Αρχιτεκτονική Υπολογιστών

Ενότητα 4: Απόδοση & Επιτάχυνση.
Νόμος του Amdahl. Συστοιχία καταχωρητών.
Υπερβαθμωτοί επεξεργαστές. VLIW. IA64.

Δρ. Μηνάς Δασυγένης
mdasyg@ieee.org
Εργαστήριο Ρομποτικής, Ενσωματωμένων και Ολοκληρωμένων Συστημάτων
http://arch.ece.uowm.gr/mdasyg
Τελευταία Επικαιροποίηση: 2023

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ψηφιακά Μαθήματα στο Πανεπιστήμιο Δυτικής Μακεδονίας**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σκοπός ενότητας

- Η κατανόηση των διαδικασιών σύγκρισης.
- Η εμβάθυνση σε θέματα της ISA.
- Η παρουσίαση και ανάλυση δημοφιλών αρχιτεκτονικών.
- Η παρουσίαση τεχνικών αύξησης των επιδόσεων.

Απόδοση & Επιτάχυνση

Υπάρχουν πολλά μετρικά σύγκρισης απόδοσης

- **Χρόνος εκτέλεσης εργασίας (ExTime)**
  - Χρόνος εκτέλεσης, χρόνος απόκρισης, καθυστέρηση
- **Αποστολή ανά μέρα, ώρα, εβδομάδα, λεπτό, δευτερόλεπτο... (Performance)**
  - Απόδοση, εύρος ζώνης

Αεροπλάνο	DC για Παρίσι	Ταχύτητα	Επιβάτες	Απόδοση (pmph)
Boeing 747	6.5 hours	610 mph	470	286,700
BAD/Sud Concodre	3 hours	1350 mph	132	178,200

Τελικά τι σημαίνει ότι το Χ είναι πιο γρήγορο από το Υ;

Οι υπολογιστές μπορούν να συγκριθούν ως προς διάφορα μετρικά:

- Μείωση χρόνου απόκρισης (response time);
- Χρόνο εκτέλεσης (execution time);
- Ρυθμό ολοκλήρωσης (throughput);

Η πιο δημοφιλής σύγκριση είναι ως προς τη διεκπεραιωτική ικανότητα

«Ο x είναι n φορές πιο γρήγορος από τον y»

$$\frac{\text{ExTime(Y)}}{\text{ExTime(X)}} = \frac{\text{Performance(X)}}{\text{Performance(Y)}}$$

Αντιστρόφως ανάλογος της απόδοσης!

- Ταχύτητα Concorde vs Boeing 747
- Απόδοση Boeing 747 vs Concorde
- Το κόστος είναι και αυτό μία πολύ σημαντική παράμετρος στην εξίσωση το οποίο είναι ο λόγος που τα concorde ανήκουν στο παρελθόν!

Μέτρηση απόδοσης

- **Response (execution) time:** ο χρόνος από την αρχή ως την ολοκλήρωση ενός event
- **Throughput:** συνολική εργασία που γίνεται για δοσμένο χρονικό διάστημα
- "Ο Χ είναι πιο γρήγορος από τον Υ n φορές":

$$n = \frac{\text{Execution time}_Y}{\text{Execution time}_X} = \frac{\frac{1}{\text{Performance}_Y}}{\frac{1}{\text{Performance}_X}} = \frac{\text{Performance}_X}{\text{Performance}_Y}$$

Βελτίωση των επιδόσεων

- Η αύξηση της απόδοσης σημαίνει μείωση του χρόνου εκτέλεσης.
- Για να μην υπάρχει σύγχυση.... χρησιμοποιούμε τον όρο **Βελτίωση**.
- Σε κάθε περίπτωση:

Ο υπολογιστής που εκτελεί το ίδιο ποσό εργασίας σε λιγότερο χρόνο, είναι ο γρηγορότερος.

Πως ορίζεται ο χρόνος εκτέλεσης;

- Ο χρόνος **ρολογιού τοίχου** *(wall-clock time)*: συνολικός χρόνος για να ολοκληρωθεί μια εργασία, μαζί με Ε/Ε και όλες τις επιβαρύνσεις του ΛΣ.
- Χρόνος CPU: χρόνος που ασχολείται η CPU με τη συγκεκριμένη διεργασία χωρίς να προσμετράται ο χρόνος Ε/Ε. Χωρίζεται σε:
  - Χρόνος CPU χρήστη για τη διεργασία.
  - Χρόνος CPU λειτουργικού συστήματος για τη διεργασία.
    Παράδειγμα: Σε unix δώστε time <εντολή>
    π.χ. `time ls`
    `0.007u 0.007s 0:00.19`

Κατανόηση της απόδοσης του προγράμματος

Η πιο σημαντική μέτρηση για την απόδοση είναι ο συνολικός **παρελθών χρόνος** *(elapsed time)* μετρημένος με ένα **ρολόι τοίχου** *(wall-clock time)*.

- **Ο χρόνος wall-clock είναι διαφορετικός από το χρόνο clock, γιατί στον πρώτο συμπεριλαμβάνονται και όλες οι καθυστερήσεις**
  (τεχνητές ή όχι, αναμονή για πόρους, καθυστέρηση επικοινωνίας, κτλ).

Ερώτηση αυτοεξέτασης

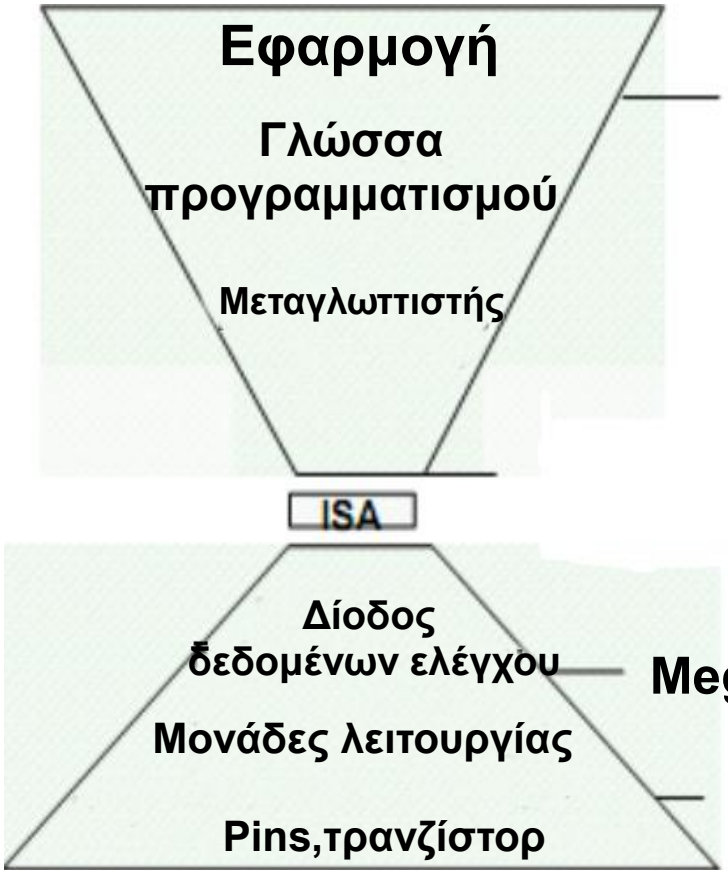
Η απόδοση του υπολογιστή Γ είναι 4 φορές καλύτερη από την απόδοση του υπολογιστή Β, που εκτελεί μια συγκεκριμένη εφαρμογή σε 28 δευτερόλεπτα.

- **Πόσο χρόνο θα χρειαστεί ο υπολογιστής για να εκτελέσει αυτή την εφαρμογή;**

Με ποιον τρόπο μπορούμε να μετρήσουμε την απόδοση;

- Μετροπρογράμματα, Ίχνη εκτέλεσης.
- Συσκευές/Υλικό: κόστος, καθυστέρηση, έκταση, εκτίμηση ισχύος.
- Προσομοίωση (σε πολλά επίπεδα):
  - ISA, RT, πύλη, κύκλωμα
- Θεωρία ουρών.
- Κανόνες χειρισμού.
- Θεμελιώδης “νόμοι”/αρχές.
- **Η αυστηρή κατανόηση των περιορισμών από οποιοδήποτε εργαλείο μέτρησης.**

Η απόδοση μπορεί να μετρηθεί σε διάφορα επίπεδα



Απαντήσεις ανά μήνα
Λειτουργίες ανά δευτερόλεπτο

Megabytes ανά δευτερόλεπτο

Κύκλοι ανά δευτερόλεπτο (clock rate)

Η αύξηση της απόδοσης μεταφράζεται σε $$$

- Οι εταιρίες υπολογιστών ευδοκιμούν ή αποτυγχάνουν ανάλογα με τη σχέση τιμής προς απόδοση.
- Προσπαθούν να έχουν όσο το δυνατόν καλύτερες επιδόσεις σύμφωνα με συγκεκριμένα προγράμματα μέτρησης.
- Μερικές φορές αυτές οι εταιρίες βελτιστοποιούν τα συστήματά τους για να έχουν καλές μετρήσεις ως προς αυτά τα μετροπρογράμματα, απέχοντας όμως από τις πραγματικές εφαρμογές.

