

Απόδοση ΚΜΕ

(Μέτρηση και τεχνικές βελτίωσης απόδοσης)

<http://mixstef.github.io/courses/comparch/>

Μ.Στεφανιδάκης



Απόδοση (Κεντρικής) Μονάδας Επεξεργασίας

- Απόδοση υπολογιστικού συστήματος
 - Η απόδοση εξαρτάται από **όλα** τα επιμέρους τμήματα του συστήματος
 - **Υλικό και λογισμικό**
- Απόδοση (Κεντρικής) Μονάδας Επεξεργασίας
 - Πόσο **γρήγορα** εκτελείται ένα πρόγραμμα;
 - Πώς επηρεάζει η αρχιτεκτονική της (Κ)ΜΕ την απόδοση;
 - Πόσο **γρηγορότερα** εκτελείται ένα πρόγραμμα μετά από μια **αρχιτεκτονική αλλαγή**;

Χρόνος απόκρισης – Ρυθμός Ολοκλήρωσης

- Χρόνος απόκρισης (response time)
 - Συνολικός χρόνος για την ολοκλήρωση των εργασιών ενός προγράμματος (από την έναρξη μέχρι τη λήξη)
- Ρυθμός ολοκλήρωσης (throughput)
 - Ρυθμός ολοκλήρωσης έργου σε συγκεκριμένο χρόνο
- Τα δύο μεγέθη είναι αλληλένδετα
 - Συνήθως η βελτίωση του ενός επιδρά θετικά και στο άλλο

Χρόνος Εκτέλεσης (Execution Time)

- Χρόνος εκτέλεσης στην (Κ)ΜΕ
 - Ο χρόνος για τον οποίο η ΚΜΕ εκτελεί εντολές του προγράμματος
 - Όχι χρόνος για αναμονή Ε/Ε ή για άλλες διεργασίες
- Συνιστώσες
 - Χρόνος προγράμματος χρήστη
 - Για το πρόγραμμα καθεαυτό
 - Χρόνος συστήματος
 - Λειτουργίες ΛΣ για την εξυπηρέτηση του προγράμματος

Εκτέλεση προγράμματος

- Χρόνος εκτέλεσης ενός προγράμματος (execution time)
 - Αύξηση απόδοσης \Leftrightarrow Μείωση χρόνου εκτέλεσης
 - Εκτέλεση σε έναν υπολογιστή X:

$$\text{Απόδοση}(X) = \frac{1}{\text{Χρόνος Εκτέλεσης}(X)}$$

Σύγκριση εκτέλεσης σε δύο συστήματα

- Συγκρίνοντας αποδόσεις για την εκτέλεση του ίδιου προγράμματος

- Έστω υπολογιστές X και Y

- Εάν:

$$\text{Απόδοση}(X) > \text{Απόδοση}(Y)$$

- Τότε (και αντίστροφα):

$$\text{Χρόνος Εκτέλεσης}(X) < \text{Χρόνος Εκτέλεσης}(Y)$$

Σύγκριση εκτέλεσης σε δύο συστήματα

- Υπολογισμός του λόγου των χρόνων εκτέλεσης
 - Όταν σε υπολογιστές X και Y για το ίδιο πρόγραμμα είναι:

$$\frac{\text{Χρόνος Εκτέλεσης}(Y)}{\text{Χρόνος Εκτέλεσης}(X)} = n$$

- Τότε:

Ο X είναι n φορές γρηγορότερος από τον Y

- Παράδειγμα
 - Ο X εκτελεί ένα πρόγραμμα σε 10 sec και ο Y σε 15 sec.
Πόσο πιο γρήγορος είναι ο X;

Σύγκριση εκτέλεσης σε δύο συστήματα

- Υπολογισμός του λόγου των χρόνων εκτέλεσης
 - Όταν σε υπολογιστές X και Y για το ίδιο πρόγραμμα είναι:

$$\frac{\text{Χρόνος Εκτέλεσης}(Y)}{\text{Χρόνος Εκτέλεσης}(X)} = n$$

- Τότε:

Ο X είναι n φορές γρηγορότερος από τον Y

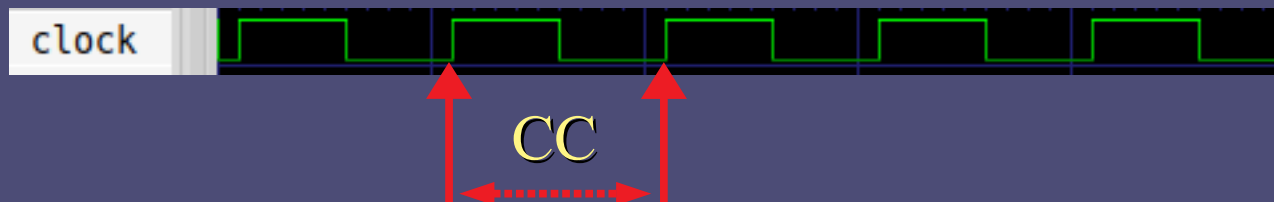
- Παράδειγμα

- Ο X εκτελεί ένα πρόγραμμα σε 10 sec και ο Y σε 15 sec.
Πόσο πιο γρήγορος είναι ο X;
 - Ο X είναι 1.5 φορές γρηγορότερος

