v.18-03-2024

ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΡΑΚΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΑΘΗΜΑ
ΟΡΓΑΝΩΣΗ ΥΠΟΛΟΓΙΣΤΩΝ (206ΕΥΥΚ)
ΠΠΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΕΑΡΙΝΟ 2023-2024

Διάλεξη Νº1:

Εισαγωγή στον M68000

Δ. Καραμπατζάκης, Επίκουρος Καθηγητής

email. dkara@cs.ihu.gr

Δήλωση προσβασιμότητας

Σε αυτό το μάθημα όλες/οι οι φοιτήτριες/τές απολαμβάνουν – και αντίστοιχα υποχρεούνται να σέβονται – το δικαίωμα της ίσης μεταχείρισης. Δεν είναι ανεκτή και αποδεκτή κανενός τύπου και μορφής διάκριση με κριτήρια την εθνικότητα, τη φυλή, την καταγωγή, τη γλώσσα, το φύλο, τη θρησκεία, την ηλικία, την υγεία, τη σωματική ικανότητα, την ιδιωτική ζωή, τον γενετήσιο προσανατολισμό, τη σωματική ικανότητα και την οικονομική και κοινωνική κατάσταση στην οποία αυτοί βρίσκονται.

Το Πανεπιστήμιο άγρυπνα μεριμνά για τη διασφάλιση της αρχής των ίσων ευκαιριών και της ίσης μεταχείρισης. Οι κοινωνικές προκαταλήψεις και οι ιδεολογικές παρωπίδες είναι έννοιες τελείως ξένες με την επιστημονική πρόοδο την οποία το Πανεπιστήμιο είναι ταγμένο να υπηρετεί.

Ο Διδάσκων

Πληροφορίες για το Μάθημα

Διδάσκων:

Δημήτρης Καραμπατζάκης, Επίκουρος Καθηγητής

Αναλογικά και Ψηφιακά Ηλεκτρονικά Συστήματα

Μέλος Εργαστηρίου Βιομηχανικών και Εκπαιδευτικών Ενσωματωμένων Συστημάτων

Επικοινωνία / πληροφορίες:

Email. dkara@cs.ihu.gr

web. http://www.internetofthings.gr/

Ώρες Γραφείου:

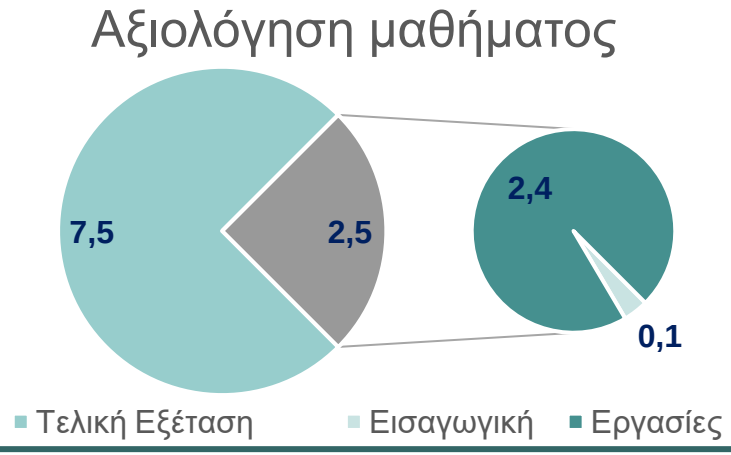
μετά από συνεννόηση με email στο ΦΕ 315 (πάνω από αιθ. Α1)

Πληροφορίες για το Μάθημα (Γενικές)

- **Κάθε Τρίτη, Πέμπτη 12.00 π.μ. - 14.00 μ.μ. μάθημα θεωρίας στο Μεγάλο Αμφιθέατρο (μπορεί να αλλάζει με ανακοινώσεις).**
- **Η διαχείριση του μαθήματος θα γίνει με χρήση της υπηρεσίας https://courses.cs.ihu.gr**
- **Όλοι οι φοιτητές πρέπει να έχουν λογαριασμό στο uregister.**
- **Η ιστοσελίδα με τις πληροφορίες του μαθήματος: http://iees.cs.ihu.gr/?page_id=3209**
- **Υλικό του μαθήματος στο moodle: https://moodle.cs.ihu.gr/**

Πληροφορίες για το Μάθημα (Αξιολόγηση)

- **Η βαθμολογία είναι 75% από την τελική εξέταση και 25% από τις ατομικές εργασίες (1 σετ ασκήσεων) που θα δοθούν για το σπίτι.**
- **Η τελική εξέταση είναι με ανοιχτό το κύριο σύγγραμμα του μαθήματος.**
- **Ο βαθμός του μαθήματος (ΒΜ = ΓΕ*0,75 + ΣΑ*0,25) πρέπει να είναι τουλάχιστον πέντε (5).**



Πληροφορίες για το Μάθημα (Μονάδες)

- **Κωδικός Μαθήματος:** 206ΕΥΥΚ
- **Εξάμηνο:** 2ο
- **Τύπος Μαθήματος:** Υποβάθρου, Ανάπτυξης Δεξιοτήτων
- **Είδος Μαθήματος:** Υποχρεωτικό (ΥΠ)
- **Διδασκαλία Θεωρίας:** 3 ώρες/εβδομάδα
- **Διδασκαλία Φροντιστήριο:** 1 ώρες/εβδομάδα
- **Πιστωτικές μονάδες ECTS: 7**
- **Γλώσσα διδασκαλίας και Εξετάσεων:** Ελληνικά

Πληροφορίες για το Μάθημα (Φόρτος)

Δραστηριότητα	**Φόρτος εργασίας εξαμήνου**
Διαλέξεις	78 ώρες
Φροντιστηριακές Ασκήσεις	26 ώρες
Γραπτές Εξετάσεις	2 ώρες
Γραπτές Εργασίες	34 ώρες
Αυτοτελής Μελέτη	35 ώρες
Σύνολο	**175 ώρες (7 ECTS)**

Κύριο Σύγγραμμα Μαθήματος (ΕΥΔΟΞΟΣ)

Οργάνωση και Σχεδίαση Υπολογιστών

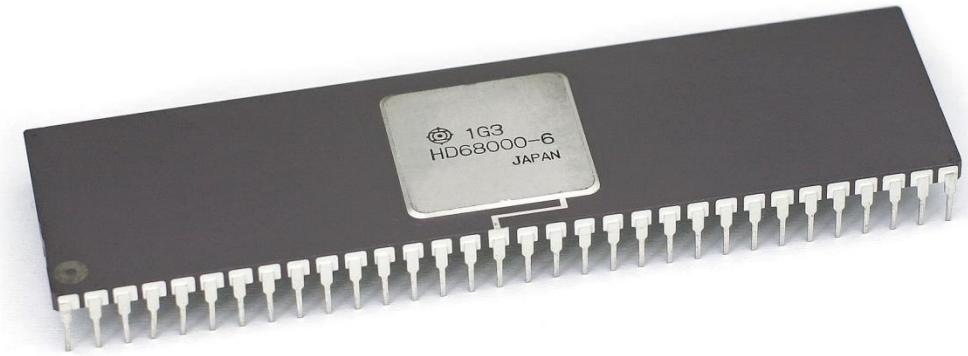
Συγγραφέας: Πογαρίδης Δημήτριος

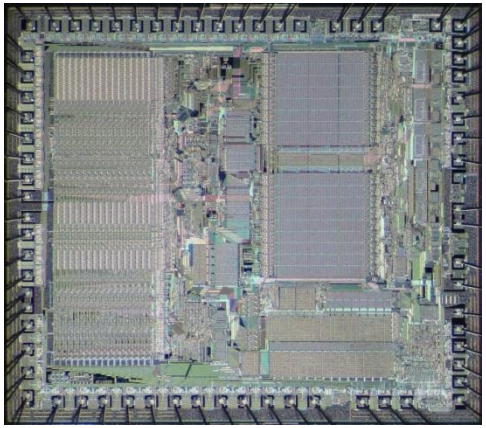
Έτος Έκδοσης: 2019

Κωδικός στον Εύδοξο: **86192986**

Λογισμικό - Αναπτυξιακό

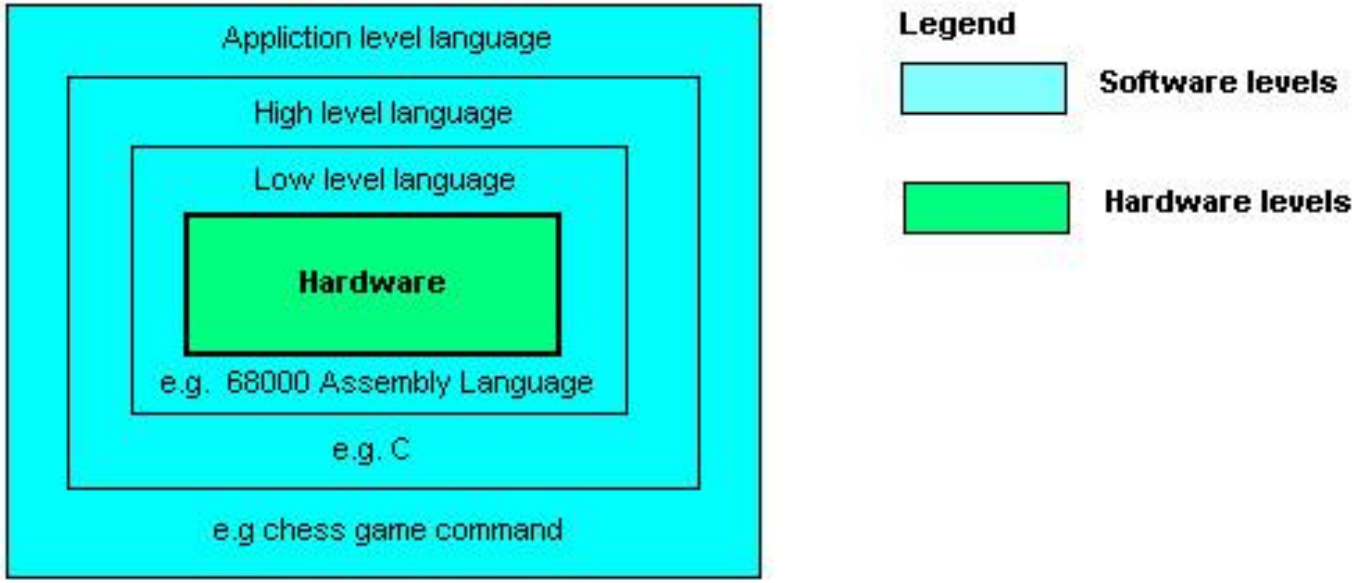
- **Α’ μέρος μαθήματος (CISC):**
  - Assembly για τον Motorola68000
  - Λογισμικό easy68k http://www.easy68k.com/
- **Β’ μέρος μαθήματος (RISC):**
  - Υλοποίηση σχεδιάσεων σε αναπτυξιακό Arduino (προαιρετική αγορά του υλικού σύμφωνα με τις οδηγίες)
  - Λογισμικό Arduino IDE https://www.arduino.cc/en/Main/Software
  - Η γλώσσα προγραμματισμού (C++) και οι εντολές που υποστηρίζει είναι διαθέσιμες στο: https://www.arduino.cc/reference/en/

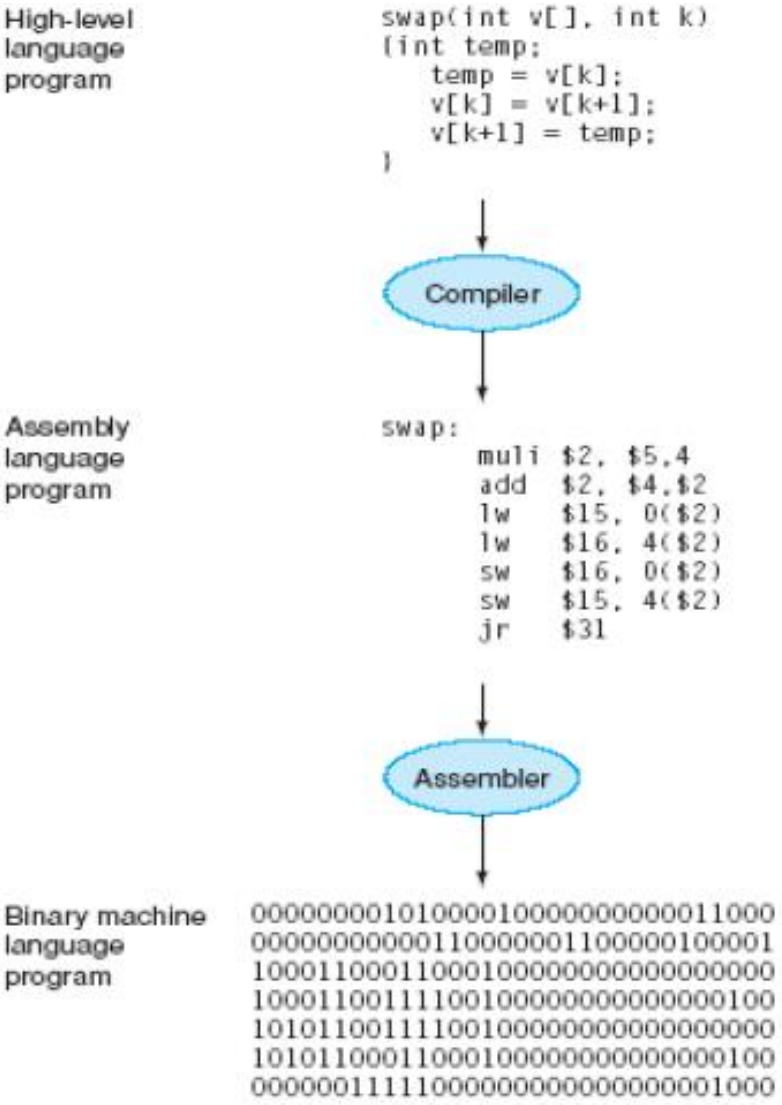




Μια πρώτη προσέγγιση στον CISC Επεξεργαστή M68000

Επίπεδα Λογισμικού/Υλικού
SW/HW Levels



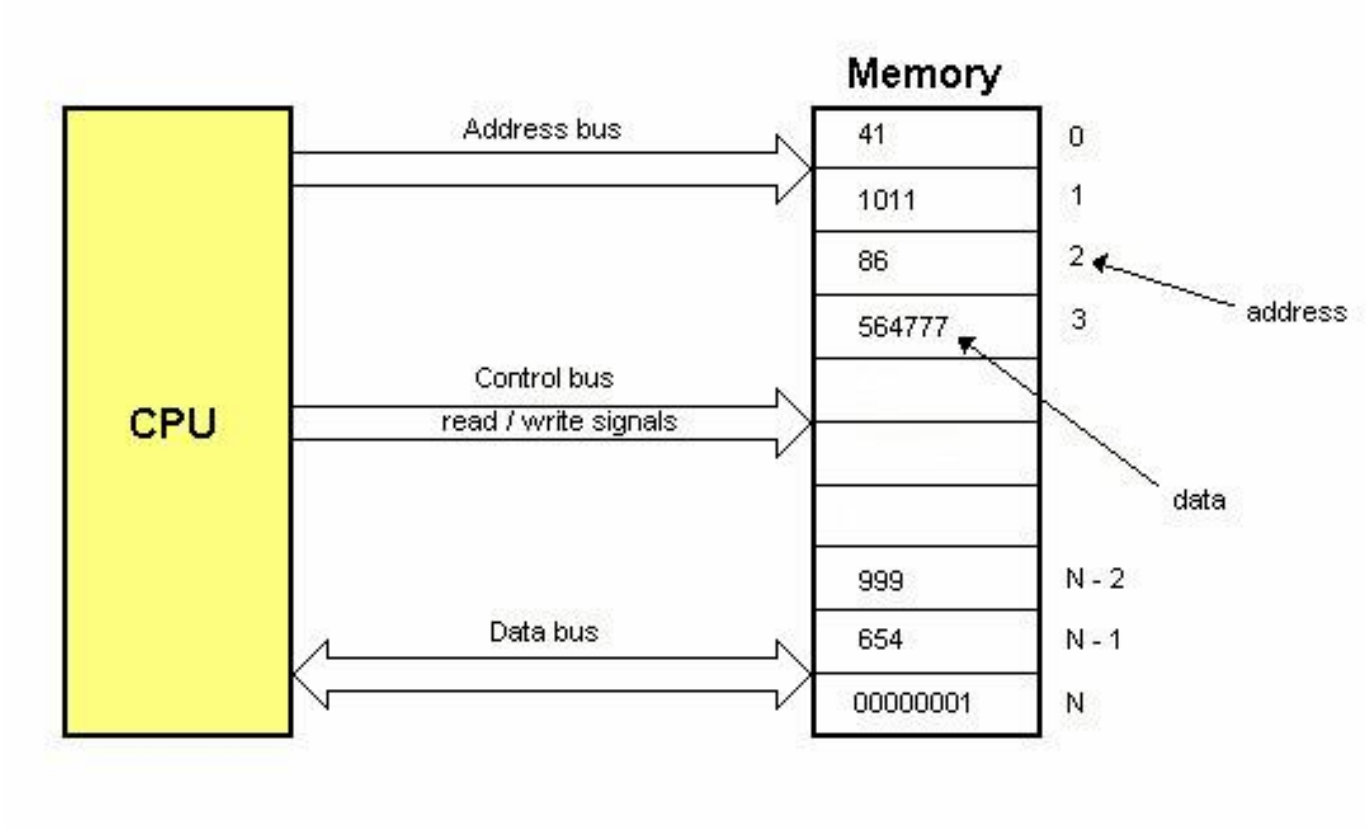

High-level language program
swap(int v[], int k)
{int temp;
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
}
Compiler
Assembly language program
swap:
muli $2, $5,4
add $2, $4,$2
lw $15, 0($2)
lw $16, 4($2)
sw $16, 0($2)
sw $15, 4($2)
jr $31
Assembler
Binary machine language program
00000000101000010000000000011000
00000000000011000000110000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000

Υπολογιστική Μηχανή Von Neumann

- Οι υπολογιστικές μηχανές υπάρχουν σήμερα σε πολλές διαφορετικές μορφές. Αυτό το μάθημα αρχικά επικεντρώνεται στη λεγόμενη Μηχανή Von Neumann. Ο Von Neumann ήταν επιστήμονας που παρουσίασε στη δεκαετία του 1940 τα γενικά χαρακτηριστικά των βασικών αρχιτεκτονικών υπολογιστών του σήμερα.
- Η υπολογιστική μηχανή Von Neumann βασίζεται στο κοινό σύστημα μνήμης που αποθηκεύει τόσο τις εντολές (instructions, commands) που καθορίζουν το πρόγραμμα (program) του υπολογιστή όσο και τα δεδομένα (data) που λειτουργούν με αυτές τις εντολές.

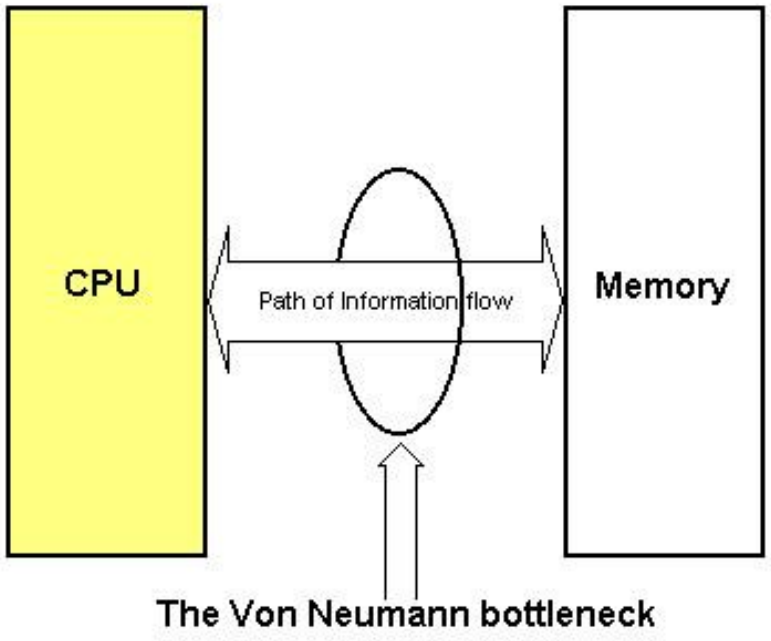
Επικοινωνία ΚΜΕ με Μνήμη

CPU < --- > Memory

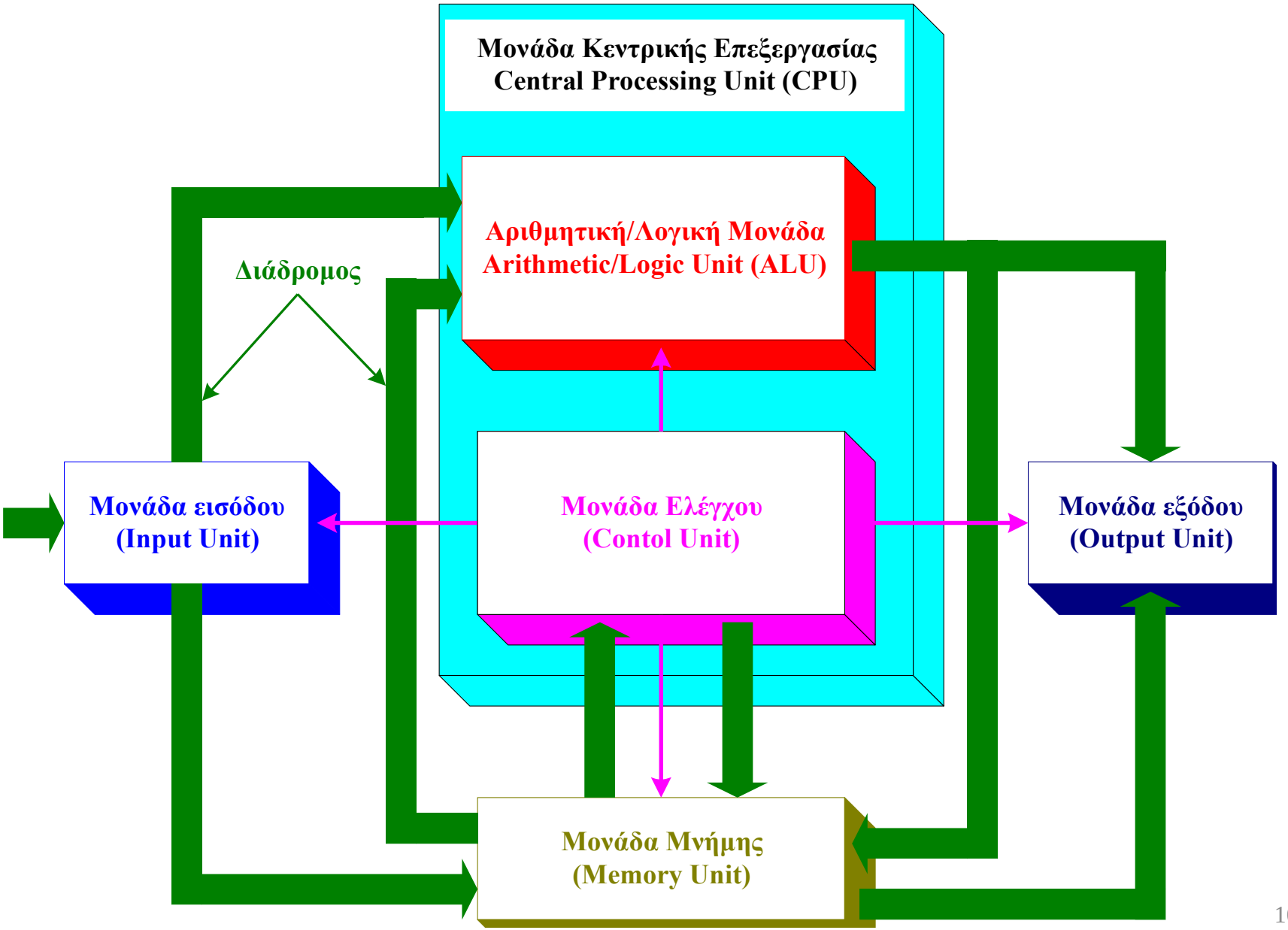


Υπολογιστική Μηχανή Von Neumann

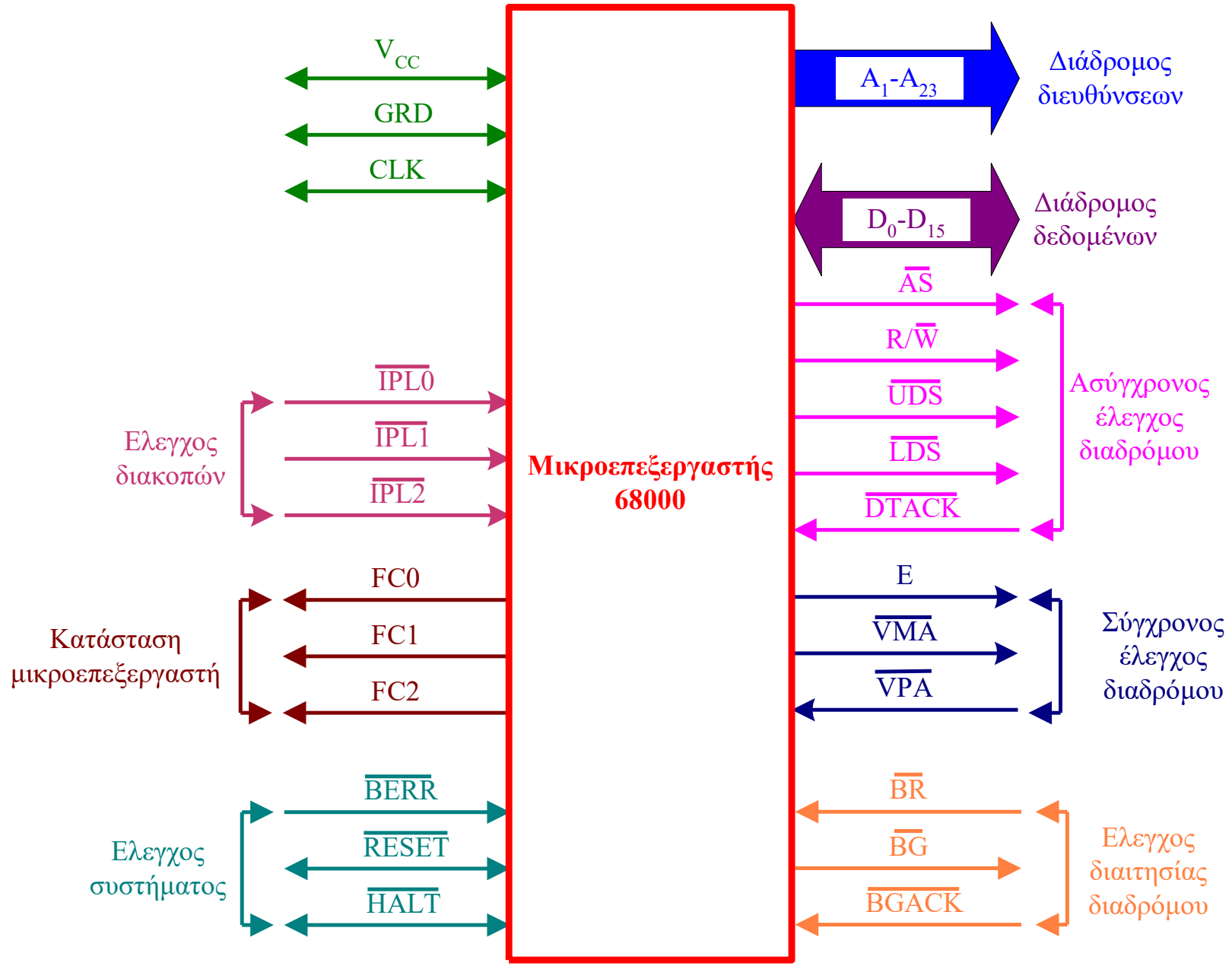
- Δεδομένου ότι όλες αυτές οι πληροφορίες πρέπει να ανταλλάσσονται συνεχώς μεταξύ της CPU και της μνήμης, η διαδρομή που ενώνει την CPU και τη μνήμη ονομάζεται συμφόρηση (Bottleneck) Von Neumann.
- Η συμφόρηση του Von Neumann είναι ένας από τους περιοριστικούς παράγοντες της μηχανής Von Neumann, καθώς το πλάτος του διαδρόμου δεδομένων θέτει ένα όριο στο μέγιστο δυνατό όγκο και ταχύτητα μεταφοράς των δεδομένων μεταξύ της CPU και της μνήμης.



Βασική Δομή Υπολογιστή

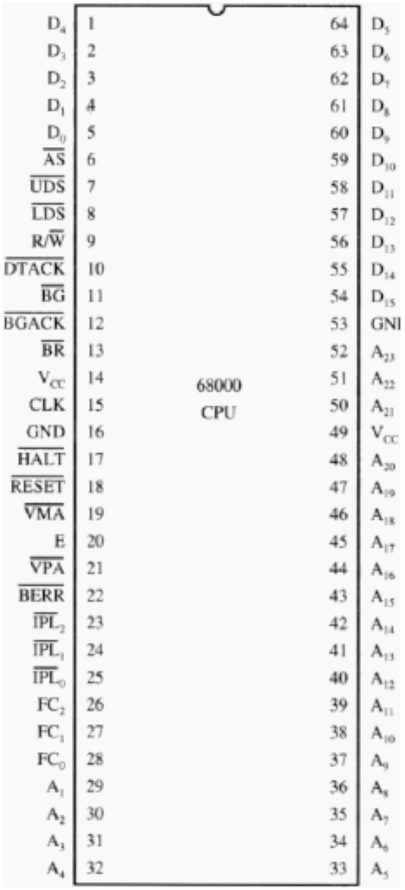


Ο μικροεπεξεργαστής M68000



Χαρακτηριστικά M68000

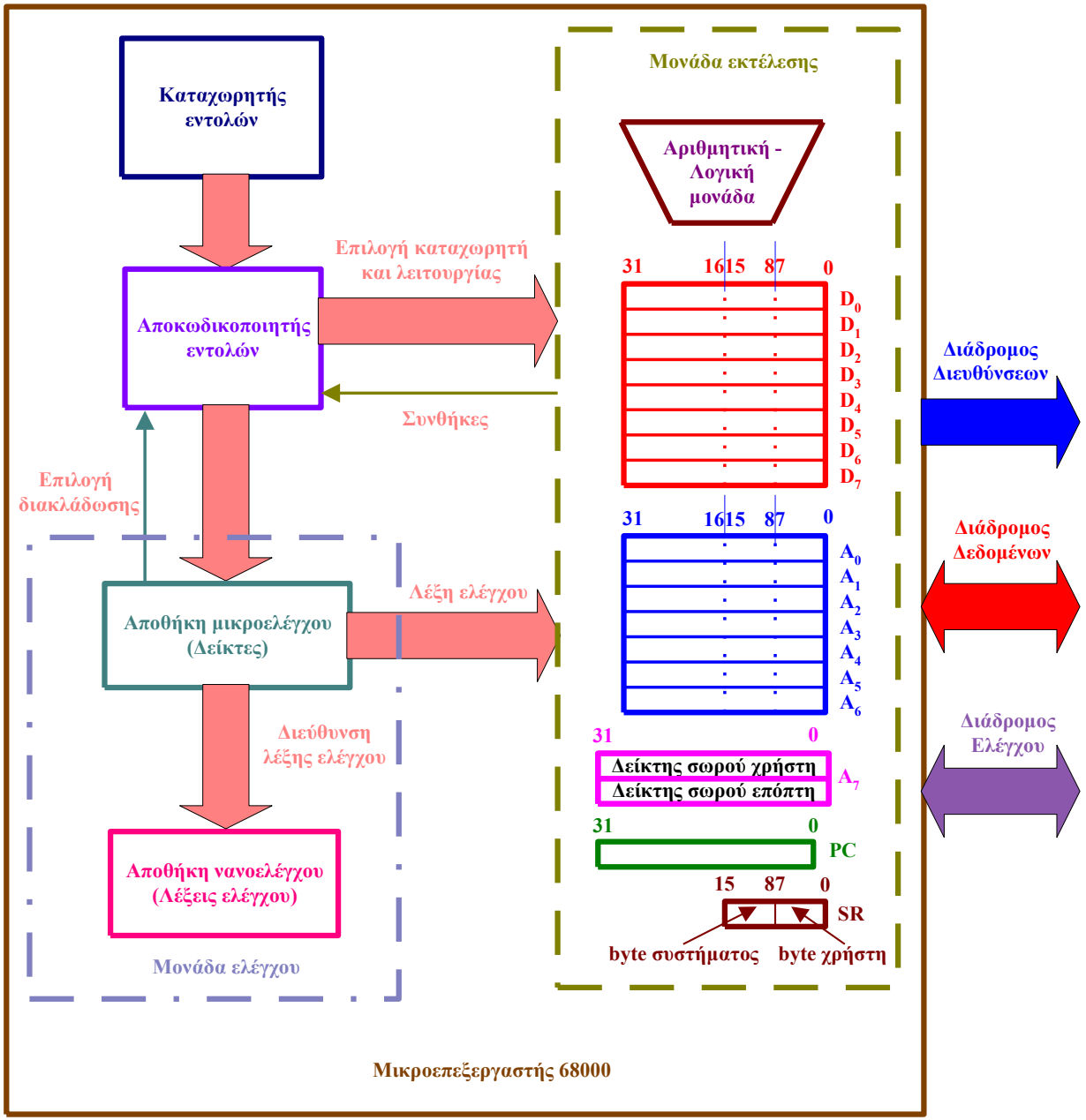
- **32-bit Καταχωρητές Δεδομένων και Διευθύνσεων (Registers)**
- **16-bit Διάδρομο Δεδομένων (Data Bus)**
- **24-bit Διάδρομο Διευθύνσεων (Address Bus)**
- **14 Μεθόδους Διευθυνσιοδότησης (Addressing Modes)**
- **Μετρητής Προγράμματος (Program Counter)**
- **Σετ εντολών με 56 εντολές (Instruction Set)**
- **5 βασικούς τύπους δεδομένων (Data types)**
- **7 επίπεδα διακοπών (Interrupt levels)**
- **Ταχύτητα Ρολογιού 4MHz -12.5MHz (Clock speed)**
- **Σύγχρονη και Ασύγχρονη Μεταφορά Δεδομένων**
- **Αντιστοίχιση I/O στη Μνήμη (Memory Mapped I/O)**



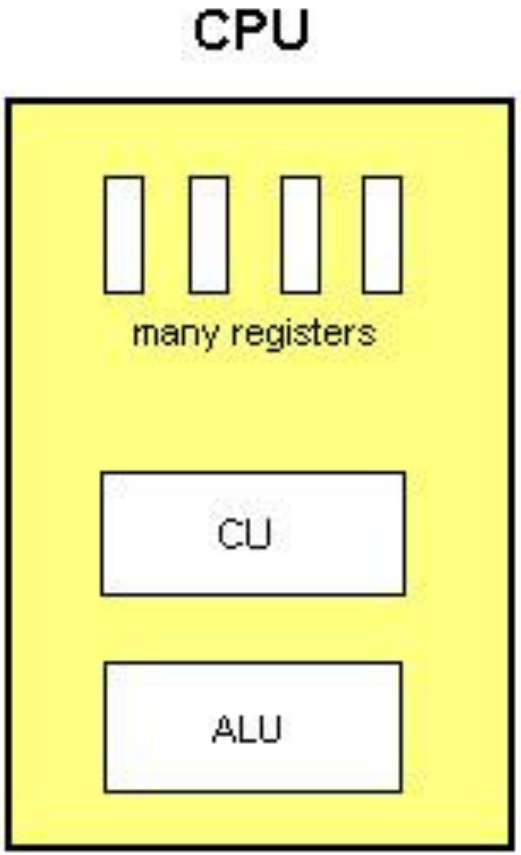
Τι είναι η «Αρχιτεκτονική Μικροεπεξεργαστή»

- Για τη δική μας δουλειά **η Αρχιτεκτονική είναι το λογισμικό ή το μοντέλο προγραμματισμού** του επεξεργαστή, δηλαδή:
  - Οι **καταχωρητές (Registers)** της ΚΜΕ που είναι διαθέσιμοι στον προγραμματιστή.
  - Οι βασικές **εντολές (Commands)** που μπορεί να εκτελέσει η ΚΜΕ.
  - Οι τρόποι που οι εντολές μπορούν **να καθορίσουν μια θέση μνήμης (Addressing Modes)**.
  - Ο τρόπος **που οργανώνεται η πληροφορία στη μνήμη (Memory and Data Organization).**
  - Πως η ΚΜΕ αποκτά πρόσβαση και ελέγχει τις **περιφερειακές συσκευές (Peripheral Devices).**

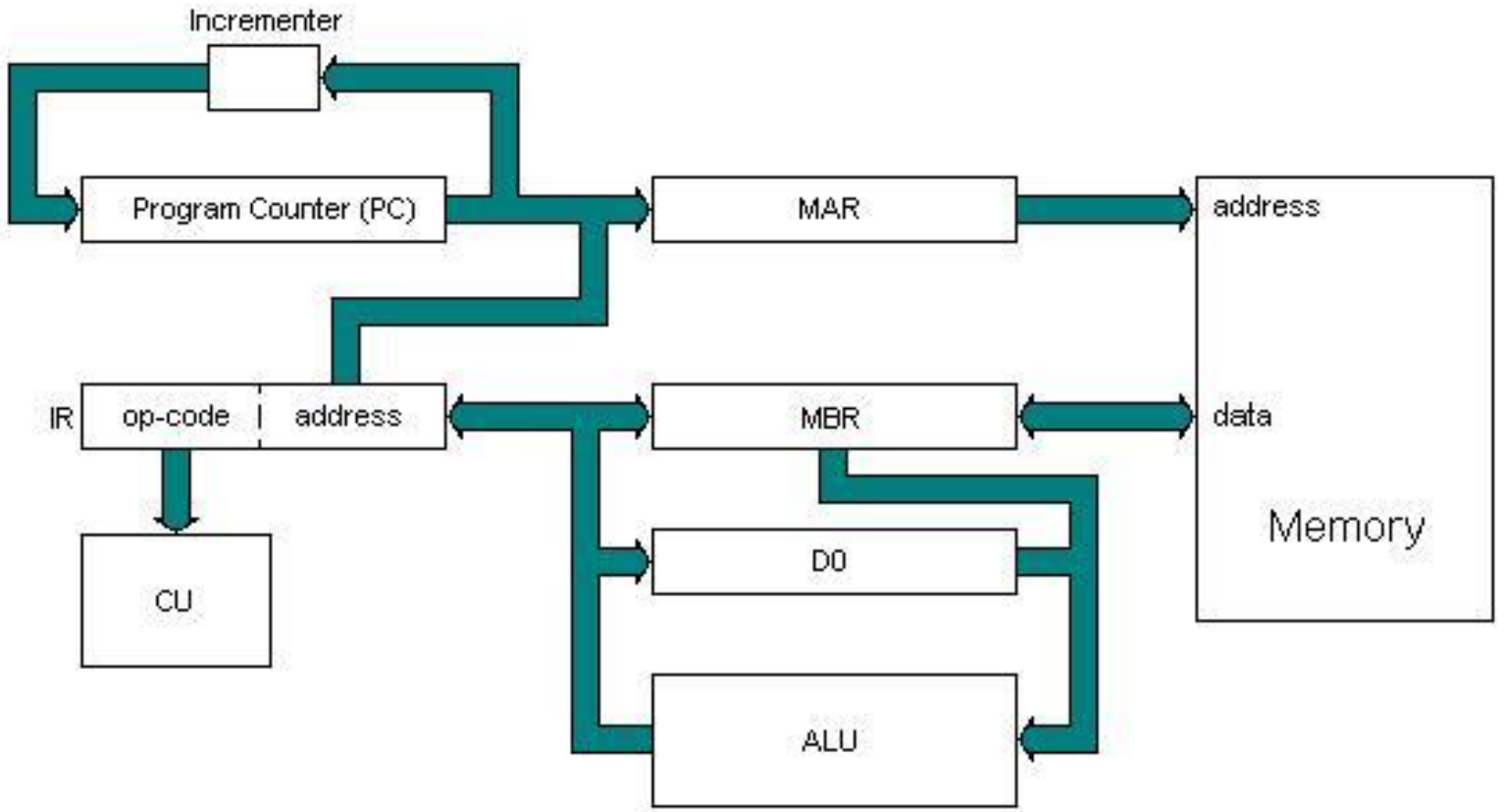
Μοντέλο προγραμματισμού του M68000



Μια απλή Κ.Μ.Ε. - CPU



Ένα απλό μοντέλο



Ένα απλό μοντέλο

Οι λειτουργικές του μονάδες είναι:

MAR: Ο καταχωρητής διευθύνσεων μνήμης, ο οποίος διατηρεί στη μνήμη τη διεύθυνση της επόμενης τοποθεσίας στην οποία θα έχετε πρόσβαση. Τα περιεχόμενα του MAR δείχνουν σε αυτήν την τοποθεσία. Για παράδειγμα, εάν τα περιεχόμενα του MAR είναι το 200, τότε η τοποθεσία με διεύθυνση 200 θα είναι η επόμενη στην οποία θα έχει πρόσβαση η CPU.