Ποιες κατηγορίες προγραμμάτων χρησιμοποιούνται για μέτρηση των επιδόσεων;

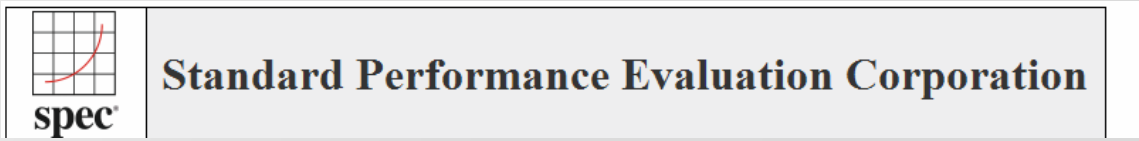
- **Real application:** compilers, software για επεξεργασία κειμένου κλπ.
  - Παίρνουν είσοδο, έξοδο, πιο κοντά στην πραγματικότητα.
- **Modified applications:**
  - Τροποποίηση με σκοπό να αυξήσουν την δυνατότητα μεταφοράς και να εστιάσουν σε μια όψη της απόδοσης.
- **Kemels:** Livermore Loops, Linpack.
  - Απομόνωση απόδοσης ανεξάρτητων χαρακτηριστικών.
- **Toy Benchmarks:** Puzzle, Quicksort
  - Μόνο για πρώτη εντύπωση.
- **Synthetic Benchmarks:** Whetstone, Dhrystone
  - Εξομοίωση της συμπεριφοράς ενός γενικής χρήσης προγράμματος.

Προγράμματα μέτρησης απόδοσης

- Μετροπρογράμματα τυπικού υπολογιστή : εντατικής χρήσης CPU & εντατικής χρήσης GPU.
- Μετροπρογράμματα διακομιστή: εντατικής χρήσης CPU.
- Μετροπρογράμματα ενσωματωμένων συστημάτων: λόγω μεγάλης ποικιλίας σε ενσωματωμένες εφαρμογές και απαιτήσεις, μη ρεαλιστικό ένα απλό σύνολο από αναφορές.

Τι έχει επικρατήσει τελικά;

- Συλλογές από μετροπρογράμματα.
- Έχουν ποικιλία εφαρμογών.
- Η αδυναμία ενός προγράμματος μειώνεται από την παρουσία των άλλων.
- Ένα από τα πιο δημοφιλή είναι SPEC (spec.org).



- Σημαντική είναι η ικανότητα αναπαραγωγής. Δίνουμε λεπτομέρειες του συστήματος για να μπορέσει κάποιος να βγάλει τα ίδια αποτελέσματα.
- Το ΛΣ και ο compiler επηρεάζουν την απόδοση.

Ο νόμος του Amdahl

Νόμος του Amdahl

Ο νόμος υποστηρίζει ότι η βελτίωση της απόδοσης που επιτυγχάνεται με τη χρήση κάποιας πιο γρήγορης μεθόδου εκτέλεσης, **περιορίζεται από το κλάσμα του χρόνου** στον οποίο μπορεί να χρησιμοποιηθεί η μέθοδος αυτή.

Αρχές σχεδιασμού υπολογιστών

- Κάνε την συνηθισμένη κατάσταση γρήγορη
- **Amdahl‘ s law:**

$$\text{Speedup}_{\text{overall}} = \frac{\text{Performance for entire task using the enhancement when possible}}{\text{Performance for entire task without using the enhancement}}$$

$$\text{Speedup}_{\text{overall}} = \frac{\text{Execution time for entire task without using the enhancement}}{\text{Execution time for entire task using the enhancement when possible}}$$

$$\text{Execution time}_{\text{new}} = \text{Execution time}_{\text{old}} \times \left((1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}} \right)$$

$$\text{Speedup}_{\text{overall}} = \frac{\text{Execution time}_{\text{old}}}{\text{Execution time}_{\text{new}}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

Επιτάχυνση

Επιτάχυνση λόγω της ενίσχυσης Ε:

$$\text{Speedup(E)} = \frac{\text{ExTime w/o E}}{\text{ExTime w/ E}} = \frac{\text{Performance w/}}{\text{Performance w/o}}$$

- Υποθέτουμε ότι η ενίσχυση Ε επιταχύνει κλασματικά το έργου με συντελεστή S, και το υπόλοιπο του έργου είναι ανεπηρέαστο.

Νόμος του Amdahl-παράδειγμα

- Ένα πρόγραμμα απαιτεί 100sec για να εκτελεστεί, με 20sec να κατανέμονται στον επεξεργαστή, 60 sec στη μνήμη DDR *(533Mhz)*, 10sec στο σκληρό δίσκο HDD, και 10sec στην κάρτα γραφικών *(υποθέτουμε ότι δεν υπάρχει παραλληλία)*.
- Αν αντικαταστήσουμε το HDD με SSD που είναι 10 φορές πιο γρήγορος, τότε ο χρόνος θα μειωθεί σε 20 + 60 + 1 + 10 = 91 ➔ Βελτίωση κατά 9% Επιτάχυνση κατά 1.098.
- Αν αντικαταστήσουμε τη μνήμη DDR με μνήμη 1333Mhz *(~2.5 φορές πιο γρήγορη)*, τότε ο χρόνος θα μειωθεί σε 20 + 24 + 10 + 10 = 64% ➔ Βελτίωση κατά 36% / Επιτάχυνση κατά 1,562.

Παράδειγμα με νόμο του Amdahl (1/2)

- Η μονάδα πράξεων πραγματικών αριθμών βελτιώθηκε κατά 2Χ. Αλλά, μόνο το 10% των εντολών χρησιμοποιεί εντολές πράξεων πραγματικών αριθμών.

$$ExTime_{new} =$$

$$Speedup_{overall} =$$

Παράδειγμα με νόμο του Amdahl (2/2)

- Το κυμαινόμενο σημείο οδηγιών βελτιώθηκε για να τρέξει στο 2X:Αλλά μόνο το 10% των πραγματικών οδηγιών είναι FP.

$$\text{ExTime}_{new} = \text{ExTime}_{old} \times (0.9 + .1/2) = 0.95 \times \text{ExTime}_{old}$$

$$\text{Speedup}_{overall} = \frac{1}{0.95} = 1.053$$

CPI *(**c**ycles **p**er **i**nstruction)*

κύκλοι ρολογιού ανά εντολή

- **"Μέσο όρο των κύκλων ανά εντολή"**
  CPI=(CPU time* κύκλο ρολογιού)/Αρίθμηση εντολής
  =κύκλοι /αριθμό εντολής:

$$\text{CPU time} = \text{CycleTime} * \sum_{i=1}^{n} \text{CPI}_i * \text{I}_i$$

- **"Συχνότητα χρήσης":**

$$\text{CPI} = \sum_{i=1}^{n} \text{CPI}_i * \text{F}_i \quad \text{where} \quad \text{F}_i = \frac{\text{I}_i}{\text{Instruction Count}}$$

Επένδυση πόρων όπου ο δαπανάται χρόνος!

Παράδειγμα απόδοσης

ο Α έχει χρόνο κύκλου ρολογιού 250ps και CPI 2,0.

ο Β έχει χρόνο κύκλου ρολογιού 500ps και CPI 1,2.

- **Ποιος είναι πιο γρήγορος;**
- Έστω εκτελείται το ίδιο πρόγραμμα με I εντολές.
- Αριθμός Κύκλων ρολογιού CPU A: Ix2,0.
- Αριθμός Κύκλων ρολογιού CPU B: Ix1,2.
- **Χρόνος CPU A:**
  Cycles CPU A x κύκλος ρολογιού = 250 * 2 * I = 500 I
- **Χρόνος CPU B:**
  Cycles CPU B x κύκλος ρολογιού = 500 * 1.2 * I = 600 I

➔ Το Α είναι πιο γρήγορο, γιατί είναι 500*I !

Παράδειγμα υπολογισμού CPI
(η κάθε κατηγορία εντολών έχει διαφορετικό CPI)

Base Machine (Reg / Reg)

Op	Freq	Cycles	CPI(i)	(% Time)
ALU	50%	1	.5	(33%)
Load	20%	2	.4	(27%)
Store	10%	2	.2	(13%)
Branch	20%	2	.4	(27%)
			1.5	

Typical Mix

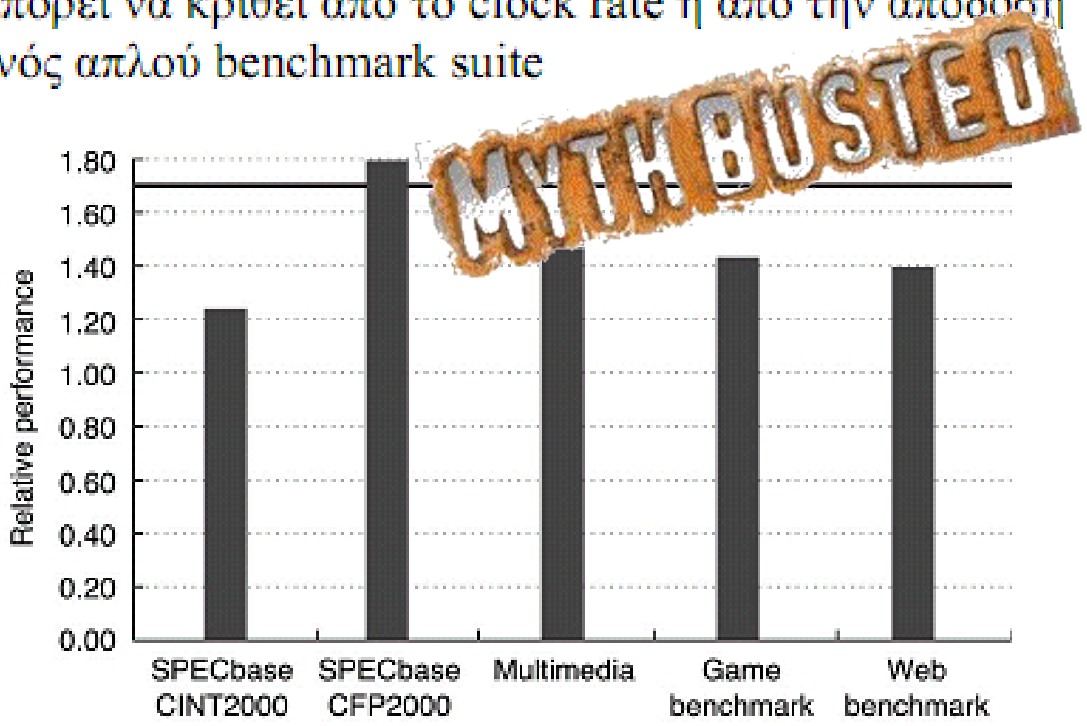
Το CPI μπορεί να είναι μικρότερο από 1 *(superscalar)*.