$$\frac{15 \text{ sec}}{10 \text{ sec}} = 1,5$$

Βασικά μεγέθη μέτρησης χρόνου εκτέλεσης

- Κύκλος ρολογιού (περίοδος)
 - Clock Cycle (CC)
 - Η **διάρκεια** ενός κύκλου ρολογιού (περίοδος ρολογιού) κατά τον οποίο η ΚΜΕ εκτελεί τις μικρότερες βασικές λειτουργίες
 - **Απόλυτα σταθερό μέγεθος**
- Ρολόι (clock)
 - Περιοδικό σήμα (εναλλάσσεται συνεχώς μεταξύ 0 και 1)
 - Ο παλμός κάθε υπολογιστικού συστήματος, συγχρονίζει τις λειτουργίες του συστήματος



Βασικά μεγέθη μέτρησης χρόνου εκτέλεσης

Κύκλοι ρολογιού ανά εντολή

▪ Clocks Per Instruction (CPI)

- Οι απαιτούμενοι κύκλοι ρολογιού για την ολοκλήρωση μιας εντολής
- Ενδεχομένως διαφορετικό μέγεθος ανά τύπο εντολής
- Σε προσεγγιστικούς υπολογισμούς χρησιμοποιείται ένα μέσο CPI
- Σε λεπτομερείς υπολογισμούς χρησιμοποιούνται μεγέθη από προσομοιώσεις ή μετρήσεις μέσω μετροπρογραμμάτων

• Αριθμός εντολών

▪ Instruction Count (IC)

- Ο αριθμός των εντολών ενός προγράμματος

Χρόνος εκτέλεσης στην (Κ)ΜΕ

- Χρόνος Εκτέλεσης για ένα πρόγραμμα

$$\text{ExecTime} = \text{IC} \times \text{CPI} \times \text{CC}$$

κύκλοι ρολογιού
συνολικού προγράμματος

περίοδος ρολογιού
(χρόνος)

- Τι μπορεί να κάνει ο σχεδιαστής ΚΜΕ για να βελτιώσει την απόδοση;
 - Να μειώσει τον κύκλο ρολογιού (CC)
 - Να μειώσει τον αριθμό κύκλων ανά εντολή (CPI)
 - Ο αριθμός εντολών δεν αλλάζει

Παράδειγμα

| Τύπος εντολής | A | B | C |
|---------------|---|---|---|
| CPI | 1 | 2 | 3 |

| Ακολουθία κώδικα | A | B | C |
|------------------|---|---|---|
| 1 | 2 | 1 | 2 |
| 2 | 4 | 1 | 1 |

[Patterson-Hennessy “Computer Organization and Design”, 3rd ed]

- Έχουμε να διαλέξουμε μεταξύ 2 ακολουθιών εντολών
 - Ποια ακολουθία εκτελεί τις περισσότερες εντολές;
 - Ποια είναι ταχύτερη;
 - Ποιο το μέσο CPI σε κάθε περίπτωση;

Παράδειγμα

| Τύπος εντολής | A | B | C |
|---------------|---|---|---|
| CPI | 1 | 2 | 3 |

| Ακολουθία κώδικα | A | B | C |
|------------------|---|---|---|
| 1 | 2 | 1 | 2 |
| 2 | 4 | 1 | 1 |

[Patterson-Hennessy “Computer Organization and Design”, 3rd ed]

- Έχουμε να διαλέξουμε μεταξύ 2 ακολουθιών εντολών
 - Ποια ακολουθία εκτελεί τις περισσότερες εντολές;
 - Ποια είναι ταχύτερη;
 - Ποιο το μέσο CPI σε κάθε περίπτωση;

$$A: 2+1+2 = 5$$

$$B: 4+1+1 = 6$$

Παράδειγμα

| Τύπος εντολής | A | B | C |
|---------------|---|---|---|
| CPI | 1 | 2 | 3 |

| Ακολουθία κώδικα | A | B | C |
|------------------|---|---|---|
| 1 | 2 | 1 | 2 |
| 2 | 4 | 1 | 1 |

[Patterson-Hennessy “Computer Organization and Design”, 3rd ed]

- Έχουμε να διαλέξουμε μεταξύ 2 ακολουθιών εντολών

- Ποια ακολουθία εκτελεί τις περισσότερες εντολές;

$$\text{κύκλοι ρολογιού} = \sum (\text{CPI}_i * \text{IC}_i)$$

$$\text{A: } 2+1+2 = 5$$

$$\text{B: } 4+1+1 = 6$$

- Ποια είναι ταχύτερη;

$$\text{A: } 2*1+1*2+2*3 = 10$$

$$\text{B: } 4*1+1*2+1*3 = 9$$

- Ποιο το μέσο CPI σε κάθε περίπτωση;