MBR: Ο καταχωρητής προσωρινής μνήμης, ο οποίος διατηρεί τα δεδομένα που μόλις διαβάστηκαν από τη μνήμη ή τα δεδομένα που πρόκειται να εγγραφούν στη μνήμη. Όλες οι πληροφορίες που ρέουν μέσα ή έξω από τη μνήμη πρέπει να περάσουν μέσω του MBR. Στην Επιστήμη των Υπολογιστών, ο όρος buffer χρησιμοποιείται συχνά για να αναφέρεται σε μια συσκευή που χρησιμοποιείται για την προσωρινή διατήρηση δεδομένων σε αναμονή για επεξεργασία ή δεδομένων που μόλις υποβλήθηκαν σε επεξεργασία και περιμένουν να αποθηκευτούν.

IR: Ο καταχωρητής εντολών, ο οποίος διατηρεί την πιο πρόσφατη εντολή ανάγνωσης από τη μνήμη ενώ αποκωδικοποιείται από το CU.

PC: Ο μετρητής προγράμματος, που ονομάζεται επίσης μετρητής εντολών, ο οποίος είναι ένας καταχωρητής που διατηρεί τη διεύθυνση στη μνήμη της επόμενης εντολής που θα εκτελεστεί. Μετά την ανάκτηση μιας εντολής από τη μνήμη, ο υπολογιστής αυξάνεται αυτόματα για να κρατήσει τη διεύθυνση ή να δείξει την επόμενη εντολή που θα εκτελεστεί.

D0: Ένα μητρώο δεδομένων γενικής χρήσης (Καταχωρητής) που χρησιμοποιείται για τη διατήρηση δεδομένων κάθε είδους. Τα δεδομένα που διατηρεί είτε πρόκειται να χρησιμοποιηθούν από τον επεξεργαστή είτε είναι το αποτέλεσμα μιας λειτουργίας που εκτελείται από τον επεξεργαστή.

ALU: Η Αριθμητική & Λογική Μονάδα - εκτελεί υπολογισμούς και συγκρίσεις (Αριθμητικές / λογικές πράξεις).

CU: Η Μονάδα Ελέγχου - ερμηνεύει το μοτίβο bit της εντολής (τον op-code της) που είναι αποθηκευμένο στον καταχωρητή IR και είναι υπεύθυνη για τη λήψη των απαραίτητων μέτρων για την εκτέλεσή της κάθε εντολής.

RTL (1)

RTL (Register Transfer Language - Γλώσσα Μεταφοράς Καταχωρητών), έγινε δημοφιλής από τους Hill & Peterson. Η RTL είναι ένας ψευδοκώδικας που βοηθά να ορίσουμε τη δράση των εντολών της γλώσσας Assembly. Αυτός ο ψευδοκώδικας είναι πολύ απλός, έχει πολύ λίγους κανόνες, μιμείται γλώσσες υψηλού επιπέδου και θα μας βοηθήσει σε μεγάλο βαθμό στην κατανόηση της ροής δεδομένων στον υπολογιστή.

Εδώ έχουμε τους κανόνες που θα μας χρειαστούν:

- Αγκύλες [ ] (Square brackets) Αναφέρεται στα περιεχόμενα του καταχωρητή ή της θέσης μνήμης που περιέχει. Να θυμάστε ότι οι καταχωρητές και οι θέσεις μνήμης χρησιμοποιούνται για αποθήκευση δεδομένων, π.χ. [MBR] σημαίνει το περιεχόμενο του καταχωρητή MBR.

- Αριθμοί και Συμβολικά Ονόματα (Numbers and symbolic names) από μόνοι τους αντιπροσωπεύουν μια πραγματική αριθμητική τιμή (literal, immediate value),
π.χ. η έκφραση [MBR] = 8 σημαίνει ότι τα περιεχόμενα του καταχωρητή προσωρινής μνήμης είναι ίσα με την αριθμητική τιμή 8. Το 8 σημαίνει τον αριθμό 8, ενώ το [8], το οποίο είναι συντομογραφία για [M (8)], σημαίνει το περιεχόμενο της θέσης μνήμης 8.

RTL (2)

-Οι θέσεις μνήμης στο σύστημα μνήμης αναφέρονται ως M (διεύθυνση) όπου η διεύθυνση δηλώνει την πραγματική διεύθυνση της θέσης.

Για παράδειγμα, το [M (128)] = 16 σημαίνει ότι το περιεχόμενο της θέσης μνήμης 128 είναι η τιμή 16. Σημειώστε ότι δε χρειάζεται να καθορίσετε μια θέση όταν εργάζεστε με καταχωρητές, επειδή ένας καταχωρητής είναι μία θέση, ενώ η μνήμη είναι μια σειρά τοποθεσιών. Σημειώστε επίσης πως η σημείωση RTL για τοποθεσίες μνήμης μοιάζει με τη σημείωση στη C για συστοιχίες. Αυτό δεν είναι σύμπτωση, αντίθετα. Η διεύθυνση μιας θέσης είναι ισοδύναμη με το ευρετήριο ενός πίνακα στη C, και το M σημαίνει απλά τη συστοιχία θέσεων μνήμης. Οι πίνακες στη C υλοποιούνται στον υπολογιστή διατηρώντας τον απαραίτητο αριθμό διαδοχικών θέσεων μνήμης.

-Η μεταφορά πληροφοριών μεταξύ καταχωρητών ή θέσεων μνήμης υποδεικνύεται από ένα βέλος που δείχνει προς τα αριστερά: ←

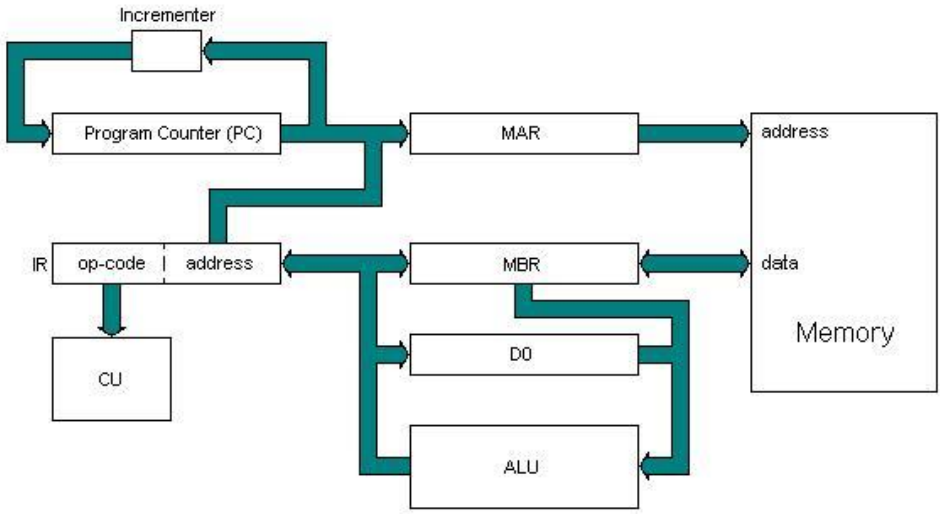
Για παράδειγμα, το [M (1234)] ← 64 σημαίνει ότι τα περιεχόμενα της θέσης μνήμης 1234 αντικαθίστανται από την αριθμητική τιμή 64. Το βέλος που δείχνει προς τα αριστερά είναι ισοδύναμο με τον χειριστή εκχώρησης στην Pascal ": =". Για παράδειγμα, η δήλωση Pascal A: = B + C μπορεί να γραφτεί σε RTL ως [M (A)] ← [M (B)] + [M (C)], το περιεχόμενο της θέσης μνήμης C προστίθεται στο περιεχόμενο της θέσης μνήμης B και το αποτέλεσμα αποθηκεύεται στη θέση μνήμης A. Τα γράμματα A, B και C εδώ είναι συμβολικές παραστάσεις πραγματικών αριθμητικών τιμών που αντιπροσωπεύουν διευθύνσεις θέσεων μνήμης.

RTL (3)

100	means “#100” or “the number 100”
[M(4)]	means “contents stored in memory location 4”
[M(4)] = 100	means “memory location 4 contains #100”
[M(4)] ← 25	means “load number 25 into memory location 4”
[PC] ← 4	means “load number 4 into PC”
[M(4)]← 100+[M(4)]	means “add #100 to contents of location 4 and save”

Instruction			RTL
	MOVE.W	#100,D0	[D0] ← 100
	MOVE.W	$100,D0	[D0] ← [M($100)]
	ADD.W	D1,D0	[D0] ← [D0] + [D1]
	MOVE.W	D1,100	[M(100)] ← D1
DATA	DC.B	20	[DATA] ← 20
	BRA	LABEL	[PC] ← label

Κύκλος Φόρτωσης / Εκτέλεσης
Fetch / Execute cycle



FETCH

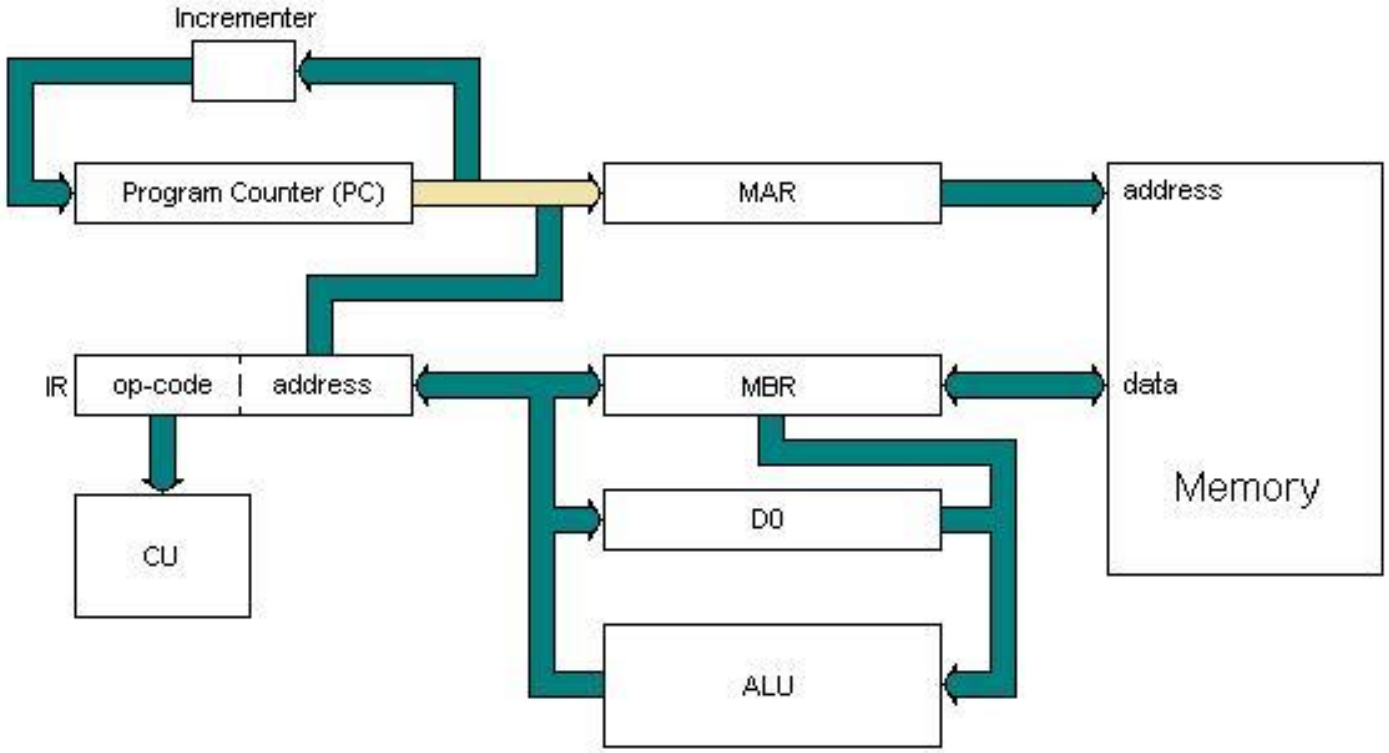
[MAR] <- [PC]
[PC] <- [PC] + 1
[MBR] <- [M([MAR])]
[IR] <- [MBR]

EXECUTE

[MAR] <- [IR(Address_Field)]
[MBR] <- [M([MAR])]
[D0] <- [D0] + [MBR]

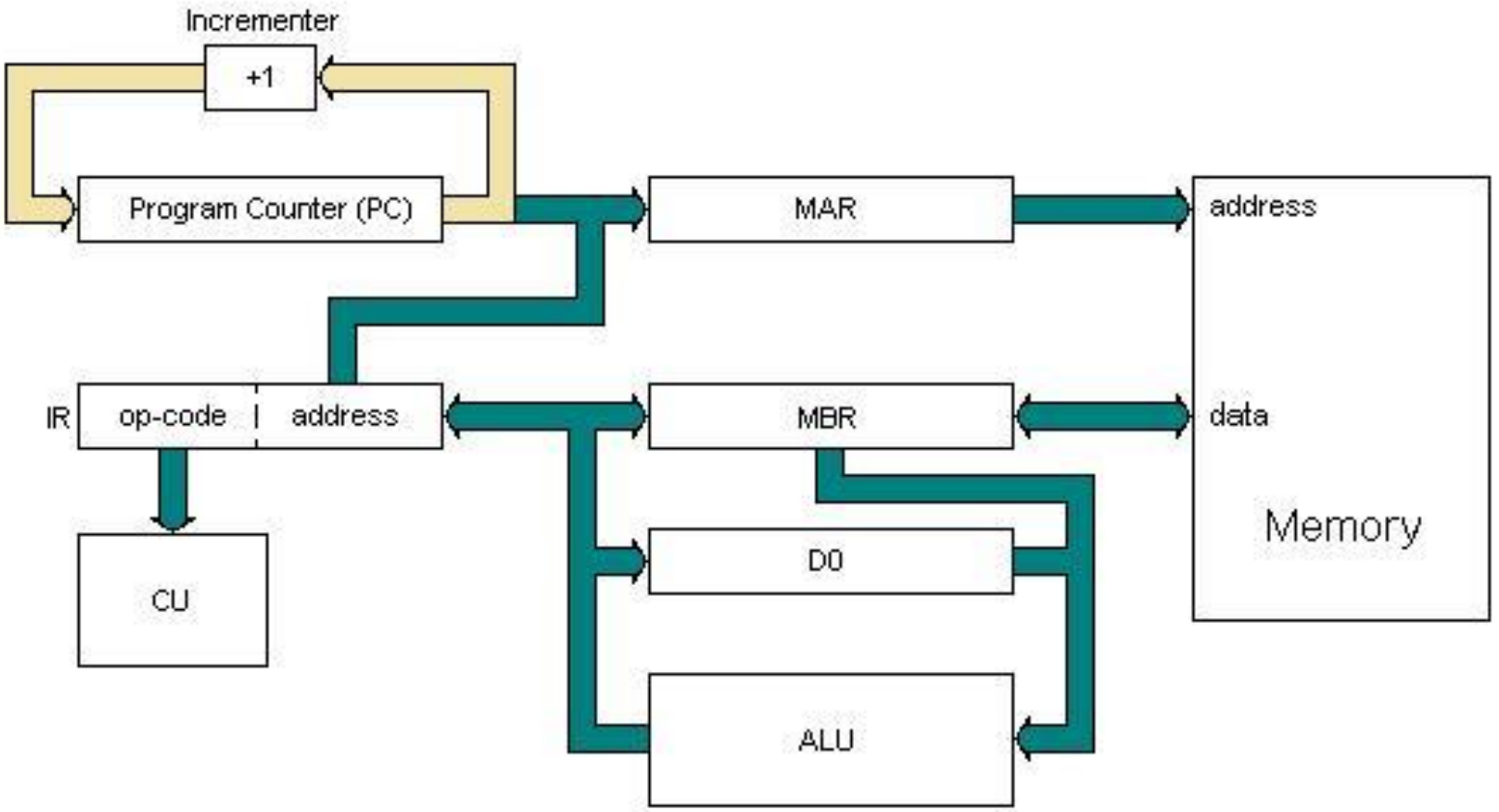
Fetch cycle: step 1

[MAR] <- [PC]



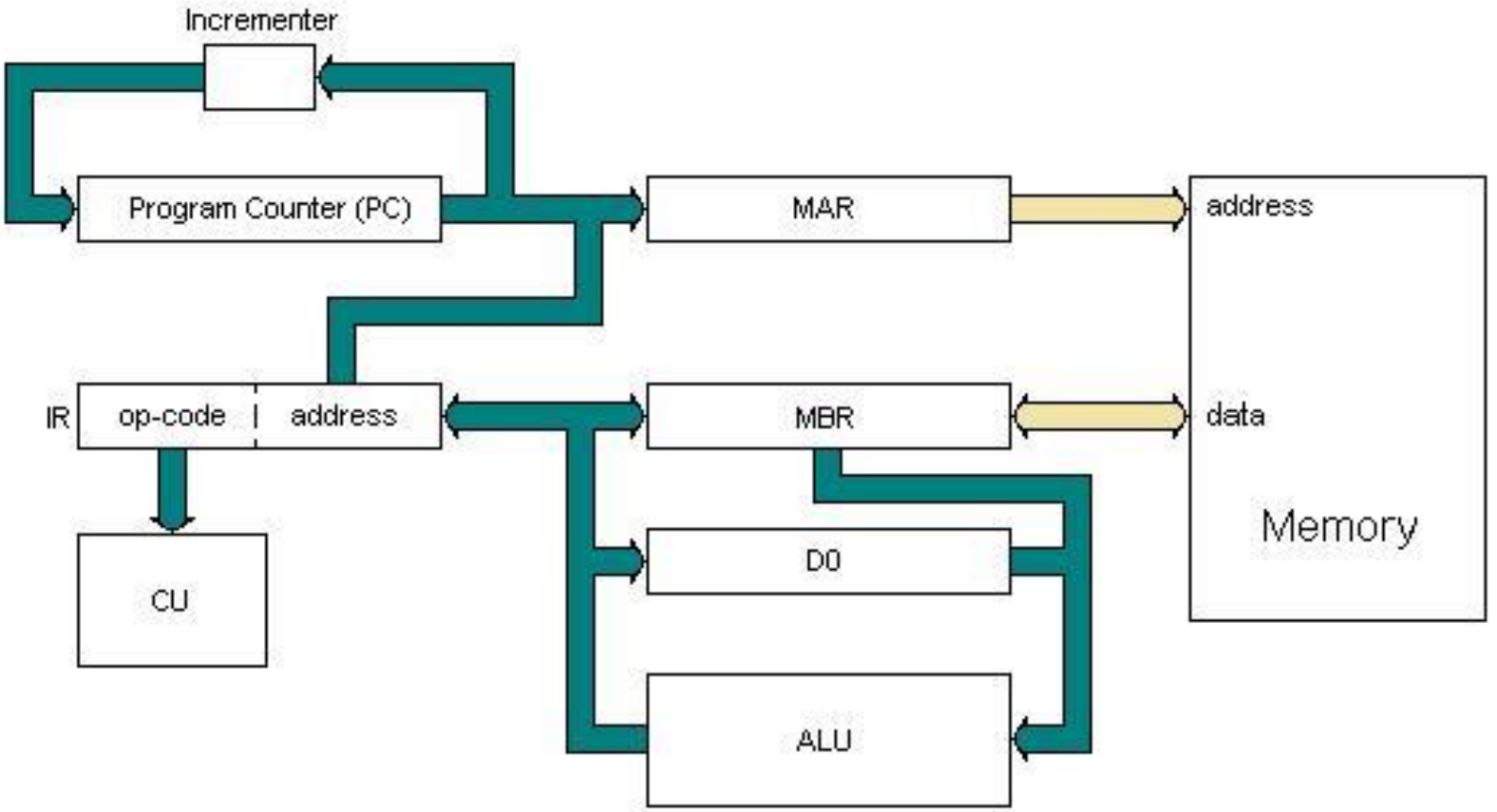
Fetch cycle: step 2

[PC] <- [PC] + 1



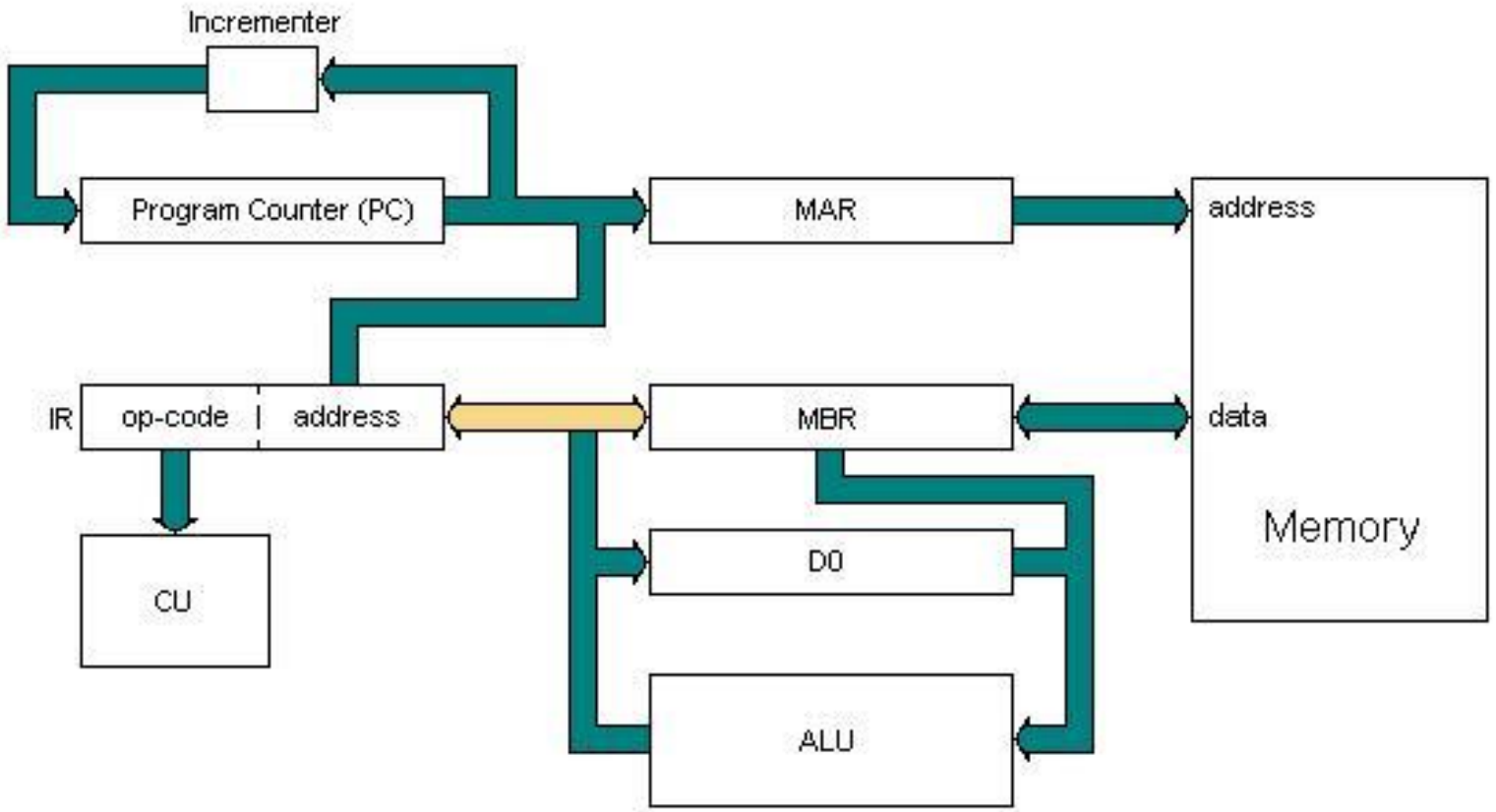
Fetch cycle: step 3

[MBR] <- [M([MAR])]



Fetch cycle: step 4

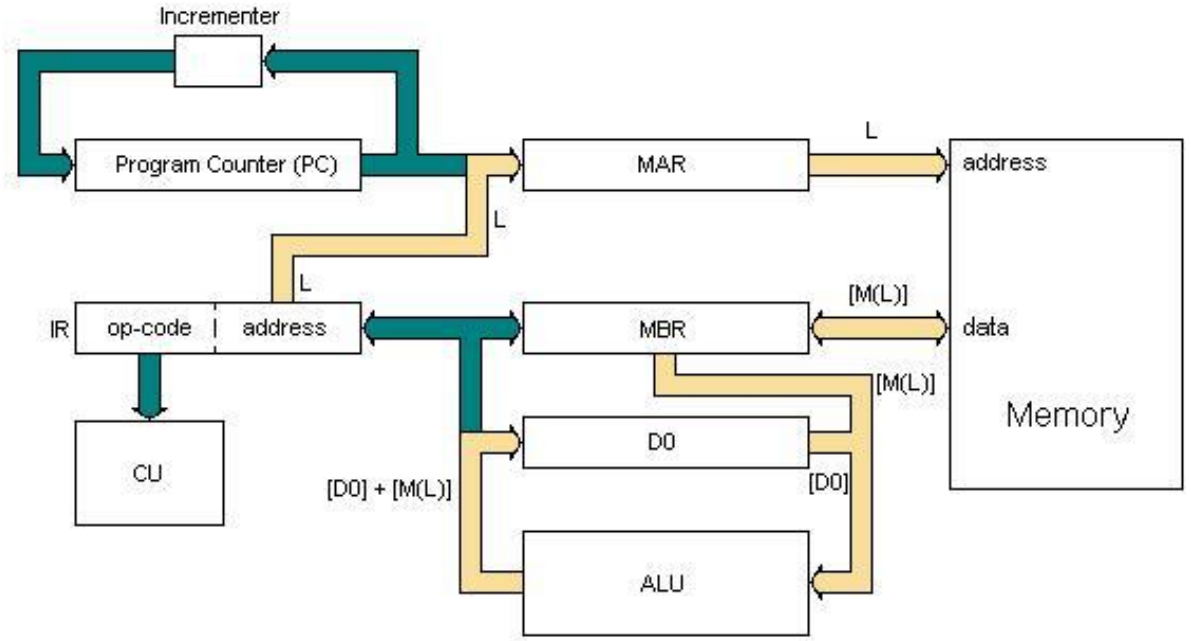
[IR] <- [MBR]



Κύκλος Εκτέλεσης – Execution Cycle

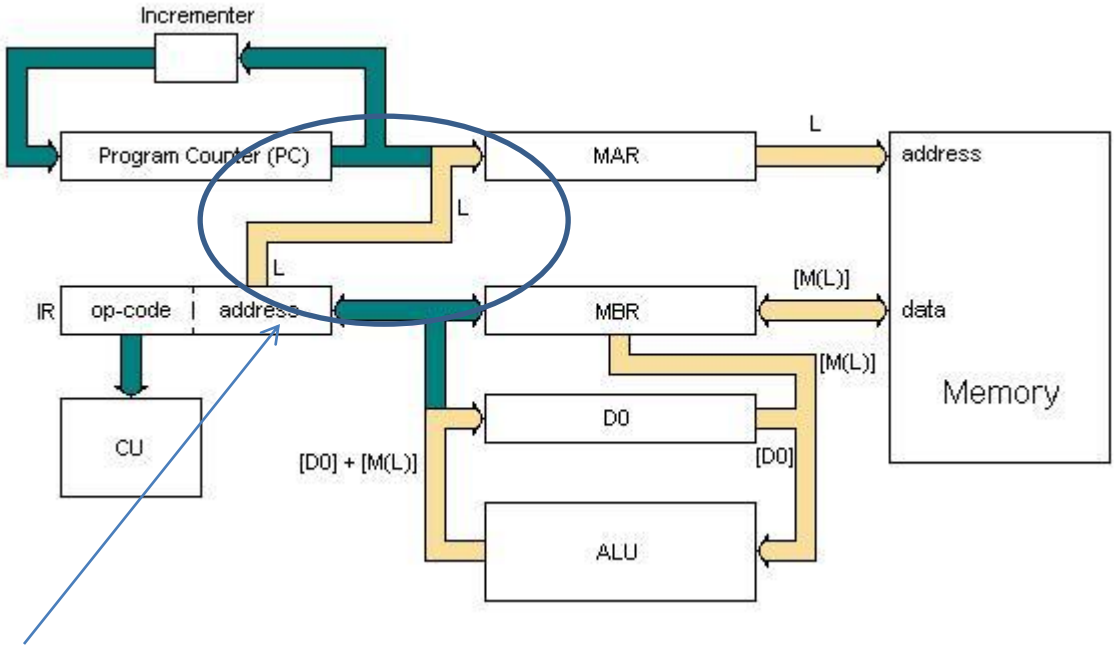
Ας υποθέσουμε ότι η εντολή που μόλις τραβήχτηκε από τη μνήμη είναι:
Προσθέστε τα περιεχόμενα της θέσης μνήμης L στα περιεχόμενα του καταχωρητή δεδομένων D0 και αποθηκεύστε το αποτέλεσμα στο D0.

Η μορφή της γλώσσας Assembly αυτής της εντολής είναι: **ADD.L (L),D0**



Η μορφή RTL αυτής της εντολής θα είναι: **[D0] <- [D0] + [M (L)]**

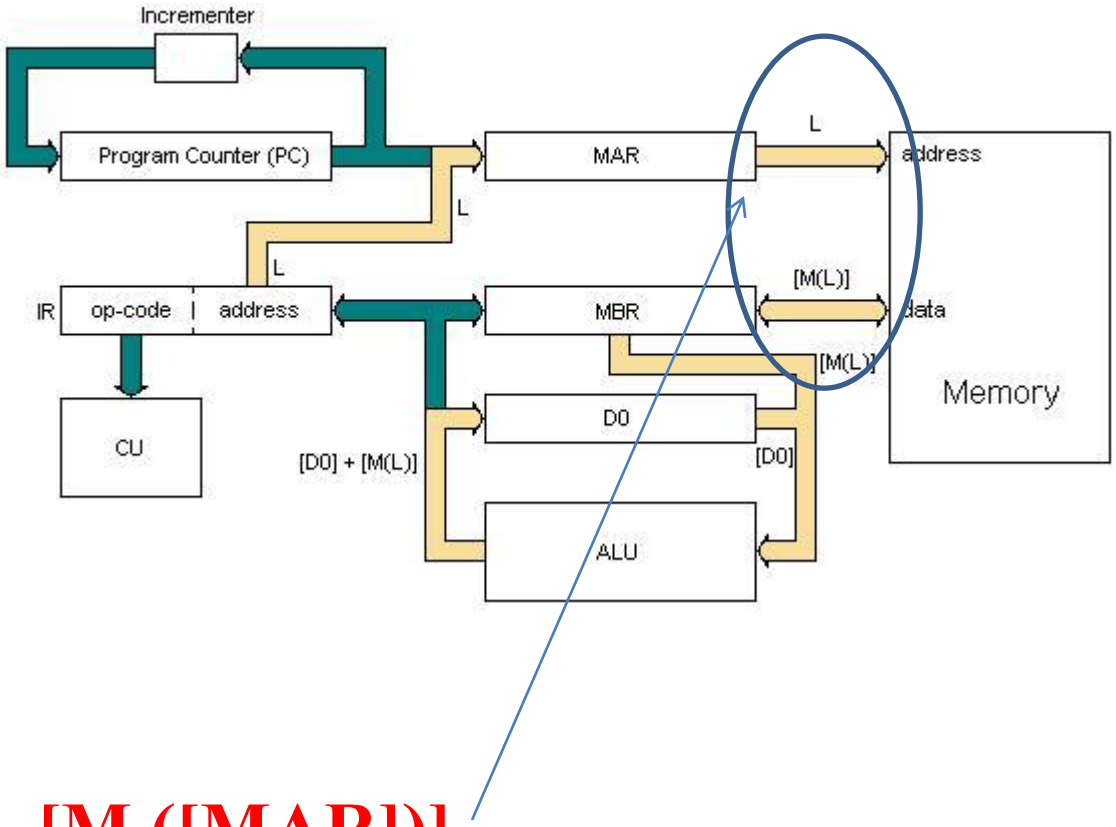
Κύκλος Εκτέλεσης – Execution Cycle



[MAR] <- [IR (Πεδίο_διεύθυνσης)]

Το πρώτο βήμα του κύκλου εκτέλεσης είναι επομένως η φόρτωση του **MAR** με την **L**, η οποία είναι η διεύθυνση του τελεστή στη μνήμη.
Το **L** αποθηκεύεται στο πεδίο διεύθυνσης του IR.