Σε αυτή την περίπτωση χρησιμοποιούμε το IPC *(Instructions per Clock)*.

Αν CPI = 0,5 τότε IPC = 2,0

Σφάλματα και παγίδες

Σφάλμα: η σχετική απόδοση δύο επεξεργαστών με την ίδια αρχιτεκτονική συνόλου εντολών (ISA) μπορεί να κριθεί από το clock rate ή από την απόδοση ενός απλού benchmark suite



Απόδοση Pentium 4 σχετική με Pentium III

Συστοιχία καταχωρητών *(register file)*

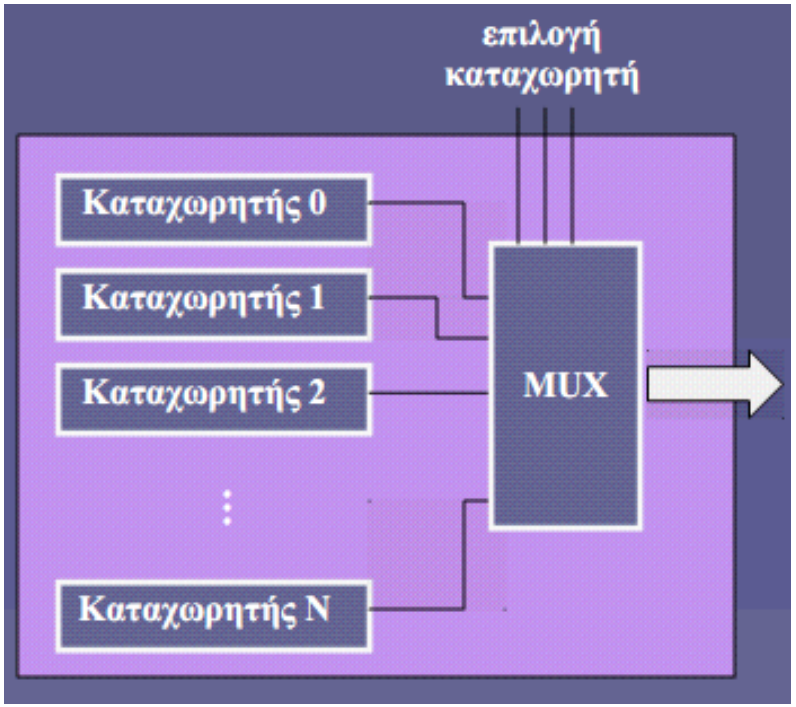
Συστοιχία καταχωρητών: πολύ σημαντικό τμήμα του επεξεργαστή

- **Τι ονομάζουμε register file *(συστοιχία καταχωρητών)*;**

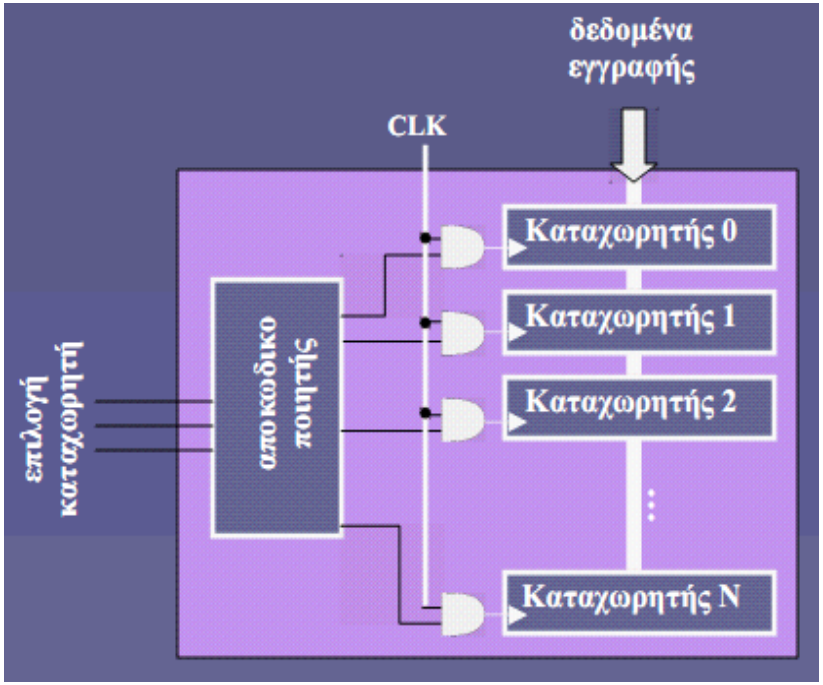
Register file ή συστοιχία καταχωρητών είναι ένα βασικό στοιχείο μιας ΚΜΕ. Συγκεκριμένα, ονομάζονται **οι θέσεις της μνήμης** μέσα στον επεξεργαστή που φυλάσσονται όλοι οι καταχωρητές του συστήματος που φαίνονται στον προγραμματιστή. Παρέχεται η δυνατότητα ταυτόχρονης λειτουργίας (ανάγνωσης ή εγγραφής) σε περισσότερους από έναν καταχωρητές (μνήμη πολλαπλών εισόδων/εξόδων).

Ανάγνωση και εγγραφή στο register file

Ανάγνωση στο register file



Εγγραφή στο register file



Επιπρόσθετα θέματα της ISA

Λεπτομέρειες Σχεδιασμού ISA

- Προκειμένου να σχεδιαστεί μια νέα ISA ποιοι πρέπει να συνεργαστούν;
  - Οι σχεδιαστές της αρχιτεκτονικής *(hardware)* με τους κατασκευαστές των μεταγλωττιστών *(software).*
- Τι σημαίνει και γιατί απαιτείται η συμβατότητα προς τα πίσω σε μια αρχιτεκτονική;
  - Νεότεροι επεξεργαστές εκτελούν το λογισμικό παλαιότερων επεξεργαστών ίδιας αρχιτεκτονικής.
  - Διαφυλάσσεται η επένδυση στο software.
  - Υιοθετούνται πιο εύκολα οι νέοι επεξεργαστές.

Ορισμός ISA

- Τι ορίζεται στο ISA μιας αρχιτεκτονικής;
  - Μοντέλο Μνήμης.
  - Καταχωρητές.
  - Τύποι δεδομένων.
  - Εντολές.
- Πως ορίζεται το ISA;
  - Σε επίσημα έγγραφα που περιέχουν κανονιστικές και πληροφοριακές ενότητες.
  - Είτε εσωτερικά σε μια εταιρία *(αν δε θέλει άλλες εταιρίες να κατασκευάσουν ίδιας αρχιτεκτονικής chip)*.

Τυπικές Καταστάσεις επεξεργαστών

Ερ. Ποιες είναι οι τυπικές καταστάσεις λειτουργίας των σύγχρονων επεξεργαστών; Ποιες καταστάσεις λειτουργίας βρίσκουμε στη IA32;

- Κατάσταση πυρήνα *(επιτρέπονται όλες οι λειτουργίες)*.
- Κατάσταση χρήστη *(περιορισμένη εκτέλεση εντολών)*.
- Το IA32 έχει:
  - Πραγματική κατάσταση.
  - Ιδεατή 8086 κατάσταση.
  - Προστατευμένη λειτουργία.
- Ο 8086 έχει:
  - Πραγματική κατάσταση.

Στοίχιση λέξεων

Τι σημαίνει και γιατί απαιτείται η χρήση στοιχισμένων λέξεων στη μνήμη;

- Τα Byte ομαδοποιούνται σε λέξεις των 4 byte *(32bit)* ή των 8 byte *(64bit).*
- Αρκετές αρχιτεκτονικές απαιτούν οι λέξεις να είναι στοιχισμένες στα φυσικά τους όρια. Για παράδειγμα λέξεις των 4Byte να ξεκινάνε στη διεύθυνση 0 ή 4,8 κτλ.
- Αν οι λέξεις δε ξεκινάνε στα φυσικά τους όρια τότε είναι αστοίχιστες και προκαλούν σημαντική επιβάρυνση στην απόδοση, γιατί απαιτούνται 2 προσβάσεις στις 2 γραμμές της μνήμης.
- Η IA-32 ακολουθώντας την x86 ISA επιτρέπει πρόσβαση ανά byte και δεν έχει τέτοιες απαιτήσεις.

Το πρόβλημα R-A-W

Είναι το πρόβλημα **Read-after-write**, δηλαδή η ανάγνωση μιας τιμής ύστερα από εγγραφή. Υπό κανονικές συνθήκες η τιμή που διαβάζεται σε κάτι που έχει γραφεί, πρέπει να έχει την ίδια τιμή.

π.χ. `mov ax,10; mov apotelesma,ax`
θα πρέπει να γράψει στη θέση μνήμης apotelesma την τιμή 10. Όμως, σύγχρονοι επεξεργαστές μπορούν να εκτελούν εντολές εκτός σειράς *(out-of-order execution, OOO exec)* που σημαίνει ότι αν εκτελεστεί πρώτα η 2η εντολή τότε θα έχουμε πρόβλημα ασυνέπειας.