Παράδειγμα

| Τύπος εντολής | A | B | C |
|---------------|---|---|---|
| CPI | 1 | 2 | 3 |

| Ακολουθία κώδικα | A | B | C |
|------------------|---|---|---|
| 1 | 2 | 1 | 2 |
| 2 | 4 | 1 | 1 |

[Patterson-Hennessy “Computer Organization and Design”, 3rd ed]

- Έχουμε να διαλέξουμε μεταξύ 2 ακολουθιών εντολών

- Ποια ακολουθία εκτελεί τις περισσότερες εντολές;

$$\text{κύκλοι ρολογιού} = \sum (\text{CPI}_i * \text{IC}_i)$$

$$\text{A: } 2+1+2 = 5$$

$$\text{B: } 4+1+1 = 6$$

- Ποια είναι ταχύτερη;

$$\text{A: } 2*1+1*2+2*3 = 10$$

$$\text{B: } 4*1+1*2+1*3 = 9$$

- Ποιο το μέσο CPI σε κάθε περίπτωση;

$$\text{A: } 10/5 = 2$$

$$\text{B: } 9/6 = 1.5$$

Συσχέτιση με λογισμικό

- Ενώ το υλικό και η αρχιτεκτονική συνόλου εντολών (ISA) καθορίζει και τα τρία μεγέθη (IC, CPI και CC), τι συμβαίνει με το λογισμικό;
- **Αλγόριθμος**
 - Καθορίζει το IC
 - Ενδεχομένως καθορίζει το CPI, ευνοώντας ορισμένους τύπους εντολών (π.χ. κινητής υποδιαστολής)
- **Γλώσσα προγραμματισμού - Μεταγλωττιστής**
 - Καθορίζει το IC (μετάφραση εντολών υψηλού επιπέδου)
 - Καθορίζει το CPI απαιτώντας/χρησιμοποιώντας συγκεκριμένους τύπους εντολών

Μετροπρογράμματα

- **Benchmarks**
 - Για τη μέτρηση της απόδοσης
 - Και τη σύγκριση μεταξύ υπολογιστών
 - Θα πρέπει να αντιπροσωπεύουν τις πραγματικές εφαρμογές
 - Υπό ρεαλιστικές συνθήκες εκτέλεσης και δεδομένα εισόδου
 - Χωρίς “εσωτερικές” ειδικές βελτιστοποιήσεις
 - Δυνατότητα επανάληψης μέτρησης
 - Διαφορετικά για ανόμοιες κλάσεις υπολογιστών
 - PCs, servers, embedded systems...

Ο «νόμος» του Amdahl

- «Η βελτίωση της συνολικής απόδοσης ενός συστήματος μέσω της εισαγωγής ενός νέου χαρακτηριστικού, περιορίζεται από το βαθμό χρήσης αυτού του νέου χαρακτηριστικού»
- Ερμηνεία – συνέπειες
 - Οι περισσότερο χρησιμοποιούμενες περιπτώσεις πρέπει να είναι γρήγορες
 - Δεν ωφελεί η βελτιστοποίηση των σπάνιων περιπτώσεων
 - Η μη χρήση του νέου χαρακτηριστικού βελτίωσης εμποδίζει να επιτύχουμε την «τέλεια» απόδοση
 - Ακόμα κι όταν το ποσοστό μη χρήσης είναι ελάχιστο

Ο «νόμος» του Amdahl – Παράδειγμα

- Ένα πρόγραμμα τρέχει για 100 sec σε έναν υπολογιστή και εκτελεί πολλαπλασιασμούς για 80 sec. Πόσο πρέπει να βελτιώσω τη ταχύτητα του πολλαπλασιασμού για να πενταπλασιάσω τη συνολική απόδοση;

Ο «νόμος» του Amdahl – Παράδειγμα

- Ένα πρόγραμμα τρέχει για **100 sec** σε έναν υπολογιστή και εκτελεί **πολλαπλασιασμούς για 80 sec**. Πόσο πρέπει να βελτιώσω τη **ταχύτητα του πολλαπλασιασμού** για να **πενταπλασιάσω τη συνολική απόδοση**;
 - Το πρόγραμμα εκτελεί **80 sec** πολλαπλασιασμούς και **20 sec** άλλες πράξεις
 - Για να πενταπλασιαστεί η συνολική απόδοση, ο λόγος των χρόνων εκτέλεσης (παλιό/νέο) θα πρέπει να είναι **5**
 - Έστω ότι το κύκλωμα πολλαπλασιασμού γίνεται **n φορές ταχύτερο**. Οι πολλαπλασιασμοί τότε θα εκτελούνται σε **80/n sec**