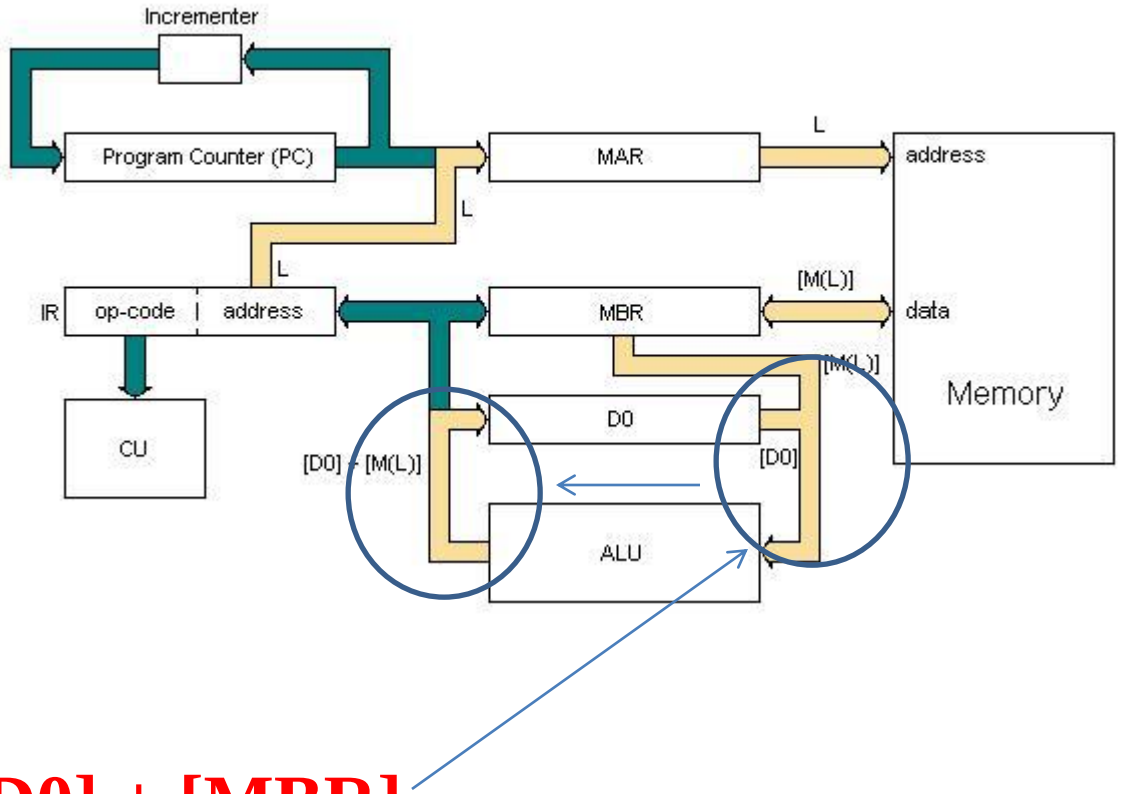
Κύκλος Εκτέλεσης – Execution Cycle



[MBR] <- [M ([MAR])]

Το δεύτερο βήμα είναι να διαβάσει τα δεδομένα από τη θέση μνήμης, η διεύθυνση του οποίου δίνεται από τα περιεχόμενα του **MAR**, και να τα αποθηκεύσει στον **MBR.**

Κύκλος Εκτέλεσης – Execution Cycle



[D0] <- [D0] + [MBR]

Τώρα που έχουμε την τιμή του τελεστή, το μόνο που έχει απομείνει είναι να εκτελέσει την πρόσθεση και να αποθηκεύσουμε το αποτέλεσμα στο καταχωρητή δεδομένων D0: **αυτό είναι το τρίτο βήμα**.

Κύκλος Φόρτωσης/Εκτέλεσης
Fetch/Execute cycle

FETCH

[MAR] <- [PC]
[PC] <- [PC] + 1
[MBR] <- [M([MAR])]
[IR] <- [MBR]

EXECUTE

[MAR] <- [IR(Address_Field)]
[MBR] <- [M([MAR])]
[D0] <- [D0] + [MBR]

Ανακεφαλαίωση

- Οι γλώσσες προγραμματισμού των υπολογιστών είναι οργανωμένες σε μια ιεραρχική δομή. Η γλώσσα Assembly είναι στο χαμηλότερο επίπεδο (low level) της ιεραρχίας αυτής.
- Τα προγράμματα γλωσσών υψηλού επιπέδου (high level) είναι ουσιαστικά ανεξάρτητα από το σύστημα, δηλαδή μπορούν να τρέξουν σχεδόν σε οποιονδήποτε υπολογιστή, ενώ τα προγράμματα γλώσσας Assembly εξαρτώνται πολύ από το σύστημα, δηλαδή ένα πρόγραμμα γλώσσας Assembly γραμμένο για έναν επεξεργαστή μπορεί να εκτελεστεί μόνο σε αυτόν τον επεξεργαστή ή σε μια ομάδα/οικογένεια από στενά συνδεδεμένους συμβατούς επεξεργαστές. Ωστόσο, η φιλοσοφία πίσω από το σχεδιασμό ενός προγράμματος γλώσσας Assembly είναι η ίδια από το ένα σύστημα στο άλλο.
- Η γλώσσα Assembly είναι μια αναγνώσιμη από τον άνθρωπο αναπαράσταση του κώδικα μηχανής με τον οποίο δουλεύουν οι υπολογιστές. Στη γλώσσα Assembly, οι δυαδικά κωδικοποιημένες εντολές αντικαθίστανται από μνημονικά και οι διευθύνσεις και οι σταθερές συνήθως γράφονται σε συμβολική μορφή (όπως και σε γλώσσες υψηλού επιπέδου). Η γλώσσα Assembly είναι η "μητρική" γλώσσα του μηχανήματος, ο φυσικός τρόπος να μιλάμε με τον επεξεργαστή.
- Τίποτα δεν αντικατοπτρίζει καλύτερα την αρχιτεκτονική και την οργάνωση ενός συστήματος υπολογιστή από τη γλώσσα Assembly του.

Ανακεφαλαίωση

- Η υπολογιστική μηχανή Von Neumann βασίζεται στο σύστημα κοινής μνήμης που αποθηκεύει τόσο τις εντολές (instructions) που καθορίζουν την πορεία εκτέλεσης του προγράμματος του υπολογιστή όσο και τα δεδομένα (data) που επεξεργάζονται από αυτές τις εντολές.
- Η μνήμη ενός υπολογιστή είναι ένας πίνακας ή μια συλλογή πολλών μονάδων/θέσεων αποθήκευσης που ονομάζονται θέσεις μνήμης. Κάθε θέση μνήμης έχει μια μοναδική διεύθυνση που χρησιμοποιείται για να αναφέρεται σε αυτήν τη θέση και τα περιεχόμενά της.
- Η Κεντρική Μονάδα Επεξεργασίας (CPU), που ονομάζεται επίσης επεξεργαστής, είναι υπεύθυνη για την ανάγνωση/ανάκληση (fetch) εντολών από τη μνήμη και την εκτέλεσή τους (execution). Είναι ο «εγκέφαλος» του υπολογιστή. Συνδέεται στη μνήμη μέσω τριών διαύλων (bus): του διαύλου διευθύνσεων, του διαύλου δεδομένων και του διαύλου ελέγχου.
- Ο δίαυλος διευθύνσεων μεταδίδει τη διεύθυνση της θέσης μνήμης στην οποία η CPU θέλει να έχει πρόσβαση. Ο δίαυλος δεδομένων χρησιμοποιείται για την μεταφορά δεδομένων από την CPU στη μνήμη και αντίστροφα. Ο δίαυλος ελέγχου χρησιμοποιείται από τη CPU για να ενημερώσει το σύστημα μνήμης εάν λαμβάνει χώρα ένας κύκλος ανάγνωσης (read cycle) ή ένας κύκλος εγγραφής (write cycle). Κατά τη διάρκεια ενός κύκλου ανάγνωσης, τα δεδομένα διαβάζονται από τη μνήμη και αποστέλλονται στη CPU, και κατά τη διάρκεια ενός κύκλου εγγραφής, τα δεδομένα αποστέλλονται από τη CPU στη μνήμη που πρόκειται να αποθηκευτεί.
- Ένα πρόγραμμα είναι μια ακολουθία εντολών με σχετικά ορίσματα. Αυτές οι εντολές και τα σχετικά ορίσματά τους μεταφράζονται σε δυαδικό κώδικα μηχανής (συνήθως αναπαρίστανται με δεκαεξαδικούς αριθμούς) που μπορεί να κατανοήσει ο υπολογιστής και αποθηκεύονται στη μνήμη.

Ανακεφαλαίωση

- Η CPU αποτελείται από:
  - Καταχωρητές (Registers), συνήθως είναι οργανωμένοι σε Αρχεία Καταχωρητών (Register files). Οι καταχωρητές είναι ειδικές, μικρές συσκευές υψηλής ταχύτητας που χρησιμοποιούνται για την αποθήκευση πληροφοριών. Το γεγονός ότι οι καταχωρητές βρίσκονται στην CPU μειώνει σημαντικά τον χρόνο που απαιτείται για την πρόσβαση στο περιεχόμενό τους. Οι καταχωρητές χρησιμοποιούνται για χώρο αποθήκευσης κατά την εκτέλεση ενός προγράμματος, επειδή είναι πολύ πιο εύχρηστο και πιο γρήγορο να χειριστείς αριθμούς που είναι αποθηκευμένοι σε καταχωρητές από τους ίδιους αριθμούς που είναι αποθηκευμένοι στη μνήμη. Μόνο τα προγράμματα της γλώσσας Assembly μπορούν να χειριστούν καταχωρητές. Υπάρχουν διάφοροι τύποι Καταχωρητών: μερικοί από αυτούς είναι καταχωρητές γενικού σκοπού, ενώ μερικοί από αυτούς προορίζονται για συγκεκριμένες εργασίες.
  - την Αριθμητική & Λογική Μονάδα (ALU) που εκτελεί υπολογισμούς και συγκρίσεις.
  - την Μονάδα Ελέγχου (CU) που είναι η μονάδα λήψης αποφάσεων. Ερμηνεύει το μοτίβο bit μιας εντολής και αποφασίζει ποιες ενέργειες πρέπει να γίνουν.

Ανακεφαλαίωση

- Ο τρόπος με τον οποίο μια CPU τύπου μηχανής Von Neumann εκτελεί μια εντολή είναι η παρακάτω:
  - Όταν μια εντολή διαβάζεται από τη μνήμη και αποθηκεύεται στον καταχωρητή εντολών (IR), στη συνέχεια αποκωδικοποιείται από την CPU από τον αποκωδικοποιητή εντολών (ID) και ενημερώνεται η μονάδα ελέγχου (CU).
  - Η CPU εξετάζει τον κωδικό λειτουργίας της εντολής (opcode), ο οποίος λέει ποια είναι η λειτουργία της.
  - Στη συνέχεια, εάν η εντολή χρειάζεται πρόσθετα δεδομένα από τη μνήμη, η CPU διαβάζει ξανά τη μνήμη στη διεύθυνση που καθορίζεται στα πεδία διεύθυνσης τελεστών (operands).
  - Αφού η CPU έχει όλα τα δεδομένα που χρειάζεται, μπορεί να εκτελέσει τη λειτουργία που καθορίζεται από τον κωδικό λειτουργίας. Εάν η εντολή απαιτεί την εγγραφή δεδομένων στη μνήμη, π.χ. εάν το αποτέλεσμα μιας αφαίρεσης πρέπει να αποθηκευτεί, τότε τα δεδομένα αποθηκεύονται στη μνήμη.
- Το μονοπάτι που ενώνει τη CPU και τη μνήμη έχει ονομαστεί συμφόρηση Von Neumann. Το σημείο συμφόρησης Von Neumann είναι ένας από τους περιοριστικούς παράγοντες της μηχανής Von Neumann, καθώς το πλάτος της διαδρομής πληροφοριών θέτει ένα όριο στον μέγιστο δυνατό όγκο και ταχύτητα ροής πληροφοριών μεταξύ της CPU και της μνήμης.

Τι είναι η «Αρχιτεκτονική Μικροεπεξεργαστή»

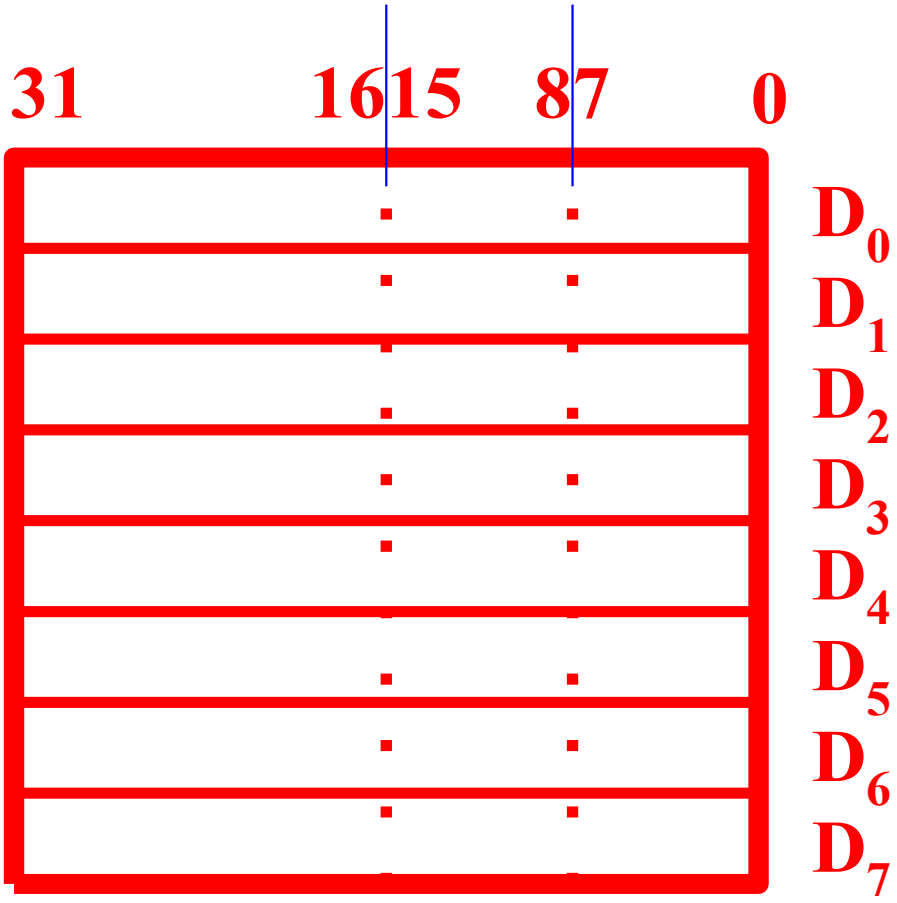
- Για τη δική μας δουλειά **η Αρχιτεκτονική είναι το λογισμικό ή το μοντέλο προγραμματισμού** του επεξεργαστή, δηλαδή:
  - Οι **καταχωρητές (Registers)** της ΚΜΕ που είναι διαθέσιμοι στον προγραμματιστή.
  - Οι βασικές **εντολές (Commands)** που μπορεί να εκτελέσει η ΚΜΕ.
  - Οι τρόποι που οι εντολές μπορούν **να καθορίσουν μια θέση μνήμης (Addressing Modes)**.
  - Ο τρόπος **που οργανώνεται η πληροφορία στη μνήμη (Memory and Data Organization).**
  - Πως η ΚΜΕ αποκτά πρόσβαση και ελέγχει τις **περιφερειακές συσκευές (Peripheral Devices).**

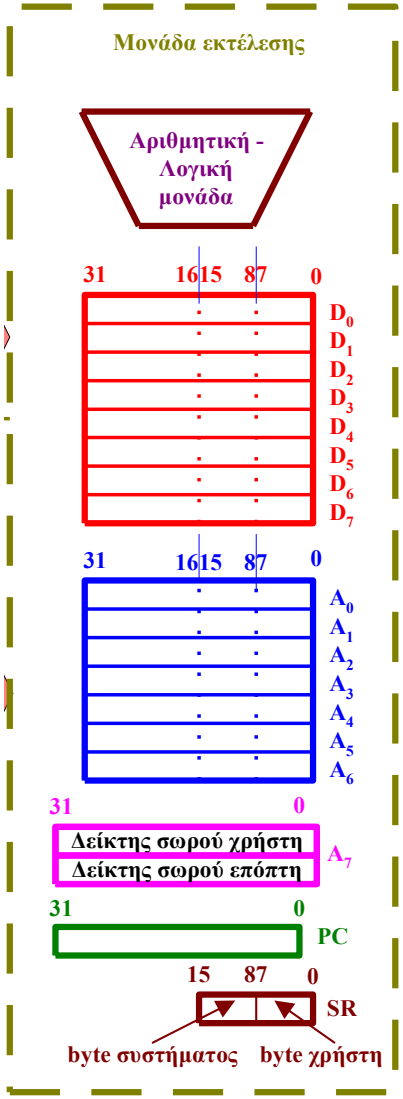
Καταχωρητές του M68000

Οι οκτώ καταχωρητές διευθύνσεων *A0-A7* γενικής χρήσης με μήκος τριάντα δύο (32) ψηφίων, που κρατούν πληροφορίες διεύθυνσης.

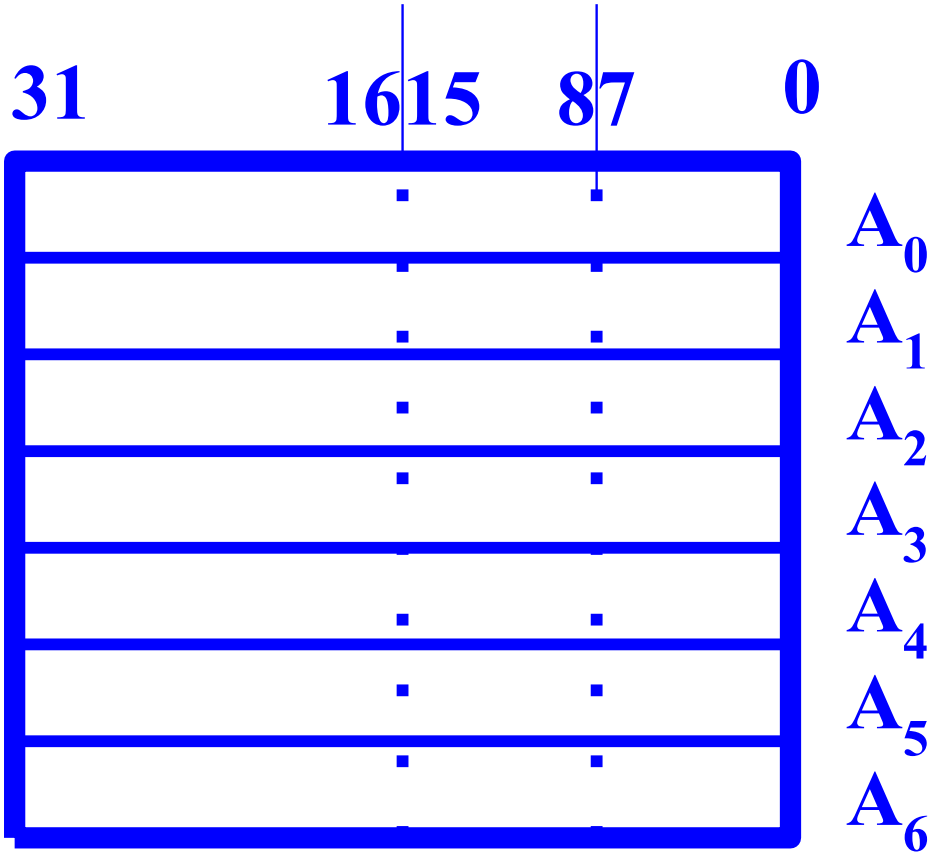
Οι οκτώ καταχωρητές δεδομένων *D0-D7* έχουν μήκος τριάντα δύο (32) ψηφίων και είναι καταχωρητές γενικής χρήσης.

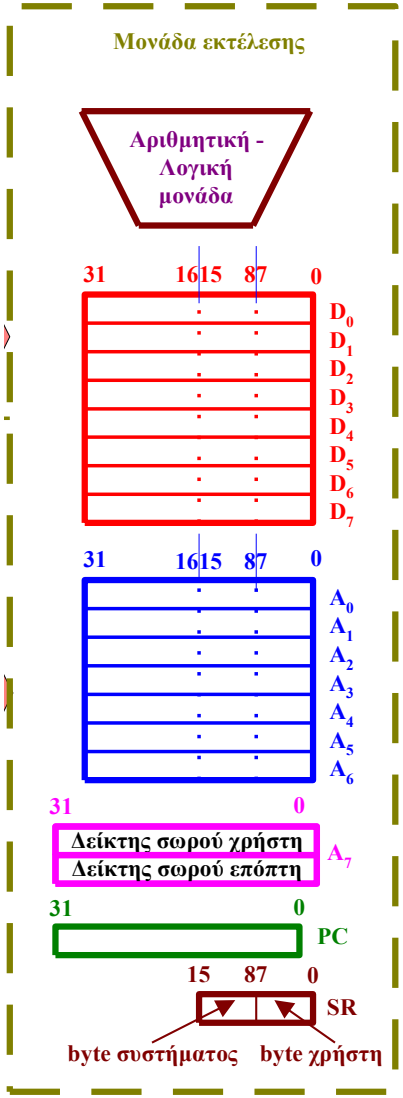
Καταχωρητές δεδομένων (Data Register)





Καταχωρητές διευθύνσεων (Address Registers)



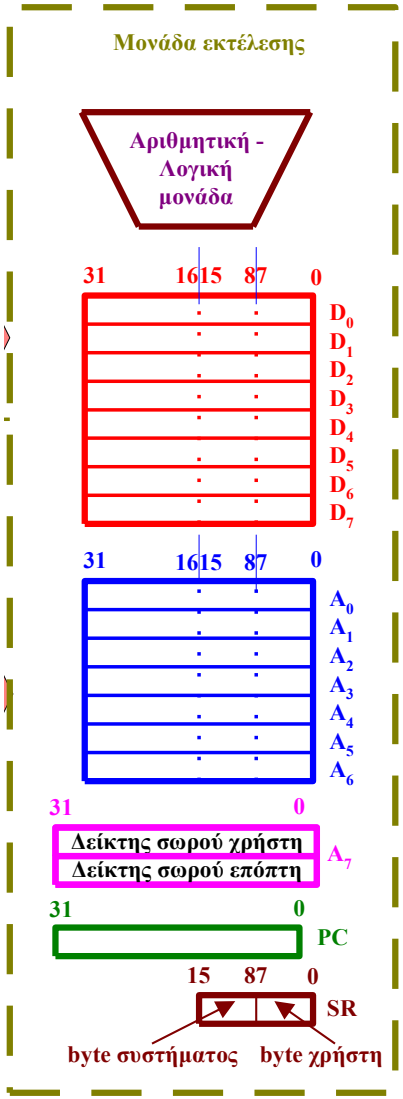


Καταχωρητές διευθύνσεων (A7)

d31 d0

Δείκτης σωρού χρήστη
Δείκτης σωρού επόπτη

A_7



Καταχωρητές διευθύνσεων

Οι δύο καταχωρητές δείκτες σωρού ονομάζονται δείκτης σωρού χρήστη (user stack pointer) και δείκτης σωρού επόπτη (supervisor stack pointer). Επειδή μόνον ένας από τους καταχωρητές αυτούς είναι ενεργοποιημένος κάθε φορά, αυτοί φαίνονται ως ένας καταχωρητής SP (*A7*).

Ο δείκτης σωρού χρήστη είναι ενεργοποιημένος όταν ο 68000 λειτουργεί σε κατάσταση χρήστη και δείχνει την κορυφή του σωρού χρήστη, που υπάρχει στο μέρος της μνήμης χρήστη, ενώ ο δείκτης σωρού επόπτη είναι ενεργοποιημένος όταν ο 68000 λειτουργεί σε κατάσταση επόπτη και δείχνει την κορυφή του σωρού επόπτη, που υπάρχει στο μέρος της μνήμης επόπτη.

Καταχωρητές διευθύνσεων

Οι δείκτες σωρού χρησιμοποιούνται για να δείχνουν τη διεύθυνση στο σωρό στην οποία μπορούν να αποθηκευτούν προσωρινά περιεχόμενα δεδομένων, διευθύνσεων επιστροφής ή άλλες παράμετροι που χρησιμοποιούνται σε λειτουργίες όπως η κλήση υπορουτίνας.

Επιπλέον, ο δείκτης σωρού επόπτη χρησιμοποιείται από τις κλήσεις επόπτη όπως εξαιρέσεις λογισμικού, διακοπές και εσωτερικές εξαιρέσεις.

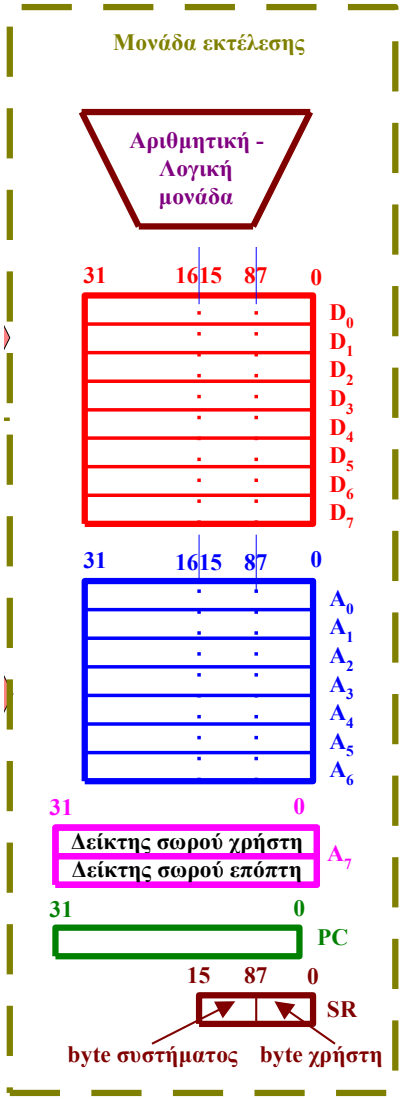
Μετρητής Προγράμματος (PC - Program Counter)

Ο Μετρητής Προγράμματος κρατά τη διεύθυνση που δείχνει την επόμενη προς εκτέλεση εντολή και αυτόματα αυξάνει κατά 2 (τουλάχιστον) το περιεχόμενό του κάθε φορά που γίνεται ανάκληση μιας εντολής.

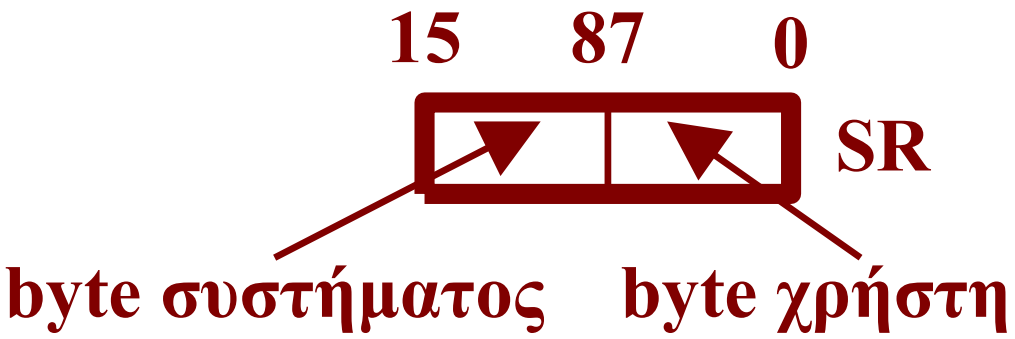
Με τον τρόπο αυτό δείχνει πάντα στην επόμενη λέξη μιας εντολής πολλών λέξεων, που είναι ο άμεσος τελεστέος προέλευσης, ή στην επόμενη εντολή του προγράμματος.

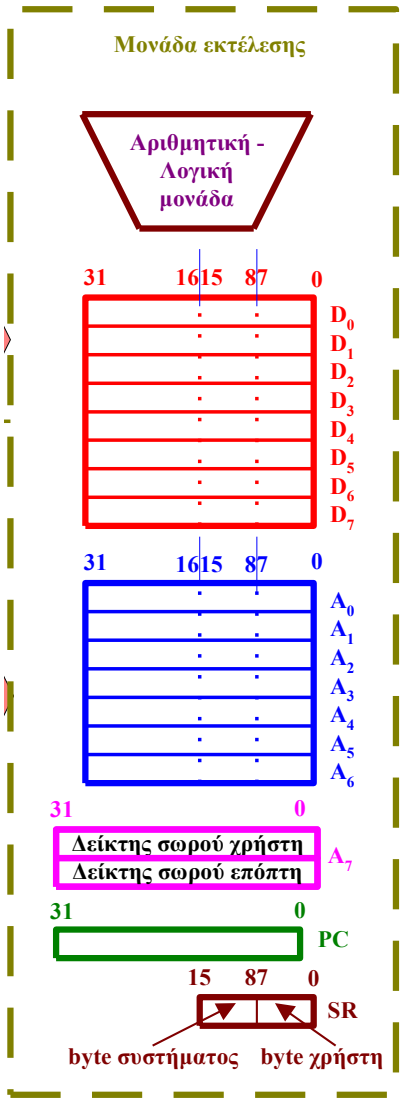
Επειδή, η εντολή ενός προγράμματος ξεκινά από άρτιο αριθμό διεύθυνσης στη μνήμη, ο PC δείχνει πάντα σε άρτιο αριθμό.

Μετρητής Προγράμματος

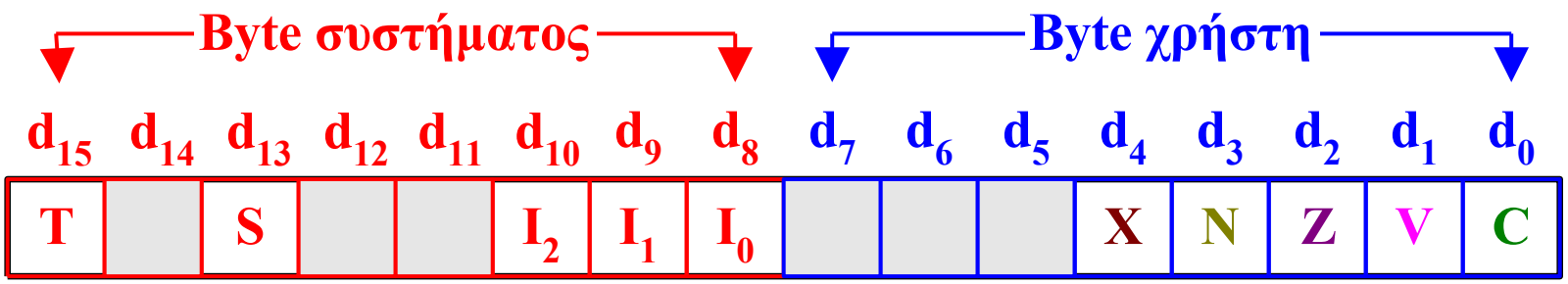


Καταχωρητής κατάστασης (Status Register - SR)



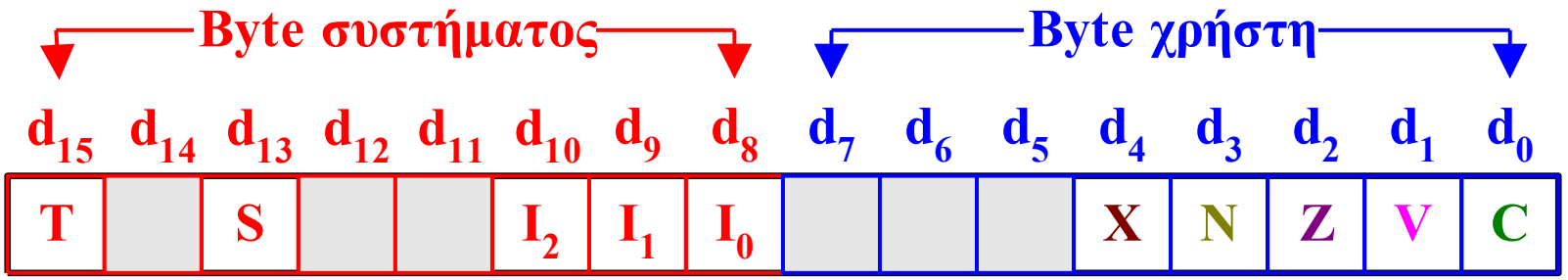


Καταχωρητής κατάστασης



Ο δείκτης κρατουμένου *C* γίνεται "1" όταν η προηγούμενη πρόσθεση έδωσε κρατούμενο ή η προηγούμενη αφαίρεση (ή σύγκριση) χρειάστηκε δανικό. Διαφορετικά παίρνει την τιμή "0". Επίσης, κρατά το ψηφίο που βγαίνει από έναν καταχωρητή κατά την εκτέλεση των εντολών ολίσθησης και περιστροφής.

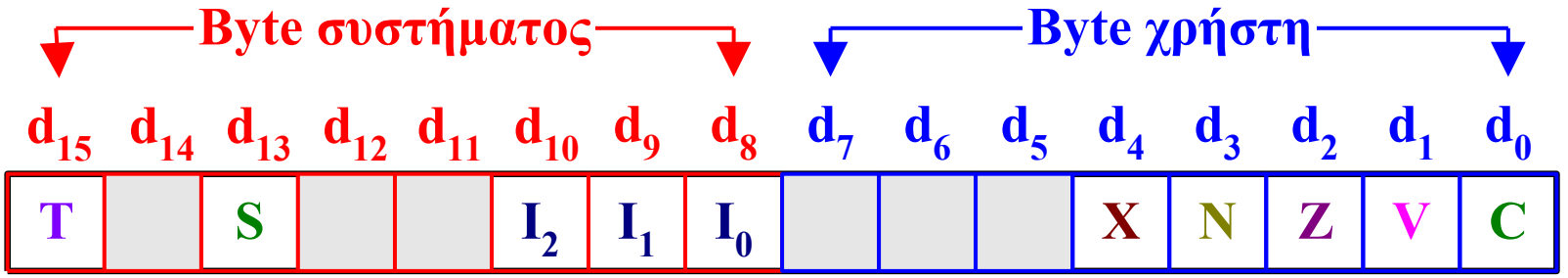
Ο δείκτης υπερχείλισης *V* γίνεται "1" όταν μια προσημασμένη αριθμητική πράξη δίνει λάθος αποτέλεσμα. Διαφορετικά παίρνει την τιμή "0". Επίσης, ο δείκτης υπερχείλισης γίνεται "1" αν μετά την εκτέλεση μιας εντολής αριθμητικής ολίσθησης αλλάζει το περισσότερο σημαντικό ψηφίο. Διαφορετικά παίρνει την τιμή "0".



Ο δείκτης μηδενός *Z* παίρνει την τιμή "1" αν το αποτέλεσμα της προηγούμενης πράξης είναι μηδέν. Διαφορετικά παίρνει την τιμή "0".

Ο δείκτης άρνησης *N* παίρνει την τιμή "1" αν το αποτέλεσμα της προηγούμενης πράξης είναι αρνητικός αριθμός. Διαφορετικά παίρνει την τιμή "0".

Ο δείκτης επέκτασης *X* παίρνει την τιμή του δείκτη κρατουμένου και χρησιμοποιείται ως το ψηφίο κρατουμένου για τις πράξεις πολλαπλής ακρίβειας και πολλαπλασιασμού.

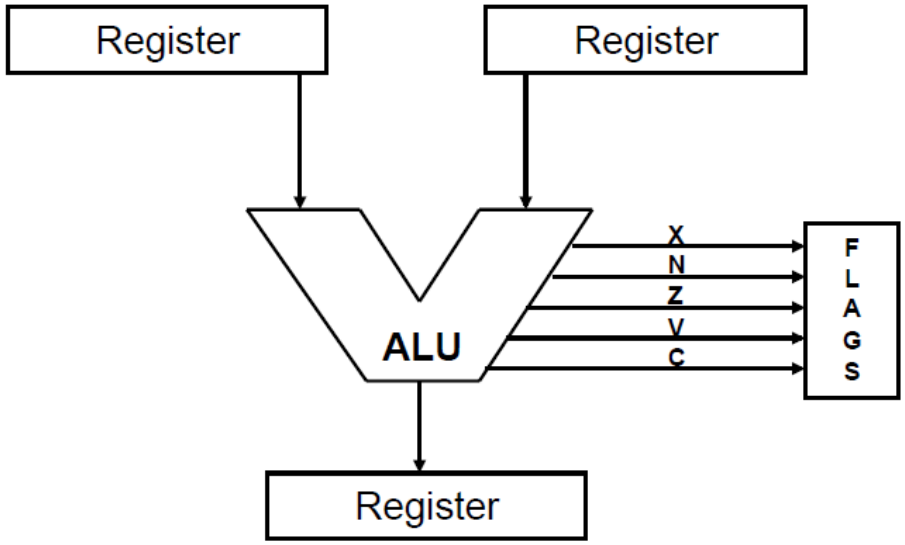


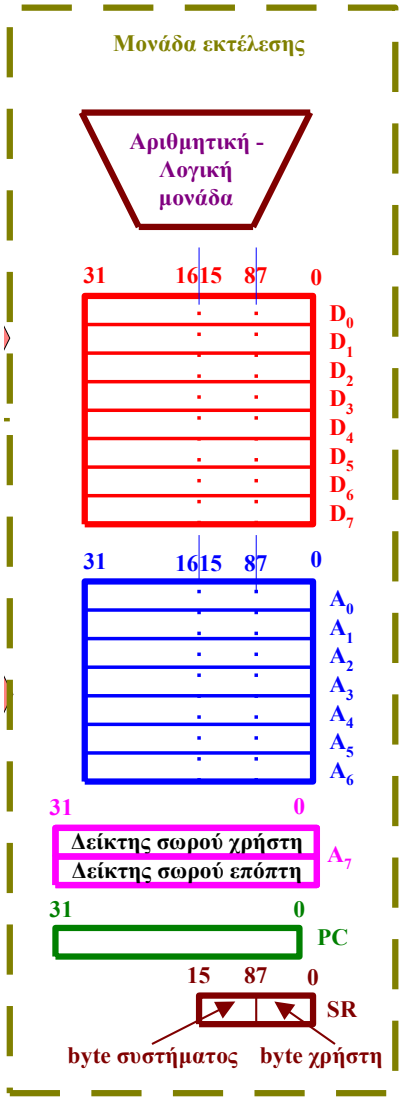
Ο δείκτης μάσκας διακοπής αποτελείται από τρία ψηφία $I_2I_1I_0$ ο κώδικας των οποίων καθορίζει ποια από τις αιτήσεις για διακοπή θα εξυπηρετηθεί και ποια όχι. Οι αιτήσεις για διακοπή που έχουν τιμή μεγαλύτερη από εκείνη που ορίζεται από τα ψηφία $I_2I_1I_0$ θα εξυπηρετηθούν ενώ οι άλλες θα αγνοηθούν.

