ΗΥ425 - Αρχιτεκτονική Υπολογιστών - M1

Χ. Σωτηρίου

20 Σεπτεμβρίου 2001

Αρχιτεκτονική Υπολογιστών

καλύπτει και τις τρείς όψεις της σχεδίασης υπολογιστών:

- αρχιτεκτονική συνόλου εντολών - σύνορο μεταξύ λογισμικού/υλικού.
- οργάνωση - υψηλού επιπέδου δομή της σχεδίασης.
- υλικό (*hardware*) - λογική σχεδίαση και τεχνολογία πακεταρίσματος.

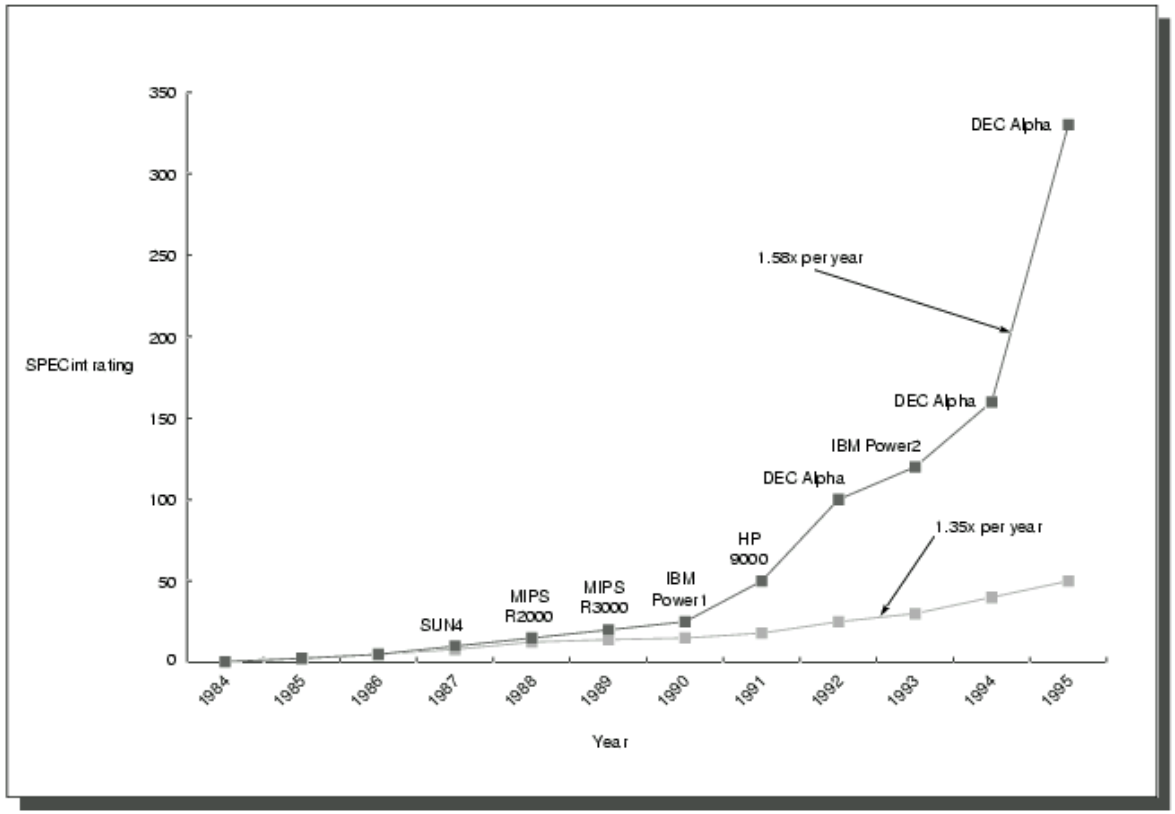
Ο ρόλος του Αρχιτέκτονα Υπολογιστών

- να αποφασίσει για τα σημαντικά χαρακτηριστικά ενός νέου μηχανήματος.
- να σχεδιάσει ένα νέο μηχάνημα με την μέγιστη απόδοση μέσα στους περιορισμούς κόστους.