Ο «νόμος» του Amdahl – Παράδειγμα

- Ένα πρόγραμμα τρέχει για **100 sec** σε έναν υπολογιστή και εκτελεί **πολλαπλασιασμούς** για **80 sec**. Πόσο πρέπει να βελτιώσω τη **ταχύτητα του πολλαπλασιασμού** για να **πενταπλασιάσω** τη **συνολική απόδοση**;
 - Το πρόγραμμα εκτελεί **80 sec** πολλαπλασιασμούς και **20 sec** άλλες πράξεις
 - Για να **πενταπλασιαστεί** η συνολική απόδοση, ο λόγος των χρόνων εκτέλεσης (παλιό/νέο) θα πρέπει να είναι **5**
 - Έστω ότι το κύκλωμα πολλαπλασιασμού γίνεται **n φορές ταχύτερο**. Οι πολλαπλασιασμοί τότε θα εκτελούνται σε **80/n sec**

$$\frac{\text{Χρόνος Εκτέλεσης(παλιό)}}{\text{Χρόνος Εκτέλεσης(νέο)}} = \frac{100 \text{ sec}}{80/n + 20 \text{ sec}} = 5$$

υπάρχει
λύση για
το n;

ΚΜΕ ενός κύκλου (single-cycle)

- **CPI = 1**
 - Σε κάθε έναν κύκλο ρολογιού ολοκληρώνεται μια εντολή ή
 - κάθε εντολή απαιτεί έναν κύκλο ρολογιού
- **Πόσο πρέπει να είναι το CC;**
 - Ίσο με τη διάρκεια της μεγαλύτερης λειτουργίας
 - Της πιο χρονοβόρας εντολής μηχανής
 - Μη αποδοτικό σχήμα
 - Όλες οι εντολές μηχανής δεν απαιτούν τον ίδιο χρόνο

Υποθετικό παράδειγμα σε ΚΜΕ ενός κύκλου

| Εντολή | IF | ID | EX | DM | WB | Σύνολο |
|------------------------|-----------|-----------|-----------|-----------|-----------|---------------|
| Αριθμητική | 200 | 50 | 100 | 0 | 50 | 400 ps |
| Διακλάδωση | 200 | 50 | 100 | 0 | 0 | 350 ps |
| Ανάγνωση μνήμης | 200 | 50 | 100 | 200 | 50 | 600 ps |
| Εγγραφή μνήμης | 200 | 50 | 100 | 200 | 0 | 550 ps |

[Patterson-Hennessy “Computer Organization and Design”, 3rd ed]

- Για να μπορεί να ολοκληρωθεί κάθε είδος εντολών το CC θα πρέπει να είναι 600 ps
 - Ποια η βελτίωση της απόδοσης αν ήταν δυνατή η χρήση μεταβλητού CC; (προσοχή: πρακτικά αδύνατο!)
 - Θεωρείστε ότι το πρόγραμμα έχει τους εξής τύπους εντολών: 25% ανάγνωσης, 10% εγγραφής, 45% αριθμητικές, 20% διακλάδωσης

Υποθετικό παράδειγμα σε ΚΜΕ ενός κύκλου

| Εντολή | IF | ID | EX | DM | WB | Σύνολο |
|-----------------|-----|----|-----|-----|----|--------|
| Αριθμητική | 200 | 50 | 100 | 0 | 50 | 400 ps |
| Διακλάδωση | 200 | 50 | 100 | 0 | 0 | 350 ps |
| Ανάγνωση μνήμης | 200 | 50 | 100 | 200 | 50 | 600 ps |
| Εγγραφή μνήμης | 200 | 50 | 100 | 200 | 0 | 550 ps |

25% ανάγνωσης, 10% εγγραφής, 45% αριθμητικές, 20% διακλάδωσης

- Και στις δύο περιπτώσεις είναι ίδιος ο αριθμός εντολών (IC) και ο αριθμός κύκλων ρολογιού ανά εντολή (CPI=1)
- Συνεπώς η απόδοση υπολογίζεται μόνο από τον λόγο των CC:

$$\frac{\text{Χρόνος Εκτέλεσης(παλιό)}}{\text{Χρόνος Εκτέλεσης(νέο)}} = \frac{\text{CC(παλιό)}}{\text{CC(νέο)}} = \frac{600\text{ps}}{\text{CC(νέο)}}$$

πώς υπολογίζεται;

Υποθετικό παράδειγμα σε ΚΜΕ ενός κύκλου

| Εντολή | IF | ID | EX | DM | WB | Σύνολο |
|-----------------|-----|----|-----|-----|----|--------|
| Αριθμητική | 200 | 50 | 100 | 0 | 50 | 400 ps |
| Διακλάδωση | 200 | 50 | 100 | 0 | 0 | 350 ps |
| Ανάγνωση μνήμης | 200 | 50 | 100 | 200 | 50 | 600 ps |
| Εγγραφή μνήμης | 200 | 50 | 100 | 200 | 0 | 550 ps |