Ο δείκτης επόπτη *S* χρησιμοποιείται για να επιλέξει την κατάσταση λειτουργίας του μικροεπεξεργαστή. Αν δηλαδή ο μικροεπεξεργαστής θα λειτουργεί σε κατάσταση χρήστη οπότε παίρνει την τιμή "0" (clear) ή κατάσταση επόπτη οπότε παίρνει την τιμή "1" (set).

Ο δείκτης *T* παίρνει την τιμή "1" όταν το πρόγραμμα εκτελείται βήμα προς βήμα. Διαφορετικά παίρνει την τιμή "0".

Μονάδα Εκτέλεσης (Execution Unit)





Κυκλώματα ελέγχου εκτέλεσης εντολών (Control Unit)

Ο καταχωρητής εντολών (IR – Instruction Register) κρατά τους κωδικούς των εντολών που ανακαλούνται απ' τη μνήμη κατά την εκτέλεση του προγράμματος.

Ο αποκωδικοποιητής εντολών (ID – Instruction Decoder), που επικοινωνεί με τον καταχωρητή εντολών, αναλαμβάνει την αποκωδικοποίηση των εντολών και επικοινωνεί με τη μονάδα ελέγχου (CU – Control Unit) και τη μονάδα εκτέλεσης (EU – Execution Unit).

Η πληροφορία που στέλνεται στη μονάδα εκτέλεσης ονομάζεται στατική μικροεντολή (micro-instruction static) γιατί δεν εξαρτάται από το χρονισμό εκτέλεσης της εντολής.

Ταυτόχρονα, ο αποκωδικοποιητής τροφοδοτεί τη μονάδα ελέγχου με μια διεύθυνση εκκίνησης μικροακολουθίας (micro-sequence starting address). Έτσι, η μονάδα ελέγχου είναι υπεύθυνη να καθορίσει την ακολουθία με την οποία θα γίνονται οι επιμέρους λειτουργίες που εκτελούνται από τη μονάδα εκτέλεσης και προκαλούν την εκτέλεση της λειτουργίας που περιγράφεται από την εντολή.

Τι είναι η «Αρχιτεκτονική Μικροεπεξεργαστή»

- Για τη δική μας δουλειά **η Αρχιτεκτονική είναι το λογισμικό ή το μοντέλο προγραμματισμού** του επεξεργαστή, δηλαδή:
  - Οι **καταχωρητές (Registers)** της ΚΜΕ που είναι διαθέσιμοι στον προγραμματιστή.
  - Οι βασικές **εντολές (Commands)** που μπορεί να εκτελέσει η ΚΜΕ.
  - Οι τρόποι που οι εντολές μπορούν **να καθορίσουν μια θέση μνήμης (Addressing Modes)**.
  - Ο τρόπος **που οργανώνεται η πληροφορία στη μνήμη (Memory and Data Organization).**
  - Πως η ΚΜΕ αποκτά πρόσβαση και ελέγχει τις **περιφερειακές συσκευές (Peripheral Devices).**

Σετ Εντολών – Instruction Set

• Το σύνολο των εντολών που υποστηρίζει ένα επεξεργαστής ονομάζεται **Σετ Εντολών (Instruction Set)**.

• Ο M68000 έχει ένα σύνολο **56 εντολών** που μπορείτε να χρησιμοποιήσετε (Instructions - Commands). Οι εντολές αυτές (Executable Instructions) είναι αυτές του μεταφράζονται από τον Assembler σε κώδικα μηχανής.

• Μπορούν να είναι εντολές που αλλάζουν τα δεδομένα, ορίζουν από που ο επεξεργαστής διαβάζει ή γράφει, και πολλά άλλα πράγματα.

• Οι εντολές κατηγοριοποιούνται σύμφωνα με τη βασική τους λειτουργία:

– Μεταφοράς Δεδομένων (Data Transfer)
– Αριθμητικές (Arithmetic)
– Λογικής (Logic)
– Ολίσθησης (Shifts & Rotates)
– Επεξεργασίας ψηφίων (Bit Manipulation)
– BCD
– Ελέγχου Προγράμματος (Program Control)
– Ελέγχου Συστήματος (System Control)

Μνημονικό	Περιγραφή	Μνημονικό	Περιγραφή
ABCD	Add decimal with extend	**MOVE**	Move source to destination
ADD	Add binary	**MULS**	Sign multiply
AND	Logical AND	**MULU**	Unsigned multiply
ASL	Arithmetic shift left	**NBCD**	Negate decimal with extend
ASR	Arithmetic shift right	**NEG**	Negate
Bcc	Branch conditionally	**NOP**	No operation
BCHG	Bit test and change	**NOT**	One's complement
BCLR	Bit test and clear	**OR**	Logical OR
BRA	Branch always	**PEA**	Push effective address
BSET	Bit test and set	**RESET**	Reset external devices
BSR	Branch to subroutine	**ROL**	Rotate left
BTST	Bit test	**ROR**	Rotate right
CHK	Check register with bounds	**ROXL**	Rotate left through extend
CLR	Clear operand	**ROXR**	Rotate right through extend
CMP	Compare	**RTE**	Return from exception
DBcc	Decrement and branch conditionally	**RTR**	Return and restore
DIVS	Signed divide	**RTS**	Return from subroutine
DIVU	Unsigned divide	**SBCD**	Subtract decimal with extend
EOR	Exclusive OR	**Scc**	Set conditionally
EXG	Exchange registers	**STOP**	Stop processor
EXT	Sign extend	**SUB**	Subtract binary
JMP	Jump to effective address	**SWAP**	Swap data register halves
JSR	Jump to subroutine	**TAS**	Test and set operand
LEA	Load effective address	**TRAP**	Trap
LINK	Link stack	**TRAPV**	Trap on overflow
LSL	Logical shift left	**TST**	Test
LSR	Logical shift right	**UNLK**	Unlink stack

Κάποιες βασικές εντολές

Εντολή	RTL Περιγραφή	Περιγραφή
MOVE #N,D1	[D1] <- N	Register D1 is loaded with the number N.
MOVE D1,L	[M(L)]<- [D1]	The contents of register D1 are copied to memory location L.
MOVE D1,D2	[D2] <- [D1]	The contents of register D1 are copied to register D2.
ADD D3,D7	[D7] <- [D3] + [D7]	The contents of register D3 are added to the contents of register D7, and the result is stored in register D7.
ADD #N,D0	[D0] <- [D0] + N	The number N is added to the contents of register D0 and the result is stored in D0.
SUB #N,D0	[D0] <- [D0] - N	The number N is subtracted from the contents of register D0 and the result is stored in D0.
SUB D1,D5	[D5] <- [D5] - [D1]	The contents of register D1 are subtracted from the contents of register D5, and the result is stored in register D5.
CMP #N,D2	[D2] - N	Subtract the number N from the contents of register D2. The result is discarded and the CCR is set up.
CMP D1,D2	[D2] - [D1]	Subtract the contents of D1 from the contents of D2. The result is discarded, and the CCR is set up.
BEQ X	IF CCR(Z) = 1 THEN [PC]<- X	Branch to location X if the Z bit of the CCR is set, i.e. if the previous operation yielded zero as result.
BNE X	IF CCR(Z) = 0 THEN [PC]<- X	Branch to location X if the Z bit of the CCR is cleared, i.e. if the previous operation didn't yield zero as result.

Μνημονικά - Mnemonics

Στην πληροφορική, οι εντολές διαβάζονται από τον επεξεργαστή σε δυαδικά ψηφία (bits), εδώ είναι ένα παράδειγμα μιας εντολής:

0011 0000 0011 1100 0000 0100 1111 0000

Τα μνημονικά "mnemonics" είναι απλά λέξεις γραμμένες στη γλώσσα των ανθρώπων ώστε να κάνουν πιο εύκολο των προγραμματισμό. Εδώ έχουμε την ίδια εντολή αλλά με μνημονικό:

MOVE.W #$04F0,D0 **(Motorola syntax)**

Μνημονικά - Mnemonics

MOVE.W #$04F0,D0

Μια εντολή Assembly M68000 αποτελείται από τέσσερα (4) μέρη:

Εντολή (Instruction - Command): Υπάρχουν 56 διαθέσιμες όπως: MOVE, ADD, SUB, DIVU, MULU, BRA ή JMP.

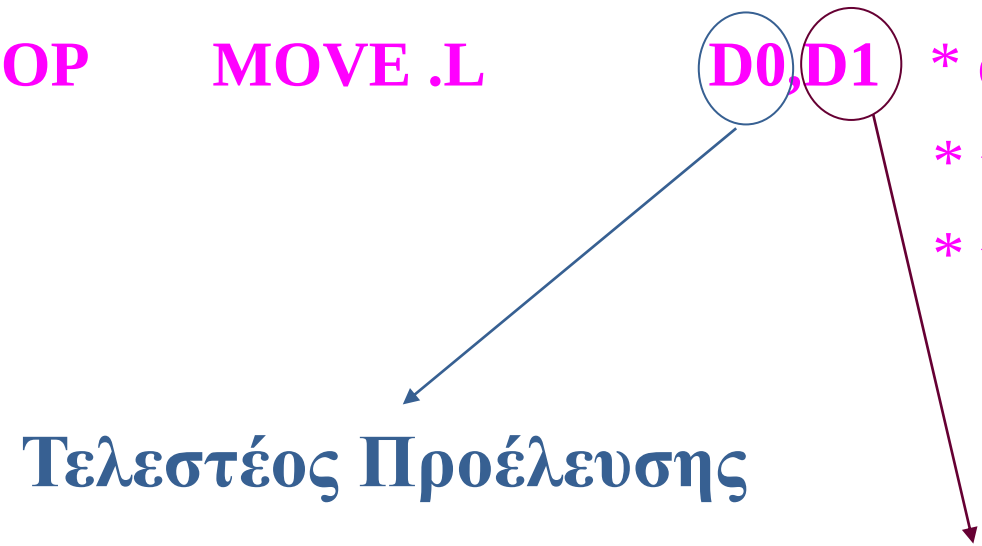
Μέγεθος (Size): Εδώ ορίζουμε τι μέγεθος δεδομένων θα διαχειριστεί η εντολή μας. Έχουμε .B για byte, .W για word, ή .L για long-word.

Τελεστέος Πηγής ή Προέλευσης (Source Operand): Από πού διαβάζεται η τιμή ή ποια τιμή πρόκειται να χρησιμοποιηθεί.

Τελεστέος Προορισμού (Destination Operand): Όπου η τιμή μετακινείται ή επεξεργάζεται/χειρίζεται.

Δείγμα Εντολής του M68000

LABEL (Ετικέτα)	**OPCODE** (Κωδικός Εντολής)	**OPERANDS** (Τελεστέοι)	**COMMENTS** (Σχόλια)
LOOP	**MOVE .L**	**D0,D1**	* αντέγραψε το περιεχόμενο του * καταχωρητή D0 στον * καταχωρητή D1.



Τελεστέος Προέλευσης

Τελεστέος Προόρισμού

Παράδειγμα

MOVE.B #$2C,$0000001E

Offset	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
00000000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	2C	00
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
…																

Τα σύμβολα # και $

Θα έχετε παρατηρήσει επίσης τα σύμβολα # και **$** που εμφανίζονται πριν από τα 2C και 0000001E, το σύμβολο $ λέει στον Assembler ότι ο αριθμός είναι ένας δεκαεξαδικός "hex" αριθμός και όχι δεκαδικός "decimal", αν δεν είχατε το σύμβολο $ για παράδειγμα:
Ο Assembler θα μετέτρεπε το 32 (decimal) σε 0010 0000 (binary) όταν θα το διάβαζε.
Το 0010 0000 είναι το 20 στο hex, επομένως αν γράψουμε 32 είναι το ίδιο με το $20.
Αν θέλετε να γράψετε δυαδικούς αριθμούς μπορείτε να χρησιμοποιήσετε το σύμβολο %.

```
MOVE.B    #%00100000,$0000001E

MOVE.B    #$20,$0000001E

MOVE.B    #32,$0000001E
```

Και τα τρία παραπάνω είναι το ίδιο πράγμα σε δυαδικό (%), σε hex ($) και σε δεκαδικό αντίστοιχα.

Το σύμβολο # από την άλλη πλευρά, είναι για να πείτε στον Assembler ότι ο αριθμός είναι μια "άμεση" τιμή, δηλαδή μια αριθμητική τιμή.

Τι είναι λοιπόν μια "άμεση" τιμή. Κρατήστε αυτή τη σκέψη για μια στιγμή και ας δούμε ένα παράδειγμα χωρίς το σύμβολο #:

MOVE.B $00000010,$0000002D

Αυτό θα διαβάσει το byte που κρατήθηκε στο offset 00000010 και θα το αντιγράψει για να αντισταθμίσει το 0000002D, αν το byte στο offset 00000010 ήταν 49, τότε το 0000002D θα είναι τώρα 49:

Offset	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
00000000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000010	49	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	49	00	00
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
…																

- ✓ **"άμεση" τιμή ή "immediate" ή "literal" είναι μια αριθμητική τιμή.**
- ✓ **Το σύμβολο # λέει στον M68000 ότι ο αριθμός δεν είναι διεύθυνση/offset.**

Τι είναι η «Αρχιτεκτονική Μικροεπεξεργαστή»

- Για τη δική μας δουλειά **η Αρχιτεκτονική είναι το λογισμικό ή το μοντέλο προγραμματισμού** του επεξεργαστή, δηλαδή:
  - Οι **καταχωρητές (Registers)** της ΚΜΕ που είναι διαθέσιμοι στον προγραμματιστή.
  - Οι βασικές **εντολές (Commands)** που μπορεί να εκτελέσει η ΚΜΕ.
  - Οι τρόποι που οι εντολές μπορούν **να καθορίσουν μια θέση μνήμης (Addressing Modes)**.
  - Ο τρόπος **που οργανώνεται η πληροφορία στη μνήμη (Memory and Data Organization).**
  - Πως η ΚΜΕ αποκτά πρόσβαση και ελέγχει τις **περιφερειακές συσκευές (Peripheral Devices).**

Ένα απλό Σύστημα Μνήμης

Ας εξετάσουμε την ιεραρχία ενός συστήματος μνήμης ενός απλού υπολογιστή.

Αποτελείται από:

- **Καταχωρητές (εντός της CPU),**
- **Μνήμη (Cache, SRAM εντός της CPU),**
- **Φυσική RAM εκτός ολοκληρωμένων (DDR) και**
- **συσκευές αποθήκευσης (Σκληρούς δίσκους κλπ.).**

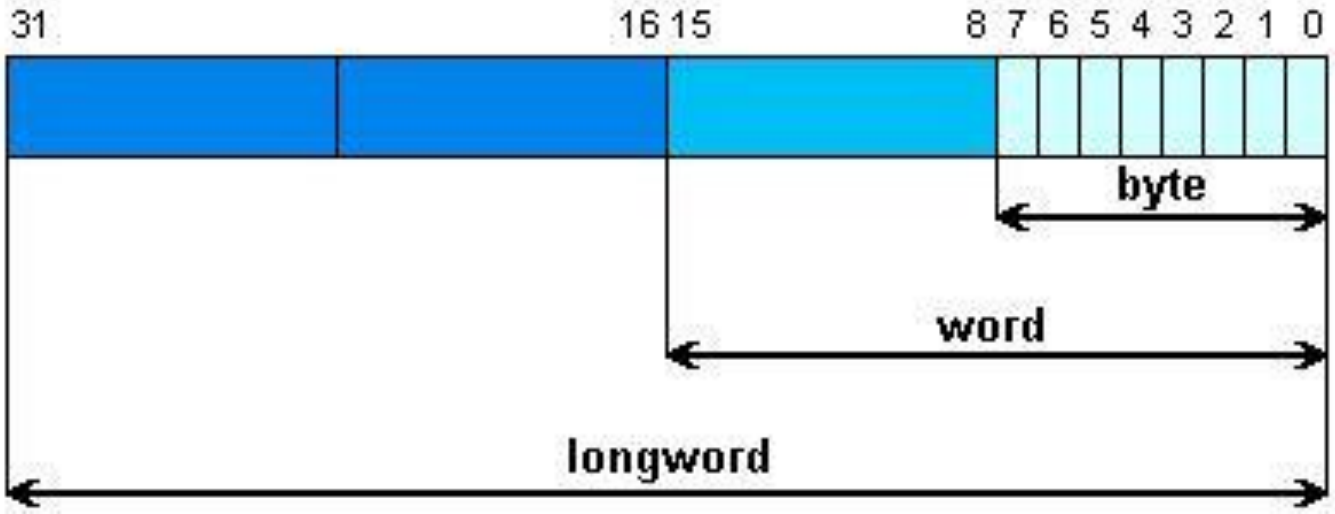
Τα δεδομένα μέσα σε ένα Καταχωρητή (Data in a Register)

Υπάρχουν δύο (2) τρόποι / **endianess (Indianess)** με τους οποίους μια σειρά bytes μπορεί να αποθηκευτεί στις θέσεις μνήμης (η κάθε θέση αποθηκεύει 1 byte):

• **Big-endian**: Το σημαντικότερο byte (MSB) αποθηκεύεται στην "μικρότερη" θέση μνήμης.

• **Little-endian**: Το λιγότερο σημαντικότερο byte (LSB) αποθηκεύεται στην "μικρότερη" θέση μνήμης.

Η αποθήκευση στους **καταχωρητές** στον M68000 είναι **little-endian.**



Η Μνήμη RAM

Η μνήμη ενός υπολογιστή είναι ένας πίνακας ή μια συλλογή από πολλές μονάδες αποθήκευσης που ονομάζονται τοποθεσίες μνήμης. Αν θέλετε, μπορείτε να δείτε τη μνήμη ως ένα μεγάλο κουτί υποδιαιρεμένο σε πολλά μικρά πλαίσια - τις θέσεις μνήμης. Μια απλή μνήμη διαθέτει:

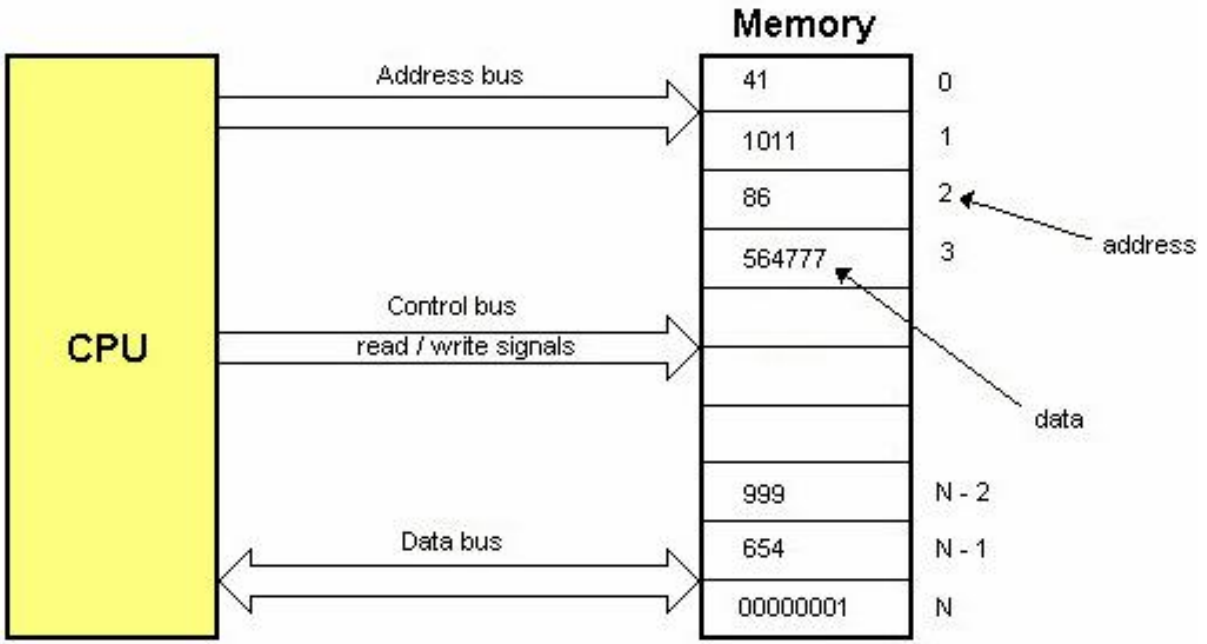
α) ένα διάδρομο διευθύνσεων εισόδου (η θύρα διεύθυνσης – Address Bus),
β) έναν διάδρομο δεδομένων εισόδου / εξόδου (η θύρα δεδομένων – Data Bus)
και
γ) έναν διάδρομο ελέγχου που ορίζει αν είναι σε κατάσταση ανάγνωσης ή εγγραφής (Read/Write – Control Bus).

Κάθε μια θέση μνήμης έχει μια μοναδική διεύθυνση, η οποία χρησιμοποιείται για να αναφέρεται στη συγκεκριμένη τοποθεσία. Τα περιεχόμενα μίας θέσης μνήμης ονομάζονται δεδομένα.

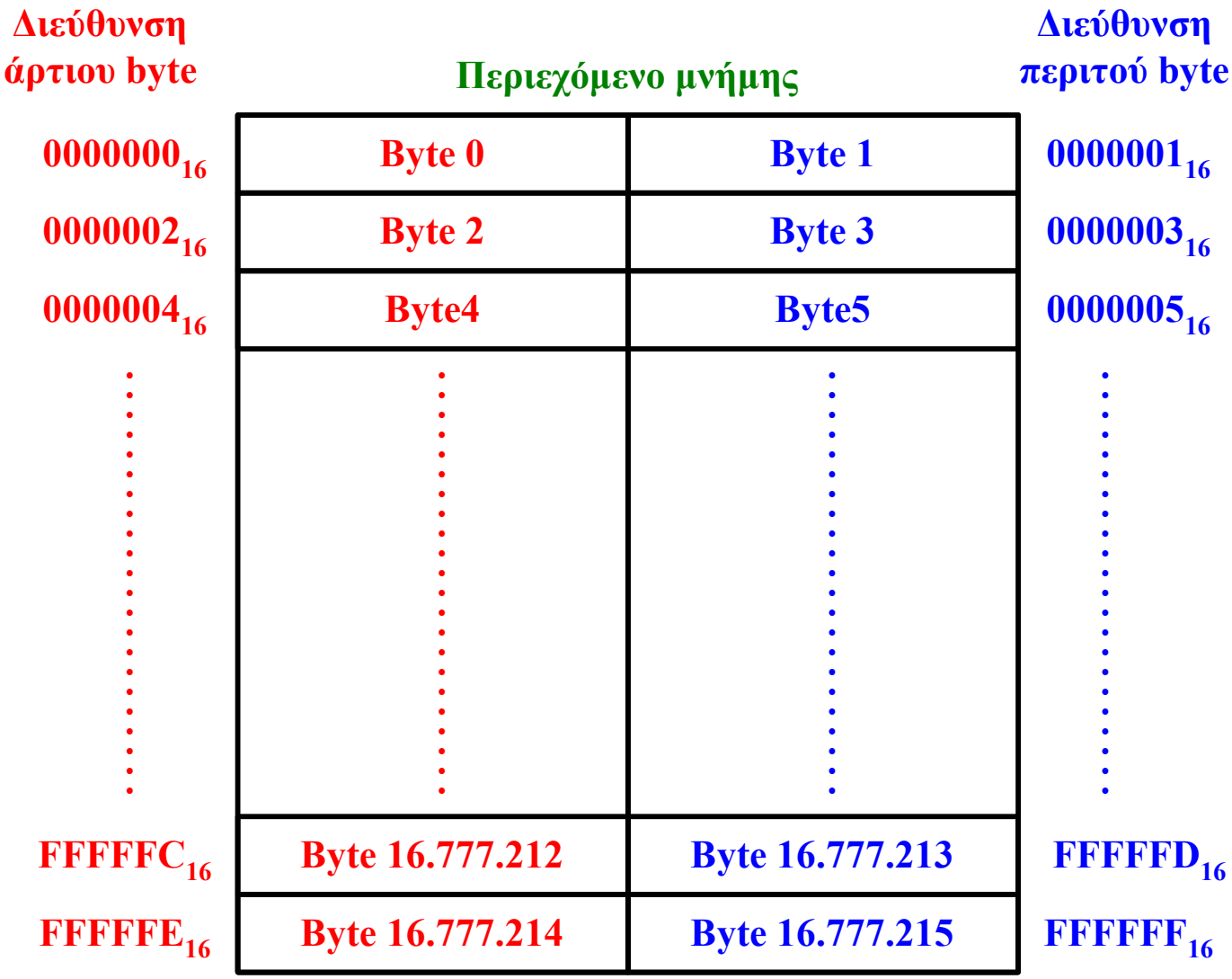
Ο M68000 διαθέτει καταχωρητές διευθύνσεων των 32-bit αλλά χρησιμοποιεί μόνο τα 24 από αυτά και μπορεί να υποστηρίξει μέχρι 16 MB μνήμης.

Η Μνήμη RAM

Ο M68000 στη μνήμη RAM αποθηκεύει με **big-endian**, κάθε φορά που απευθύνεται σε μνήμη ή αποστέλλει / αποθηκεύει λέξεις κατά βούληση, το πιο σημαντικό byte - το byte που περιέχει το σημαντικότερο bit **MSB** - αποθηκεύεται **πρώτα** (**έχει τη χαμηλότερη διεύθυνση**) ή αποστέλλεται πρώτα, ή αποστέλλονται σε φθίνουσα σειρά σημασίας, με το λιγότερο σημαντικό byte - αυτό που περιέχει το **LSB** ελάχιστο σημαντικό αποθηκευμένο στο τελευταίο bit (**που έχει την υψηλότερη διεύθυνση**) ή έχει σταλεί τελευταίο. Η μνήμη είναι **byte-addressable**, δηλαδή κάθε θέση στη μνήμη έχει μια μοναδική διεύθυνση μεγέθους byte και είναι άμεσα προσπελάσιμη.

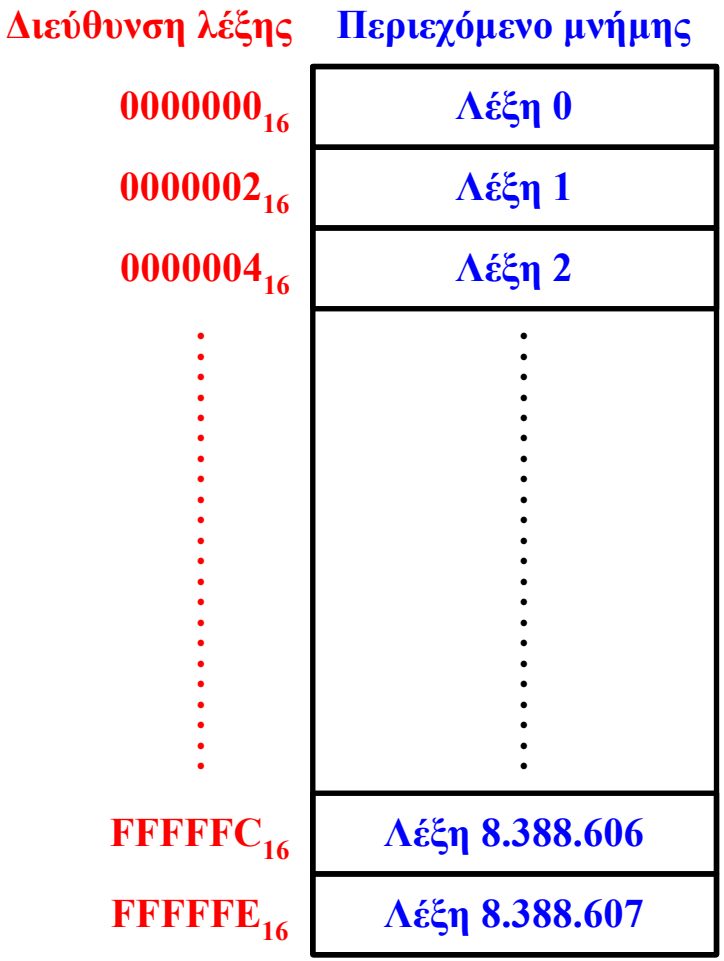


Χώρος διευθύνσεων byte



Ένα **byte** μπορεί να αποθηκευτεί σε **άρτια ή περιττή** θέση μνήμης.

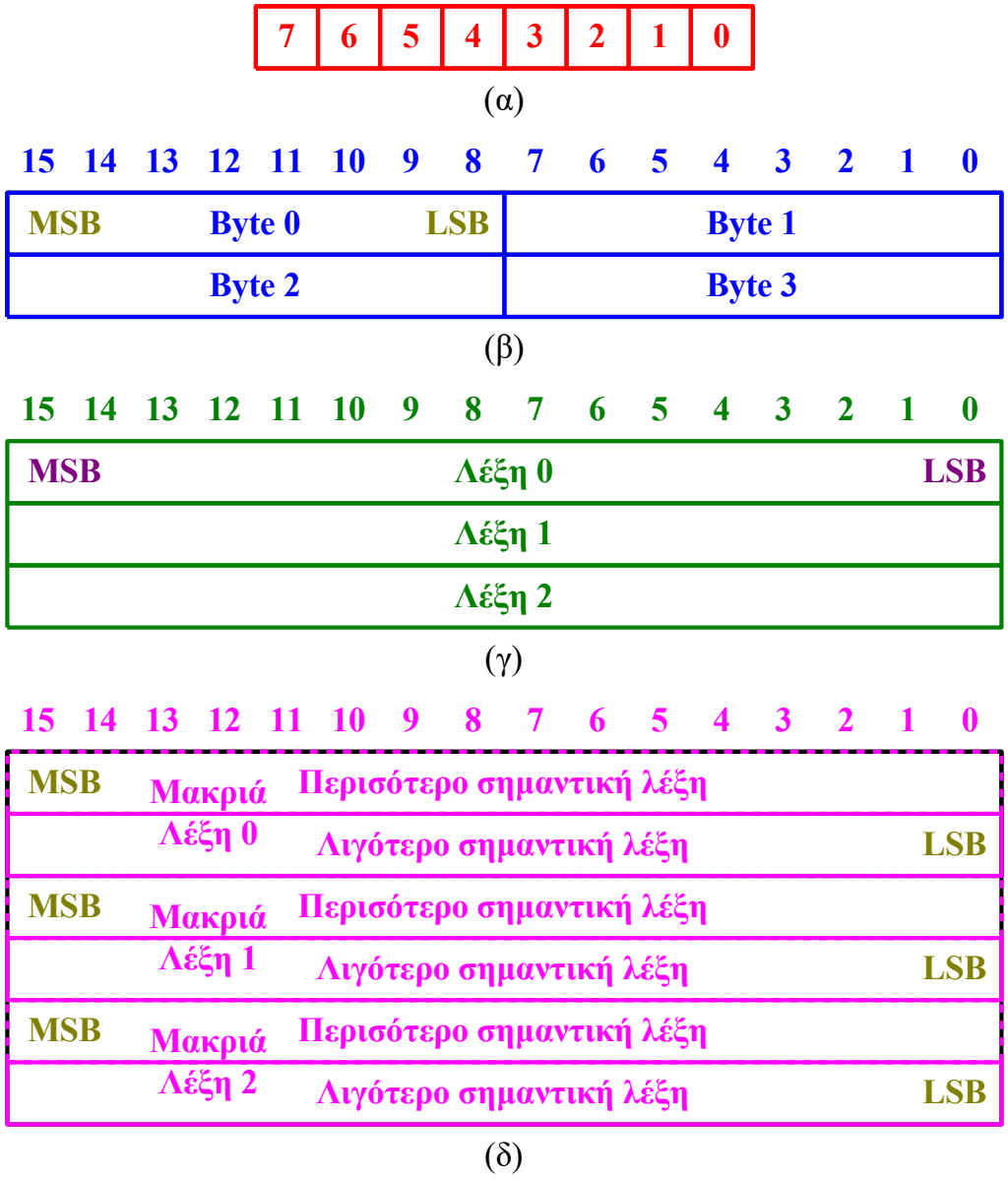
Χώρος διευθύνσεων word



Τα **words** μπορούν να αποθηκευτούν **ΜΟΝΟ** σε **άρτιες** θέσεις μνήμης.
Αν προσπαθήσουμε να το αποθηκεύσουμε σε περιττή τότε προκαλούμε ένα “TRAP”.

Οργάνωση δεδομένων στη μνήμη

Τα **longwords** μπορούν να αποθηκευτούν **MONO** σε **άρτιες** θέσεις μνήμης και απαιτούν δύο προσβάσεις στη μνήμη αφού χρειάζονται δύο words.



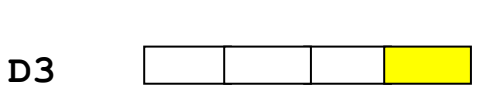
Τα δεδομένα μέσα στη Μνήμη (Data in the Memory)

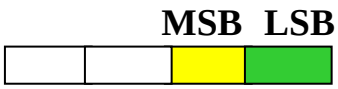
Διάβασε ένα **“byte”** στη διεύθυνση offset 00000013
Διάβασε ένα **“long-word”** στη διεύθυνση offset 00000034
Γράψε ένα **“word”** 20 4F στη διεύθυνση offset 0000000A

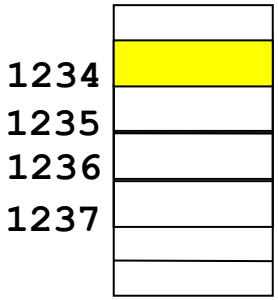
Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000000	00	AC	5E	00	00	00	00	00	00	00	00	00	00	00	00	00
00000010	00	00	00	01	07	77	00	00	00	00	00	00	00	00	00	00
00000020	00	00	00	00	00	00	00	00	00	00	47	00	00	00	00	00
00000030	00	00	11	20	**F3**	**44**	**F5**	**39**	C2	00	00	00	00	00	00	00
etc																

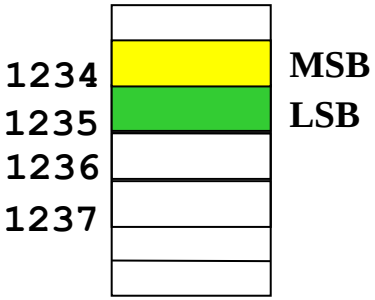
Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000000	00	AC	5E	00	00	00	00	00	00	00	20	4F	00	00	00	00
00000010	00	00	00	01	07	77	00	00	00	00	00	00	00	00	00	00
00000020	00	00	00	00	00	00	00	00	00	00	47	00	00	00	00	00
00000030	00	00	11	20	F3	44	F5	39	C2	00	00	00	00	00	00	00
etc																

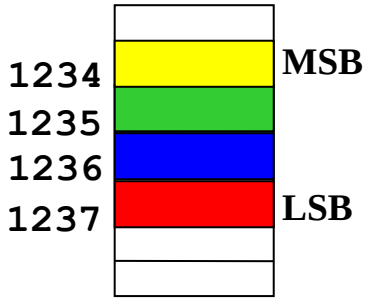
bytes, words, longwords











`MOVE.B $1234,D3`

[D3(0:7)] ← [M($1234)]

`MOVE.W $1234,D3`

[D3(0:15)] ← [M($1234)]

`MOVE.L $1234,D3`

[D3] ← [M($1234)]

Τι είναι η «Αρχιτεκτονική Μικροεπεξεργαστή»

- Για τη δική μας δουλειά **η Αρχιτεκτονική είναι το λογισμικό ή το μοντέλο προγραμματισμού** του επεξεργαστή, δηλαδή:
  - Οι **καταχωρητές (Registers)** της ΚΜΕ που είναι διαθέσιμοι στον προγραμματιστή.
  - Οι βασικές **εντολές (Commands)** που μπορεί να εκτελέσει η ΚΜΕ.
  - Οι τρόποι που οι εντολές μπορούν **να καθορίσουν μια θέση μνήμης (Addressing Modes)**.
  - Ο τρόπος **που οργανώνεται η πληροφορία στη μνήμη (Memory and Data Organization).**
  - Πως η ΚΜΕ αποκτά πρόσβαση και ελέγχει τις **περιφερειακές συσκευές (Peripheral Devices).**

Addressing Modes
Μέθοδοι Διευθυνσιοδότησης

- Ερώτηση: Που διαμένει ο κ. Κώστας?
- Απάντηση:

Ο κ. Κώστας μένει στην οδό Σμύρνης 33

ή

Ο κ. Κώστας μένει μετά από το γήπεδο του τένις δύο σπίτια αριστερά …

Στην πρώτη περίπτωση δίνουμε την **απόλυτη** διεύθυνση και στη δεύτερη τη **σχετική** θέση σύμφωνα με κάτι γνωστό. Έτσι κάπως δουλεύουν και οι υπολογιστές.

Μέθοδοι Διευθυνσιοδότησης

Οι μέθοδοι διευθυνσιοδότησης (Addressing Modes) αφορούν στο τρόπο με τον οποίο ο επεξεργαστής προσπελαύνει τους τελεστέους που χρησιμοποιούνται απ΄ τις εντολές.