Λειτουργικά Χαρακτηριστικά Αρχιτεκτονικών

Λειτουργική Απαίτηση	Τυπικά χαρακτηριστικά που απαιτούνται/υποστηρίζονται
Περιοχή Εφαρμογής	Στόχος του Υπολογιστή
Γενικού σκοπού	Ισορροπημένη απόδοση για σύνολο εργασιών
Επιστημονική	Υψηλή απόδοση για αριθμητικές πράξεις ρητών (FP)
Βιομηχανική	Υποστήριξη για δεκαδική αριθμητική, βάσεις δεδομένων και συναλλαγές.
Συμβατότητα Λογισμικού	Ορίζει το υπάρχον λογισμικό για την μηχανή
στην γλώσσα προγραμματισμού	πιο ευλύγιστος τρόπος· απαιτείται νέος μεταγλωτιστής
συμβατότητα στον κώδικα μηχανής	ορισμένη αρχιτεκτονική συνόλου εντολών, λίγη ευλυγισία αλλα εύκολο τρέξιμο υπαρχόντων προγραμμάτων.
Απαιτήσεις Λειτουργικού	Απαραίτητα χαρακτηριστικά για υποστήριξη ΛΣ
Μέγεθος διαστήματος διευθύνσεων	πολύ σημαντικό· περιορίζει εφαρμογές
Διαχείριση μνήμης	απαραίτητη για μοντέρνα λειτουργικά· σελιδοποιημένη η διαχωρισμένη μνήμη.
Προστασία	απαιτήσεις λειτουργικού/εφαρμογών· σελιδοποιημένη η διαχωρισμένη προστασία
Υποστήριξη Προδιαγραφών	Συγκεκριμένες προδιαγραφές μπορεί να απαιτούνται
Αριθμητική ρητών (FP)	$IEEE$, DEC, IBM
Δίαυλος Εις/Εξ	για συσκευές Εις/Εξ: VME, $SCSI$, $FireWire$
Λειτουργικά συστήματα	$UNIX$, DOS ή εσωτερικό
Δίκτυα	υποστήριξη για δίκτυα: $Ethernet$, ATM
Γλώσσες προγραμματισμού	C, $C++$, $Java$, $Fortran$ 77

Αύξηση Απόδοσης Μικροεπεξεργαστών



Τάσεις Χρήσης Υπολογιστών

- αυξανόμενη χρήση μνήμης - αύξηση επί 1.5 με 2 τον χρόνο (1/2 με 1 *bit* το χρόνο).
- αντικατάσταση της γλώσσας ***assembly*** απο γλώσσες υψηλού επιπέδου.
- συνεχής ανάπτυξη της τεχνολογίας μεταγλώτισης - έξυπνοι μεταγλωτιστές.

Τάσεις Τεχνολογίας Κατασκευής

- Τεχνολογία Ολοκληρωμένων Κυκλωμάτων: αύξηση πυκνότητας τρανζίστορ: 50% το χρόνο, μέγεθος ολοκληρωμένου 10-25% το χρόνο. Συνολικά: 60-80% αύξηση τρανζίστορ το χρόνο - πρόβλημα: μεταλλικές συνδέσεις.

- Μνήμη *DRAM*: αύξηση χωρητικότητας λίγο κάτω απο 60% τον χρόνο. Ο κύκλος πρόσβασης μειώνεται πολύ αργά: περίπου 1/3 σε 10 χρόνια.

- Τεχνολογία Μαγνητικών Δίσκων: τελευταία η αύξηση πυκνότητας είναι περίπου 50% τον χρόνο. Ο χρόνος πρόσβασης αυξήθηκε κατα 1/3 σε 10 χρόνια.