25% ανάγνωσης, 10% εγγραφής, 45% αριθμητικές, 20% διακλάδωσης

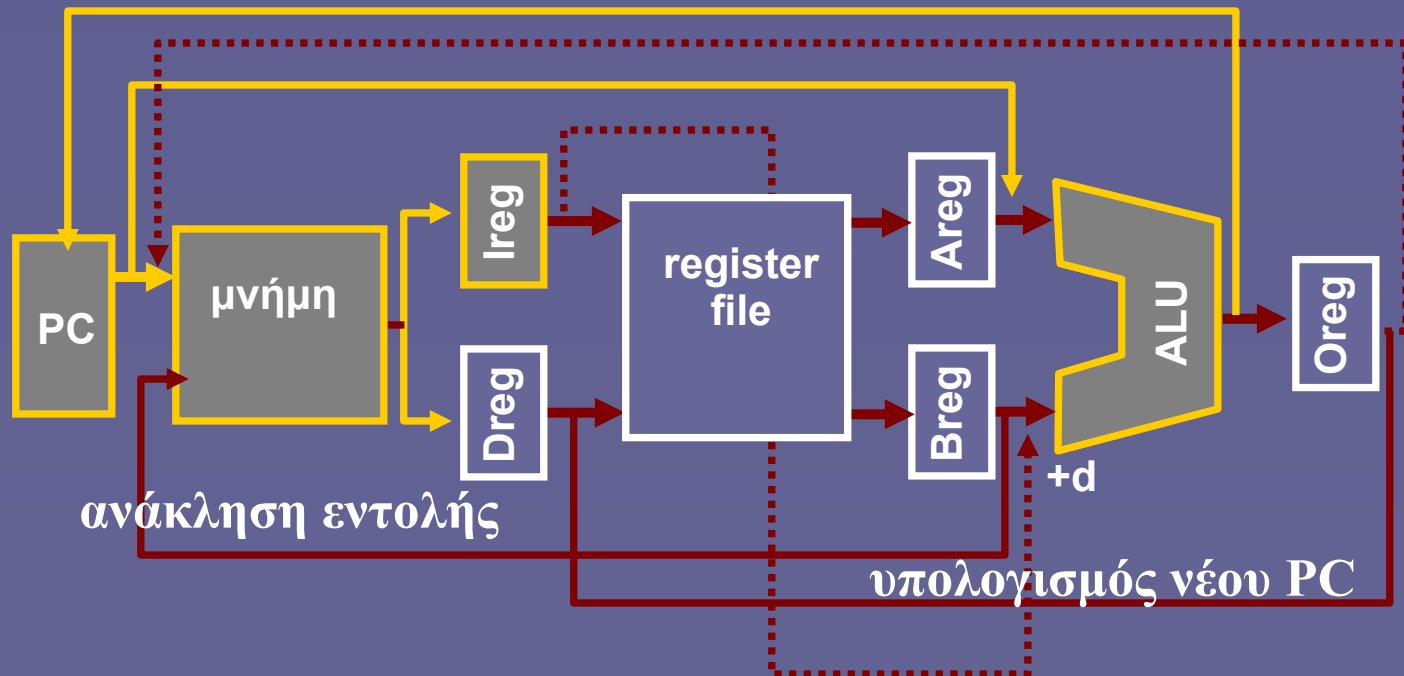
- CC(νέο) = σταθμισμένος μέσος όρος CC ανά τύπο εντολής, ανάλογα με τη συχνότητα εμφάνισης του τύπου εντολών =
 $0,45*400 + 0,2*350 + 0,25*600 + 0,1*550 = 455\text{ps}$

$$\frac{\text{Χρόνος Εκτέλεσης(παλιό)}}{\text{Χρόνος Εκτέλεσης(νέο)}} = \frac{\text{CC(παλιό)}}{\text{CC(νέο)}} = \frac{600\text{ps}}{455\text{ps}} = 1,32$$

ΚΜΕ πολλαπλών κύκλων (multi-cycle)