Αντιμετώπιση

Το πρόβλημα R-A-W αντιμετωπίζεται με:

- Σειροποίηση όλων των αιτήσεων μνήμης (βλάπτει απόδοση).
- Εντολή assembly SYNC για εγγραφή στη μνήμη.
- Ανίχνευση από το υλικό RAW ή WAR, ώστε να μη γίνει OOO για αυτές τις εντολές.

Κατηγορίες καταχωρητών

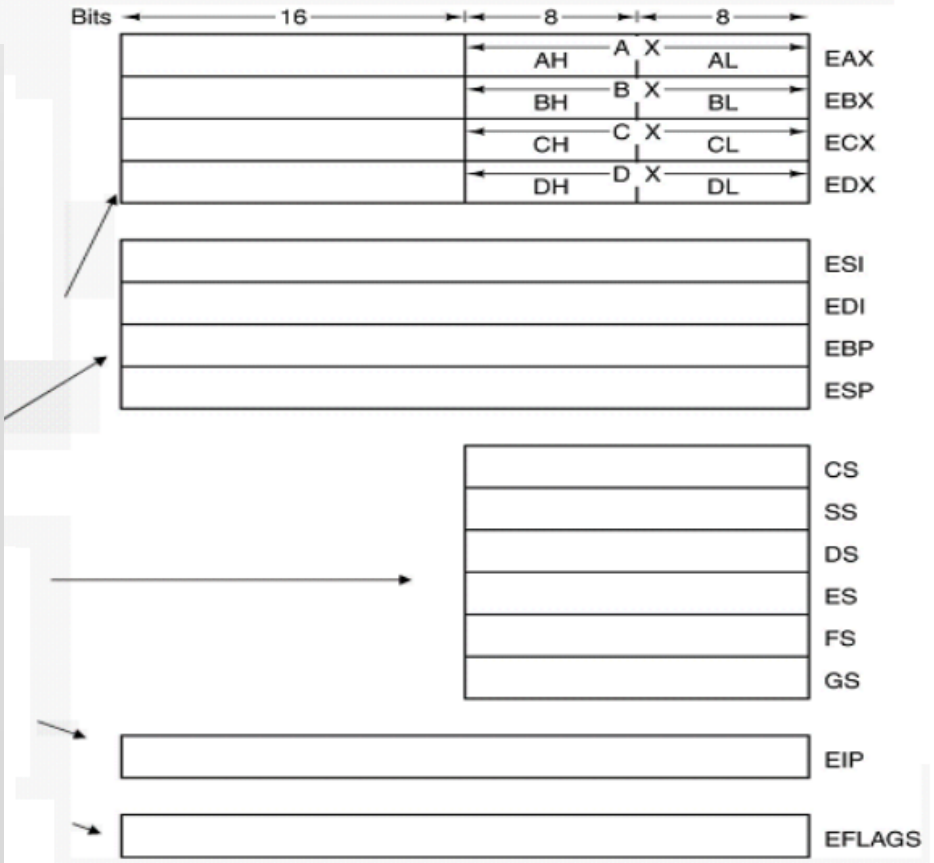
Καταχωρητές Γενικής Χρήσης, Καταχωρητές Ειδικής Χρήσης, Μη ορατοί καταχωρητές

- **8086 *(16bit)*:**
  - Γενικοί: AX, BX, CX, DX.
  - Ειδικοί: CS, DS, ES, SS, DI, SI, BP, SP, SR, IP.
- **IA32 *(32bit)*:**
  - Γενικοί: EAX, EBX, ECX, EDX.
  - Ειδικοί: CS, DS, ES, SS, EDI, ESI, EBP, ESP, EFLAGS, EIP.
- **Ενδεικτικά μη ορατοί καταχωρητές είναι:**
  - Memory Address Register.
  - Instruction register.
  - ALU Input Register 1.
  - ALU Input Register 2.
  - ALU Output Register ....

Καταχωρητές IA-32 (π.χ. Pentium)

Καταχωριστές Pentium (1)

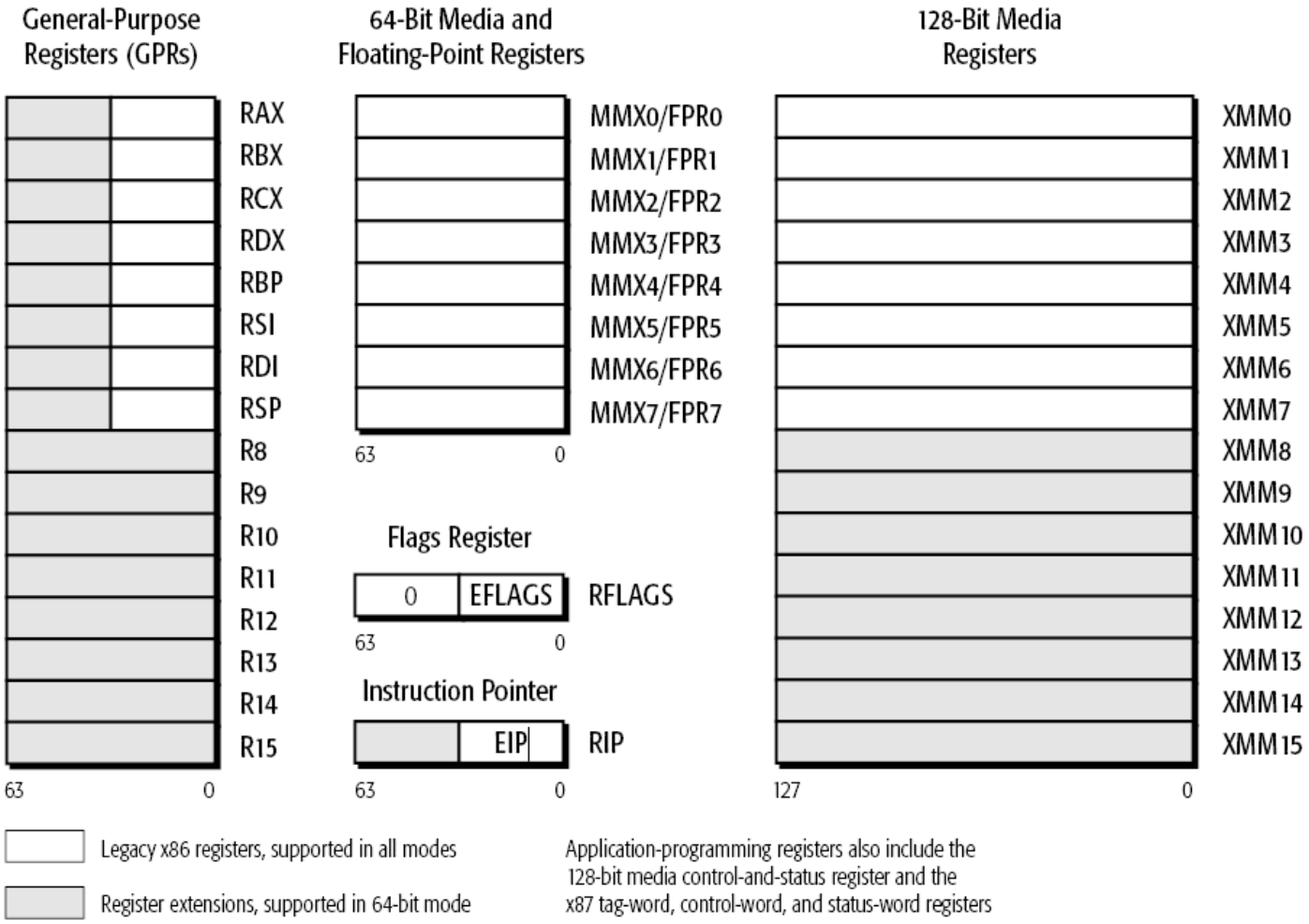
- Γενικοί καταχωρητές, αλλά με κατά περίπτωση ειδικές χρήσεις (mul, div, strings).
- Δείκτες Τοπικών Μεταβλητών, Στοίβας, κλπ.
- Παλιοί δείκτες Τμημάτων (segments).
- Μετρητής προγράμματος.
- Καταχωρητής κατάστασης.



Η αρχιτεκτονική AMD64

- Η AMD x86-64 ή εν συντομία AMD64 είναι συμβατή προς τα πίσω με τη x86 (IA32).
- Η Intel υλοποίησε την AMD64 και την ονόμασε EM64T.
- 64bit χώρος διευθύνσεων.
- 64bit καταχωρητές (υποστηρίζονται και 32bit/16bit).
- 8 γενικού τύπου καταχωρητές R (-AX,...)
- 8 καταχωρητές 128bit (XMM?).
- 8 64bit καταχωρητές (MMX?).
- 64bit καταχωρητές επέκτασης.

Καταχωρητές AMD64 *(π.χ. Intel i7)*



Επιδόσεις αρχιτεκτονικής AMD64

- Αν μια εφαρμογή x86 εκτελεστεί σε λειτουργικό σύστημα AMD64, τότε στην καλύτερη περίπτωση δε θα χειροτερέψει η απόδοση εκτέλεσης.
- Μάλιστα, δεδομένου ότι τα 64bit συστήματα χρησιμοποιούν χαμηλότερες συχνότητες λειτουργίας (λόγω πολλαπλών πυρήνων) από τα 32bit συστήματα, τότε αναμένεται μεγάλη μείωση της απόδοσης.
- Μόνο αν γίνει re-compile για 64bit λειτουργικό σύστημα θα υπάρξουν οφέλη.
- Περισσότερα οφέλη θα υπάρξουν αν η εφαρμογή τροποποιηθεί και παραλληλοποιηθεί.

