φόρτωσε τον καταχωρητή αυτόν με το ζεύγος
     (αριθμός_σελίδας, πίνακας_σελίδων[αριθμός_σελίδας].
                                          διεύθυνση_πλαισίου);
     MAR = καταχωρητής_συσχέτισης.διεύθυνση[9] +
                                                 αριθμός_λέξης;
}
else MAR = καταχωρητής_συσχέτισης.διεύθυνση +
                                                 αριθμός_λέξης;
```

Κάθε φορά που διανέμεται η ΚΜΕ σε μία νέα διεργασία (βλ. Κεφ. 5.1), τα περιεχόμενα της συνειρμικής μνήμης πρέπει να διαγραφούν, διότι η νέα διεργασία μπορεί ν' αναφέρεται στους ίδιους αριθμούς σελίδων με την προηγούμενη. Επειδή η διανομή της ΚΜΕ πρέπει να γίνεται όσο το δυνατόν ταχύτερα, αν η διαγραφή

---

[8] Ή `αριθμός_πλαισίου`.

[9] Ή `αριθμός_πλαισίου*μέγεθος_σελίδας` (ο πολλαπλασιασμός υλοποιείται με μετατόπιση προς τ' αριστερά).

ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

ΣΥΣΤΗΜΑΤΑ ΥΠΟΛΟΓΙΣΤΩΝ ΤΟΜΟΣ ΙΙ

Το βιβλίο αυτό καλύπτει:

- Ιστορική αναδρομή των λειτουργικών συστημάτων
- Παράλληλο προγραμματισμό
- Ταυτόχρονες διεργασίες
- Επικοινωνία διεργασιών
- Αδιέξοδα
- Δομή λειτουργικών συστημάτων και πυρήνων τους
- Διαχείριση εισόδου/εξόδου
- Διαχείριση μνήμης
- Χρονοπρογραμματισμό της Κεντρικής Μονάδας Επεξεργασίας
- Συστήματα αρχείων
- Μηχανισμούς προστασίας και διαχείρισης εργασιών
- Εισαγωγή στα συστήματα πραγματικού χρόνου και την αξιοπιστία.

Περιέχει επίσης λεπτομερή τεχνική περιγραφή του λειτουργικού συστήματος UNIX, καθώς και πολλές λυμένες ασκήσεις σε TURBO Pascal, C και C-shell.

*Ο **Γιάννης Κάβουρας** είναι Καθηγητής στο Τμήμα Πληροφορικής του Οικονομικού Πανεπιστημίου Αθηνών (ΟΠΑ). Σπούδασε Φυσικές Επιστήμες στο Πανεπιστήμιο Αθηνών, και κατέχει τους μεταπτυχιακούς τίτλους Diploma in Computing Science (University of Glasgow), Master of Science (University of London, ICS) και Ph.D. (University of Glasgow) στην Επιστήμη Υπολογιστών (Computer Science). Έχει εργαστεί ως "Lecturer" στο Glasgow College of Technology (τώρα Glasgow Caledonian University), στο Heriot-Watt University στο Εδιμβούργο και στο University of Glasgow, καθώς και ως "Επιστήμων Ερευνητής" στο Ε.ΚΕ.Φ.Ε. "Δημόκριτος". Διατέλεσε Επίτιμος (Άμισθος) Πρόξενος της Ελλάδος στη Γλασκώβη και στην περιοχή Strathclyde της Σκωτίας. Προϊστάμενος του Κέντρου Υπολογιστών του ΟΠΑ, επανειλημμένως Πρόεδρος του Τμήματος Πληροφορικής του Πανεπιστημίου αυτού και Πρόεδρος του Γνωμοδοτικού Τεχνικού Συμβουλίου του Υπουργείου Εσωτερικών, Δημόσιας Διοίκησης και Αποκέντρωσης. Υπήρξε κριτής επιστημονικών άρθρων για την Association of Computer Machinery (ACM) και άλλους ερευνητικούς φορείς, μέλος της ACM, μέλος και "Fellow" της British Computer Society (BCS), και μέλος της ΕΠΥ. Έχει, μεταξύ άλλων, γράψει πανεπιστημιακά συγγράμματα στους τομείς του Προγραμματισμού και της Οργάνωσης Υπολογιστών, καθώς και των Λειτουργικών και Κατανεμημένων Συστημάτων, οι οποίοι περιλαμβάνονται στα επιστημονικά του ενδιαφέροντα.*

Επισκεφθείτε μας στο Internet:
www.klidarithmos.gr

εκδόσεις ΚΛΕΙΔΑΡΙΘΜΟΣ
Δαμασκού 4, Στάθμός Λαρίσης, 10440 ΑΘΗΝΑ, Τηλ. 210-5237035

ISBN 978-960-461-244-4