Ο μικροεπεξεργαστής M68000 έχει δεκατέσσερις (14) μεθόδους διευθυνσιοδότησης χωρισμένες στις παρακάτω έξι (6) ομάδες:

- ***Διευθυνσιοδότηση απευθείας δεδομένων (immediate data addressing).***
- ***Άμεση διευθυνσιοδότηση καταχωρητή (register direct addressing).***
- ***Απόλυτη διευθυνσιοδότηση δεδομένων (absolute data addressing).***
- ***Συνεπαγόμενη διευθυνσιοδότηση (implied addressing).***
- ***Σχετική διευθυνσιοδότηση μετρητή προγράμματος (program counter relative addressing).***
- ***Έμμεση διευθυνσιοδότηση καταχωρητή (register indirect addressing).***

Τελεστέοι - Operands

- Οι τελεστέοι μπορεί να είναι:
  - Καταχωρητές (Registers)
  - Σταθερές (Constants)
  - Διευθύνσεις Μνήμης (Memory addresses)
- Οι τελεστέοι ορίζουν μεθόδους διευθυνσιοδότησης, όπως:
  - Dn: data register direct **MOVE.W D0, D1**
  - An: address register indirect **MOVE.W (A0),D1**
  - #n: immediate **MOVE.W #10,D1**
  - N: absolute **MOVE.W $1000,D1**
- Οι τελεστέοι μπορούν να οριστούν με πέντε (5) τύπους δεδομένων:
  - Decimal: προκαθορισμένος τύπος δεδομένων
  - Hexadecimal: πρόθεμα $
  - Octal: πρόθεμα @
  - Binary: πρόθεμα %
  - ASCII: μέσα σε μονά εισαγωγικά ‘ABC’
- **Ενεργός Διεύθυνση (Effective address)**:
  Η πραγματική διεύθυνση που χρησιμοποιεί η εντολή.
  - Παράδειγμα:
    - Ο καταχωρητής δεδομένων D1 στον επεξεργαστή
    - Η διεύθυνση $10000 στη μνήμη

Παράδειγμα

Μορφή Εντολής:

CLR.s <ea>

Παράδειγμα:

CLR.W D1 ;Clears lower word of D1

Αποτέλεσμα:

Μορφή Εντολής:

```
MOVE.s    <ea>,<ea>
```

Παράδειγμα:

```
MOVE.W    D0,D1     ;Copy lower word of D0 to D1
```

Αποτέλεσμα:

Καταχωρητές Διευθύνσεων – Address Registers

- Χρησιμοποιούνται για την αποθήκευση διευθύνσεων (είναι δείκτες σε, pointers) των θέσεων στη μνήμη.
- Έχουν πλάτος 32 bits, καθώς οι διευθύνσεις στη μνήμη είναι αριθμοί 32 bit.
- Στην περίπτωση του επεξεργαστή M68000, ακόμα και αν όλα τα bits διεύθυνσης συμμετέχουν σε εσωτερικές λειτουργίες, μόνο τα πρώτα 24 bits διεύθυνσης (bits διεύθυνσης 0 έως 23) συμμετέχουν στην επιλογή μίας θέσης στη μνήμη.
- Οι συνήθως χρησιμοποιούμενες εντολές όπως MOVE, ADD, CMP, SUB έχουν ειδικές παραλλαγές που έχουν σχεδιαστεί για χρήση με καταχωρητές διευθύνσεων. Αυτές οι παραλλαγές είναι MOVEA, ADDA, CMPA, SUBA, αντίστοιχα, και πρόκειται να χρησιμοποιηθούν όταν ο τελεστής προορισμού είναι ένας καταχωρητής διευθύνσεων.
- Για παράδειγμα, ακόμη και αν το MOVE.L A0,D1 είναι νόμιμο, MOVE.L D1, A0 δεν είναι. Πρέπει να γράψετε το MOVEA.L D1, A0.
- Αυτή η παραλλαγή των εντολών δημιουργήθηκε για να αναγκάσει τον προγραμματιστή να κάνει τη διάκριση μεταξύ δεδομένων που κρατούν διευθύνσεις και των καταχωρητών δεδομένων για να αποφύγει τυχαίες αλλοιώσεις του περιεχομένου των καταχωρητών διευθύνσεων.

Καταχωρητές Διευθύνσεων –Address Registers

Παρόλο που οι καταχωρητές δεδομένων επιτρέπουν στον χρήστη να εκτελεί πράξεις byte, word και longword στα δεδομένα τους, οι καταχωρητές διευθύνσεων επιτρέπουν μόνο λειτουργίες longword ή word για το περιεχόμενό τους (καμία λειτουργία byte).

Αυτό έχει νόημα επειδή το περιεχόμενο ενός καταχωρητή δεδομένων μπορεί να είναι σχεδόν οτιδήποτε, ενώ ένας καταχωρητής διευθύνσεων περιέχει διεύθυνση 32-bit που είναι μια ενιαία οντότητα που δεν μπορεί να υποδιαιρεθεί σε μικρότερες μονάδες. Συνεπώς, οι πράξεις byte δεν έχουν σημασία όσον αφορά τους καταχωρητές διευθύνσεων.

Οι εντολές με μέγεθος word δεν είναι σχετικές αλλά επιτρέπονται επειδή ο τελεστέος 16-bit μιας εντολής με μέγεθος .W αυτόματα επεκτείνεται με πρόσημο σε 32-bit πριν εκτελεστεί η εντολή.

Για παράδειγμα, εάν έχετε ADDA.W #$6,A2 ο τελεστής 16-bit του $0006 επεκτείνεται σε 32 bit $00000006 πριν προστεθεί στα περιεχόμενα του A2. Έτσι, οι εντολές με τελεστέο μεγέθους word δίνουν ένα αποτέλεσμα 32-bit και επηρεάζουν το σύνολο των 32-bit περιεχομένων του καταχωρητή διευθύνσεων προόρισμού.

Καταχωρητές Διευθύνσεων –Address Registers

Όταν ένας τελεστής επεκτείνεται στα 32 bit, επεκτείνεται και το πρόσημο του. Αυτό σημαίνει ότι η επέκταση πρέπει να διατηρήσει το πρόσημο του τελεστή, **επειδή στον M68000 οι διευθύνσεις θεωρούνται προσημασμένες τιμές**.

Για παράδειγμα, αν έχετε ADDA.W #$FFF8,A1 ο τελεστής 16-bit από $FFF8 δεν επεκτείνεται σε $0000FFF8 αλλά σε $FFFFFFF8. Επομένως, αν το πιο σημαντικό bit του τελεστή είναι 0, τα bits 16 έως 31 ορίζονται στο 0, αν το πιο σημαντικό bit είναι 1, τα bits 16 έως 31 ορίζονται στο 1. Αυτό δηλώνει πώς χειρίζονται οι προσημασμένοι αριθμοί στη αριθμητική με συμπληρώματα ως προς 2. Ελέγξτε ότι στην αριθμητική τιμή του συμπληρώματος ως προς 2 τόσο το $\$FFF8_{16}$ όσο και το $\$FFFFFFF8_{16}$ ισούνται με το -8_{10}.

Γιατί οι τιμές των διευθύνσεων είναι προσημασμένες; Εάν προσθέσετε μια θετική τιμή στο, ας πούμε, A0, τότε το A0 θα δείξει μια θέση προς την υψηλότερη μνήμη (προς μνήμη με μεγαλύτερες διευθύνσεις). Εάν προσθέσετε μια αρνητική τιμή στα περιεχόμενα του A0, τότε το A0 θα δείχνει προς τη χαμηλότερη μνήμη (μνήμη με χαμηλότερες διευθύνσεις). Έτσι, μπορείτε να σκεφτείτε θετικές διευθύνσεις που υποδηλώνουν κίνηση προς τα εμπρός στη μνήμη (προς την υψηλότερη μνήμη) και αρνητικές διευθύνσεις που υποδηλώνουν κίνηση προς την κατώτερη μνήμη.

Εάν, ο τελεστής προορισμού είναι καταχωρητής διευθύνσεων, τα περιεχόμενα του CCR στον Καταχωρητή Κατάστασης (Status Register) δεν επηρεάζονται, εκτός αν η εντολή είναι CMPA (σύγκριση με τον καταχωρητή διευθύνσεων).

Κωδικοποίηση Συμπληρώματος ως προς 2

Για το διάνυσμα: $\vec{x} = [x_{w-1}, x_{w-2}, \dots, x_0]$

Ισχύει: $$B2T_w(\vec{x}) \doteq -x_{w-1}2^{w-1} + \sum_{i=0}^{w-2} x_i 2^i$$

Το περισσότερο σημαντικό bit, x_{w-1}, ονομάζεται bit προσήμου (sign bit). Όταν είναι 0 είναι θετικός, όταν είναι 1 είναι αρνητικός

Παράδειγμα:

$$B2T_4([0001]) = -0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 0 + 0 + 0 + 1 = 1$$

$$B2T_4([0101]) = -0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 0 + 4 + 0 + 1 = 5$$

$$B2T_4([1011]) = -1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = -8 + 0 + 2 + 1 = -5$$

$$B2T_4([1111]) = -1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = -8 + 4 + 2 + 1 = -1$$

[1 1 1 1]

Πρόσημο - αρνητικό

Μέτρο/τιμή

[1 1 1]
[0 0 0]
+1

[0 0 1]

Δείγμα Εντολής του M68000

LABEL (Ετικέτα)	**OPCODE** (Κωδικός Εντολής)	**OPERANDS** (Τελεστέοι)	**COMMENTS** (Σχόλια)
LOOP	**MOVE .B**	**D0,D3**	* αντέγραψε το περιεχόμενο του * καταχωρητή D0 στον * καταχωρητή D3.

Τελεστέος Προέλευσης

Τελεστέος Προόρισμού

Άμεση διευθυνσιοδότηση καταχωρητή (register direct addressing)

- Η πιο απλή διευθυνσιοδότηση.
- Ο Τελεστέος Προέλευσης (Source) ή Προορισμού (Destination) είναι καταχωρητής δεδομένων ή διευθύνσεων.
- Παραδείγματα:

Instruction: MOVE.B D0,D3

	Register	Contents
Before:	D0	10204FFF
	D3	1034F88A
After:	D0	10204FFF
	D3	1034F8FF

Only bits 0-7 affected

Instruction: MOVEA.L A3,A0

	Register	Contents
Before:	A0	00200000
	A3	0004F88A
After:	A0	0004F88A
	A3	0004F88A

Move to address register

32 bits are moved

Instruction: MOVE.W D4,D7

	Register	Contents
Before:	D4	100030FF
	D7	8E552900
After:	D4	100030FF
	D7	8E5530FF

Instruction: MOVEA A3,A0

	Register	Contents
Before:	A0	00200000
	A3	0004F88A
After:	A0	FFFFF88A
	A3	0004F88A

Assembler -> .W

Παράδειγμα

Η εντολή:

Διεύθυνση στη μνήμη	**Εντολή**
$400400	**MOVE.L A0, D0**

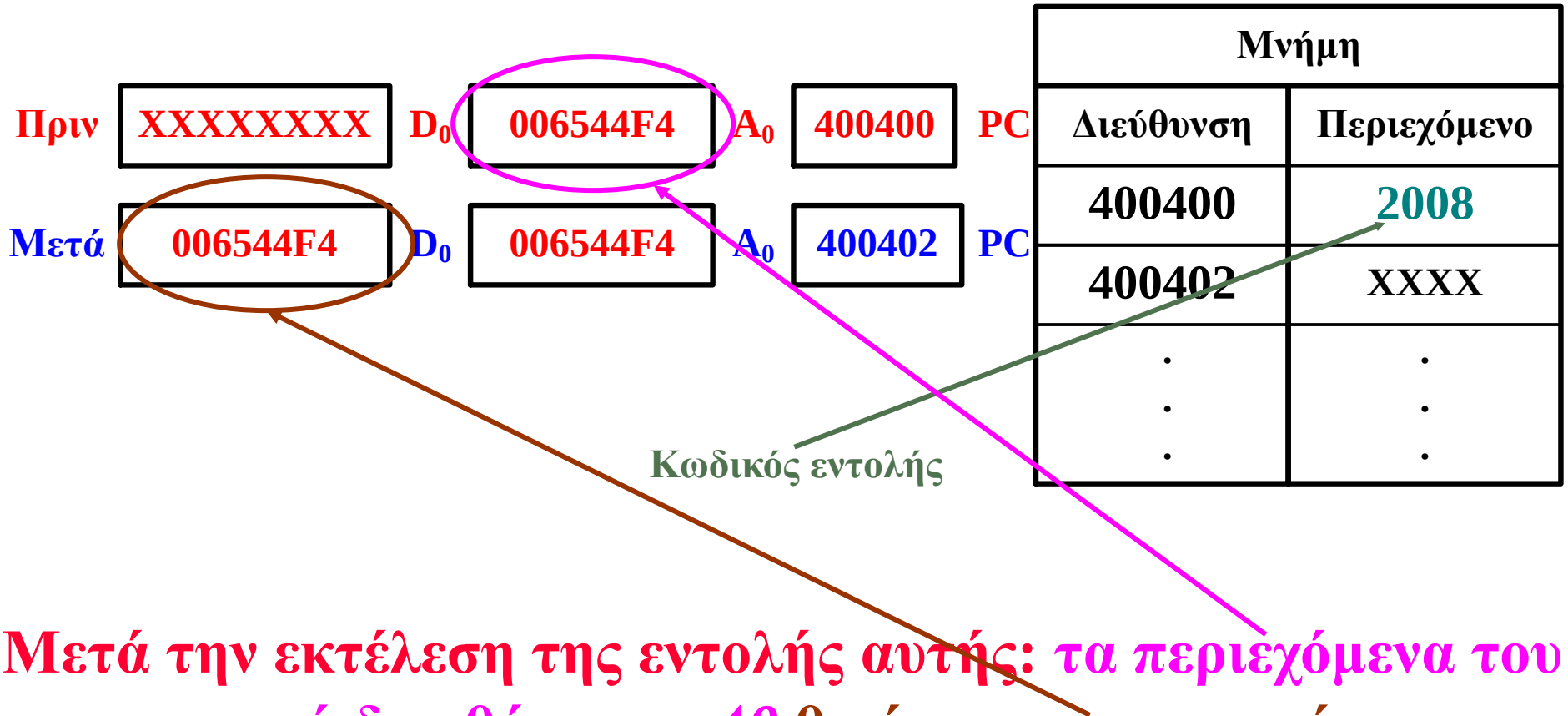
μεταφράζεται ως: "αντέγραψε τη μακριά λέξη από τον καταχωρητή διευθύνσεων *A0* στον καταχωρητή δεδομένων *D0*" και χρησιμοποιεί:

Άμεση διευθυνσιοδότηση καταχωρητή διευθύνσεων για τον τελεστέο προέλευσης, που βρίσκεται στον καταχωρητή διευθύνσεων *A0*

και άμεση διευθυνσιοδότηση καταχωρητή δεδομένων για τον τελεστέο προορισμού, που είναι ο καταχωρητής δεδομένων *D0*

Χάρτης μνήμης

MOVE.L A0,D0



Μετά την εκτέλεση της εντολής αυτής: τα περιεχόμενα του καταχωρητή διευθύνσεων *A0* θα έχουν αντιγραφεί στον καταχωρητή δεδομένων *D0*.

Συμπεράσματα

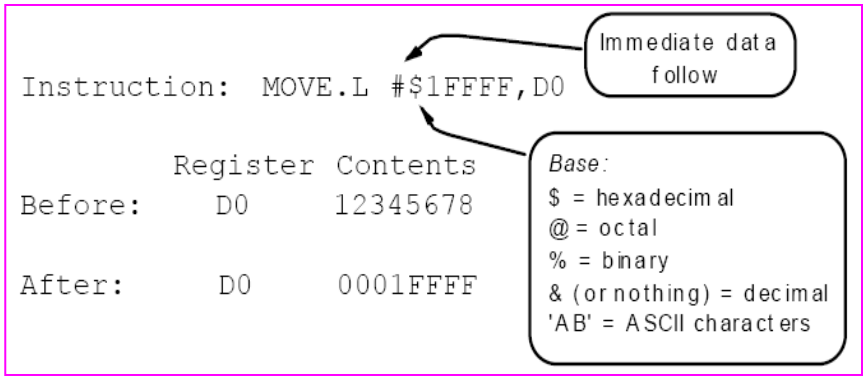
Η άμεση διευθυνσιοδότηση καταχωρητή χρησιμοποιεί κοντές εντολές διότι χρειάζονται μόνο τρία ψηφία για να καθοριστεί ένας από τους οκτώ καταχωρητές δεδομένων.

Η άμεση διευθυνσιοδότηση καταχωρητή είναι γρήγορη γιατί δε χρειάζεται να προσπελαστεί η εξωτερική μνήμη.

Οι προγραμματιστές χρησιμοποιούν άμεση διευθυνσιοδότηση καταχωρητή για να κρατούν μεταβλητές που προσπελαύνονται συχνά.

Διευθυνσιοδότηση απευθείας δεδομένων (immediate data addressing)

- Μπορεί να χρησιμοποιηθεί μόνο για διευθυνσιοδότηση Τελεστέων Προέλευσης. Ο προορισμός των δεδομένων θα πρέπει να προσδιοριστεί με τη διευθυνσιοδότηση του Τελεστέου Προορισμού.
- Σημειώνεται με το σύμβολο # μπροστά από τον Τελεστέο Προέλευσης.




```
Instruction: MOVE.B #@72,D5

        Register Contents
Before:   D5      12345678
After :   D5      1234563A
```

```
Instruction: MOVE.W #$9E00,D5

        Register Contents
Before:   D5      12345678
After :   D5      12349E00
```

```
Instruction: MOVE.L #1,D5

        Register Contents
Before:   D5      12345678
After :   D5      00000001
```

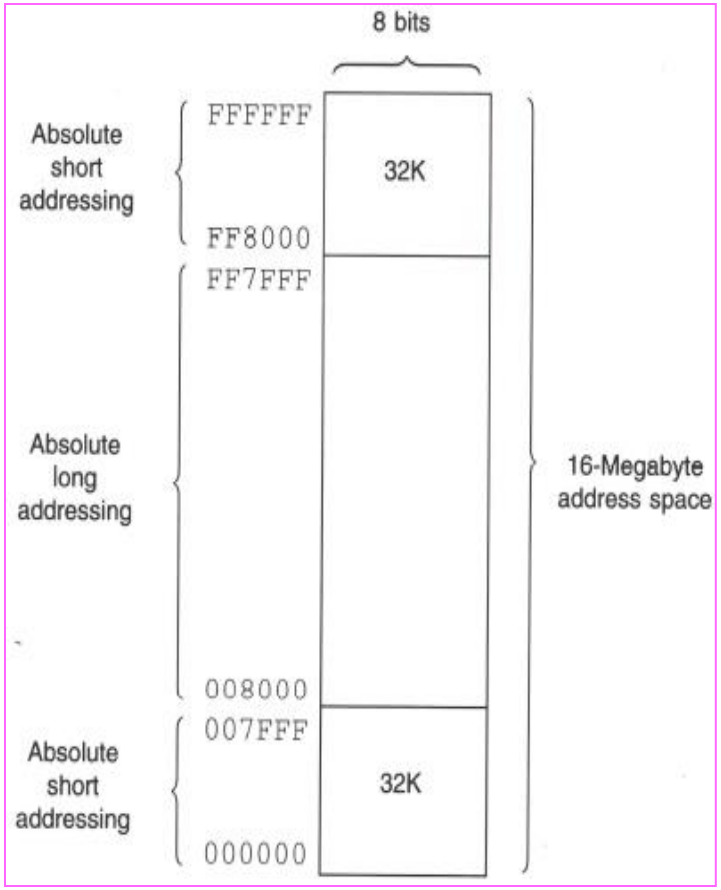
Πολύ χρήσιμο για να φορτώνεις σταθερές (constants)

Απόλυτη διευθυνσιοδότηση δεδομένων (absolute data addressing)

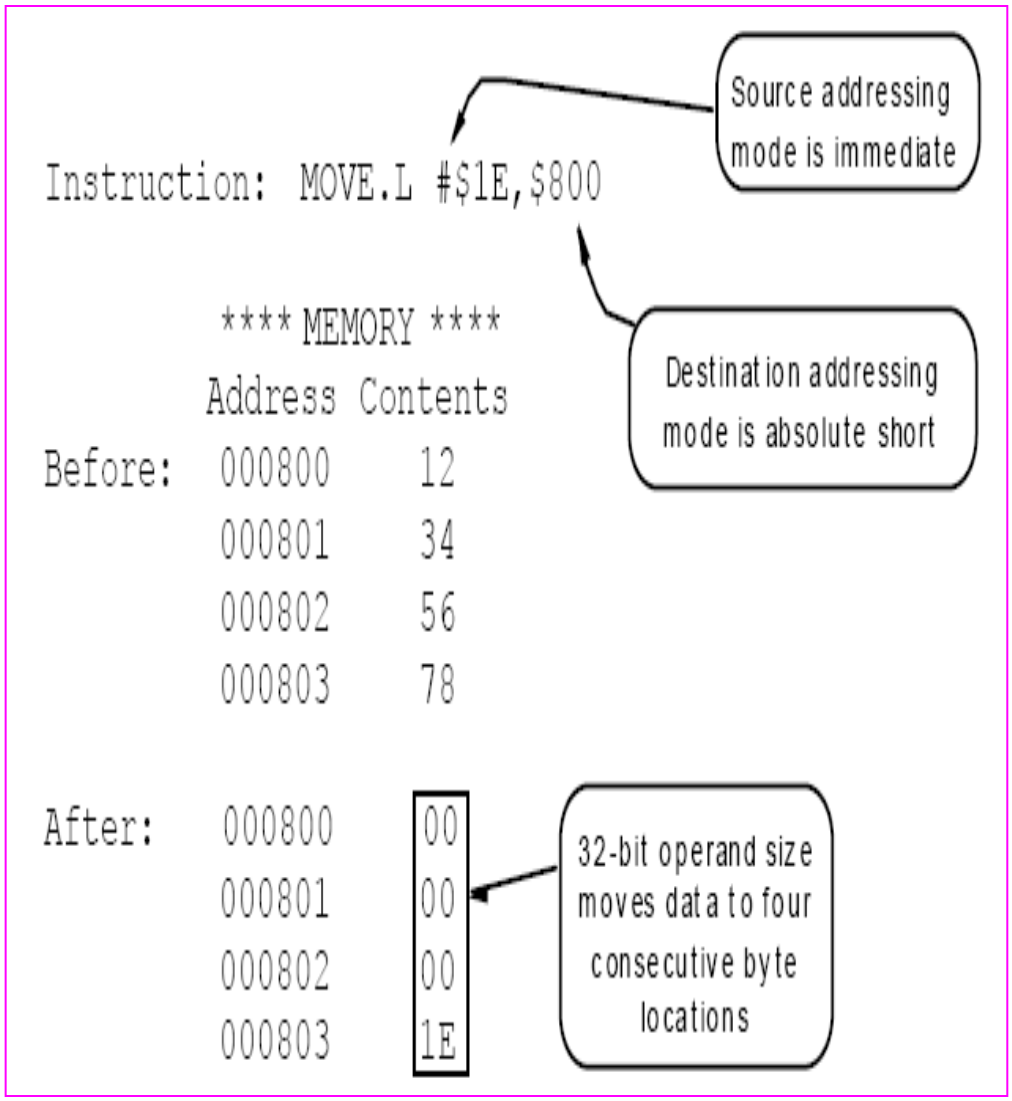
- **Ενεργός Διευθυνση** (Effective Address): Η θέση μνήμης που καθορίζεται από την εντολή.
- Οι περισσότερες διευθύνσεις ορίζονται στο δεκαεξαδικό (hex) άρα σχεδόν πάντα το σύμβολο‘$’ θα το βρίσκουμε μπροστά από εντολές που έχουν τελεστές με διευθύνσεις μνήμης.
- **Στην απόλυτη διευθυνσιοδότηση δεδομένων η εντολή δίνει τη διεύθυνση του τελεστέου στη μνήμη.**
- **Η ενεργός διεύθυνση του τελεστέου εμπεριέχεται στην εντολή και μάλιστα ακολουθεί αμέσως μετά τον κωδικό της εντολής.**
- Υπάρχουν δύο εκδόσεις για απόλυτη διευθυνσιοδότηση:
  - Απόλυτη Κοντή **(Absolute short)** και
  - Απόλυτη Μακρυά **(Absolute long)**.
- **Όταν η ενεργός διεύθυνση αποτελείται από τέσσερα δεκαεξαδικά ψηφία (λέξη 16 δυαδικών ψηφίων) τότε η μέθοδος διευθυνσιοδότησης αναφέρεται ως *απόλυτη διευθυνσιοδότηση κοντών δεδομένων (absolute short data addressing).***
- **Όταν η ενεργός διεύθυνση αποτελείται από πέντε ή έξι δεκαεξαδικά ψηφία (λέξη 20 ή 24 δυαδικών ψηφίων) τότε η μέθοδος διευθυνσιοδότησης αναφέρεται ως *απόλυτη διευθυνσιοδότηση μακριών δεδομένων (absolute long data addressing).***

Απόλυτη διευθυνσιοδότηση δεδομένων (absolute data addressing)

- Το αν ο τρόπος απόλυτης διευθυνσιοδότησης είναι μακρύς ή κοντός καθορίζεται συνήθως από τον Assembler.
- Η Απόλυτη Κοντή διευθυνσιοδότηση παράγεται συνήθως από τον Assembler και όχι απευθείας από τον προγραμματιστή.
- Ο Assembler κάνει αυτό για να δημιουργήσει μικρότερο κώδικα αλλά περιορίζει το εύρος μνήμης στο οποίο μπορείτε να έχετε πρόσβαση ($000000-$007FFF και $FF8000-$FFFFFF).
- Ως απόλυτο κοντό χρησιμοποιεί μια λέξη 16-bit που με αυτόματο τρόπο γίνετε επέκταση προσήμου σε 24-bit (32-bit) και μπορεί να χρησιμοποιείτε μόνο για πρόσβαση στη μνήμη στο κάτω ή στο επάνω μέρος της μνήμης όπως φαίνεται δίπλα. Το απόλυτο κοντό δε μπορεί να χρησιμοποιηθεί για την πρόσβαση στο χώρο της μνήμης μεταξύ των δύο που προαναφέρθηκαν.
- Το απόλυτο μακρύ μπορεί να χρησιμοποιηθεί στον χώρο μνήμης $008000-$FFFFFF και είναι σε πλήρη χρήση και των 24-bit (32-bit).



Absolute Short Addressing Mode



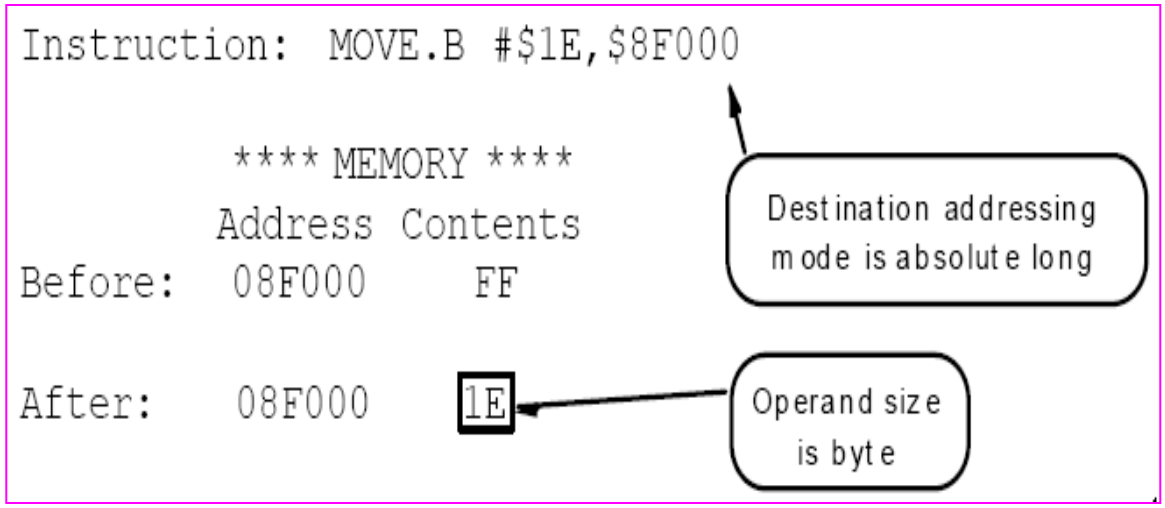
Instruction: MOVE.B $3C00,D1

	Register	Content
Before:	D1	11111111
	*****Memory*****	
	Address	Content
	003BFE	12
	003BFF	AB
	003C00	4F
	003C01	1D
	003C02	CC
After :	D1	1111114F

Instruction: MOVE.W $9AE0,D2

	Register	Content
Before:	D2	AAAAAAAA
	*****Memory*****	
	Address	Content
	FF9ADE	12
	FF9ADF	34
	FF9AE0	56
	FF9AE1	78
	FF9AE2	CC
After :	D2	AAAA5678

Absolute Long Addressing Mode




```
Instruction: MOVE.B $2E000,D3
          Register Content
Before:   D3       FFFFFFFF
          *****Memory*****
          Address Content
            02DFFE    11
            02DFFF    22
            02E000    33
            02E001    44
            02E002    55
After :   D3       FFFFFF33
```

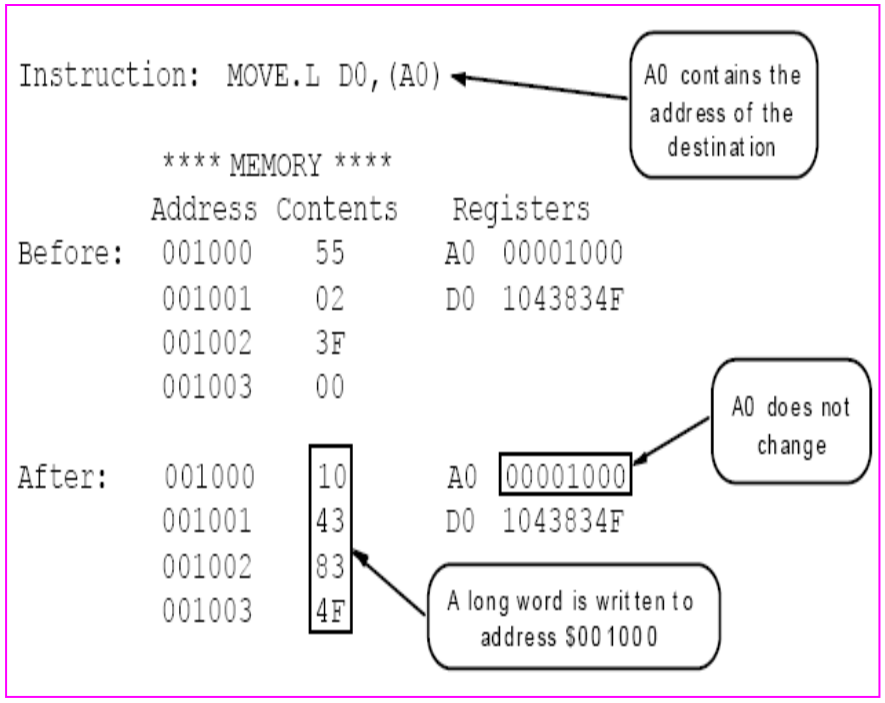
Listing File

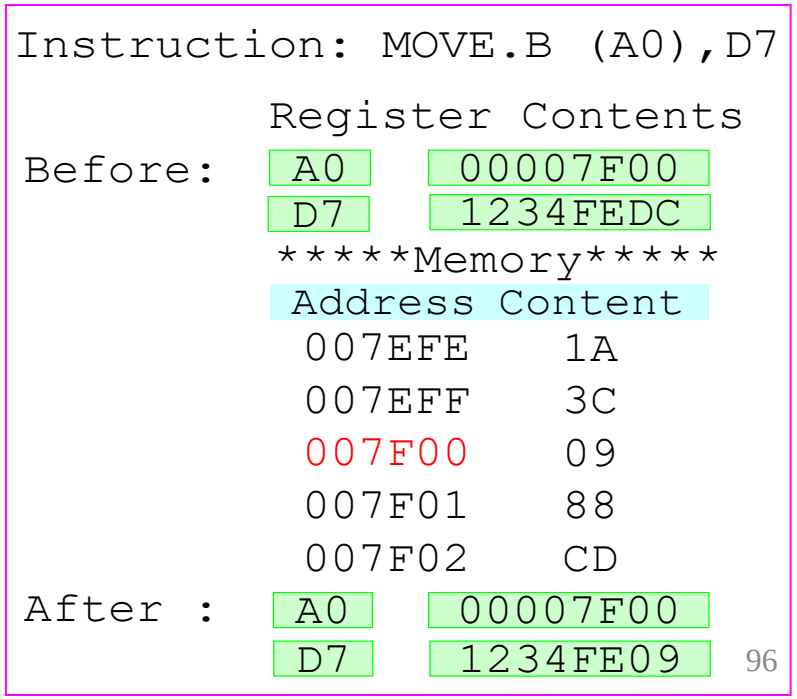
```
00000830 Starting Address
Assembler used: EASy68K Editor/Assembler v5.16.01
Created On: 29-Oct-18 1:07:41 PM

00000800                             1         ORG    $800
00000800= 12345678                   2  X      DC.L $12345678
00000830                             3         ORG    $830
00000830  23FC 0000001E 00000800     4  START: MOVE.L #$1E,$800.L
0000083A  21FC 0000001E 0804         5         MOVE.L #$1E,$804
Line 6 WARNING: Forcing SHORT addressing disables range checking of extension word
00000842  1238 0800                  6         MOVE.B $800.W,D1
Line 7 WARNING: Forcing SHORT addressing disables range checking of extension word
00000846  1438 0400                  7         MOVE.B $40400.W,D2
0000084A  1439 00400400              8         MOVE.B $400400,D2
00000850                             9         END START

No errors detected
2 warnings generated


SYMBOL TABLE INFORMATION
Symbol-name          Value
-------------------------
START                830
X                    800
```

Έμμεση διευθυνσιοδότηση καταχωρητή
Address Register Indirect Addressing (ARI) Mode

- Χρησιμοποιείται για προσπέλαση δεδομένων που είναι στη μνήμη.
- Ο Καταχωρητής διευθύνσεων κρατά τη θέση μνήμης που περιέχει τα δεδομένα του τελεστή.
- Ο M68000 αναγνωρίζει αυτή τη διευθυνσιοδότηση κάθε φορά που ο καταχωρήτης είναι μέσα σε παρενθέσεις, π.χ. (A0), (A3), και (A7).
- Παραδείγματα:

CLR.W (A0) : Η εντολή καθαρίζει τη θέση μνήμης που δείχνει ο A0.





Παράδειγμα

Στην περίπτωση αυτή ένας από τους καταχωρητές διεύθυνσης κρατά τη διεύθυνση στη μνήμη του τελεστέου προέλευσης ή προορισμού.

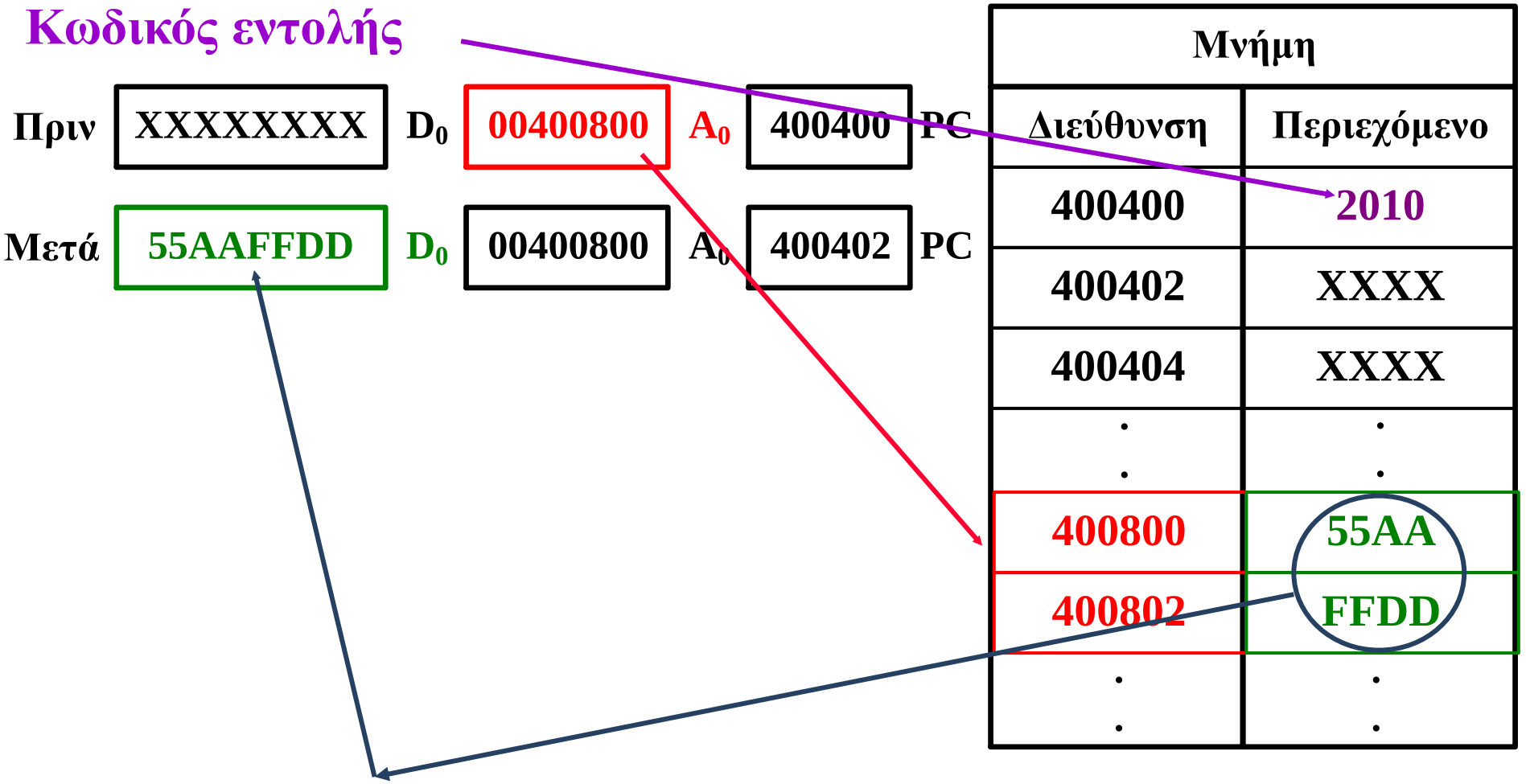
Η εντολή:

Διεύθυνση στη μνήμη	Εντολή
$400400	MOVE.L (A0),D0

μεταφράζεται ως: "αντέγραψε τη μακριά λέξη, που αρχίζει από τη διεύθυνση που εμπεριέχεται στον καταχωρητή διεύθυνσης *A0*, στον καταχωρητή δεδομένων *D0*"

Χάρτης Μνήμης

MOVE.L (A0),D0



D0=Μακριά λέξη των θέσεων $400800 έως $400803

Έμμεση μεταυξητική καταχωρητή διεύθυνσης

Στην περίπτωση αυτή ο τελεστέος προέλευσης ορίζεται ως (A0)+ και η διαφορά από την προηγούμενη μέθοδο έγκειται στο ότι το περιεχόμενο του καταχωρητή διεύθυνσης αυξάνει αυτόματα, μετά την εκτέλεση της εντολής, κατά 1, 2 ή 4 ανάλογα με το αν η εντολή αφορά σε byte, λέξη ή μακριά λέξη.