Οι τυπικές λειτουργίες μιας ISA(1/2)

- **Μετακινήσεις Δεδομένων:**
  - Load *(Καταχωρητής = Μνήμη).*
  - Store *(Μνήμη = Καταχωρητής).*
  - Καταχωρητής – Καταχωρητής .
  - Μνήμη – Μνήμη.
  - Λειτουργίες Στοίβας *(ως αδιαίρετη πράξη μετακίνησης).*
  - Λειτουργίες Εισόδου/Εξόδου *(ως μετακίνηση σε ειδική θέση).*
- **Αριθμητικές Πράξεις:**
  - Πρόσθεση, Αφαίρεση, Αντίθετο, Σύγκριση, Πολ/σμός, Διαίρεση.
  - Ακέραιοι και Πραγματικοί *(Κινητή Υποδιαστολή)*.
- **Ολίσθηση:**
  - Αριθμητική/Λογική, Δεξιά/Αριστερά.

Οι τυπικές λειτουργίες μιας ISA(2/2)

- **Λογικές Πράξεις:**
  - AND, OR, XOR, NOT, Set, Clear.
- **Έλεγχος Ροής Προγράμματος:**
  - Διακλάδωση χωρίς/με συνθήκη.
  - Κλήση/Επιστροφή υπορουτίνας.
- **Είσοδος/Έξοδος** *(ως ξεχωριστές λειτουργίες):*
  - Ανάγνωση, Εγγραφή.
- **Διακοπές και Παγίδες:**
  - Έλεγχος, Εξυπηρέτηση, Επιστροφή.
- **Επιπλέον: Συγχρονισμός και Νήματα.**
  - Strings, Γραφικά, Streaming..

Ο ειδικός καταχωρητής σημαιών *(Status Register, SR)*

- Ο μόνος που επιτρέπει πρόσβαση ανά bit.
- Ενημερώνεται μόνο από την ALU.
- Έχει bit για κρατούμενο, μηδενικό αποτέλεσμα, υπερχείλιση, αρνητικό κτλ.
- Οι εντολές σύγκρισης και διακλάδωσης χρησιμοποιούν τα bit αυτά. Για παράδειγμα αν στην εντολή `cmp AL,BL` ενεργοποιηθεί η σημαία **ZF** *(zero flag)* τότε η επόμενη εντολή `je label3` θα είναι αληθής *(je=jump if equal)* και θα αλλάξει η ροή εκτέλεσης.

Αναπαράσταση λογικών τιμών και χάρτες bit

- Μια λογική τιμή *(boolean)* έχει είτε τιμή **0** *(FALSE)* είτε **1** *(TRUE).*
- Αν και είναι ένα bit τοποθετείται σε 1 Byte επειδή οι επεξεργαστές υποστηρίζουν πρόσβαση άνα byte και όχι bit. Δλδ, το TRUE γράφεται 00000001.
- Η μόνη περίπτωση όπου καταλαμβάνει 1 bit είναι στους χάρτες bit, όπου είναι δομές ειδικού τύπου. Χρησιμοποιούνται σε πολλές περιπτώσεις, όπως στην παρακολούθηση των ελεύθερων ενοτήτων *(blocks)* του σκληρού δίσκου. Το πρώτο bit π.χ. δείχνει αν είναι ελεύθερο το πρώτο block του δίσκου.

Κριτήρια σχεδιασμού μορφής εντολών ISA

- Δυνατότητα να προστίθενται νέες εντολές.
- Μελλοντική αξιοποίηση τεχνολογίας.
- Τρέχουσα τεχνολογία και εκτίμηση μελλοντικών δυνατοτήτων.
- Οι μικρές εντολές καλύτερες από τις μεγάλες *(σε αριθμό bit)*: λιγότερο χώρο στη μνήμη, πιο εύκολη αποκωδικοποίηση, επικαλυπτόμενη εκτέλεση, καλύτερη εκμετάλλευση του εύρους ζώνης της μνήμης, πιο αποδοτική χρήση κρυφής μνήμης.
- Αριθμός εντολών που συνδέεται με τα bit του opcode.
- Αριθμός bit παραμέτρων διευθυνσιοδότησης.

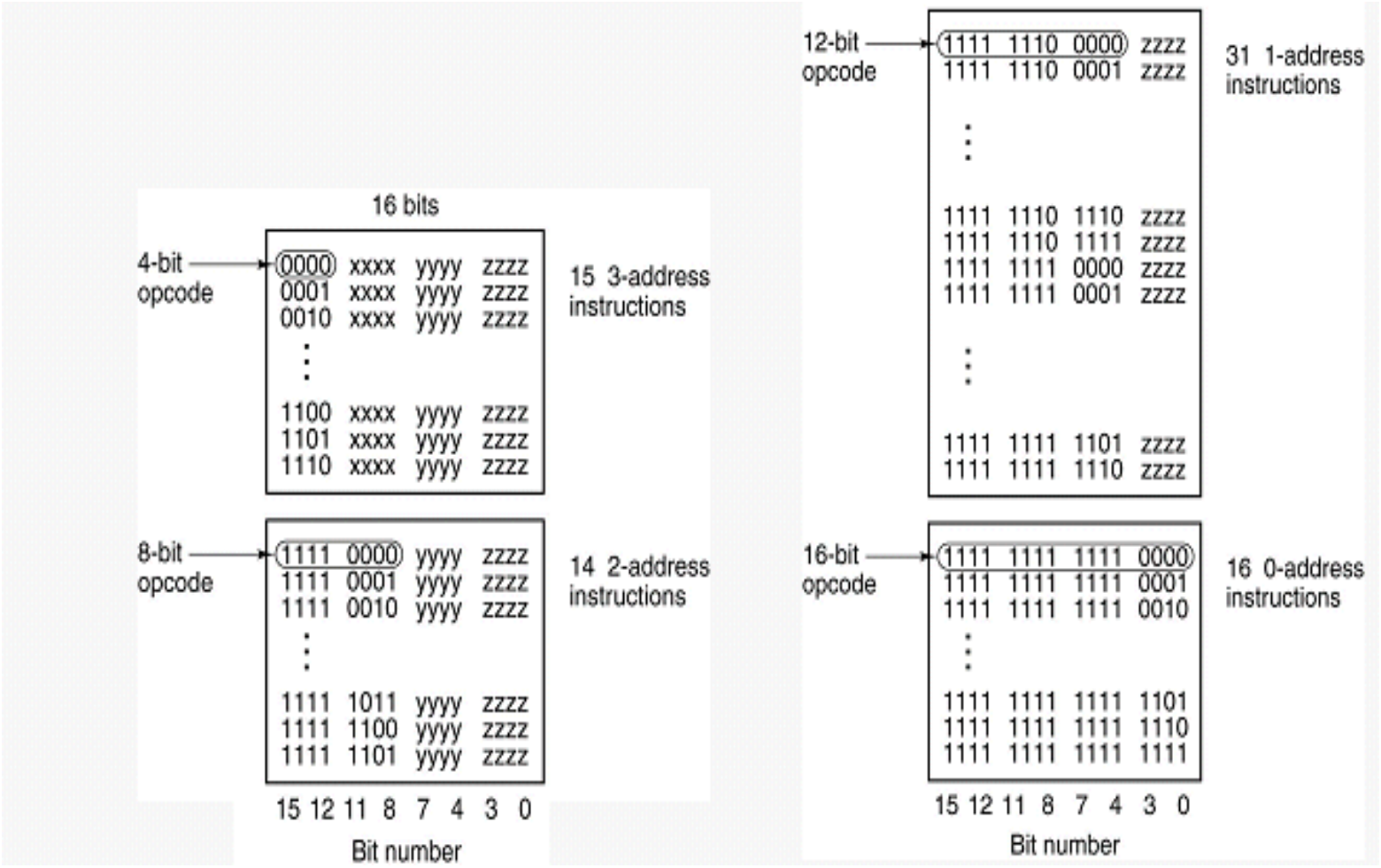
Πώς υλοποιούνται οι ανάγκες για συμβιβασμό bit op-code και operands;

- Σε αρχιτεκτονικές με σταθερό μέγεθος machine code *(π.χ. 32Bit)* όσο μεγαλύτερο σε αρ.bit είναι το machine code, τόσο μικρότερη ευελιξία έχουμε στις παραμέτρους.
- Για να δημιουργήσουμε ευελιξία χρησιμοποιούμε τον **επεκτεινόμενο** κωδικό πράξης.
- Ομαδοποιούμε τα opcodes σε ομάδες με μεταβλητό μέγεθος bit (*opcodes 4bit,8bit,12bit,16bit...)*.
- Αποφασίζουμε μια σειρά από bit που δείχνει ότι υπάρχει και επόμενα 4bit στον opcode *(π.χ. 1111).*
  - Με αυτόν τον τρόπο οι εντολές 0000 – 1110 είναι εντολές με 4bit opcode και τα υπόλοιπα στους παραμέτρους.
  - Οι εντολές 1111 0000 - 1111 1110 είναι εντολές με 8bit opcode και τα υπόλοιπα στους παραμέτρους κτλ.