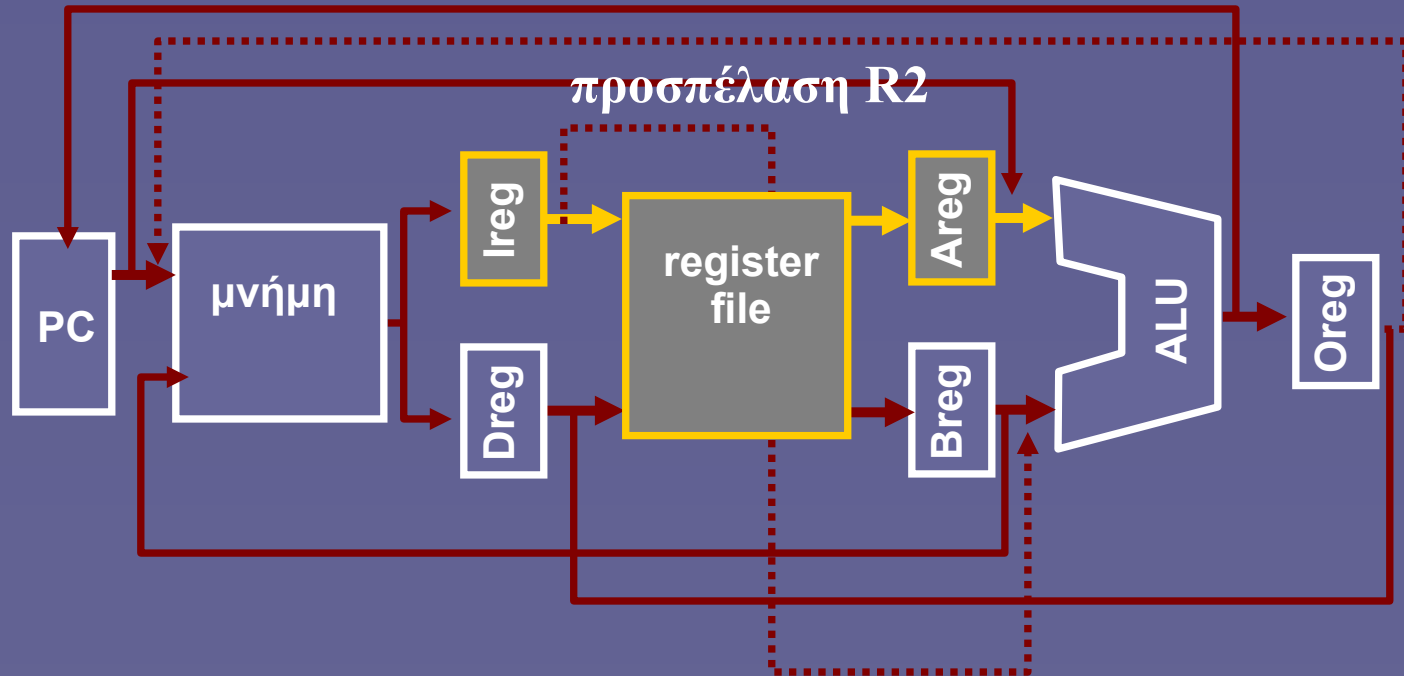
- **CPI > 1**
 - Κάθε εντολή μηχανής χωρίζεται σε έναν **αριθμό βημάτων**
 - Διαφορετικός αριθμός βημάτων για κάθε τύπο εντολών
 - Κάθε βήμα απαιτεί **έναν κύκλο ρολογιού**
- **Πόση πρέπει να είναι η περίοδος του ρολογιού (CC);**
 - Ίση με τη διάρκεια ολοκλήρωσης του μεγαλύτερου βήματος
- Καταχωρητές για τη συγκράτηση αποτελεσμάτων μεταξύ βημάτων
- Μέρη της ΚΜΕ μπορούν να χρησιμοποιηθούν για περισσότερες από μία φορές κατά την εκτέλεση μιας εντολής

Παράδειγμα: Εντολή load (βήμα IF)



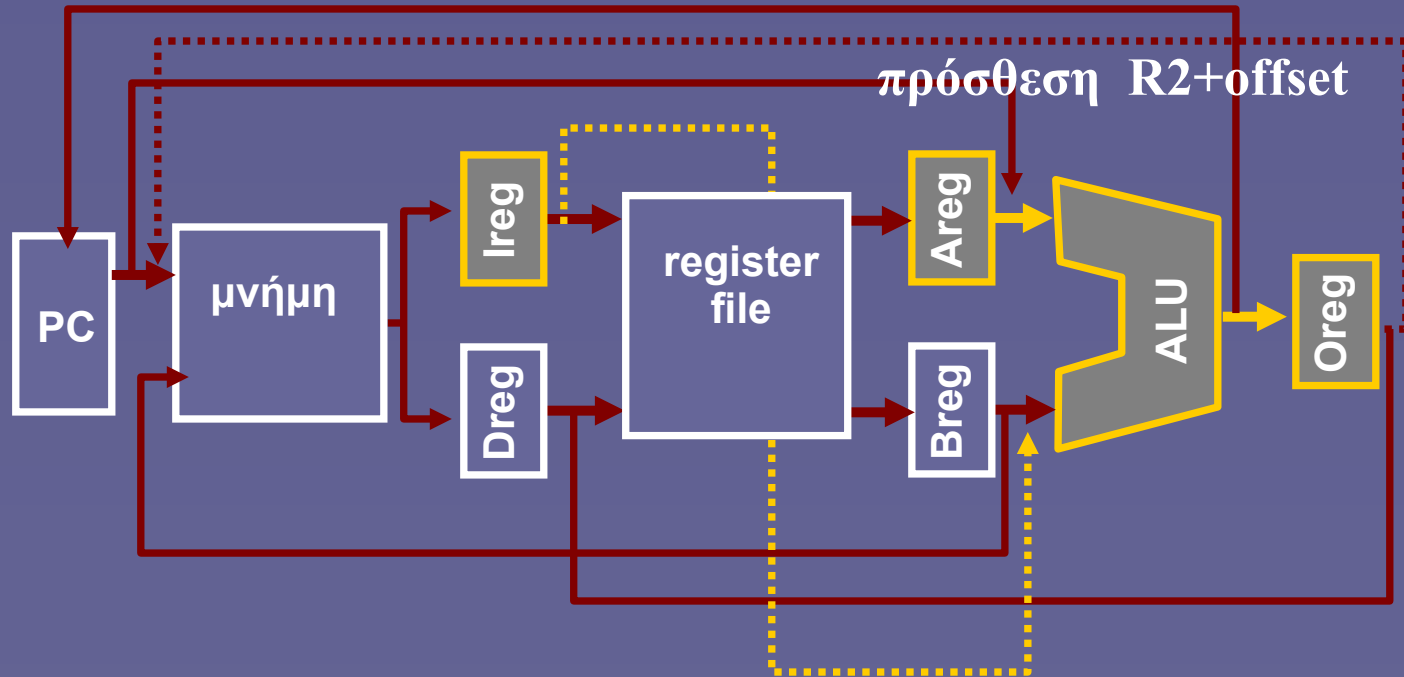
$R1 \leftarrow \text{mem}[R2 + \text{offset}]$