Με τον τρόπο αυτό η διεύθυνση δείχνει τα δεδομένα που ακολουθούν αμέσως μετά.

Έμμεση μεταυξητική καταχωρητή διεύθυνσης

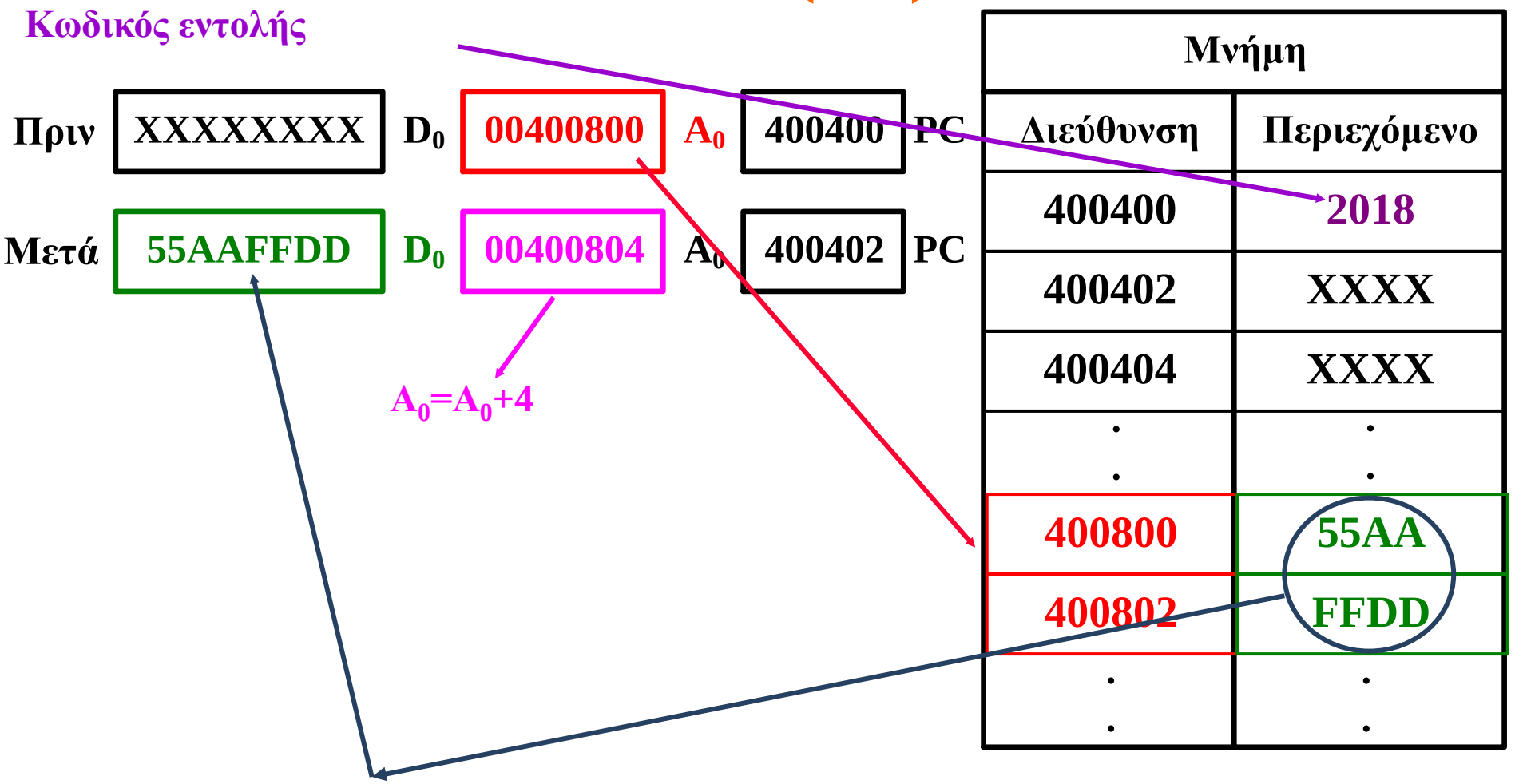
Η εντολή:

Διεύθυνση στη μνήμη	**Εντολή**
$400400	**MOVE.L (A0)+,D0**

μεταφράζεται ως: "αντέγραψε τη μακριά λέξη, που αρχίζει από τη διεύθυνση, που εμπεριέχεται στον καταχωρητή διεύθυνσης *A0*, στον καταχωρητή δεδομένων *D0* και μετά αύξησε το περιεχόμενο του καταχωρητή διεύθυνσης κατά τέσσερα"

Χάρτης Μνήμης

MOVE.L (A0)+,D0



D0=Μακριά λέξη των θέσεων $400800 έως $400803

Έμμεση προμειωτική καταχωρητή διεύθυνσης

Στην περίπτωση αυτή ο τελεστέος προέλευσης ορίζεται ως -(A0) και το περιεχόμενο του καταχωρητή διεύθυνσης μειώνεται αυτόματα, πριν απ' την εκτέλεση της εντολής, κατά 1, 2 ή 4 ανάλογα με το αν η εντολή αφορά σε byte, λέξη ή μακριά λέξη. Με τον τρόπο αυτό η νέα διεύθυνση δείχνει τα προηγούμενα δεδομένα

Έμμεση προμειωτική καταχωρητή διεύθυνσης

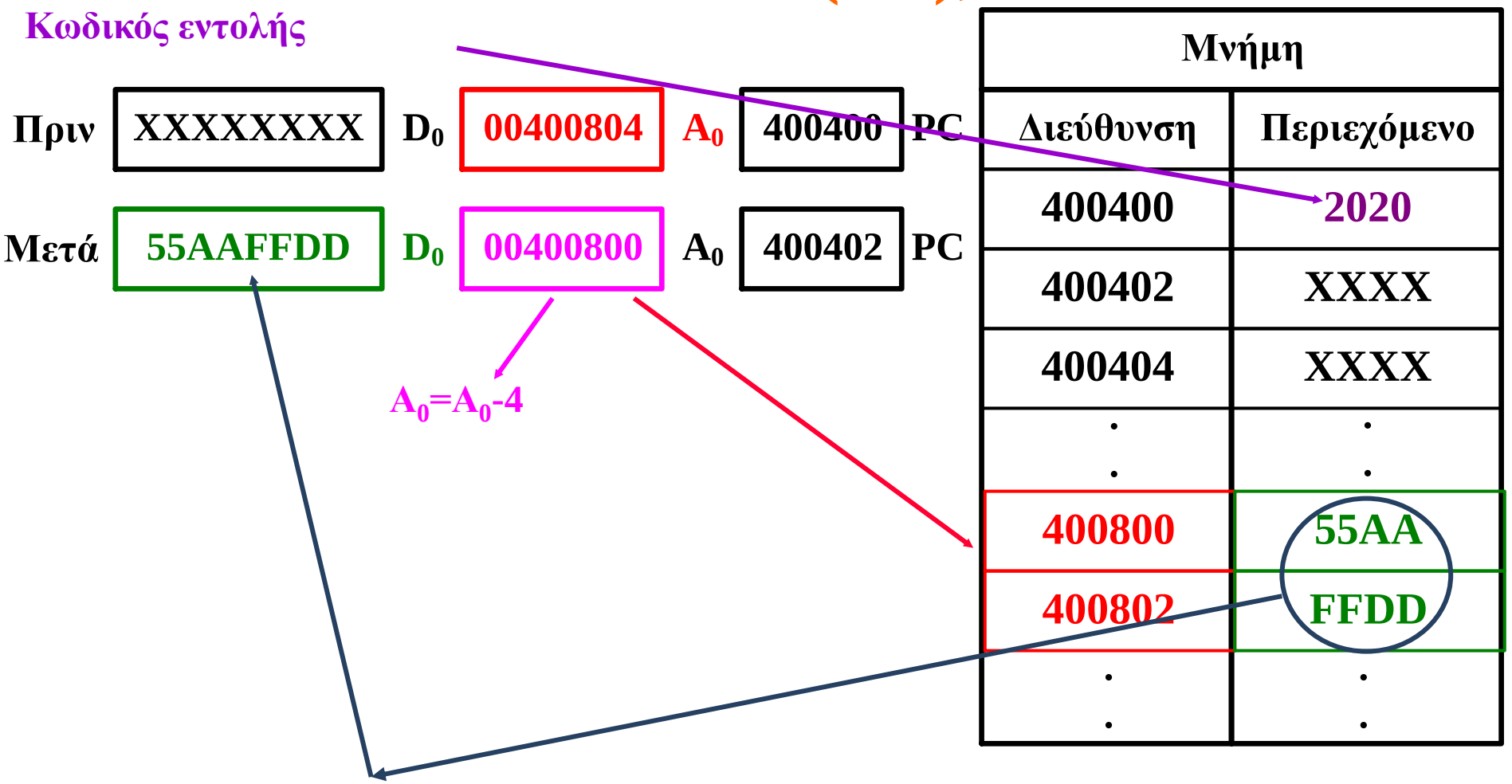
Η εντολή:

Διεύθυνση στη μνήμη	**Εντολή**
$400400	**MOVE.L -(A0),D0**

μεταφράζεται ως: "μείωσε το περιεχόμενο του καταχωρητή διεύθυνσης A_0 κατά τέσσερα και αντέγραψε τη μακριά λέξη, που αρχίζει από τη νέα διεύθυνση, που εμπεριέχεται στον καταχωρητή διεύθυνσης A_0, στον καταχωρητή δεδομένων D_0"

Χάρτης Μνήμης

MOVE.L -(A0),D0



D0=Μακριά λέξη των θέσεων $400800 έως $400803

Δείγμα Εντολής του M68000

LABEL (Ετικέτα)	**OPCODE** (Κωδικός Εντολής)	**OPERANDS** (Τελεστέοι)	**COMMENTS** (Σχόλια)
LOOP	**MOVE .L**	**D0,D1**	* αντέγραψε το περιεχόμενο του * καταχωρητή D0 στον * καταχωρητή D1.

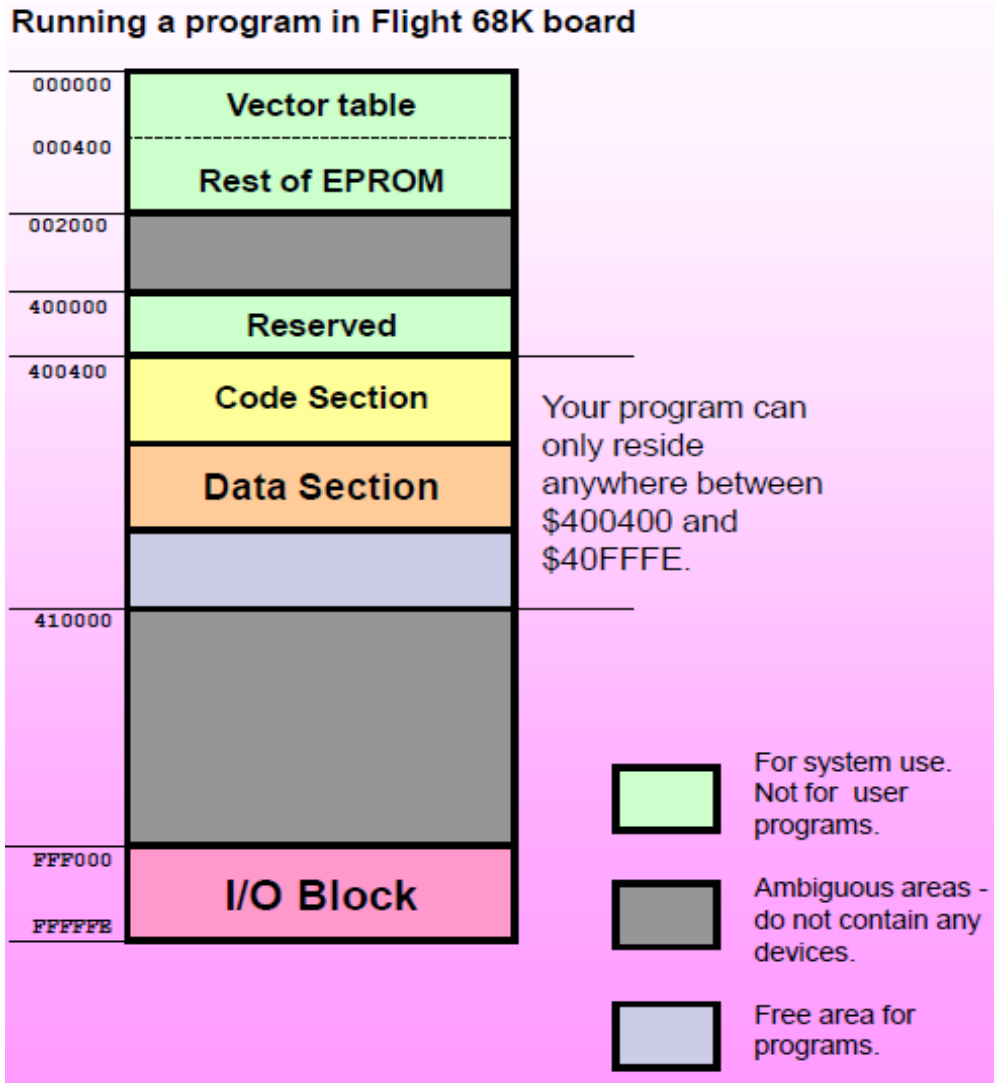
Τελεστέος Προέλευσης

Τελεστέος Προόρισμού

Δείγμα Προγράμματος του M68000

Ετικέτα	Κωδ.Εντ.	Τελεστέοι	Σχόλια
ADDNUMS:	MOVE.B	$400400,D0	*Πάρε στον D0 και το D1 τα
	MOVE.B	$400401,D1	*byte των θέσεων $400400 και
			*$400401 αντίστοιχα
	ADD.B	D0,D1	*Πρόσθεσέ τα.
	MOVE.B	D1,$400402	*Αποθήκευσε το αποτέλεσμα
			*στη θέση $400402.

Memory Map for Programs



***Microprocessor: Motorola 68K (SEE3223-06)**

Ψευδοεντολές – Assembler Directives

Η ψευδοεντολή ORG (ORiGin) καθορίζει την τιμή που θα πάρει ο μετρητής προγράμματος δείχνοντας τη θέση μνήμης, όπου θα αποθηκευτεί η πρώτη εντολή του κώδικα ή τα πρώτα δεδομένα μετά τη συμβολομετάφραση.

Η ψευδοεντολή EQU (EQUate) δίνει (εξισώνει) ένα όνομα σε μια τιμή.

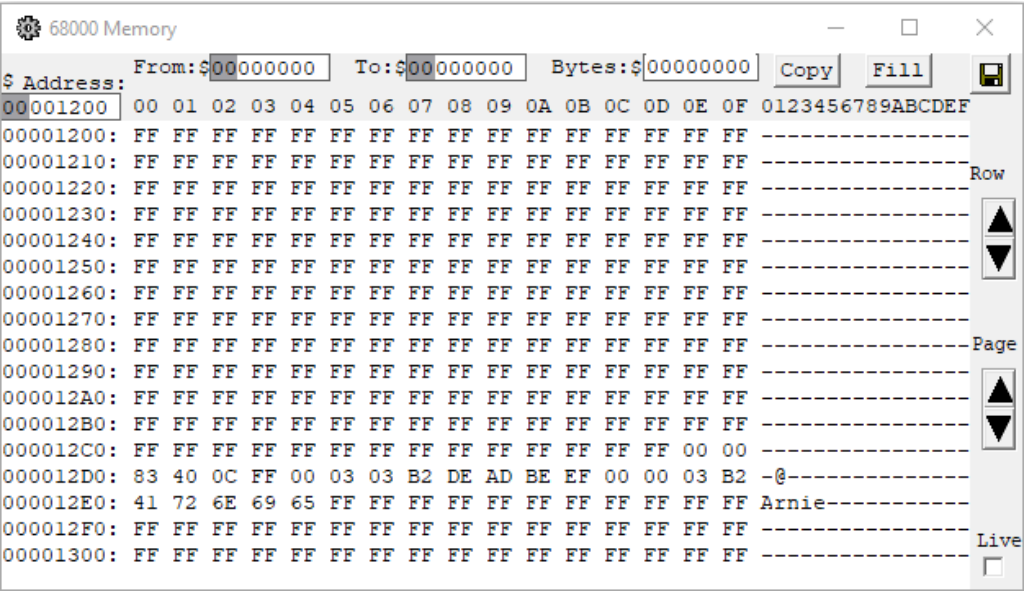
Η ψευδοεντολή DC (Define a Constant) επιτρέπει στον προγραμματιστή να καθορίσει μια σταθερά που θα φορτωθεί στη μνήμη πριν αρχίσει η εκτέλεση του προγράμματος.

Η ψευδοεντολή DS (Define Storage) κρατά ελεύθερες θέσεις μνήμης στις οποίες πρόκειται να αποθηκευτούν μεταβλητές που θα παραχθούν κατά την εκτέλεση του προγράμματος.

Η ψευδοεντολή END δηλώνει στο συμβολομεταφραστή ότι οι εντολές του προγράμματος έχουν τελειώσει και δεν υπάρχουν άλλες εντολές ή ψευδοεντολές για συμβολομετάφραση.

Οι Ψευδοεντολές δε μεταφράζονται σε κώδικα μηχανής απλά δίνουν πληροφορίες στον Assembler για το πρόγραμμα και το περιβάλλον του.

	ORG $001200	Sets the origin of the data area at address $001200
C	**DS.B 1**	Reserves one byte of memory and calls it C
X	**DS.W 1**	Reserves one word of memory and calls it X
Y	**DS.L 1**	Reserves one longword of memory and calls it Y
ARRAY	**DS.W 25**	Reserves 25 words of memory and calls this space ARRAY
TABLE	**DS.B $80**	Reserves 128 bytes of memory and calls them TABLE
ADRS	**DS.L 5**	Reserves 5 longwords for variable ADRS
START	**EQU $20C8**	Equate START to $20C8
OFFSET	**EQU 32**	Equate OFFSET to $0020
	DC.L 4*START+OFFSET	Store the longword 4*$20C8+$0020 in memory
	DC.B 12	The value $0C is stored in one byte of memory
	DC.W 3,$3B2	The values $0003 and $03B2 are stored in consecutive locations, each of them taking up 2 bytes
	DC.L $DEADBEEF	The value $DEADBEEF (4 bytes) is stored in memory
	DC.L $3B2	The value $000003B2 is stored in 4 bytes of memory
	DC.B 'Arnie'	The ASCII characters are stored as 5 bytes




```
SYMBOL TABLE INFORMATION
Symbol-name          Value
-------------------------
ADRS                 12BA
ARRAY                1208
C                    1200
OFFSET               20
START                20C8
TABLE                123A
X                    1202
Y                    1204
```

ΠΑΡΑΔΕΙΓΜΑ Μ.1

Να γραφτεί ένα πρόγραμμα που θα χρησιμοποιεί τις ψευδοεντολές που αναφέρθηκαν και θα προσθέτει δύο αριθμούς μήκους byte, που είναι αποθηκευμένοι στις θέσεις μνήμης $400400 και $400401.

Το αποτέλεσμα θα το αποθηκεύει στη θέση μνήμης $400402.

```
*Data Section
            ORG         $400400
NUM1        DC.B        5
NUM2        DC.B        7
SUM         DS.B        1
*Code Section
            ORG         $400410
ADDNUMS     MOVE.B      NUM1,D0
            MOVE.B      NUM2,D1
            ADD.B       D0,D1
            MOVE.B      D1,SUM
            END         $400410
```

Μετά τη συμβολομετάφραση ο συμβολομεταφραστής θα δώσει την παρακάτω λίστα (Listing File):

```
1  00400400                          ORG       $400400
2  00400400 05                       NUM1:     DC.B 5
3  00400401 07                       NUM2:     DC.B 7
4  00400402 00000001                 SUM:      DS.B  1
5
6  00400410                                    ORG     $400410
7  00400410 103900400400   ADDNUMS:  MOVE.B  NUM1,D0
8  00400416 123900400401             MOVE.B        NUM2,D1
9  0040041C D200                     ADD.B         D0,D1
10  0040041E 13C100400402            MOVE.B        D1,SUM
11  00400410                         END     $400410
```

Τα περιεχόμενα της μνήμης πριν απ' την εκτέλεση του προγράμματος θα είναι:

```
400400  05 07 00 00 00 00 00 00 00 00 00 00 00 00 00 00
400410  10 39 00 40 04 00 12 39 00 40 04 01 D2 00 13 C1
400420  00 40 04 02 00 00 00 00 00 00 00 00 00 00 00 00
```

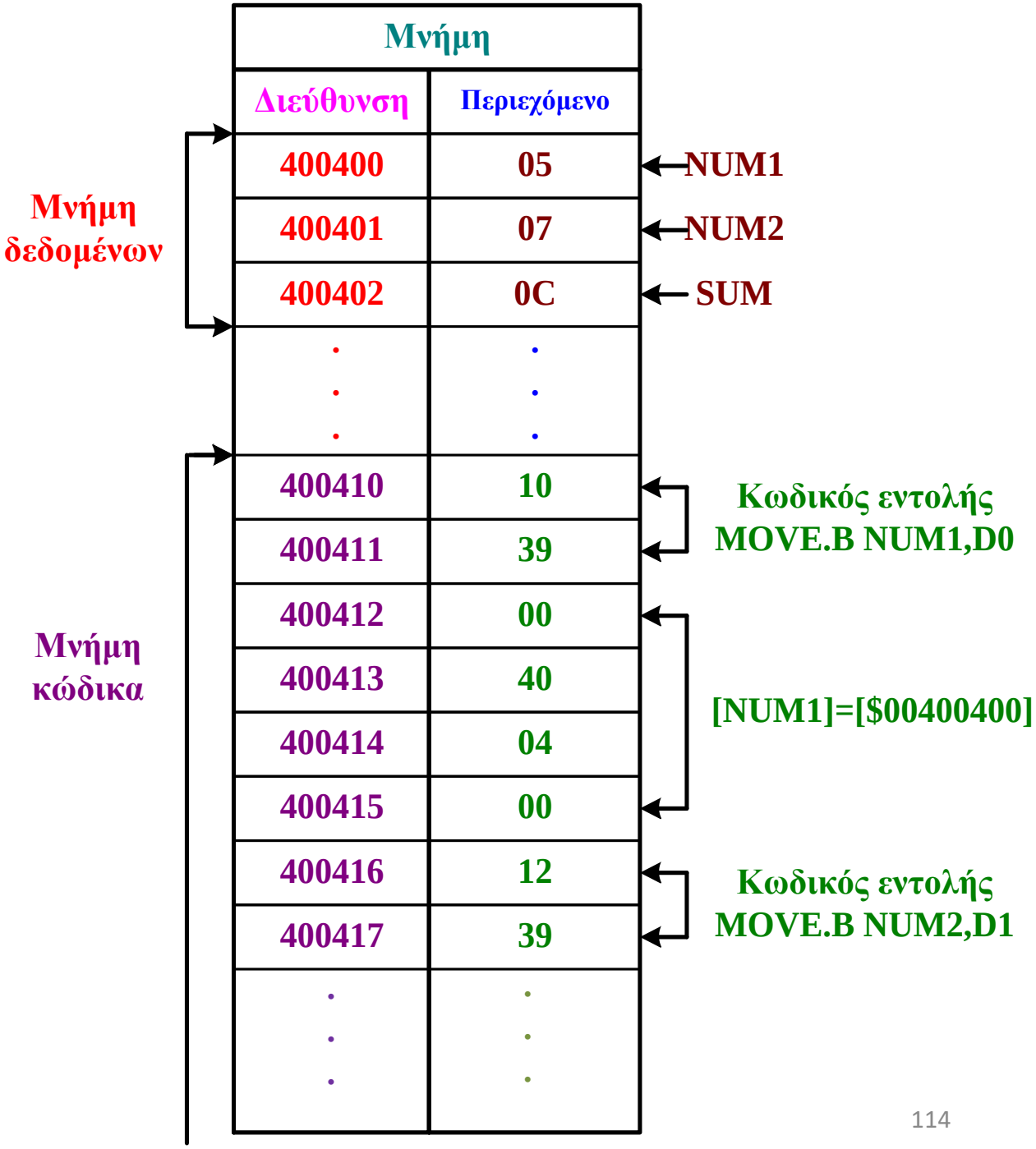
Τα περιεχόμενα της μνήμης μετά την εκτέλεση του προγράμματος θα είναι:

```
400400  05 07 0C 00 00 00 00 00 00 00 00 00 00 00 00 00
400410  10 39 00 40 04 00 12 39 00 40 04 01 D2 00 13 C1
400420  00 40 04 02 00 00 00 00 00 00 00 00 00 00 00 00
```

Περιοχή μνήμης κώδικα

Περιοχή μνήμης δεδομένων

Χαρτογράφηση μνήμης



ΠΑΡΑΔΕΙΓΜΑ Μ.2

```
 1  00400400                          ORG       $400400
 2  00400400 75134B00    NUM1:        DC.L      $75134B00
 3  00400404 3742131F    NUM2:        DC.L      $3742131F
 4
 5  00400410                          ORG       $400410
 6  00400410 203900400400             MOVE.L    NUM1,D0
 7  00400416 223900400404             MOVE.L    NUM2,D1
 8  0040041C 1200                     MOVE.B    D0,D1
 9  0040041E 3200                     MOVE.W    D0,D1
10  00400420 203C12345678             MOVE.L    #$12345678,D0
11  00400426 303C4321                 MOVE.W    #$4321,D0
12  0040042A 103C00FF                 MOVE.B    #$FF,D0
13  0040042E 72FF                     MOVEQ     #$FF,D1
14  00400430 7235                     MOVEQ     #$35,D1
15  00400432 243900400400             MOVE.L    $400400,D2
16  00400438 363900400400             MOVE.W    $400400,D3
17  0040043E 183900400400             MOVE.B    $400401,D4
18  00400410                          END       $400410
```

```
 1  00400400                          ORG         $400400
 2  00400400 75134B00    NUM1:        DC.L        $75134B00
 3  00400404 3742131F    NUM2:        DC.L        $3742131F
 4
 5  00400410                          ORG         $400410
 6  00400410 203900400400             MOVE.L      NUM1,D0
 7  00400416 223900400404             MOVE.L      NUM2,D1
 8  0040041C 1200                     MOVE.B      D0,D1
 9  0040041E 3200                     MOVE.W      D0,D1
10  00400420 203C12345678             MOVE.L      #$12345678,D0
11  00400426 303C4321                 MOVE.W      #$4321,D0
12  0040042A 103C00FF                 MOVE.B      #$FF,D0
13  0040042E 72FF                     MOVEQ       #$FF,D1
14  00400430 7235                     MOVEQ       #$35,D1
15  00400432 243900400400             MOVE.L      $400400,D2
16  00400438 363900400400             MOVE.W      $400400,D3
17  0040043E 183900400400             MOVE.B      $400401,D4
18  00400410                          END         $400410

----------->MOVE.L  $400400,D0

PC=400416      SR=2000      SS=00A00000      US=00000000      X=0

A0=00000000    A1=00000000  A2=00000000      A3=00000000      N=0

A4=00000000    A5=00000000  A6=00000000      A7=00A00000      Z=0

D0=75134B00    D1=00000000  D2=00000000      D3=00000000      V=0

D4=00000000    D5=00000000  D6=00000000      D7=00000000      C=0
```

```
 1  00400400                          ORG         $400400
 2  00400400 75134B00   NUM1:         DC.L        $75134B00
 3  00400404 3742131F   NUM2:         DC.L        $3742131F
 4
 5  00400410                          ORG         $400410
 6  00400410 203900400400             MOVE.L      NUM1,D0
 7  00400416 223900400404             MOVE.L      NUM2,D1
 8  0040041C 1200                     MOVE.B      D0,D1
 9  0040041E 3200                     MOVE.W      D0,D1
10  00400420 203C12345678             MOVE.L      #$12345678,D0
11  00400426 303C4321                 MOVE.W      #$4321,D0
12  0040042A 103C00FF                 MOVE.B      #$FF,D0
13  0040042E 72FF                     MOVEQ       #$FF,D1
14  00400430 7235                     MOVEQ       #$35,D1
15  00400432 243900400400             MOVE.L      $400400,D2
16  00400438 363900400400             MOVE.W      $400400,D3
17  0040043E 183900400400             MOVE.B      $400401,D4
18  00400410                          END         $400410
```

```
----------->MOVE.L  $400400,D0

PC=40041C       SR=2000       SS=00A00000     US=00000000     X=0

A0=00000000     A1=00000000   A2=00000000     A3=00000000     N=0

A4=00000000     A5=00000000   A6=00000000     A7=00A00000     Z=0

D0=75134B00     D1=3742131F   D2=00000000     D3=00000000     V=0

D4=00000000     D5=00000000   D6=00000000     D7=00000000     C=0
```

```
 1  00400400                          ORG         $400400
 2  00400400 75134B00    NUM1:        DC.L        $75134B00
 3  00400404 3742131F    NUM2:        DC.L        $3742131F
 4
 5  00400410                          ORG         $400410
 6  00400410 203900400400             MOVE.L      NUM1,D0
 7  00400416 223900400404             MOVE.L      NUM2,D1
 8  0040041C 1200                     MOVE.B      D0,D1
 9  0040041E 3200                     MOVE.W      D0,D1
10  00400420 203C12345678             MOVE.L      #$12345678,D0
11  00400426 303C4321                 MOVE.W      #$4321,D0
12  0040042A 103C00FF                 MOVE.B      #$FF,D0
13  0040042E 72FF                     MOVEQ       #$FF,D1
14  00400430 7235                     MOVEQ       #$35,D1
15  00400432 243900400400             MOVE.L      $400400,D2
16  00400438 363900400400             MOVE.W      $400400,D3
17  0040043E 183900400400             MOVE.B      $400401,D4
18  00400410                          END         $400410
```

```
----------->MOVE.L  $400400,D0

PC=40041E       SR=2000       SS=00A00000     US=00000000     X=0

A0=00000000     A1=00000000   A2=00000000     A3=00000000     N=0

A4=00000000     A5=00000000   A6=00000000     A7=00A00000     Z=1

D0=75134B00     D1=37421300   D2=00000000     D3=00000000     V=0

D4=00000000     D5=00000000   D6=00000000     D7=00000000     C=0
```

```
 1  00400400                          ORG         $400400
 2  00400400 75134B00    NUM1:        DC.L        $75134B00
 3  00400404 3742131F    NUM2:        DC.L        $3742131F
 4
 5  00400410                          ORG         $400410
 6  00400410 203900400400             MOVE.L      NUM1,D0
 7  00400416 223900400404             MOVE.L      NUM2,D1
 8  0040041C 1200                     MOVE.B      D0,D1
 9  0040041E 3200                     MOVE.W      D0,D1
10  00400420 203C12345678             MOVE.L      #$12345678,D0
11  00400426 303C4321                 MOVE.W      #$4321,D0
12  0040042A 103C00FF                 MOVE.B      #$FF,D0
13  0040042E 72FF                     MOVEQ       #$FF,D1
14  00400430 7235                     MOVEQ       #$35,D1
15  00400432 243900400400             MOVE.L      $400400,D2
16  00400438 363900400400             MOVE.W      $400400,D3
17  0040043E 183900400400             MOVE.B      $400401,D4
18  00400410                          END         $400410
```

```
----------->MOVE.L  $400400,D0

PC=400420       SR=2000       SS=00A00000     US=00000000     X=0

A0=00000000     A1=00000000   A2=00000000     A3=00000000     N=0

A4=00000000     A5=00000000   A6=00000000     A7=00A00000     Z=0

D0=75134B00     D1=37424B00   D2=00000000     D3=00000000     V=0

D4=00000000     D5=00000000   D6=00000000     D7=00000000     C=0
```

```
 1  00400400                          ORG         $400400
 2  00400400 75134B00    NUM1:        DC.L        $75134B00
 3  00400404 3742131F    NUM2:        DC.L        $3742131F
 4
 5  00400410                          ORG         $400410
 6  00400410 203900400400             MOVE.L      NUM1,D0
 7  00400416 223900400404             MOVE.L      NUM2,D1
 8  0040041C 1200                     MOVE.B      D0,D1
 9  0040041E 3200                     MOVE.W      D0,D1
10  00400420 203C12345678             MOVE.L      #$12345678,D0
11  00400426 303C4321                 MOVE.W      #$4321,D0
12  0040042A 103C00FF                 MOVE.B      #$FF,D0
13  0040042E 72FF                     MOVEQ       #$FF,D1
14  00400430 7235                     MOVEQ       #$35,D1
15  00400432 243900400400             MOVE.L      $400400,D2
16  00400438 363900400400             MOVE.W      $400400,D3
17  0040043E 183900400400             MOVE.B      $400401,D4
18  00400410                          END         $400410
```

```
---------->MOVE.L  $400400,D0

PC=400426       SR=2000     SS=00A00000     US=00000000     X=0

A0=00000000     A1=00000000 A2=00000000     A3=00000000     N=0

A4=00000000     A5=00000000 A6=00000000     A7=00A00000     Z=0

D0=12345678     D1=3742131F D2=00000000     D3=00000000     V=0

D4=00000000     D5=00000000 D6=00000000     D7=00000000     C=0
```

```
 1  00400400                          ORG         $400400
 2  00400400 75134B00    NUM1:        DC.L        $75134B00
 3  00400404 3742131F    NUM2:        DC.L        $3742131F
 4
 5  00400410                          ORG         $400410
 6  00400410 203900400400             MOVE.L      NUM1,D0
 7  00400416 223900400404             MOVE.L      NUM2,D1
 8  0040041C 1200                     MOVE.B      D0,D1
 9  0040041E 3200                     MOVE.W      D0,D1
10  00400420 203C12345678             MOVE.L      #$12345678,D0
11  00400426 303C4321                 MOVE.W      #$4321,D0
12  0040042A 103C00FF                 MOVE.B      #$FF,D0
13  0040042E 72FF                     MOVEQ       #$FF,D1
14  00400430 7235                     MOVEQ       #$35,D1
15  00400432 243900400400             MOVE.L      $400400,D2
16  00400438 363900400400             MOVE.W      $400400,D3
17  0040043E 183900400400             MOVE.B      $400401,D4
18  00400410                          END         $400410
```