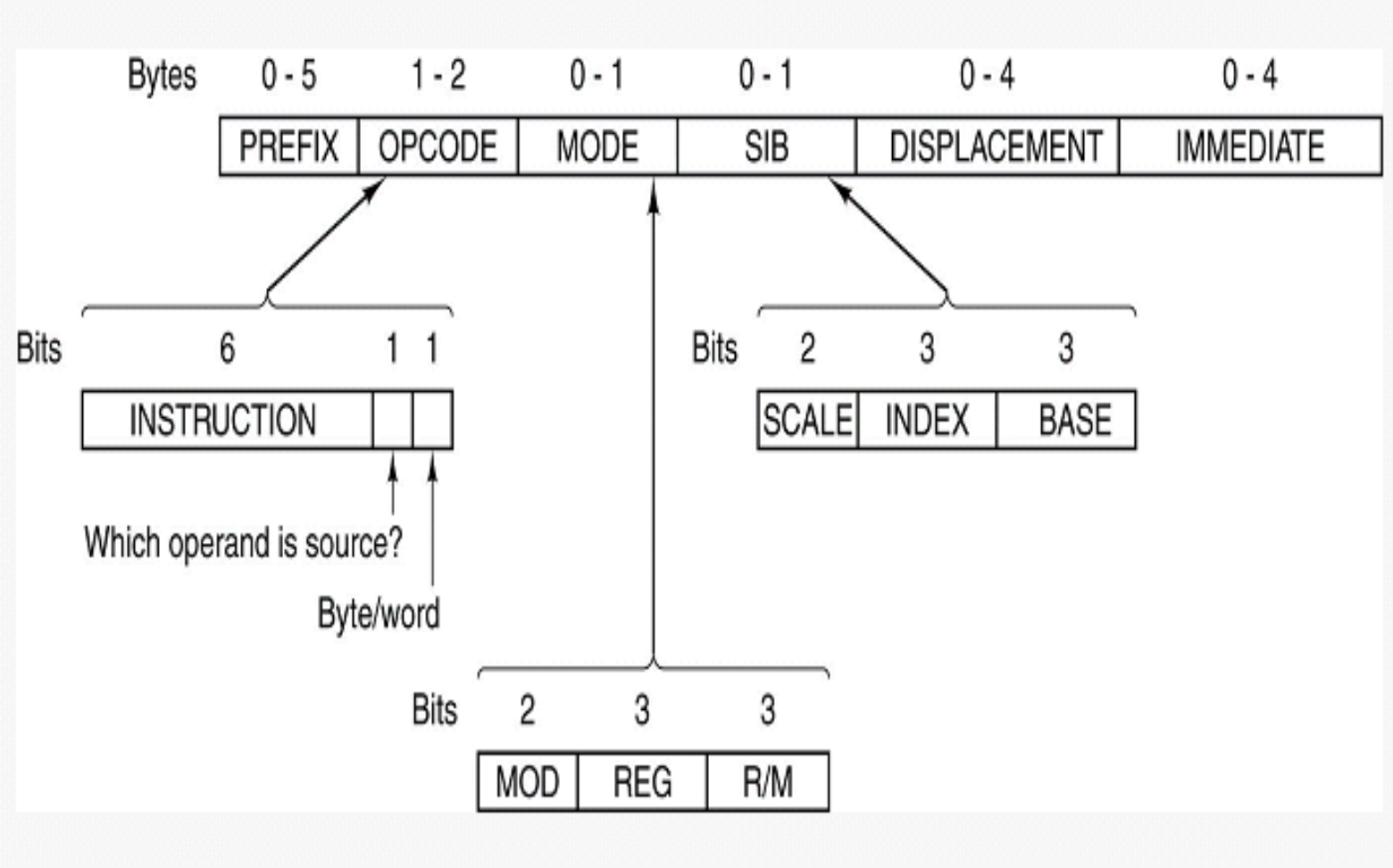
Ο επεκτεινόμενος κωδικός πράξεων (1/2)

- Μας επιτρέπει ένα συμβιβασμό ανάμεσα στο χώρο των κωδικών πράξεων *(opcodes)* και στο χώρο των παραμέτρων *(operands).*
- Διατήρηση όλων των εντολών στο ίδιο μέγεθος.
- Ελαχιστοποιείται το μέγεθος της μέσης εντολής, με την επιλογή των μικρότερων κωδικών πράξεων για τις πιο συνηθισμένες εντολές.

Ο επεκτεινόμενος κωδικός πράξεων (2/2)



Μορφή Εντολών IA-32



Η δυσκολία αποκωδικοποίησης των εντολών x86

- Η x86 ISA έχει μεταβλητό μέγεθος machine code.
- Χρησιμοποιείται μια πολύ σύνθετη κωδικοποίηση στην οποία απαιτείται να γίνει πλήρης αποκωδικοποίηση όλου του machine code για να προσδιοριστεί τι είδους πράξη πρόκειται να εκτελεστεί και επομένως τι μήκος έχει η εντολή.

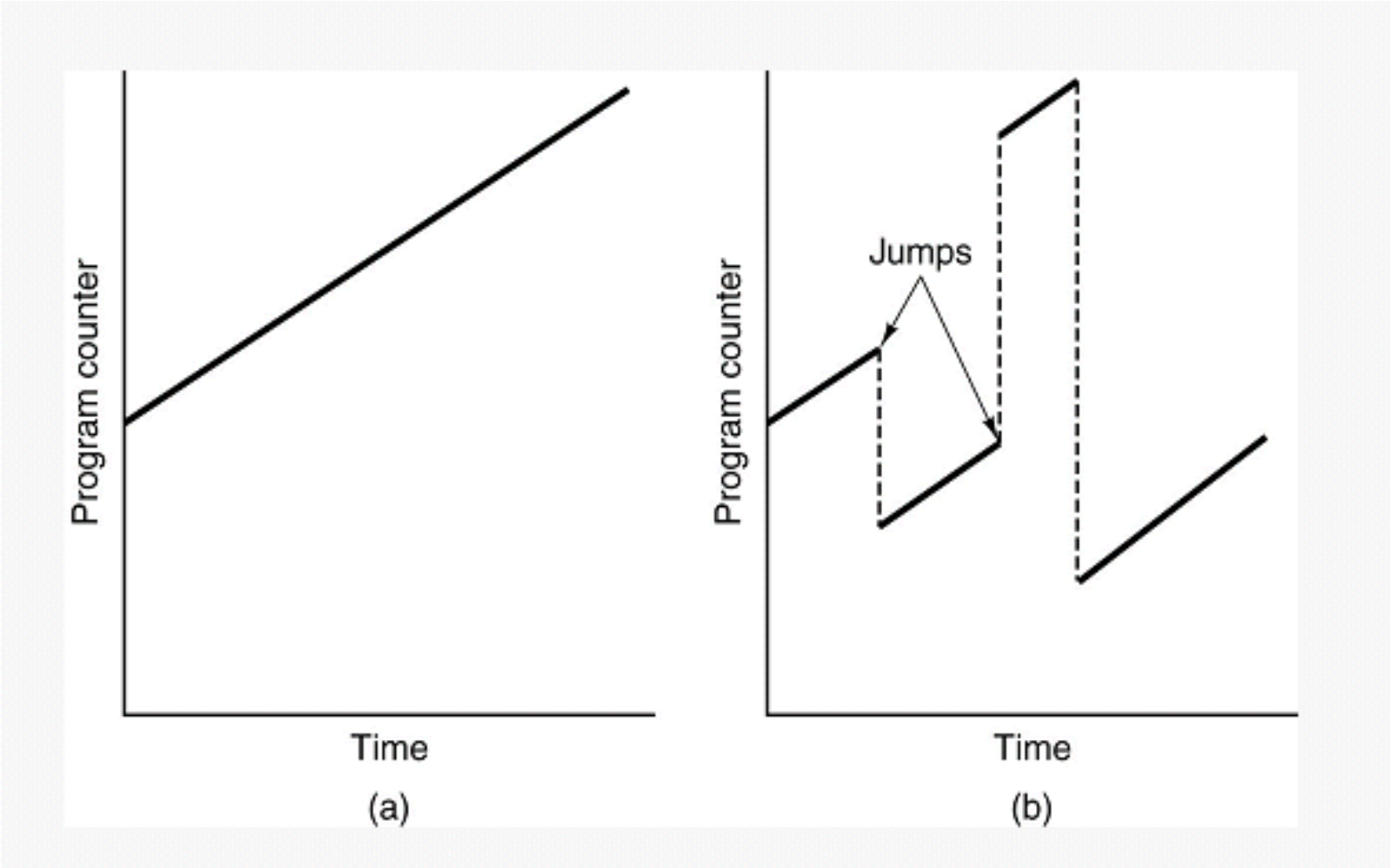
Οι διμελείς και μονομελείς πράξεις

- Μια διμελής πράξη σημαίνει ότι δέχεται δύο παραμέτρους. Για παράδειγμα η εντολή **`mov al,5`** δέχεται τις παραμέτρους al και 5.
- Μια μονομελής πράξη δέχεται μια μόνο παράμετρο. Για παράδειγμα η εντολή αύξησης κατά 1 **`inc AL`** δέχεται μόνο την παράμετρο AL.
- Μια πράξη χωρίς παραμέτρους είναι εντολή επιστροφής συνάρτησης **`ret`** η οποία δε δέχεται παράμετρο.
- Στη x86 ISA δεν υπάρχουν πράξεις με άλλο αριθμό παραμέτρων *(π.χ. 3 operands)*.