Μέτρηση και Χαρακτηρισμός Απόδοσης

- κατα τη σχεδίαση συγκρίνουμε εναλλακτικά σχέδια.
- έτσι συσχετίζουμε την απόδοση δύο μηχανών, π.χ. Χ και Ψ.
- η φράση 'το Χ είναι γρηγορότερο απο το Ψ' σημαίνει οτι το πρώτο απαιτεί μικρότερο χρόνο εκτέλεσης για μία εργασία.
- η φράση 'το Χ είναι ν φορές γρηγορότερο απο το Ψ' σημαίνει:

$$\frac{\text{Χρόνος Εκτέλεσης Ψ}}{\text{Χρόνος Εκτέλεσης Χ}} = \nu$$

επειδή ο χρόνος εκτέλεσης είναι αντίστροφος της απόδοσης, ισχύει το παρακάτω:

$$\nu = \frac{\text{Χρόνος Εκτέλεσης Ψ}}{\text{Χρόνος Εκτέλεσης Χ}} = \frac{\frac{1}{\text{Απόδοση Ψ}}}{\frac{1}{\text{Απόδοση Χ}}} = \frac{\text{Απόδοση Χ}}{\text{Απόδοση Ψ}}$$

Προγράμματα για Χαρακτηρισμό Απόδοσης

1. Πραγματικά Προγράμματα: μεταγλωτιστές, π.χ. *C*, πρόγραμματα επεξεργασίας κειμένου, π.χ. *TEX*, προγράμματα *CAD*, π.χ. *SPICE*.

2. Πυρήνες: μικρά κομμάτια απο πραγματικά προγράμματα, π.χ. ***Livermore Loops*** καί ***Linpack***.

3. Μικρά Μετρητικά Προγράμματα (***Benchmarks***): 10-100 γραμμές κώδικα: π.χ. αλγόριθμος Ερατοσθένη, *Puzzle*, *Quicksort*.

4. Συνθετικά Μετρητικά Προγράμματα: σύνθεση μικρών κομματιών προγραμμάτων για να πλησιάσουν την συχνότητα και συμπεριφορά μεγάλων πραγματικών προγραμμάτων, π.χ. *Whetstone*, *Dhrystone*.

Σύγκριση και Μέτρηση Απόδοσης

- ο πιό απλός τρόπος για να χαρακτηρίσουμε την σχετική απόδοση ειναι ο αριθμητικός μέσος των χρόνων εκτέλεσης n προγραμμάτων:

$$\frac{1}{n}\sum_{i=1}^{n}\text{Χρόνος}(i)$$

- για να δώσουμε συγκεκριμένη σημασία και βαρύτητα στα i απο τα n προγράμματα χρησιμοποιούμε τον βεβαρημένο αριθμητικό μέσο:

$$\frac{1}{n}\sum_{i=1}^{n}\text{Βάρυτητα}(i)\text{Χρόνος}(i)$$

- ενας άλλος τρόπος χαρακτηρισμού απόδοσης σχετικοποιεί τους χρόνους εκτέλεσης με ένα αναφορικό μηχάνημα (οπως π.χ. τα $SPEC\ benchmarks$ που χρησιμοποιούν σαν μέτρο τον $VAX-11/780$) και υπολογίζει το μέσο τους.

$$\sqrt{\prod_{i=1}^{n}\text{Λόγος Χρόνου Εκτέλεσης}(i)}$$

Παράδειγμα Σύγκρισης Απόδοσης

Χρόνοι εκτέλεσης δύο προγραμμάτων σε τρία μηχανήματα:

	Υπολογιστής Α	Υπολογιστής Β	Υπολογιστής Γ
Πρόγραμμα Π1 (***secs***)	1	10	20
Πρόγραμμα Π2 (***secs***)	1000	100	20
Συνολικός Χρόνος (***secs***)	1001	110	40

Σύγκριση με αριθμητικό και βεβαρημένο αριθμητικό μέσο:

	Υπολογιστής Α	Υπολογιστής Β	Υπολογιστής Γ	Βάρος Π1	Βάρος Π2
Αριθμητικός Μέσος	500.50	55.0	20.00	0.5	0.5
Βεβ. Αριθμητικός Μέσος	91.91	18.19	20.00	0.909	0.091
Βεβ. Αριθμητικός Μέσος	2.00	10.09	20.00	0.999	0.001

Παράδειγμα Σύγκρισης Απόδοσης

Σύγκριση σχετικών χρόνων εκτέλεσης:

	Σχετικά με Α			Σχετικά με Β			Σχετικά με Γ		
	Α	Β	Γ	Α	Β	Γ	Α	Β	Γ
Πρόγραμμα Π1	1.0	10.0	20.0	0.1	1.0	2.0	0.05	0.5	1.0
Πρόγραμμα Π2	1.0	0.1	0.02	10.0	1.0	0.2	50.0	5.0	1.0
Αριθμητικός μέσος	1.0	5.05	10.01	5.05	1.0	1.1	25.03	2.75	1.0
Γεωμετρικός μέσος	1.0	1.0	0.63	1.0	1.0	0.63	1.58	1.58	1.0
Συνολικός Χρόνος	1.0	0.11	0.04	9.1	1.0	0.36	25.03	2.75	1.0

- η τιμή του αριθμητικού μέσου διαφέρει ανάλογα με το αναφορικό μηχάνημα.
- στη στήλη 2 ο χρόνος στο Β είναι πέντε φορές περισσότερος αλλά το αντίστροφο συμβαίνει στην στήλη 4.
- οι γεωμετρικοί μέσοι έχουν συνοχή ανεξάρτητα απο το αναφορικό μηχάνημα.
- όμως οι γεωμετρικοί μέσοι δέν εκφράζουν τον χρόνο εκτέλεσης.

Ποσοτικές Αρχές Αρχιτεκτονικής Υπολογιστών

- ‘Κάνε την σύνηθη περίπτωση γρήγορη’.
- πηγάζει απο τον νόμο του *Amdahl*.
- ο νόμος του *Amdahl* προσδιορίζει της επιτάχυνσης που επιτυγχάνει ενα πρόγραμμα χρησιμοποιώντας μια νέα αρχιτεκτονική αναβάθμιση.
- Τοπικότητα αναφορών.
- προγράμματα ξαναχρησιμοποιούν εντολές και δεδομένα που έχουν πρόσφατα χρησιμοποιήσει.

Νόμος *Amdahl*

ξεκινά απο τον ορισμό της επιτάχυνσης ενος προγράμματος (***speedup***):

$$\text{επιτάχυνση} = \frac{\text{απόδοση χρησιμοποιώντας την αναβάθμιση όπου εφαρμόζεται}}{\text{απόδοση χωρίς την αναβάθμιση}}$$

ή

$$\text{επιτάχυνση} = \frac{\text{χρόνος εκτέλεσης χωρίς την αναβάθμιση}}{\text{χρόνος εκτέλεσης χρησιμοποιώντας την αναβάθμιση όπου εφαρμόζεται}}$$

Ο νόμος του *Amdahl* μας βοηθά να υπολογίσουμε την επιτάχυνση χρησιμοποιώντας δύο παράγοντες:

1. το ποσοστό υπολογιστικού χρόνου της αρχικής μηχανής που μπορεί να μετατραπεί για να χρησιμοποιήσει την νέα αναβάθμιση.
2. τη βελτίωση που προσφέρει η αναβάθμιση σε ιδανικές συνθήκες, δηλαδή για όλο το πρόγραμμα.

Νόμος *Amdahl*

Ο νέος χρόνος εκτέλεσης με την αναβάθμιση είναι το άθροισμα των χρόνων των μερών του προγράμματος που μπορούν και δέν μπορούν αντίστοιχα να χρησιμοποιήσουν την αναβάθμιση:

$$\text{Χρόνος Εκτέλεσης}_{\text{νέος}} = \text{Χρόνος Εκτέλεσης}_{\text{παλιός}} \left((1 - \text{Ποσοστό}_{\text{αναβάθμιση}}) + \frac{\text{Ποσοστό}_{\text{αναβάθμιση}}}{\text{Επιτάχυνση}_{\text{αναβάθμιση}}}\right)$$

η επιτάχυνση είναι ο λόγος των δύο χρόνων:

$$\text{επιτάχυνση} = \frac{\text{Χρόνος Εκτέλεσης}_{\text{νέος}}}{\text{Χρόνος Εκτέλεσης}_{\text{παλιός}}} = \frac{1}{(1-\text{Ποσοστό}_{\text{αναβάθμιση}}) + \frac{\text{Ποσοστό}_{\text{αναβάθμιση}}}{\text{Επιτάχυνση}_{\text{αναβάθμιση}}}}$$

- ο νόμος του ***Amdahl*** εκφράζει το νόμος των ελλατούντων εισπράξεων.
- όταν μια αναβάθμιση χρησιμοποιείται για ενα ποσοστό ενός προγράμματος, η επιτάχυνση που πετυχαίνεται δεν μπορεί να είναι μεγαλύτερη απο το αντίστροφο του 1 μείον αυτό το ποσοστό.