----------->MOVE.L $400400,D0

PC=40042A SR=2000 SS=00A00000 US=00000000 X=0

A0=00000000 A1=00000000 A2=00000000 A3=00000000 N=0

A4=00000000 A5=00000000 A6=00000000 A7=00A00000 Z=0

D0=12344321 D1=3742131F D2=00000000 D3=00000000 V=0

D4=00000000 D5=00000000 D6=00000000 D7=00000000 C=0

```
 1  00400400                          ORG         $400400
 2  00400400 75134B00    NUM1:        DC.L        $75134B00
 3  00400404 3742131F    NUM2:        DC.L        $3742131F
 4
 5  00400410                          ORG         $400410
 6  00400410 203900400400             MOVE.L      NUM1,D0
 7  00400416 223900400404             MOVE.L      NUM2,D1
 8  0040041C 1200                     MOVE.B      D0,D1
 9  0040041E 3200                     MOVE.W      D0,D1
10  00400420 203C12345678             MOVE.L      #$12345678,D0
11  00400426 303C4321                 MOVE.W      #$4321,D0
12  0040042A 103C00FF                 MOVE.B      #$FF,D0
13  0040042E 72FF                     MOVEQ       #$FF,D1
14  00400430 7235                     MOVEQ       #$35,D1
15  00400432 243900400400             MOVE.L      $400400,D2
16  00400438 363900400400             MOVE.W      $400400,D3
17  0040043E 183900400400             MOVE.B      $400401,D4
18  00400410                          END         $400410

----------->MOVE.L  $400400,D0

PC=40042E       SR=2000       SS=00A00000     US=00000000     X=0
A0=00000000     A1=00000000   A2=00000000     A3=00000000     N=1
A4=00000000     A5=00000000   A6=00000000     A7=00A00000     Z=0
D0=751343FF     D1=3742131F   D2=00000000     D3=00000000     V=0
D4=00000000     D5=00000000   D6=00000000     D7=00000000     C=0
```

d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0
1	1	1	1	1	1	1	1

F F

```
 1  00400400                          ORG         $400400
 2  00400400 75134B00    NUM1:        DC.L        $75134B00
 3  00400404 3742131F    NUM2:        DC.L        $3742131F
 4
 5  00400410                          ORG         $400410
 6  00400410 203900400400             MOVE.L      NUM1,D0
 7  00400416 223900400404             MOVE.L      NUM2,D1
 8  0040041C 1200                     MOVE.B      D0,D1
 9  0040041E 3200                     MOVE.W      D0,D1
10  00400420 203C12345678             MOVE.L      #$12345678,D0
11  00400426 303C4321                 MOVE.W      #$4321,D0
12  0040042A 103C00FF                 MOVE.B      #$FF,D0
13  0040042E 72FF                     MOVEQ       #$FF,D1
14  00400430 7235                     MOVEQ       #$35,D1
15  00400432 243900400400             MOVE.L      $400400,D2
16  00400438 363900400400             MOVE.W      $400400,D3
17  0040043E 183900400400             MOVE.B      $400401,D4
18  00400410                          END         $400410

----------->MOVE.L  $400400,D0

PC=400430      SR=2000      SS=00A00000      US=00000000      X=0
A0=00000000    A1=00000000  A2=00000000      A3=00000000      N=1
A4=00000000    A5=00000000  A6=00000000      A7=00A00000      Z=0
D0=751343FF    D1=FFFFFFFF  D2=00000000      D3=00000000      V=0
D4=00000000    D5=00000000  D6=00000000      D7=00000000      C=0
```

d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0
1	1	1	1	1	1	1	1
	F					F	

d_{31}	d_{30}	d_{29}	d_{28}	d_{27}	d_{26}	d_{25}	d_{24}	. . . . .	d_{11}	d_{10}	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0
1	1	1	1	1	1	1	1	. . . . .	1	1	1	1	1	1	1	1	1	1	1	1
	F				F					F				F				F		

```
 1  00400400                        ORG         $400400
 2  00400400 75134B00    NUM1:      DC.L        $75134B00
 3  00400404 3742131F    NUM2:      DC.L        $3742131F
 4
 5  00400410                        ORG         $400410
 6  00400410 203900400400           MOVE.L      NUM1,D0
 7  00400416 223900400404           MOVE.L      NUM2,D1
 8  0040041C 1200                   MOVE.B      D0,D1
 9  0040041E 3200                   MOVE.W      D0,D1
10  00400420 203C12345678           MOVE.L      #$12345678,D0
11  00400426 303C4321               MOVE.W      #$4321,D0
12  0040042A 103C00FF               MOVE.B      #$FF,D0
13  0040042E 72FF                   MOVEQ       #$FF,D1
14  00400430 7235                   MOVEQ       #$35,D1
15  00400432 243900400400           MOVE.L      $400400,D2
16  00400438 363900400400           MOVE.W      $400400,D3
17  0040043E 183900400400           MOVE.B      $400401,D4
18  00400410                        END         $400410
----------->MOVE.L  $400400,D0

PC=400432       SR=2000       SS=00A00000     US=00000000     X=0

A0=00000000     A1=00000000   A2=00000000     A3=00000000     N=0

A4=75134B2C     A5=00000000   A6=00000000     A7=00A00000     Z=0

D0=751343FF     D1=00000035   D2=00000000     D3=00000000     V=0

D4=00000000     D5=00000000   D6=00000000     D7=00000000     C=0
```

d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0
0	0	1	1	0	1	0	1

3 5

d_{31}	d_{30}	d_{29}	d_{28}	d_{27}	d_{26}	d_{25}	d_{24}	. . . . .	d_{11}	d_{10}	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0
0	0	0	0	0	0	0	0	. . . . .	0	0	0	0	0	0	1	1	0	1	0	1

0 0 0 0 0 0 3 5

```
 1  00400400                          ORG         $400400
 2  00400400 75134B00    NUM1:        DC.L        $75134B00
 3  00400404 3742131F    NUM2:        DC.L        $3742131F
 4
 5  00400410                          ORG         $400410
 6  00400410 203900400400             MOVE.L      NUM1,D0
 7  00400416 223900400404             MOVE.L      NUM2,D1
 8  0040041C 1200                     MOVE.B      D0,D1
 9  0040041E 3200                     MOVE.W      D0,D1
10  00400420 203C12345678             MOVE.L      #$12345678,D0
11  00400426 303C4321                 MOVE.W      #$4321,D0
12  0040042A 103C00FF                 MOVE.B      #$FF,D0
13  0040042E 72FF                     MOVEQ       #$FF,D1
14  00400430 7235                     MOVEQ       #$35,D1
15  00400432 243900400400             MOVE.L      $400400,D2
16  00400438 363900400400             MOVE.W      $400400,D3
17  0040043E 183900400400             MOVE.B      $400401,D4
18  00400410                          END         $400410
----------->MOVE.L  $400400,D0

PC=400438       SR=2000       SS=00A00000     US=00000000     X=0

A0=00000000     A1=00000000   A2=00000000     A3=00000000     N=0

A4=75134B2C     A5=00000000   A6=00000000     A7=00A00000     Z=0

D0=751343FF     D1=00000035   D2=75134B00     D3=00000000     V=0

D4=00000000     D5=00000000   D6=00000000     D7=00000000     C=0
```

```
 1  00400400                          ORG         $400400
 2  00400400 75134B00    NUM1:        DC.L        $75134B00
 3  00400404 3742131F    NUM2:        DC.L        $3742131F
 4
 5  00400410                          ORG         $400410
 6  00400410 203900400400             MOVE.L      NUM1,D0
 7  00400416 223900400404             MOVE.L      NUM2,D1
 8  0040041C 1200                     MOVE.B      D0,D1
 9  0040041E 3200                     MOVE.W      D0,D1
10  00400420 203C12345678             MOVE.L      #$12345678,D0
11  00400426 303C4321                 MOVE.W      #$4321,D0
12  0040042A 103C00FF                 MOVE.B      #$FF,D0
13  0040042E 72FF                     MOVEQ       #$FF,D1
14  00400430 7235                     MOVEQ       #$35,D1
15  00400432 243900400400             MOVE.L      $400400,D2
16  00400438 363900400400             MOVE.W      $400400,D3
17  0040043E 183900400400             MOVE.B      $400401,D4
18  00400410                          END         $400410
```

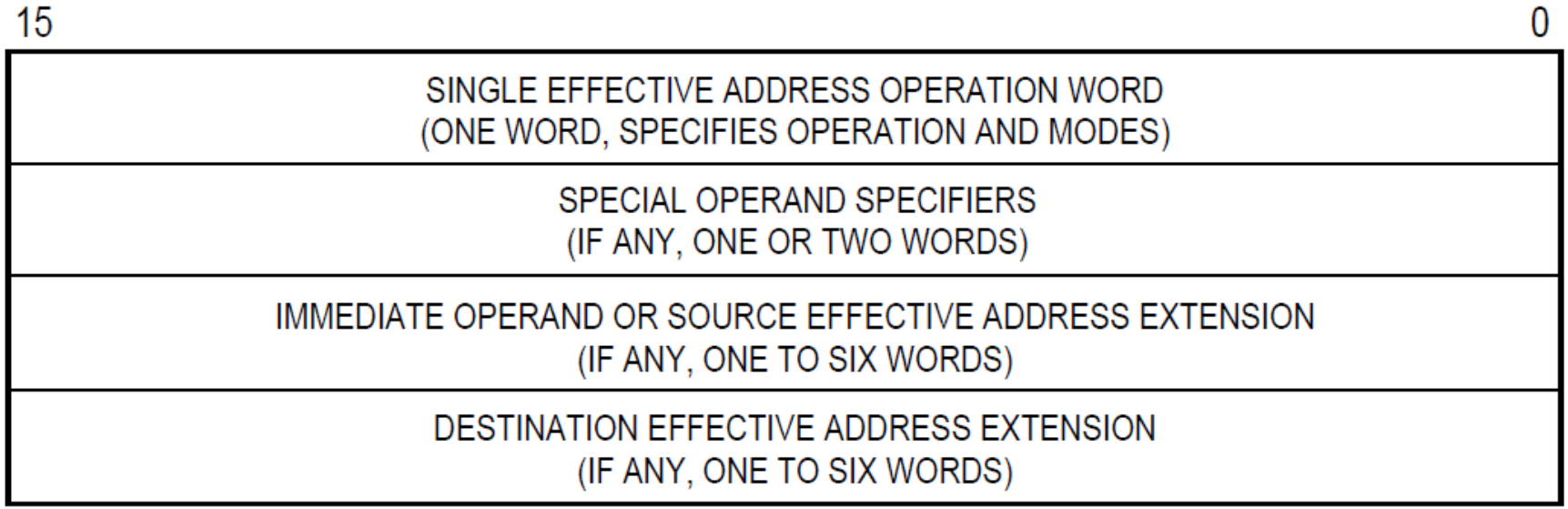
```
---------->MOVE.L   $400400,D0

PC=40043E       SR=2000       SS=00A00000     US=00000000     X=0

A0=00000000     A1=00000000   A2=00000000     A3=00000000     N=0

A4=75134B2C     A5=00000000   A6=00000000     A7=00A00000     Z=0

D0=751343FF     D1=00000035   D2=75134B00     D3=00007513     V=0

D4=00000000     D5=00000000   D6=00000000     D7=00000000     C=0
```

```
 1  00400400                          ORG         $400400
 2  00400400 75134B00    NUM1:        DC.L        $75134B00
 3  00400404 3742131F    NUM2:        DC.L        $3742131F
 4
 5  00400410                          ORG         $400410
 6  00400410 203900400400             MOVE.L      NUM1,D0
 7  00400416 223900400404             MOVE.L      NUM2,D1
 8  0040041C 1200                     MOVE.B      D0,D1
 9  0040041E 3200                     MOVE.W      D0,D1
10  00400420 203C12345678             MOVE.L      #$12345678,D0
11  00400426 303C4321                 MOVE.W      #$4321,D0
12  0040042A 103C00FF                 MOVE.B      #$FF,D0
13  0040042E 72FF                     MOVEQ       #$FF,D1
14  00400430 7235                     MOVEQ       #$35,D1
15  00400432 243900400400             MOVE.L      $400400,D2
16  00400438 363900400400             MOVE.W      $400400,D3
17  0040043E 183900400400             MOVE.B      $400401,D4
18  00400410                          END         $400410
```

```
---------->MOVE.L  $400400,D0

PC=400410       SR=2000      SS=00A00000     US=00000000     X=0

A0=00000000     A1=00000000  A2=00000000     A3=00000000     N=0

A4=75134B2C     A5=00000000  A6=00000000     A7=00A00000     Z=0

D0=751343FF     D1=00000035  D2=75134B00     D3=00007513     V=0

D4=00000013     D5=00000000  D6=00000000     D7=00000000     C=0
```

Μορφή εντολής – Instruction Format

- Μια εντολή στους επεξεργαστές της οικογένειας M68000 αποτελείτε από τουλάχιστον μία (1) λέξη word, συνήθως είναι μέχρι πέντε (5) και μπορεί να φθάσει μέχρι και έντεκα (11) λέξεις.
- Η πρώτη λέξη της εντολής, ονομάζεται **Λέξη Λειτουργίας** (**Single Effective Address Operation Word**), καθορίζει το μήκος της εντολής, τη διευθυνσιοδότηση και τη λειτουργία που θα εκτελεστεί.
- Οι υπόλοιπες λέξεις, που ονομάζονται σύντομες και πλήρους επέκτασης λέξεις (brief and full extension words) καθορίζουν με τη σειρά τους την εντολή και τους τελεστέους.
- Το γράφημα παρακάτω παρουσιάζει τη γενική σύνθεση μιας εντολής.



Μνημονικά - Mnemonics

MOVE.W #$04F0,D0

Μια εντολή Assembly M68000 αποτελείται από τέσσερα (4) μέρη:

Εντολή (Instruction - Command): Υπάρχουν 56 διαθέσιμες όπως: MOVE, ADD, SUB, DIVU, MULU, BRA ή JMP.

Μέγεθος (Size): Εδώ ορίζουμε τι μέγεθος δεδομένων θα διαχειριστεί η εντολή μας. Έχουμε .B για byte, .W για word, ή .L για long-word.

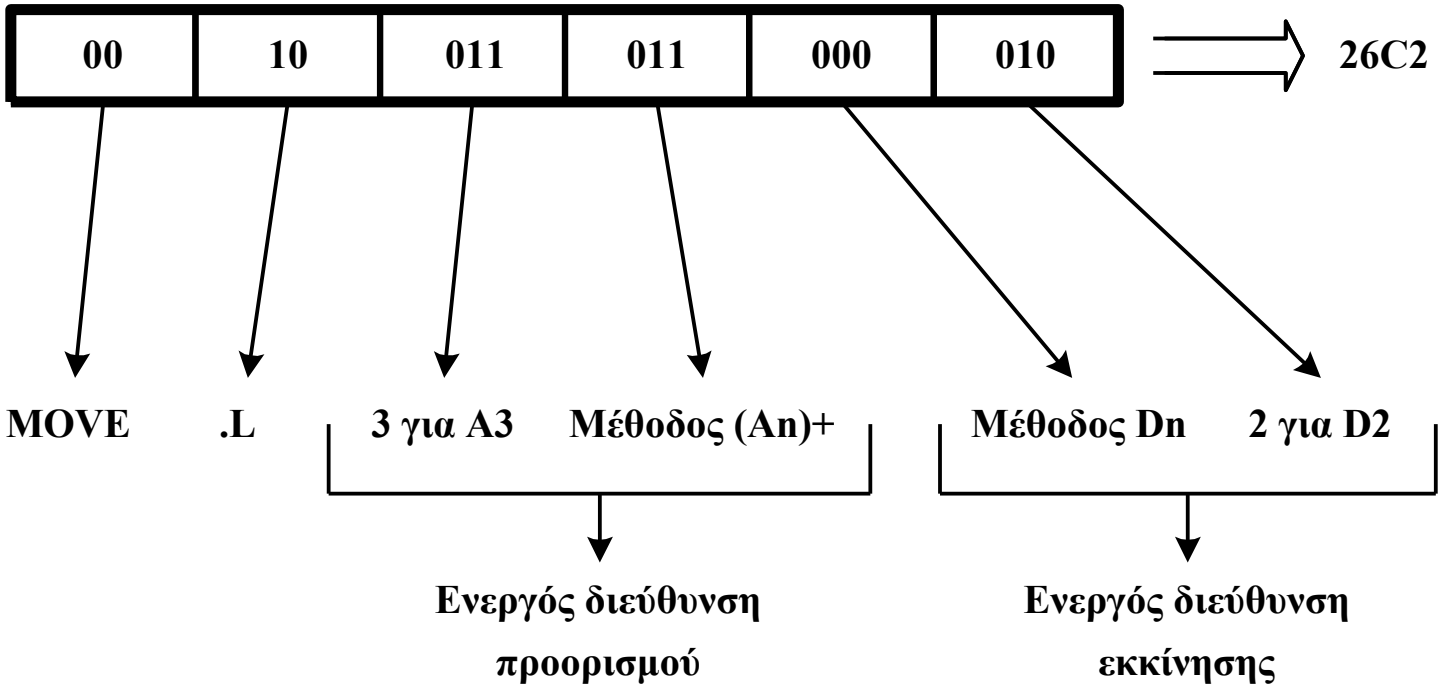
Τελεστέος Πηγής ή Προέλευσης (Source Operand): Από πού διαβάζεται η τιμή ή ποια τιμή πρόκειται να χρησιμοποιηθεί.

Τελεστέος Προορισμού (Destination Operand): Όπου η τιμή μετακινείται ή επεξεργάζεται/χειρίζεται.

Παράδειγμα

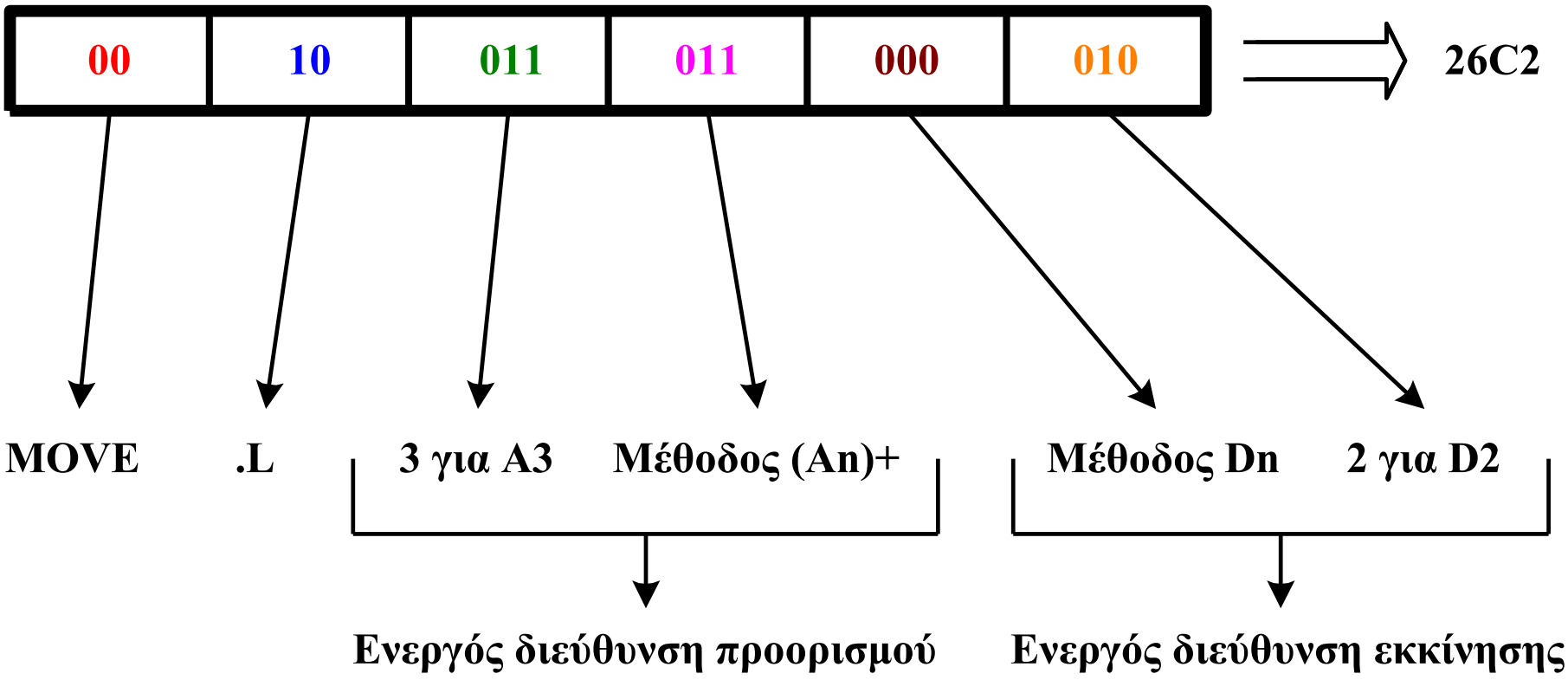
Να σχηματιστεί ο κωδικός της εντολής MOVE.L D2, (A3)+

d15	d14	d13	d12	d11	d10	d9	d8	d7	d6	d5	d4	d3	d2	d1	d0
0	0	ΜΕΓΕΘΟΣ		ΠΡΟΟΡΙΣΜΟΣ						ΠΡΟΕΛΕΥΣΗ					
				ΚΑΤΑΧΩΡΗΤΗΣ			ΜΟΡΦΗ			ΜΟΡΦΗ			ΚΑΤΑΧΩΡΗΤΗΣ		

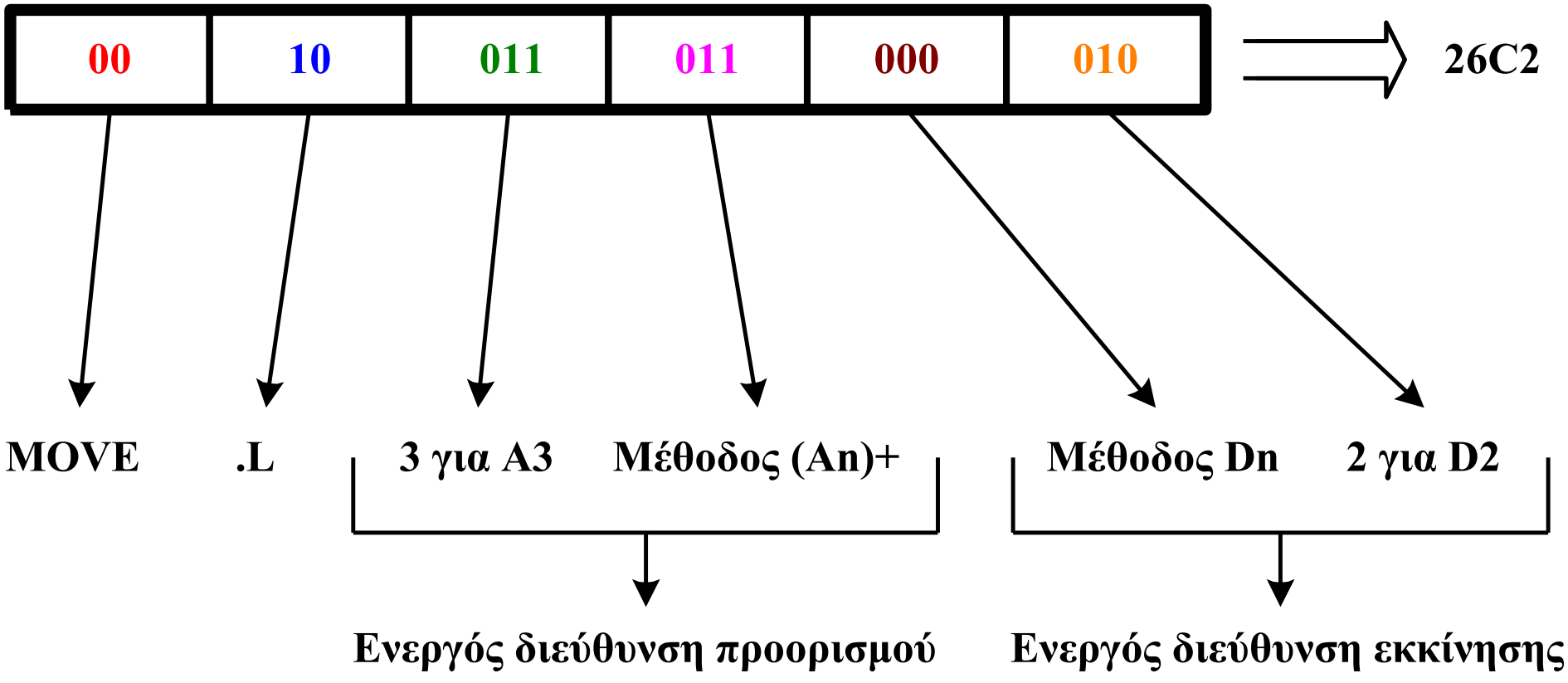


Κωδικοποίηση διευθυνσιοδοτήσεων

Addressing Modes	Syntax	Mode Field	Reg. Field	Data	Memory	Control	Alterable
Register Direct							
Data	Dn	000	reg. no.	X	—	—	X
Address	An	001	reg. no.	—	—	—	X
Register Indirect							
Address	(An)	010	reg. no.	X	X	X	X
Address with Postincrement	(An)+	011	reg. no.	X	X	—	X
Address with Predecrement	–(An)	100	reg. no.	X	X	—	X
Address with Displacement	(d_{16},An)	101	reg. no.	X	X	X	X
Address Register Indirect with Index							
8-Bit Displacement	(d_8,An,Xn)	110	reg. no.	X	X	X	X
Base Displacement	(bd,An,Xn)	110	reg. no.	X	X	X	X
Memory Indirect							
Postindexed	([bd,An],Xn,od)	110	reg. no.	X	X	X	X
Preindexed	([bd,An,Xn],od)	110	reg. no.	X	X	X	X
Program Counter Indirect with Displacement	(d_{16},PC)	111	010	X	X	X	—
Program Counter Indirect with Index							
8-Bit Displacement	(d_8,PC,Xn)	111	011	X	X	X	—
Base Displacement	(bd,PC,Xn)	111	011	X	X	X	—
Program Counter Memory Indirect							
Postindexed	([bd,PC],Xn,od)	111	011	X	X	X	X
Preindexed	([bd,PC,Xn],od)	111	011	X	X	X	X
Absolute Data Addressing							
Short	(xxx).W	111	000	X	X	X	—
Long	(xxx).L	111	001	X	X	X	—
Immediate	#<xxx>	111	100	X	X	—	—

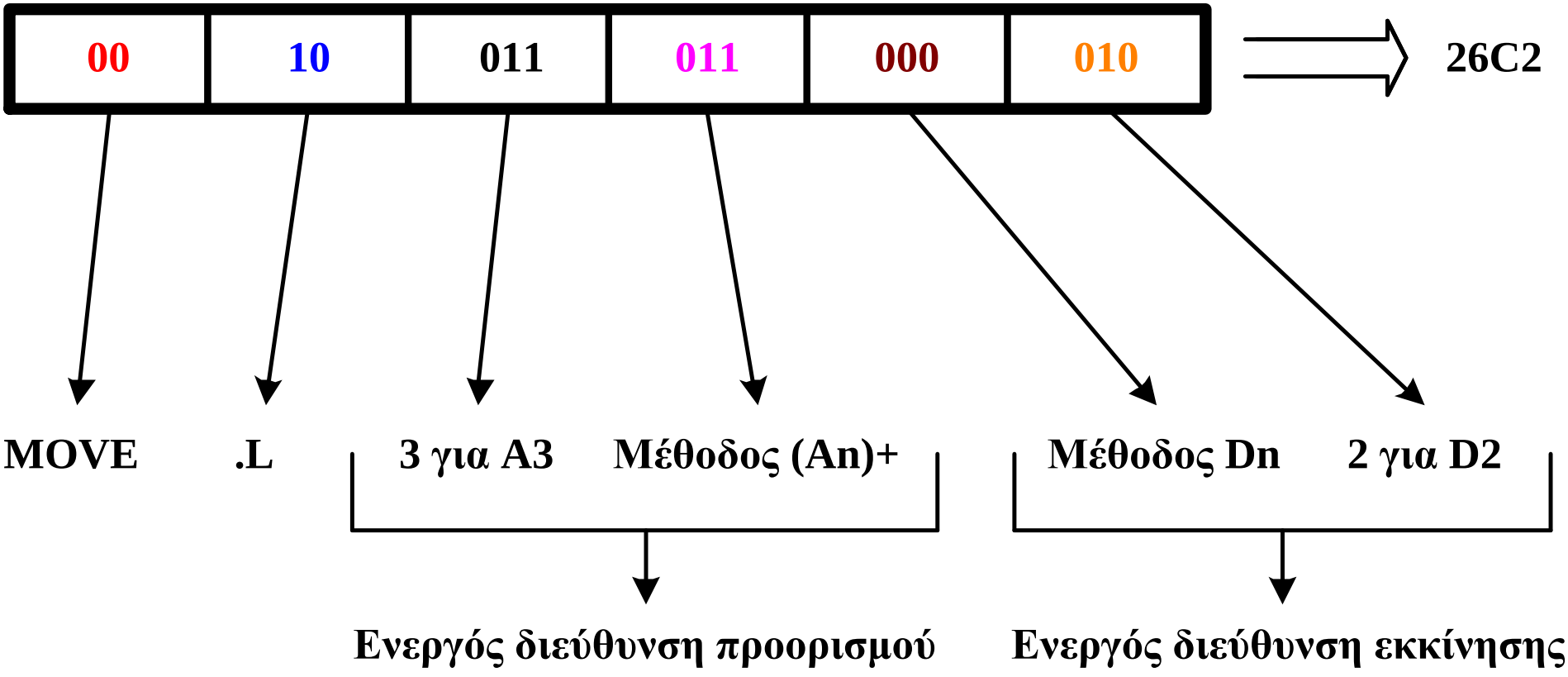


Τα ψηφία $d_{15}d_{14}$ αναφέρονται στην εντολή MOVE και έχουν σταθερή τιμή 00_2.

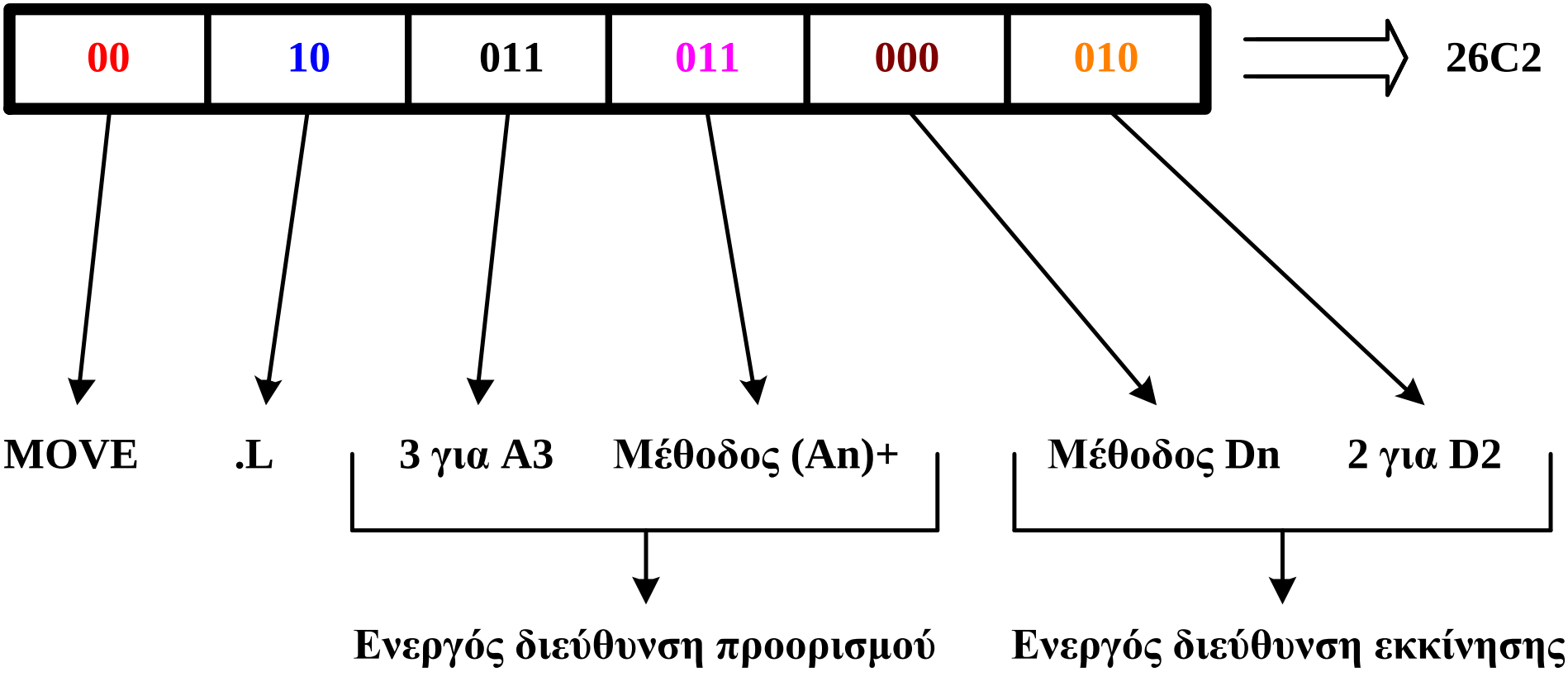


Τα ψηφία $d_{13}d_{12}$ αναφέρονται στο μέγεθος της λέξης που η εντολή πρόκειται να επεξεργαστεί. Το μέγεθος είναι μακριά λέξη και άρα τα ψηφία αυτά θα πάρουν την τιμή 10_2.

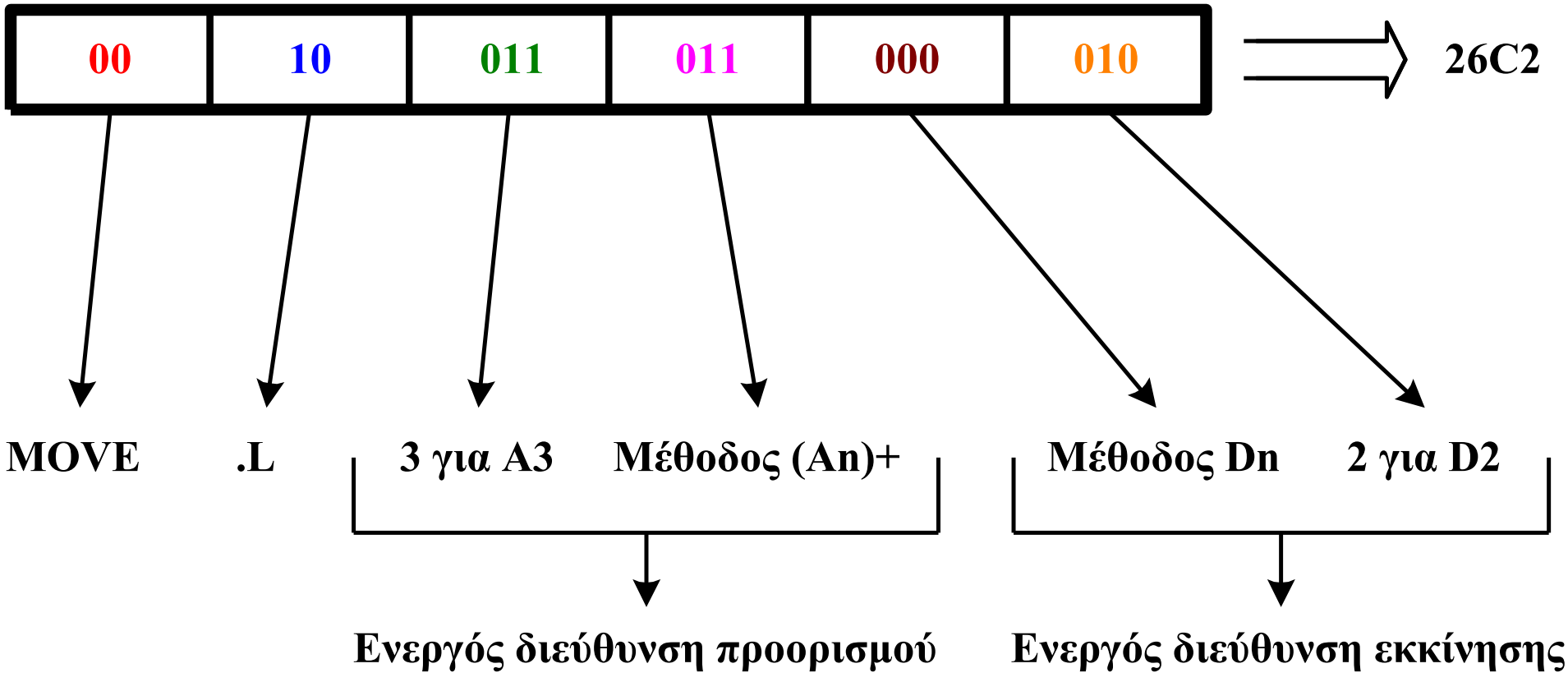
Πεδίο μεγέθους
01-Λειτουργία Byte
11-Λειτουργία Λέξης
10-Λειτουργία Μακριάς λέξης



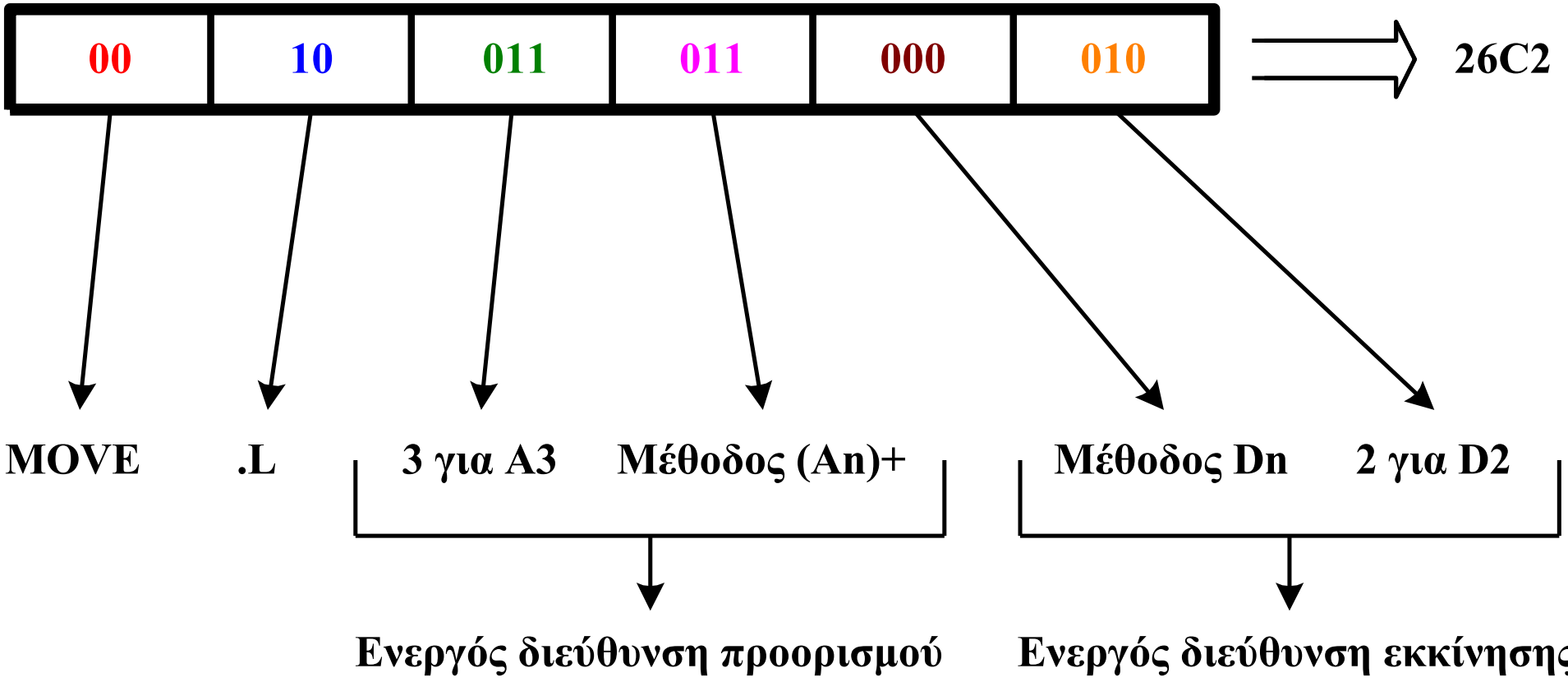
Τα ψηφία $d_{11}d_{10}d_9$ αναφέρονται στον καταχωρητή προορισμού που είναι ο *A3* και επομένως θα πάρουν την τιμή 011_2.