Παράδειγμα: Εντολή load (βήμα ID)



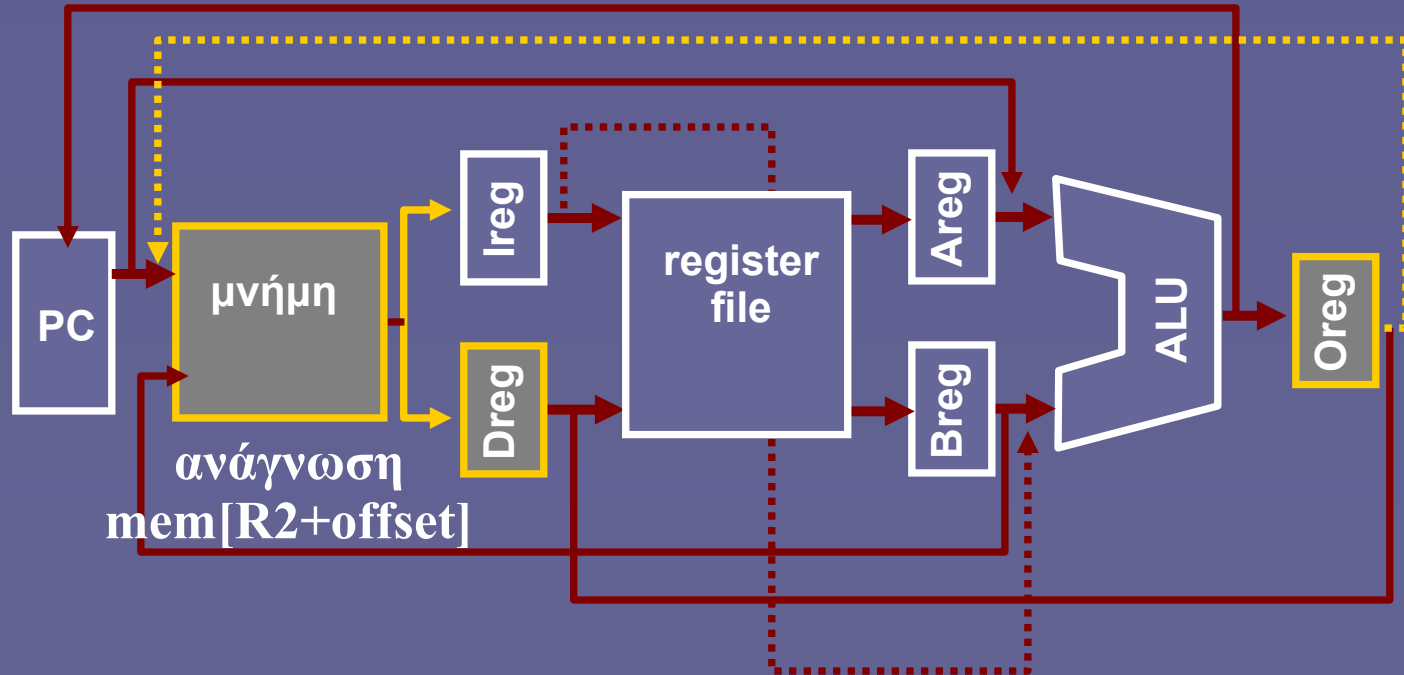
$R1 \leftarrow \text{mem}[R2 + \text{offset}]$

Παράδειγμα: Εντολή load (βήμα EX)



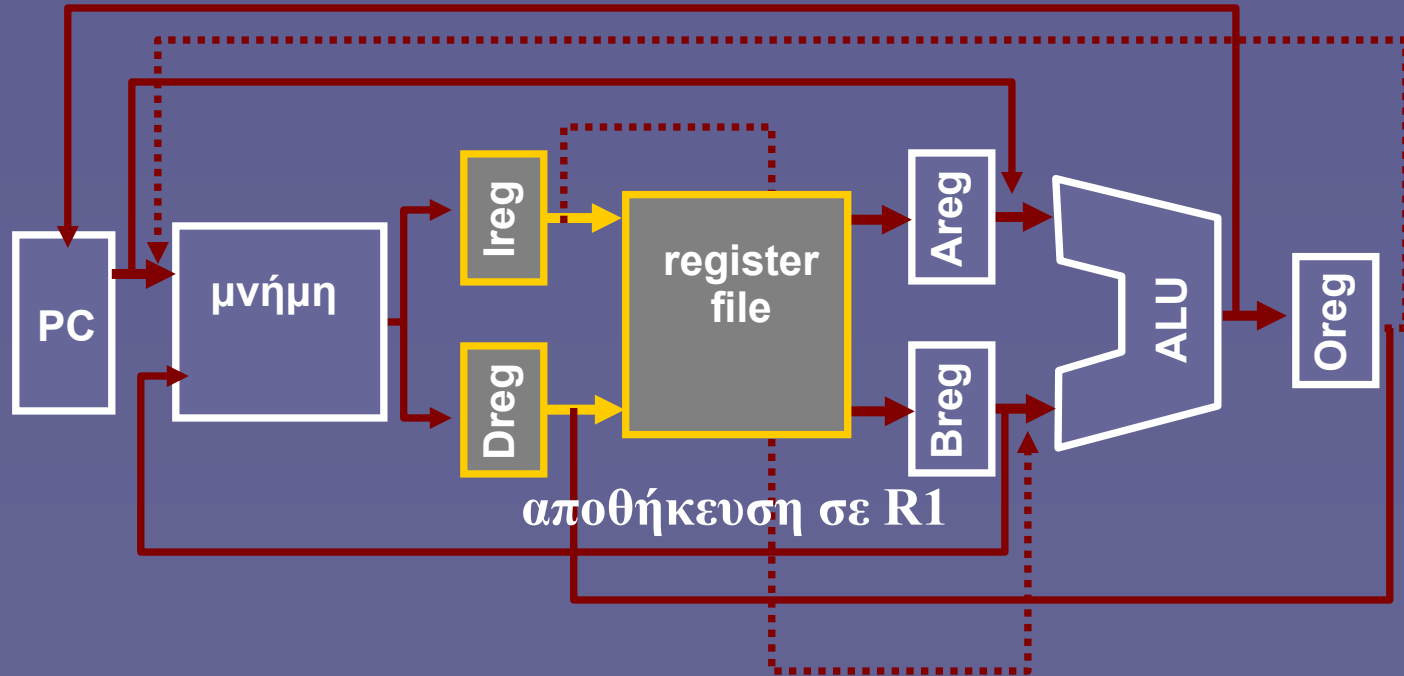
$R1 \leftarrow \text{mem}[R2 + \text{offset}]$

Παράδειγμα: Εντολή load (βήμα DM)



$R1 \leftarrow \text{mem}[R2 + \text{offset}]$

Παράδειγμα: Εντολή load (βήμα WB)



$R1 \leftarrow \text{mem}[R2 + \text{offset}]$

Μονάδα Ελέγχου ΚΜΕ πολλαπλών κύκλων

- Δημιουργία σημάτων σε κάθε βήμα εκτέλεσης εντολής
- Μέθοδοι υλοποίησης
 - Αυτόματα πεπερασμένων καταστάσεων
 - Παραγωγή σημάτων ελέγχου ανάλογα με εισόδους και τρέχουσα κατάσταση
 - Μικροπρόγραμμα
 - Καθορισμός σημάτων μέσω **μικροεντολών**
 - Εσωτερικά στην ΚΜΕ
 - Για υλοποίηση σύνθετων εντολών με πολλά βήματα και πολλαπλά περάσματα από το datapath
 - Μερικές φορές είναι εγγράψιμο (updates, patches..)

Απόδοση ΚΜΕ πολλαπλών κύκλων

- **Πλεονεκτήματα**

- Δεν απαιτείται ο μέγιστος χρόνος για το CC
- Μέρη της ΚΜΕ μπορούν να χρησιμοποιηθούν με πολλαπλό τρόπο κατά την εκτέλεση μιας εντολής

- **Μειονεκτήματα**

- Η μονάδα ελέγχου γίνεται πολυπλοκότερη
 - Η πολυπλοκότητα πιθανόν να ακυρώνει τα πλεονεκτήματα
- Σε κάθε βήμα, **μερικά τμήματα μένουν ανενεργά**
 - Πώς θα μπορούσαμε να τα εκμεταλλευτούμε;
 - (στο επόμενο μάθημα...)