Εξίσωσεις απόδοσης CPU

- οι περισότεροι επεξεργαστές υλοποιούνται με ρολόι που δουλεύει με σταθερό ρυθμό (αλλα όχι όλοι).

- το ρολόι χαρακτηρίζεται απο την περίοδο του (κύκλος).

- ο χρόνος εκτέλεσης μπορέι να οριστεί σαν συνάρτηση του κύκλου του ρολογιού:

$$\text{Χρόνος } CPU = \text{Κύκλοι Ρολογιού } CPU \text{ για ένα πρόγραμμα} \; . \; \text{Χρόνος Κύκλου}$$

ή

$$\text{Χρόνος } CPU = \frac{\text{Κύκλοι Ρολογιού } CPU \text{ για ένα πρόγραμμα}}{\text{Ρυθμός Ρολογιού}}$$

Εξίσωσεις απόδοσης CPU

μετρώντας τον αριθμό εντολών που εκτελούνται μπορούμε να υπολογίσουμε τον μέσο αριθμό κύκλων ρολογιού ανα εντολή:

$$\text{Κύκλοι-ανα-εντολή } (CPI) = \frac{\text{Κύκλοι Ρολογιού } CPU \text{ για ένα πρόγραμμα}}{\text{αριθμός εντολών } (IC)}$$

έτσι ο χρόνος CPU μπορεί να γραφεί και όπως παρακάτω:

$$\text{Χρόνος } CPU = IC.CPI.\text{Χρόνος Κύκλου}$$

η οποία εξίσωση μπορέι να αναπτυχθεί για να μας δείξει τα τρία χαρακτηριστικά που επηρεάζουν τον χρόνο του CPU:

$$\text{Χρόνος } CPU = \frac{\text{Εντολές}}{\text{Πρόγραμμα}} \cdot \frac{\text{Κύκλοι}}{\text{Εντολή}} \cdot \frac{\text{Δευτερόλεπτα}}{\text{Κύκλο}} = \frac{\text{Δευτερόλεπτα}}{\text{Πρόγραμμα}}$$

Εξίσωσεις απόδοσης CPU

- επίσεις ο χρόνος του CPU μπορεί να γραφεί σαν συνάρτηση του κάθε τύπου εντολής.
- ο αριθμός κύκλων για ένα πρόγραμμα μπορεί να γραφεί ως εξής:

$$\text{Κύκλοι } CPU = \sum_{i=1}^{n} CPI_i.IC_i$$

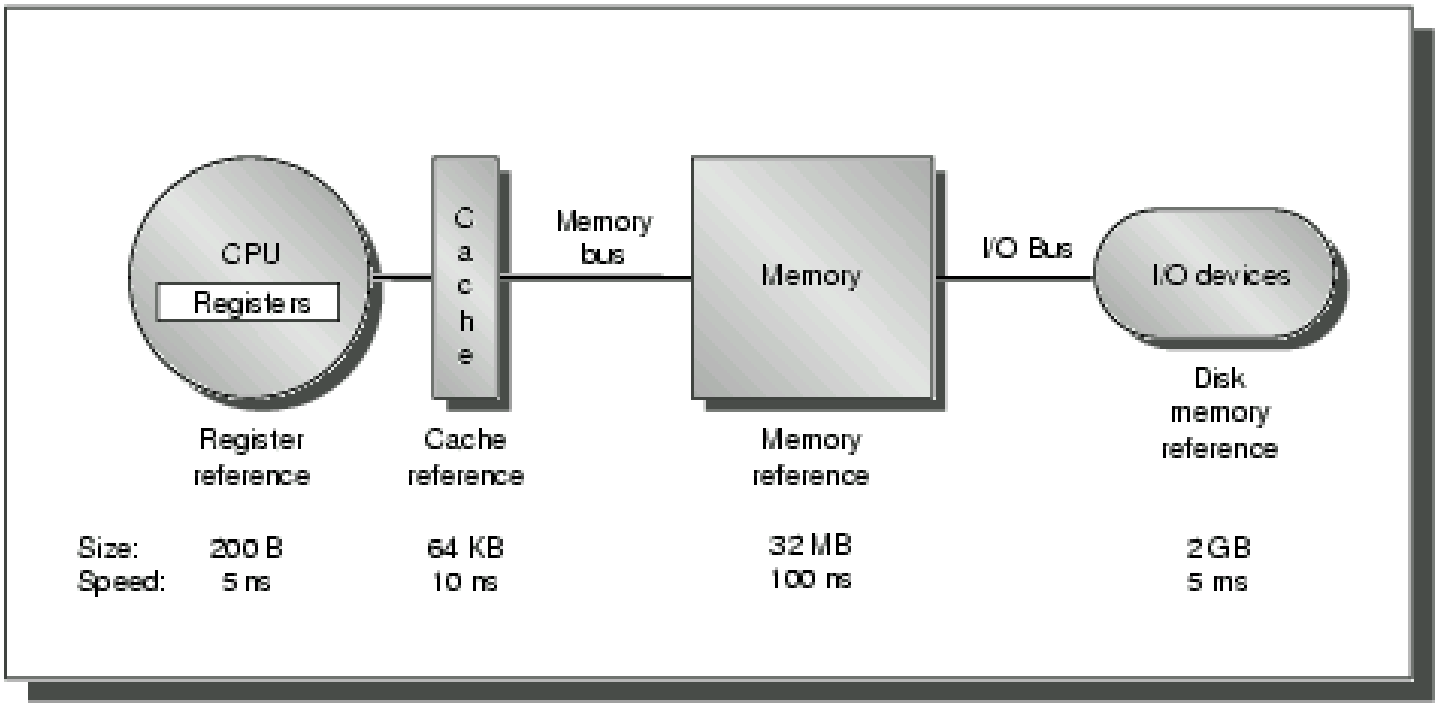
- όπου IC_i είναι ο αριθμός των εντολών τύπου i στο πρόγραμμα και CPI_i είναι ο μέσος αριθμός κύκλων για εντολή τύπου i.
- έτσι ο χρόνος του CPU υπολογίζεται απο τον αριθμό κύκλων επι τον χρόνο του κάθε κύκλου:

$$\text{Χρόνος } CPU = (\sum_{i=1}^{n} CPI_i.IC_i).\text{Χρόνος Κύκλου}$$

- τέλος, το μέσο CPI μπορεί να εκφραστεί με την παραπάνω μορφή της εξίσωσης χρόνου του CPU:

$$CPI = \frac{\sum_{i=1}^{n} CPI_i.IC_i}{\text{Αριθμός Εντολών}}$$

Ιεραρχία Μνήμης



Ιεραρχία Μνήμης, Επίπεδα

Επίπεδο	1	2	3	4
Ονομασία	Καταχωρητές	Κρυφή Μνήμη ($Cache$)	Κεντρική Μνήμη	Δίσκος
Τυπικό Μέγεθος	$< 1\ KB$	$< 4\ MB$	$< 4\ GB$	$> 1\ GB$
Τεχνολογία Υλοποίησης	Εσωτερική μνήμη με πολλαπλές θύρες	$SRAM$ στο ίδιο ή σε άλλο ολοκληρωμένο	$CMOS\ DRAM$	Μαγνητικός Δίσκος
Χρόνος πρόσβασης (σε ns)	2-5	3-10	80-400	5.000.000
Ροή (MB/sec)	4000-32.000	800-5000	400-2000	4-32
Χειρήζεται απο	Μεταγλωτιστή	Υλικό	Λειτουργικό	Λειτουργικό/Χρήστη
Αντίγραφο στο	$Cache$	Κεντρική Μνήμη	Δίσκο	Ταινία