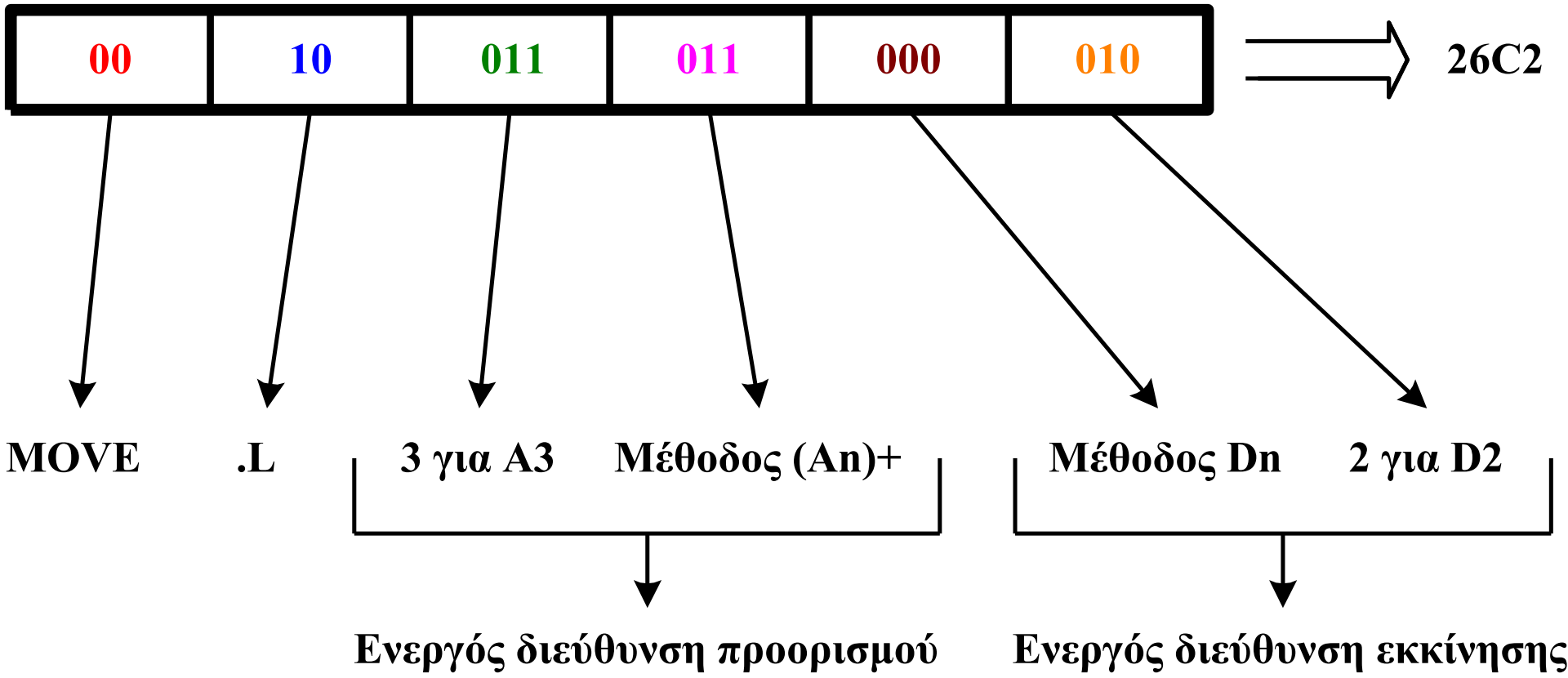
Τα ψηφία $d_8d_7d_6$ αναφέρονται στη μέθοδο διευθυνσιοδότησης του τελεστέου προορισμού που είναι έμμεση μεταυξητική καταχωρητή διεύθυνσης και επομένως θα πάρουν την τιμή 011_2.



Τα ψηφία $d_5d_4d_3$ αναφέρονται στη μέθοδο διευθυνσιοδότησης του τελεστέου προέλευσης (ή εκκίνησης) και επειδή είναι άμεση διευθυνσιοδότηση καταχωρητή δεδομένων θα πάρουν την τιμή 000_2.



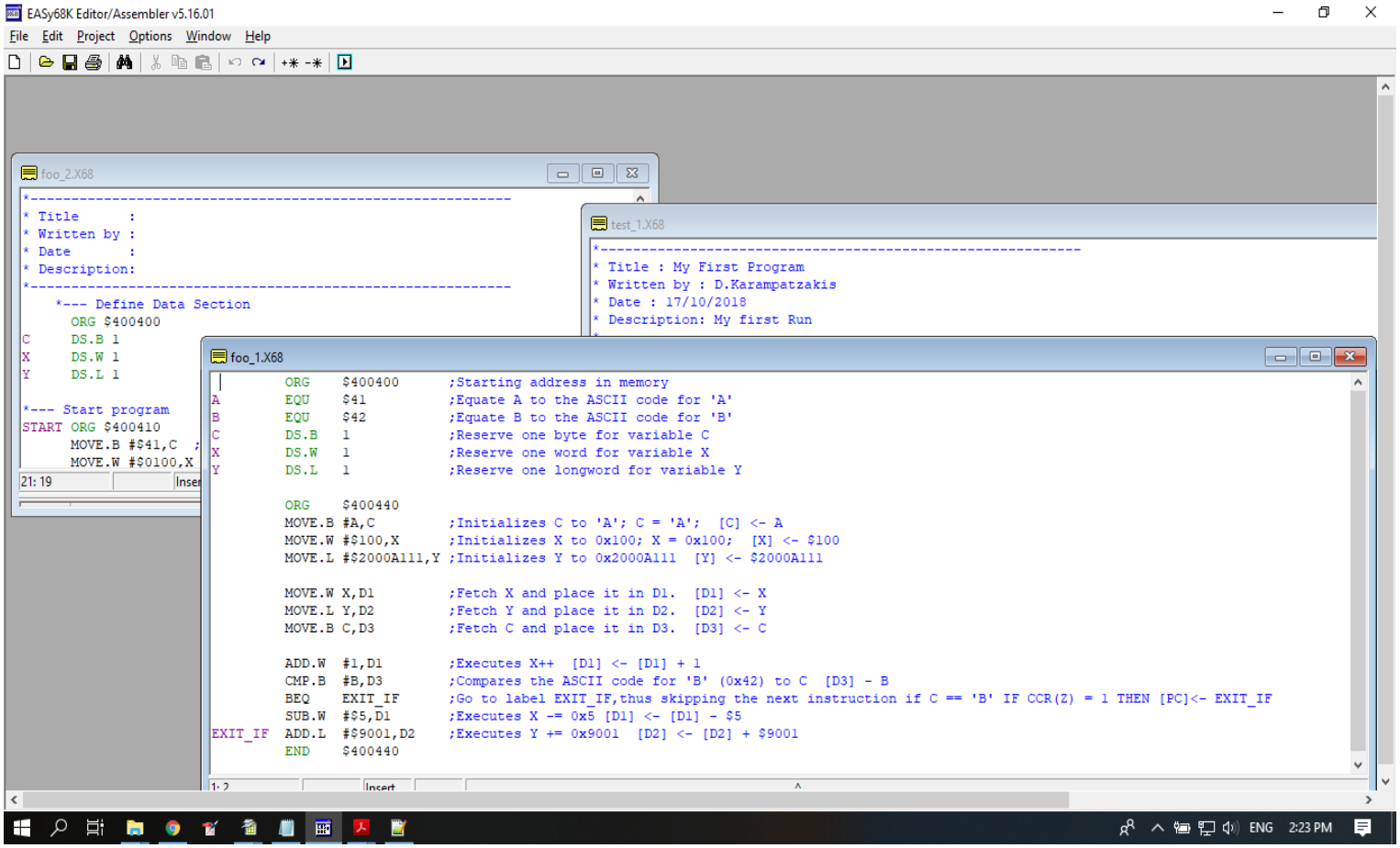
Τα τελευταία τρία ψηφία $d_2d_1d_0$ αναφέρονται στον καταχωρητή προέλευσης που είναι ο *D2* και επομένως θα πάρουν την τιμή 010_2.



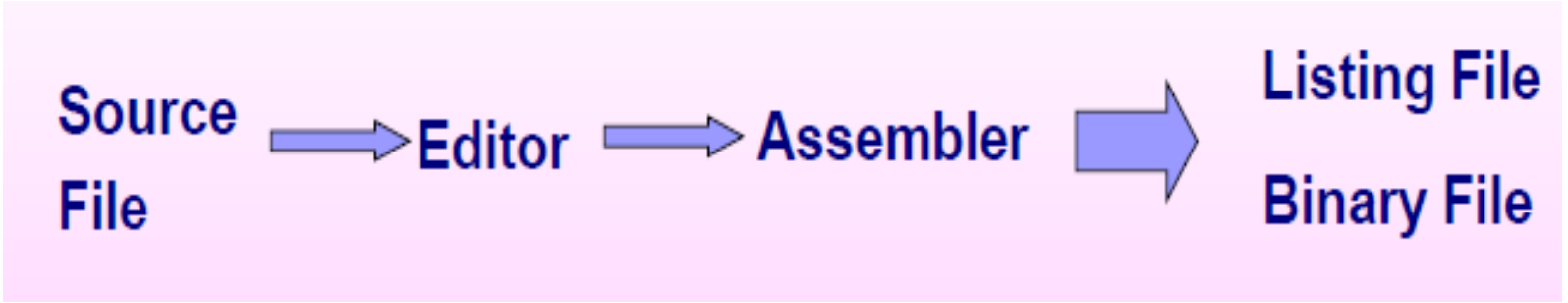
Έτσι, η λέξη που θα δώσει τον κωδικό της εντολής θα είναι:

Κωδικός εντολής=0010 0110 1100 0010_2= $26C2_{16}$

EASy68K


EASy68K Editor/Assembler v5.16.01
File Edit Project Options Window Help
foo_2.X68
* Title :
* Written by :
* Date :
* Description:
*--- Define Data Section
ORG $400400
C DS.B 1
X DS.W 1
Y DS.L 1
*--- Start program
START ORG $400410
MOVE.B #$41,C
MOVE.W #$0100,X
21: 19
test_1.X68
* Title : My First Program
* Written by : D.Karampatzakis
* Date : 17/10/2018
* Description: My first Run
foo_1.X68
ORG $400400 ;Starting address in memory
A EQU $41 ;Equate A to the ASCII code for 'A'
B EQU $42 ;Equate B to the ASCII code for 'B'
C DS.B 1 ;Reserve one byte for variable C
X DS.W 1 ;Reserve one word for variable X
Y DS.L 1 ;Reserve one longword for variable Y
ORG $400440
MOVE.B #A,C ;Initializes C to 'A'; C = 'A'; [C] <- A
MOVE.W #$100,X ;Initializes X to 0x100; X = 0x100; [X] <- $100
MOVE.L #$2000A111,Y ;Initializes Y to 0x2000A111 [Y] <- $2000A111
MOVE.W X,D1 ;Fetch X and place it in D1. [D1] <- X
MOVE.L Y,D2 ;Fetch Y and place it in D2. [D2] <- Y
MOVE.B C,D3 ;Fetch C and place it in D3. [D3] <- C
ADD.W #1,D1 ;Executes X++ [D1] <- [D1] + 1
CMP.B #B,D3 ;Compares the ASCII code for 'B' (0x42) to C [D3] - B
BEQ EXIT_IF ;Go to label EXIT_IF,thus skipping the next instruction if C == 'B' IF CCR(Z) = 1 THEN [PC]<- EXIT_IF
SUB.W #$5,D1 ;Executes X -= 0x5 [D1] <- [D1] - $5
EXIT_IF ADD.L #$9001,D2 ;Executes Y += 0x9001 [D2] <- [D2] + $9001
END $400440
Insert
ENG 2:23 PM

Assembler



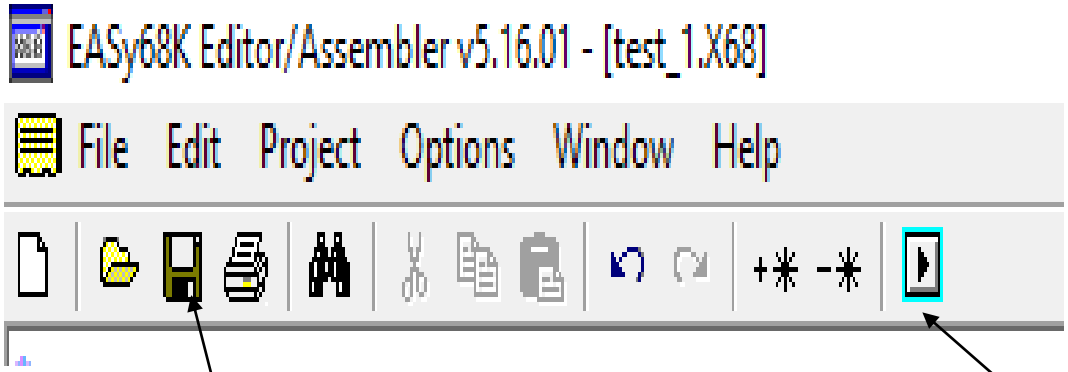
Ο **Assembler** δέχεται το πηγαίο αρχείο:

- **Source File:** με επέκταση (.X68 ή .ASM) το πηγαίο αρχείο με τον κώδικα assembly,

και παράγει δύο (2) αρχεία:

- **Object File/Binary File**: με επέκταση (.S68 ή .OBJ). Περιέχει των δυαδικό κώδικα για τις εντολές και πληροφορίες για τις διευθύνσεις των εντολών.
- **List File**: με επέκταση (.L68 ή .LST). Περιέχει τις δηλώσεις του προγράμματος assembly, τους δυαδικούς κώδικες για κάθε εντολή και τις πληροφορίες διευθύνσεων.

Save & Assemble Source

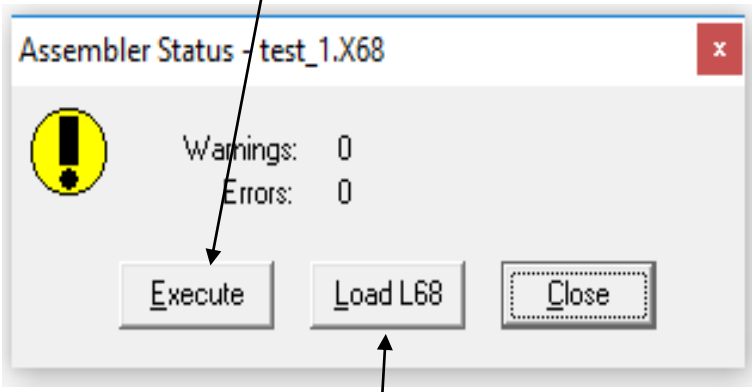


Save test_1.X68

Assemble Source (F9)

Assemble Source

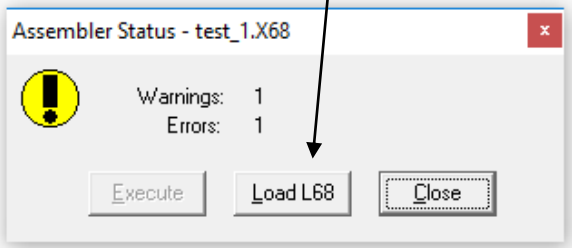
No Errors – Run Code

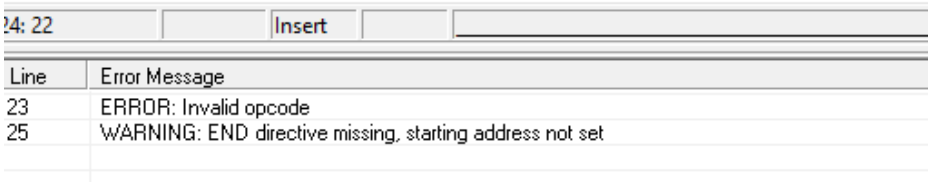


View Listing File

```
*--- Data Section
      ORG $400400
C DS.B 1
X DS.W 1
Y DS.L 1
*--- Code Section
START ORG $400410
      MOVE.B #$41,C
      MOVE.W #$0100,X
      MOVE.L #$2000A111,Y
      MOVE.B C,D0
      MOVE.W X,D1
      MOVE.L Y,D2
      MOVE.B D0,D3
      MOVE.B D1,D4
      MOVE.W D2,D5
END START
*--- End Code Section
```

View Listing File with Errors





Listing File with No Errors

```
00400410 Starting Address
Assembler used: EASy68K Editor/Assembler v5.16.01
Created On: 17-Oct-18 7:00:15 PM

00000000                             1  *-----------------------------------------------------------
00000000                             2  * Title : My First Program
00000000                             3  * Written by : D.Karampatzakis
00000000                             4  * Date : 17/10/2018
00000000                             5  * Description: My first Run
00000000                             6  *-----------------------------------------------------------
00000000                             7  *--- Data Section
00400400                             8         ORG $400400
00400400                             9  C DS.B 1
00400402                            10  X DS.W 1
00400404                            11  Y DS.L 1
00400408                            12  *--- Code Section
00400410                            13  START ORG $400410
00400410  13FC 0041 00400400      14         MOVE.B #$41,C
00400418  33FC 0100 00400402      15         MOVE.W #$0100,X
00400420  23FC 2000A111 00400404  16         MOVE.L #$2000A111,Y
0040042A  1039 00400400           17         MOVE.B C,D0
00400430  3239 00400402           18         MOVE.W X,D1
00400436  2439 00400404           19         MOVE.L Y,D2
0040043C  1600                    20         MOVE.B D0,D3
0040043E  1801                    21         MOVE.B D1,D4
00400440  3A02                    22         MOVE.W D2,D5
00400442                          23         END START
```

No errors detected
No warnings generated

```
SYMBOL TABLE INFORMATION
Symbol-name         Value
-------------------------
C                   400400
START               400410
X                   400402
Y                   400404
```

Listing File with Errors

```
00400410 Starting Address
Assembler used: EASy68K Editor/Assembler v5.16.01
Created On: 17-Oct-18 6:55:30 PM

00000000                             1  *-----------------------------------------------------------
00000000                             2  * Title : My First Program
00000000                             3  * Written by : D.Karampatzakis
00000000                             4  * Date : 17/10/2018
00000000                             5  * Description: My first Run
00000000                             6  *-----------------------------------------------------------
00000000                             7  *--- Data Section
00400400                             8         ORG $400400
00400400                             9  C DS.B 1
00400402                            10  X DS.W 1
00400404                            11  Y DS.L 1
00400408                            12  *--- Code Section
00400410                            13  START ORG $400410
00400410  13FC 0041 00400400      14          MOVE.B #$41,C
00400418  33FC 0100 00400402      15          MOVE.W #$0100,X
00400420  23FC 2000A111 00400404  16          MOVE.L #$2000A111,Y
0040042A  1039 00400400           17          MOVE.B C,D0
00400430  3239 00400402           18          MOVE.W X,D1
00400436  2439 00400404           19          MOVE.L Y,D2
0040043C  1600                    20          MOVE.B D0,D3
0040043E  1801                    21          MOVE.B D1,D4
00400440  3A02                    22          MOVE.W D2,D5
Line 23 ERROR: Invalid opcode
00400442                            23  END START
00400442                            24  *--- End Code Section
Line 25 WARNING: END directive missing, starting address not set

1 error detected
1 warning generated


SYMBOL TABLE INFORMATION
Symbol-name         Value
-------------------------
C                   400400
START               400410
X                   400402
Y                   400404
```

D0-D7
Data Registers

A0-A7
Address
Registers

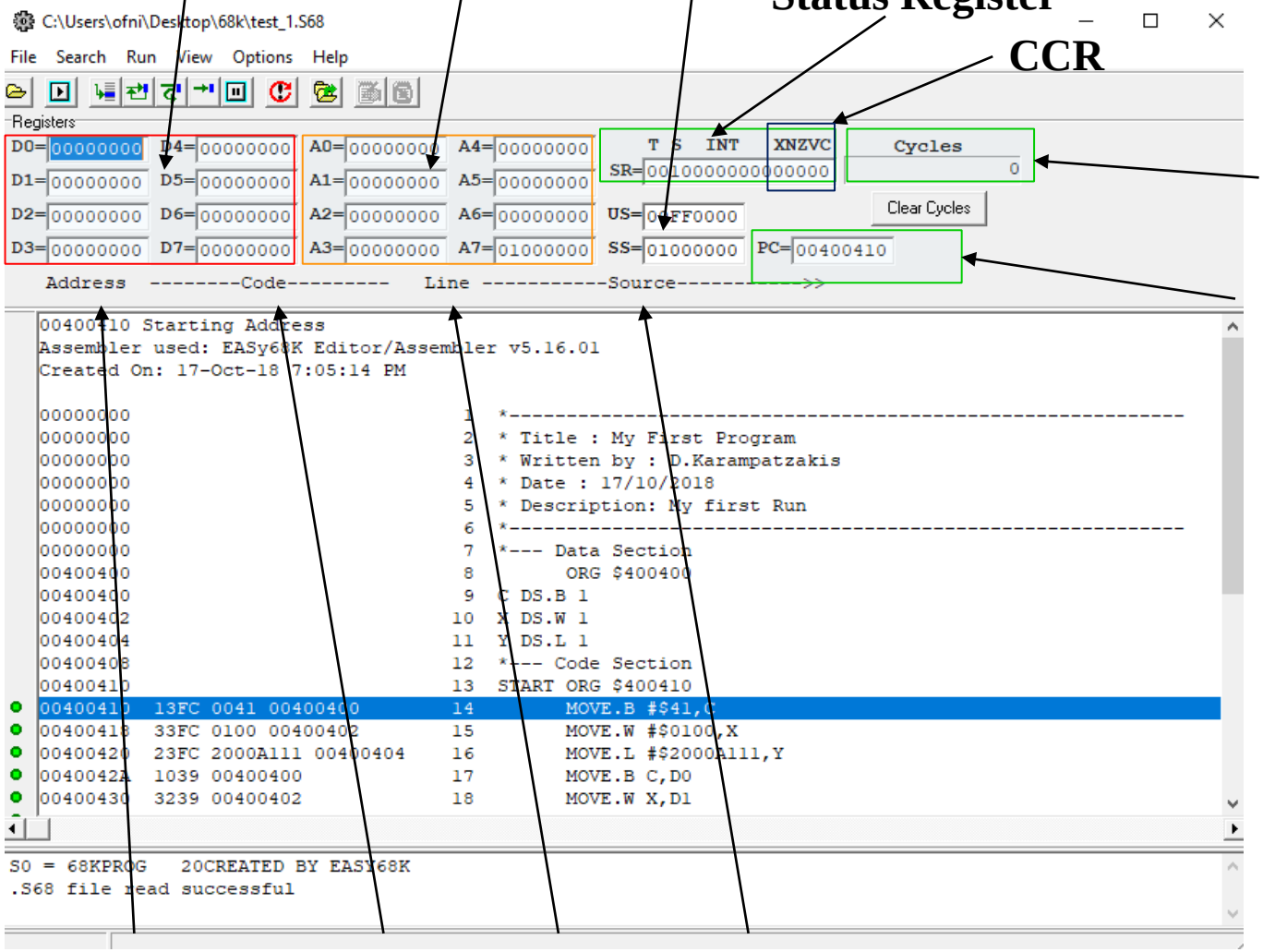
US = User Stack (The user stack is the same as A7 when the S bit in the Status Register is set to 0).
SS = System Stack (The system stack is the same as A7 when the S bit in the Status Register is set to 1).

Status Register

CCR

Execution Cycles

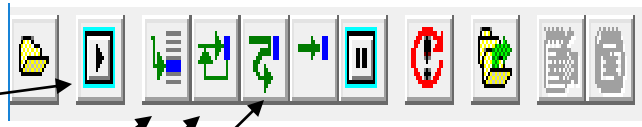
Program Counter



Address **Code** **Line** **Source**

Sim68k window

Running a Program



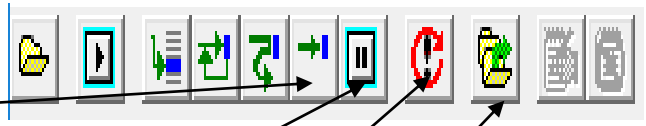
Run

To run the program, select Run from the Run Menu, press F9 or click the Run button on the toolbar. Sim68K begins executing the 68000 program at the current Program Counter location. Program execution will continue until one of the following occurs:
- The program reaches a STOP instruction.
- The program reaches a user placed Break-Point
- The user Pauses the program.
- The user Resets the simulator.
- An exception occurs.

Run To Cursor - Program execution will continue until the Program Counter reaches the Cursor line (highlighted line) or until any of the stop conditions listed above in the Run command occurs. Run To Cursor may be selected from the Run Menu or by pressing Ctrl-F9 or by clicking the Run To Cursor button.

Auto Trace - Automatically activates a Trace Into at the specified time interval. The time interval may be adjusted by selecting Auto Trace Options in the Options menu. To start Auto Trace, select Auto Trace from the Run Menu, press F10 or click the Auto Trace button.

Step Over - Executes the current instruction and positions the Program Counter at the instruction in the next line. If the current instruction is a JSR or BSR the subroutine is completely executed and the Program Counter is placed at the instruction following the JSR or BSR. To Step through a program, select Step Over from the Run Menu, press F8 or click the Step Over button.



Trace Into - Executes the current instruction and positions the Program Counter at the next instruction to be executed. If the current instruction is a JSR or BSR the program counter is placed at the first instruction of the subroutine. To Trace through a program, select Trace Into from the Run Menu, press F7 or click the Trace Into button.

When Tracing or Stepping through a program, the next line to be executed is highlighted as shown above.

The Auto Trace, Trace and Step buttons will be disabled if the program is waiting for input.

Pause

Pauses program execution and enables the menus. To Pause a running program select Pause from the Run Menu, press F6 or click the Pause button.

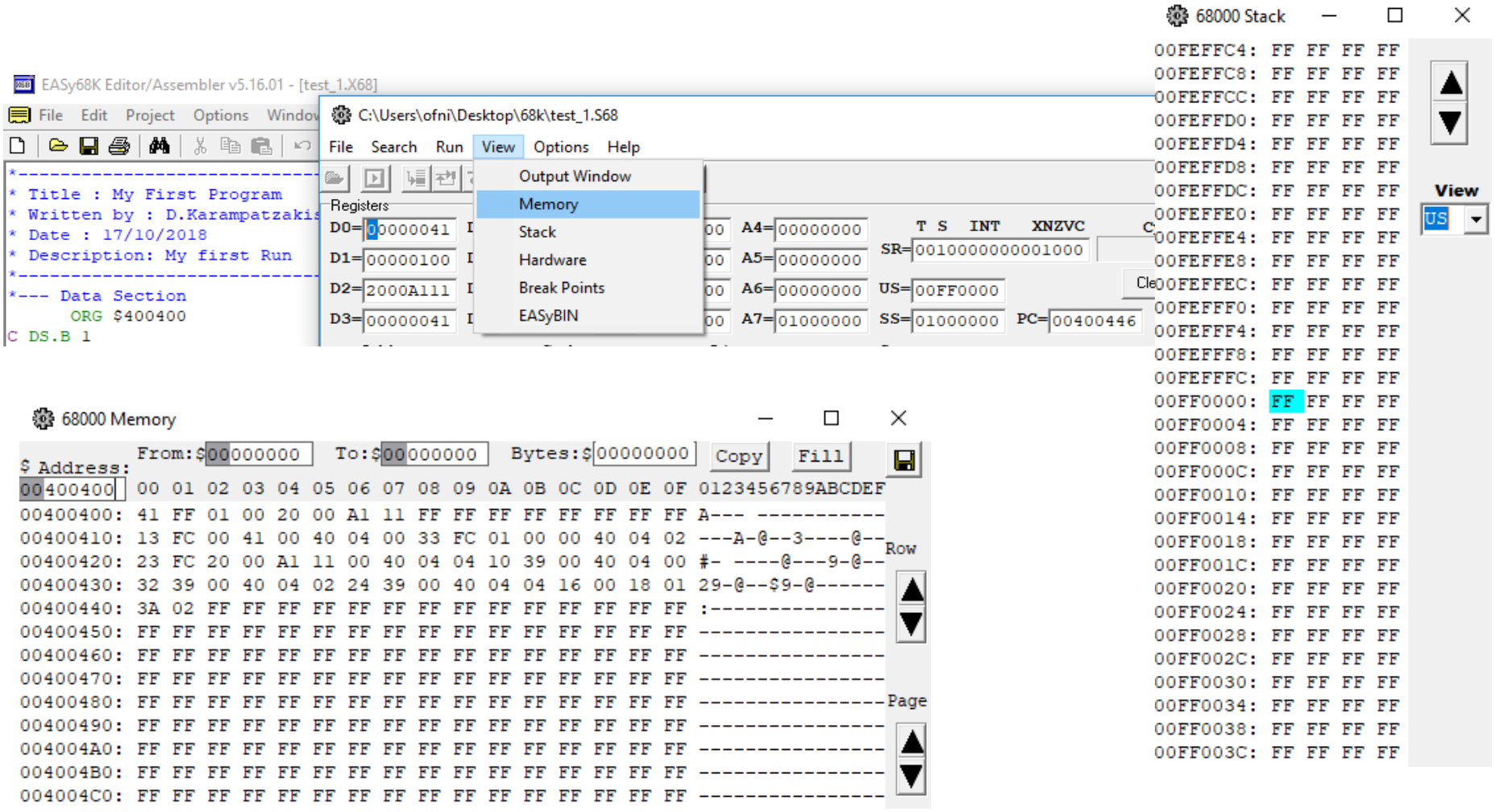
Rewind Program

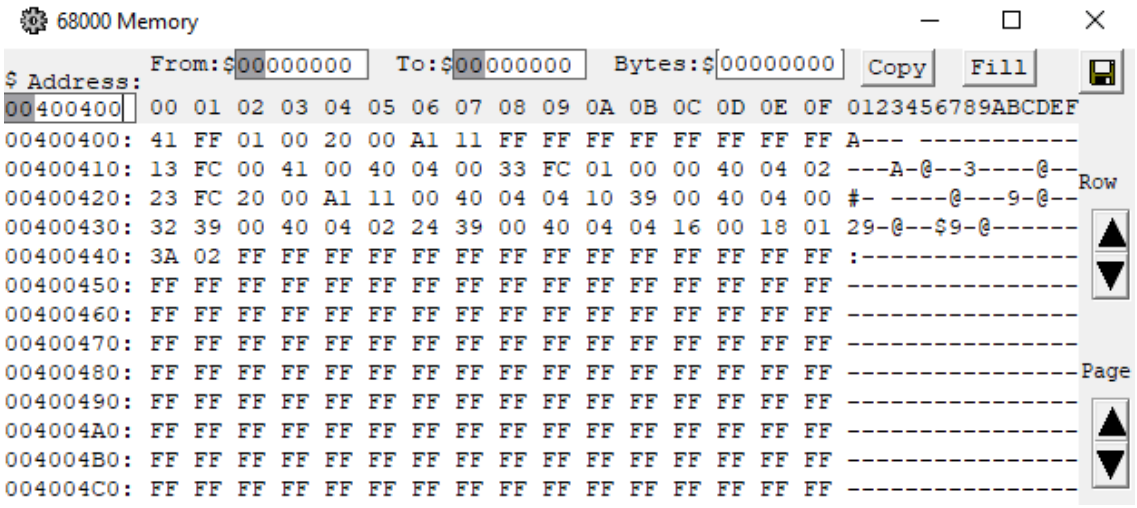
Clears the Output Window, clears the 68000 registers and places the Program Counter at the beginning of the program. To Rewind a program select Rewind Program from the Run Menu, press Ctrl+F2 or click the Rewind Program button.

Reload Program

Reloads the last program into the simulator, clears the Output Window, clears the 68000 registers and places the Program Counter at the beginning of the program. To Reload a program select Reload Program from the Run Menu, press Ctrl+F3 or click the Reload Program button.

Memory & Stack Window





The simulator provides a full 16 Mega Byte 68000 address space from address $00000000 - $00FFFFFF. Each row of the memory window displays an address followed by 16 bytes of hexadecimal memory data, followed by the ASCII representation of the 16 bytes. The contents of memory may be changed by clicking on the desired location and entering a new value. Entries may be made in Hexadecimal or ASCII.

Use the **Row** and **Page** buttons or the mouse wheel to scroll up or down through memory. To jump to a certain address, enter it in the **Address:** field.

Copy

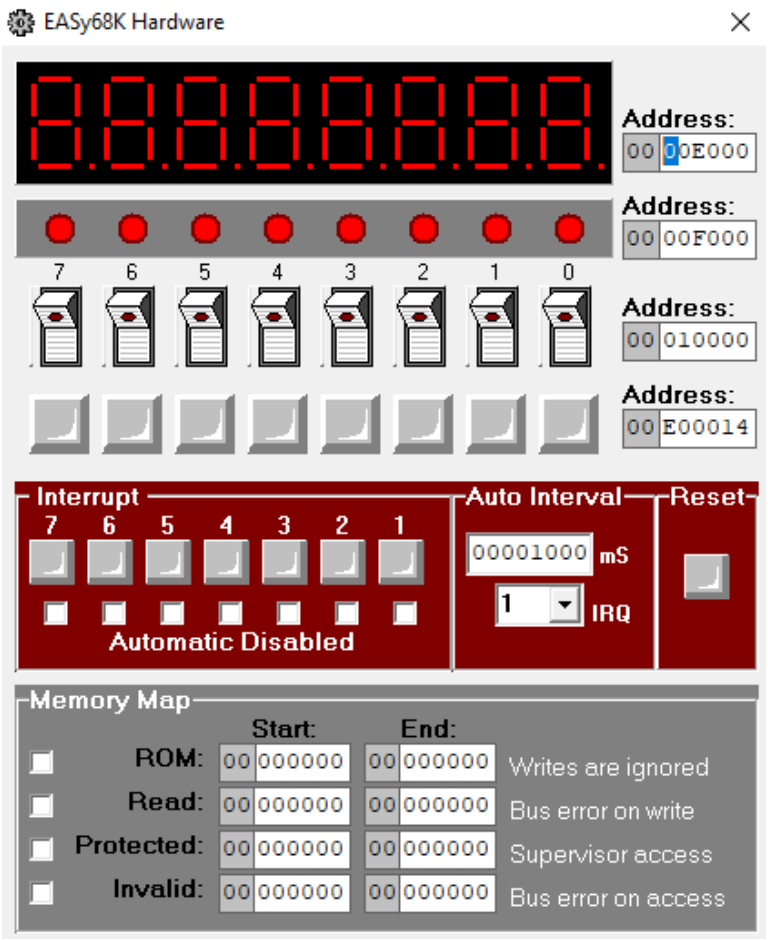
Blocks of memory may be copied. Enter the From and To address and the number of Bytes to copy and click the Copy button. Memory copy is useful when testing position independent code.

Fill

Blocks of memory may be filled. Enter the From and To address and the Bytes to fill with and click the Fill button.

Blocks of memory may be saved to a binary file. Enter the From and To address and click the Save button.

Simulator I/O Hardware Window



Αναφορές

• *Σύγγραμμα Δ. Πογαρίδης, «Σχεδίαση Συστημάτων Μικροεπεξεργαστών (68000)».*
• *University Course, COMP-573B Microcomputers, McGill University, 1998.*
• *SEE 3223 Microprocessor Systems, 68000 Architecture, OpenCourseWare, ocw.utm.my.*
• *Easy68K software, http://www.easy68k.com*