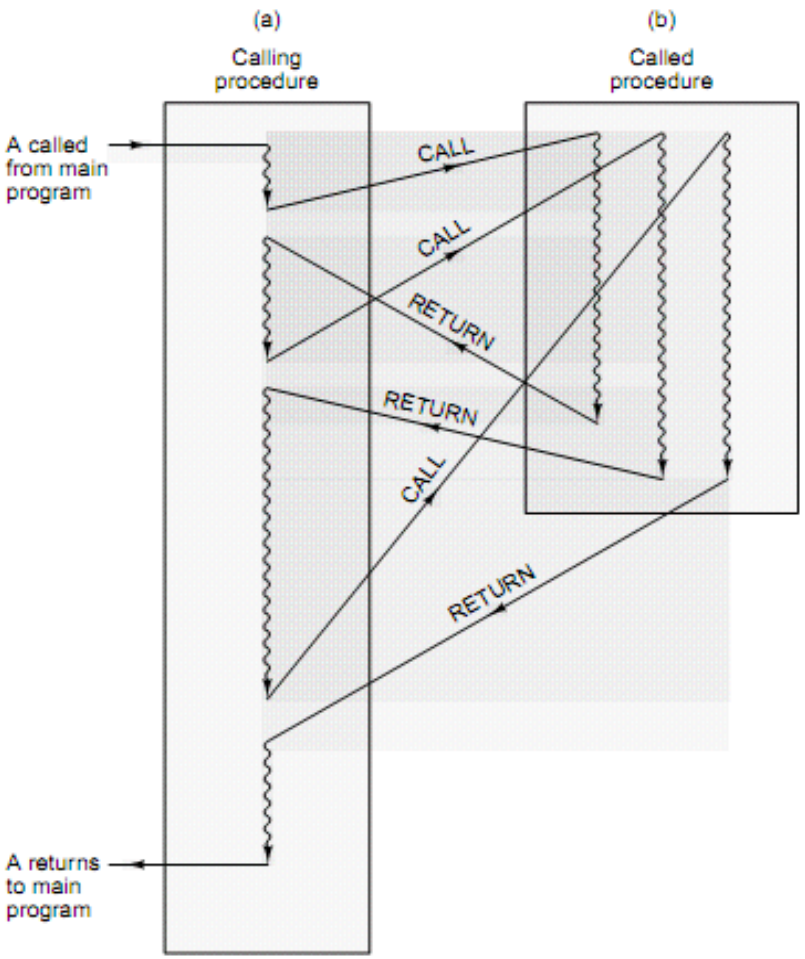
Δημιουργία μασκών bit με AND και OR

- Η AND χρησιμοποιείται για την **εξαγωγή bit** από λέξεις. Για να το πετύχουμε αυτό χρησιμοποιούμε μια μάσκα από 0 και 1. Όπου έχει 1 η μάσκα, θα εξαχθεί αυτό το bit. Για παράδειγμα, αν θέλουμε να εξάγουμε τα τέσσερα τελευταία bit θα χρησιμοποιήσουμε τη μάσκα 00001111.
- Η OR χρησιμοποιείται για να **εντάξουμε κάποια bit** σε μια λέξη. Για να το πετύχουμε αυτό χρησιμοποιούμε μια μάσκα από 1 και 0. Όπου έχει 1 η μάσκα τότε στο αποτέλεσμα θα τοποθετηθεί 1 σε αυτό το σημείο. Για παράδειγμα, αν θέλουμε να τοποθετήσουμε στα τέσσερα τελευταία bit το ένα, θα χρησιμοποιήσουμε τη μάσκα 00001111.

Ακολουθιακή ροή ελέγχου και διακλαδώσεις



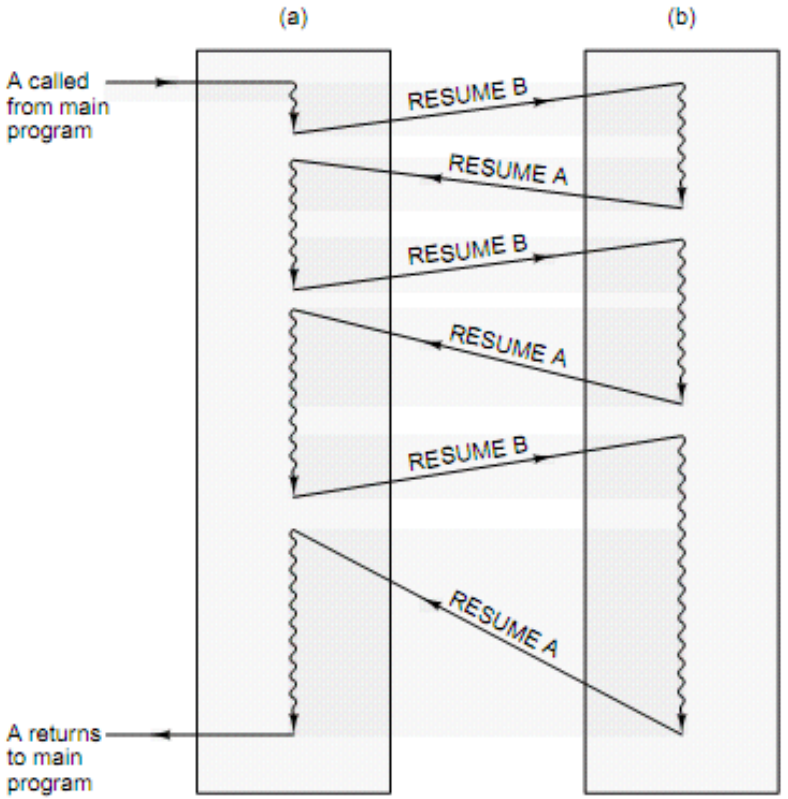
Κλήσεις υπορουτίνων



- Με την κλήση υπορουτίνων αλλάζει η ροή εκτέλεσης.
- Τροποποιείται ο PC (CS:IP για x86).
- Εκτελείται ένα άλλο κομμάτι κώδικα.
- Μια συνάρτηση αν καλεί τον εαυτό της ονομάζεται 'αναδρομική'.

Συρρουτίνες

- Σε αντίθεση με τις υπορουτίνες, κάθε φορά που καλείται η διαδικασία, συνεχίζει η εκτέλεση από το σημείο που είχε σταματήσει σε προηγούμενη εκτέλεση.
- Δημιουργείται ψευδοπαραλληλία.
- Απαιτείται software εξομοίωση για να επιτευχθεί.



Υπερβαθμωτοί επεξεργαστές *(super-scalar)*

Υπερβαθμωτοί επεξεργαστές

- Ένας υπερβαθμωτός επεξεργαστής έχει πολλαπλές ροές εκτέλεσης.
- Υλοποιεί τον παραλληλισμό σε επίπεδο εντολής *(instruction level parallelism).*
- Έχει πολλαπλές λειτουργικές μονάδες *(π.χ. ALU, shifters, Multiplier, FPU).*
- Σύγκριση με την τεχνική της διασωλήνωσης:
  - Η διασωλήνωση και η υπερβάθμωση είναι διαφορετικές τεχνικές.
  - Οι υπερβαθμωτοί επεξεργαστές χρησιμοποιούν και την τεχνική της διασωλήνωσης.

Η λειτουργία των υπερβαθμωτών επεξεργαστών

- Υπάρχει μια ακολουθιακή ροή εντολών.
- Η CPU ελέγχει για εξαρτήσεις δεδομένων ανάμεσα στις εντολές κατά τη διάρκεια εκτέλεσης *(run-time).*
- Εντολές που δεν έχουν εξαρτήσεις δίνονται σε λειτουργικές μονάδες που εκείνη τη στιγμή δε χρησιμοποιούνται.
- Όλοι οι επεξεργαστές γενικού σκοπού από το 1998 και μετά είναι υπερβαθμωτοί.
- Ο πρώτος υπερβαθμωτός επεξεργαστής ήταν ο Pentium P5.

Οι πρώτοι υπερβαθμωτοί x86 επεξεργαστές

- Ο P5 Pentium*, (nextgen)* Nx586, P6 Pentium Pro, AMD K5 ήταν οι πρώτοι x86 υπερβαθμωτοί cpu.
- Αποκωδικοποιούσαν τις macro-code σε micro-code εντολές πριν την εκτέλεση.
- Αυτό επέτρεψε:
  - τη δυναμική χρονοδρομολόγηση.
  - την αύξηση της παραλληλίας.
  - την υποθετική εκτέλεση *(speculative execution).*
  - επίτευξη ακόμη μεγαλύτερης συχνότητας ρολογιού.

Υπερβαθμωτοί Επεξεργαστές (1/2)

- Οι υπερβαθμωτοί επεξεργαστές στοχεύουν:
  - στην ακρίβεια του διαμεταβιβαστή εντολών *(instruction dispatcher)* ώστε να μη παραβιάζονται οι εξαρτήσεις.
  - στην όσο δυνατόν μεγαλύτερη χρησιμοποίηση των λειτουργικών μονάδων που διαθέτουν.
- Οι πρώτοι υπερβαθμωτοί cpu είχαν 2 ALU, 1FPU.
- Οι σύγχρονοι υπερβαθμωτοί cpu *(π.χ. PowerPC 970-Apple)* 4 ALU, 2 FPU, 2 SIMD cores.
- Επιτυγχάνουν εκτέλεση μεγαλύτερη από 1 εντολή ανά κύκλο μηχανής.

Υπερβαθμωτοί Επεξεργαστές (2/2)

- Υπάρχουν αρκετές τεχνολογίες που επιτυγχάνουν την παράλληλη εκτέλεση εντολών:
  - Διασωλήνωση.
  - Πολλαπλοί επεξεργαστές *(σε διαφορετική συσκευασία).*
  - Πολλαπλοί πυρήνες *(στην ίδια συσκευασία).*
- Το πιο σημαντικό στοιχείο είναι ο διαμεταβιβαστής εντολών *(instruction dispatcher)* που διαβάζει τη ροή των εντολών και αποφασίζει άμεσα ποιες μπορούν να εκτελεστούν από ανενεργές μονάδες.
- Υπάρχει μια κοινή ροή εντολών για όλο το σύστημα.

Στοιχεία που μειώνουν τις επιδόσεις στου superscalar

- Τα παρακάτω στοιχεία μειώνουν τις επιδόσεις των υπερβαθμωτών επεξεργαστών:
  - Ο βαθμός της παραλληλίας της ροής εντολών.
  - Η πολυπλοκότητα και η χρονική επιβάρυνση του διαμεταβιβαστή εντολών.
  - Η επεξεργασία εντολών στους βρόγχους *(δηλαδή, ποια ροή να ακολουθήσει).*
- Τα προγράμματα έχουν ποικίλους βαθμούς παραλληλίας επιπέδου εντολής.
  - Παράδειγμα:
    - `a=b+c; d=e+f` *(υπάρχει παραλληλία).*
    - `a=b+c; b=e+f` *(δεν υπάρχει παραλληλία).*

Υπάρχουν προβλήματα με την αύξηση της παραλληλίας (1/2)

- Με την αύξηση της παραλληλίας, αυξάνεται εκθετικά το κόστος ελέγχου των εξαρτήσεων.
- Το πρόβλημα οξύνεται γιατί πρέπει οι αποφάσεις να λαμβάνονται κατά το χρόνο εκτέλεσης και στη συχνότητα λειτουργίας του υπολογιστή.
- Πρέπει να ληφθεί υπόψιν η χρονική και υλική επιβάρυνση του κυκλώματος του διαμεταβιβαστή εντολών *(λογικές πύλες).*
- Το υλικό κόστος έχει βρεθεί πως είναι nk, ενώ η καθυστέρηση k2logn *(n αρ. εντολών ISA, k βαθμός παραλληλίας).*

Υπάρχουν προβλήματα με την αύξηση της παραλληλίας (2/2)

- Το γεγονός ότι αυξάνεται η καθυστέρηση αναλόγως του βαθμού της παραλληλίας θέτει ένα **φυσικό άνω όριο** στο πόσες λειτουργικές μονάδες *(ALU, FPU κ.α.)* θα τοποθετηθούν πάνω στο ίδιο ολοκληρωμένο κύκλωμα.
- Για αυτό αν και υπάρχει τεχνολογική πρόοδος που το επιτρέπει, δεν τοποθετούνται πολλαπλές λειτουργικές μονάδες.
- Εκτός από τα παραπάνω αν υπάρχουν πολλαπλές εξαρτήσεις δεδομένων, τότε η υπερβαθμωτή εκτέλεση δε θα έχει οφέλη.

Η αρχιτεκτονική IA-64
(Itanium 64bit)

Τα προβλήματα της αρχιτεκτονικής IA32 (1/2)

- Απαιτείται πολύ υλικό *(transistor)* για να μετατρέπονται οι CISC εντολές σε μικροεντολές RISC.
- Είναι προσανατολισμένη στη χρήση μνήμης. Οι διαρκείς αναφορές στη μνήμη μειώνουν τις επιδόσεις.
- Έχει μικρό σύνολο καταχωρητών. Τα αποτελέσματα πρέπει να απομακρύνονται από τους καταχωρητές.
- Δημιουργούνται εξαρτήσεις WAR, λόγω του μικρού αριθμού καταχωρητών. Υπάρχει πρόβλημα σε εκτέλεση εκτός σειράς.

Τα προβλήματα της αρχιτεκτονικής IA32 (2/2)

- Απαιτείται μεγάλη διοχέτευση, που σημαίνει ότι απαιτείται μεγάλη και ακριβή πρόγνωση διακλαδώσεων.
- Απαιτείται εικαζόμενη εκτέλεση για να αποφεύγεται η καθυστέρηση από τις λανθασμένες προγνώσεις.
- Διευθυνσιοδοτούνται μόνο 4GB μνήμης.

Η IA64 (2001) προτάθηκε για να λύσει τα προβλήματα της x86 (1/2)

- RISC 64bit.
- 2 καταστάσεις λειτουργίας.
- 64 γενικοί καταχωρητές.
- Όλες οι εντολές σταθερή μορφή.
- Πολλές διαθέσιμες λειτουργικές μονάδες.
- Δέσμη σχετικών εντολών *(ομαδοποίηση ανά 3).*
- Ο μεταφραστής καθορίζει την ομάδα που μπορεί να εκτελεστεί παράλληλα.

Η IA64 (2001) προτάθηκε για να λύσει τα προβλήματα της x86 (2/2)

- → EPIC *(explicitly parallel instruction computing).*
- → Υπολογιστική ρητής παραλληλίας εντολών.
- Απλούστερο υλικό: περισσότερα transistor διαθέσιμα π.χ. για cache.
- Χαμηλότερη κατανάλωση ενέργειας.
- Χρήση της κατηγορηματοποίησης *(predication execution).*
- Χρήση της εικαζόμενης φόρτωσης *(speculative load).*

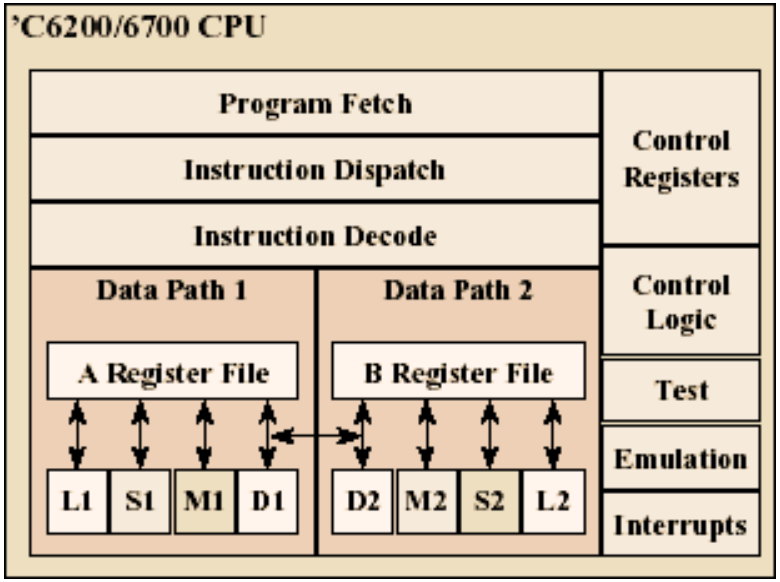
Η αρχιτεκτονική VLIW (1/2)

- VLIW = Very Large Instruction Word.
- Αρχιτεκτονική που εκμεταλλεύεται την παραλληλία εντολών.
- Διαφορετική από διασωλήνωση, υπερβάθμωση.
- Στη VLIW η παραλληλία ορίζεται κατά το χρόνο συμβολομετάφρασης *(από τον compiler ή το χρήστη).*
- Ο επεξεργαστής δε χρειάζεται το επιπρόσθετο υλικό ανίχνευσης εξαρτήσεων και χρονοδρομολόγησης για εκτέλεση εκτός σειράς ή υπερβαθμωτή εκτέλεση.
- Απαιτείται περισσότερη πολυπλοκότητα στο compiler παρά στο hardware.
- Ο αριθμός των λειτουργικών μονάδων *(alu, fpu...)* είναι φανερός στο συμβολομεταφραστή, αφού πρέπει να ξέρει τι έχει στη διάθεσή του.

Η αρχιτεκτονική VLIW (2/2)

- Υπάρχουν πολλαπλά ταυτόχρονα opcodes στο Machine Code για παραπάνω από μια λειτουργίες. Π.χ. Αν έχουμε 5 μονάδες εκτέλεσης, τότε κάθε machine code θα έχει 5 opcodes για κάθε μια μονάδα.
- Δεν υπάρχει συμβατότητα προς τα πίσω, αφού νέοι επεξεργαστές είχαν νέες επιπρόσθετες μονάδες. Κάποιοι κατασκευαστές χρησιμοποιούν μετατροπείς binary-to-binary.
- Το machine code συνήθως έχει μέγεθος από 64bit.
- Μειονέκτημα: Αν οι άλλες μονάδες δεν έχουν κάτι να εκτελέσουν τότε εκτελούν NOP *(No Operation).*
- Δε χρησιμοποιείται για general computing *(είναι δύσχρηστο),* αφού υπάρχει διαθεσιμότητα εκατομμυρίων τρανζιστορ onchip για το ειδικό hardware.
- Παραδείγματα: TriMedia, TI C6000 DSP, ST2000, ATI RADEON R600.

Παράδειγμα VLIW *(TI 6XC CPU)*



- Κάθε γραμμή 256 bit.
- 32 bit registers.
- Κάθε γραμμή έχει 8 εντολές των 32bit που κατανέμονται σε 2 διαδρομές δεδομένων.
  - 2 ALUs *(fixed point).*
  - 4 ALUs *(fixed/floating).*
  - 2 MULs *(fixed/floating).*

Έτσι σε κάθε γραμμή κώδικα υπάρχουν 8 εντολές:

ALU1	ALU2	ALU3	ALU4	ALU5	ALU6	ALU7	ALU8

Διαφορές και ομοιότητες superscalar / EPIC

- Και στις δύο τεχνικές υποστηρίζεται παράλληλη εκτέλεση σε πολλαπλές λειτουργικές μονάδες.
- **EPIC:** Η παραλληλία εντολών καθορίζεται κατά το χρόνο μετάφρασης από το compiler. Οι εξαρτήσεις δεδομένων καθορίζονται από το compiler. Οδηγεί σε απλούστερο υλικό. Υλοποιεί την τεχνική **VLIW**.
- **Superscalar:** Η παραλληλία εντολών καθορίζεται από το υλικό *(επεξεργαστή).* Απαιτείται πολύ πιο σύνθετο υλικό *(αυξημένος αριθμός transistor).* Οι εξαρτήσεις δεδομένων καθορίζονται από το υλικό.

Τέλος Ενότητας

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης