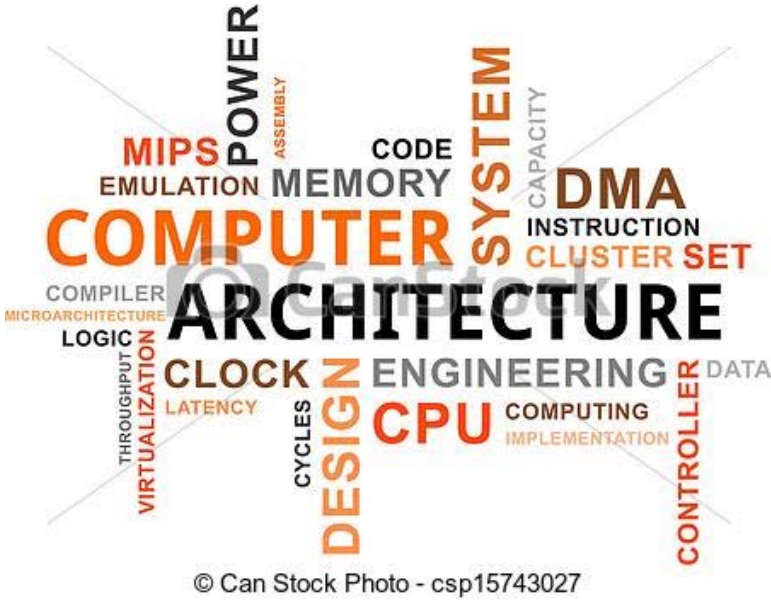
SCHOOL OF INFORMATICS



Αρχιτεκτονική Υπολογιστών

Διάλεξη 2 – Απόδοση Υπολογιστών

Γεώργιος Κεραμίδας, Επίκουρος Καθηγητής

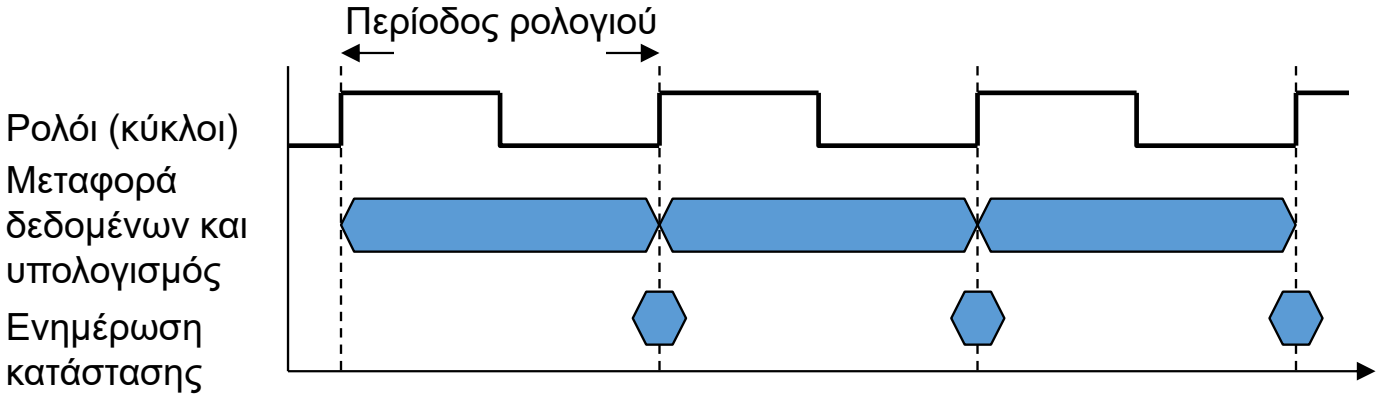
3ο Εξάμηνο, Τμήμα Πληροφορικής

Αντιστοίχιση με ύλη Βιβλίου

- **Το συγκεκριμένο σετ διαφανειών καλύπτει τα εξής κεφάλαια/ενότητες:**
  - **Κεφάλαιο 1: 1.4, 1.5, 1.6, 1.7, 1.8**

Χρονισμός CPU (clocking)

- **Η λειτουργία του ψηφιακού υλικού ρυθμίζεται από ένα ρολόι σταθερού ρυθμού**



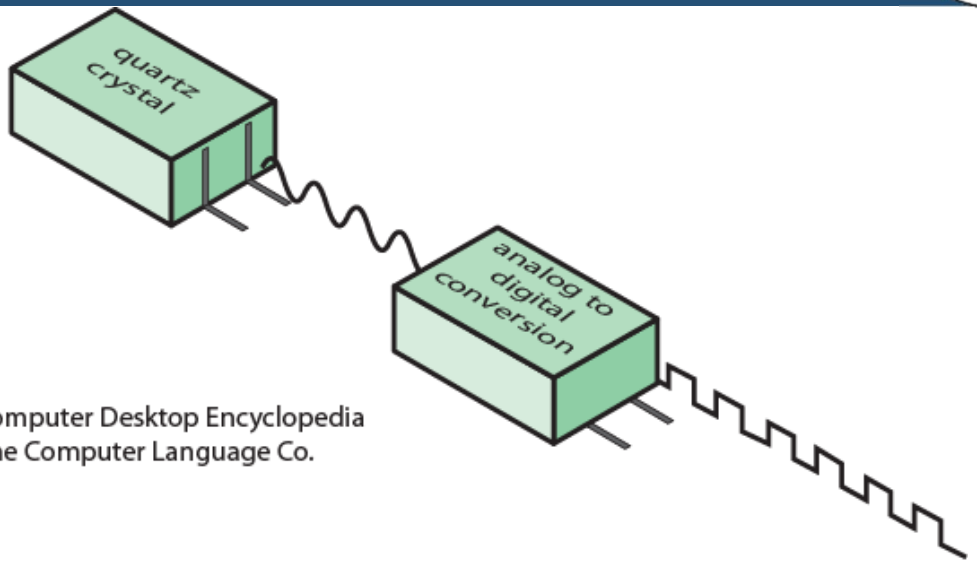
- **Περίοδος ρολογιού: η διάρκεια ενός κύκλου**
  - **π.χ., 250ps = 0.25ns = 250×10^{-12}s**
- **Συχνότητα (ρυθμός) ρολογιού: κύκλοι/second**
  - **π.χ., 4.0GHz = 4000MHz = 4.0×10^{9}Hz**

Απόδοση Η/Υ: Ρολόι συστήματος

- Ταχύτητα ρολογιού CPU: GHz
- Το κάθε γεγονός (π.χ. εντολή) εκτελείται στον χτύπο του ρολογιού

$$Cycle\ time\ \tau = \frac{1}{f}$$

- Συχνότητα ρολογιού 1GHz, κύκλος ρολογιού 1nsec



From Computer Desktop Encyclopedia
1998, The Computer Language Co.

- Όμως...διπλασιάζοντας τη συχνότητα ρολογιού, συμβαίνουν όλα με διπλάσια ταχύτητα ?

Τι συμβαίνει σε έναν κύκλο ρολογιού ?

- RISC (Reduced Instruction Set Computers)
  - Εκτελεί μία εντολή
- CISC (Complex Instruction Set Computers)
  - Απαιτεί πολλούς κύκλους για την εκτέλεση μίας εντολής
- Superscalar
  - Εκτελεί πολλαπλές εντολές ανά κύκλο
- **Θυμηθείτε: Αναφερόμαστε στον κύκλο ρολογιού του επεξεργαστή. Η μνήμη και τα περιφερειακά (π.χ. δίσκος) έχουν το "δικό τους" (διαφορετικό) ρολόι**

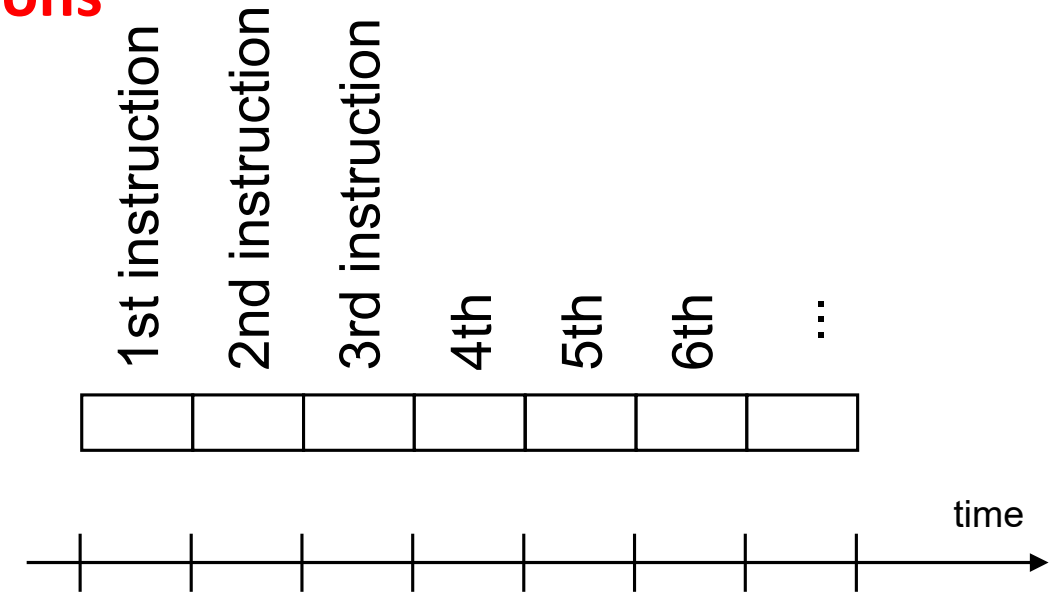
Χρόνος CPU (CPU time)

$$\text{Χρόνος CPU} = \text{Κύκλοι ρολογιού CPU} \times \text{Χρόνος κύκλου ρολογιού}$$
$$= \frac{\text{Κύκλοι ρολογιού CPU}}{\text{Συχνότητα ρολογιού}}$$

- **Η απόδοση βελτιώνεται με**
  - **Μείωση του αριθμού των κύκλων ρολογιού**
  - **Αύξηση του ρυθμού του ρολογιού**
  - **Ο σχεδιαστής του υλικού πρέπει να κάνει συχνά συμβιβασμούς μεταξύ του ρυθμού ρολογιού και του πλήθους των κύκλων ρολογιού. Ακολουθεί παράδειγμα…**

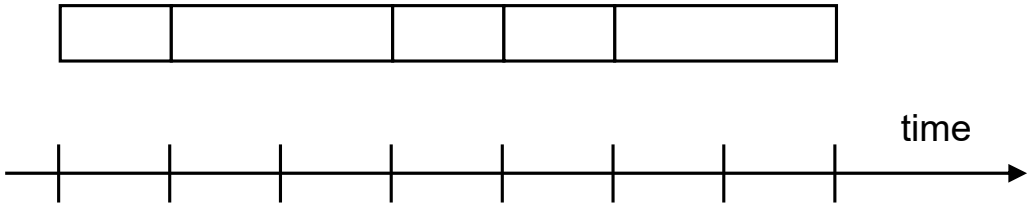
Πόσοι κύκλοι χρειάζονται για ένα πρόγραμμα

- **Έστω ότι # of cycles = # of instructions**



- **Η υπόθεση είναι λάθος:**
  - **Διαφορετικές εντολές παίρνουν διαφορετικό αριθμό από κύκλους**
  - **Γιατί ?**

Πόσοι κύκλοι χρειάζονται για ένα πρόγραμμα



- **Οι πολλαπλασιασμοί παίρνουν περισσότερους κύκλους από τις διαιρέσεις**
- **Οι πράξεις κινητής υποδιαστολής παίρνουν περισσότερο χρόνο από τις πράξεις ακεραίων**
- **Η πρόσβαση στην μνήμη παίρνει περισσότερο χρόνο από τις προσβάσεις στους καταχωρητές**
- **Προσοχή: αλλάζοντας την συχνότητα αλλάζουμε συνήθως και τους κύκλους που απαιτούνται για κάθε εντολή (γιατί αλλάζουμε το υλικό που απαιτείται για την εκτέλεση της εντολής). Ακολουθεί παράδειγμα:**

Παράδειγμα χρόνου CPU

- Υπολογιστής Α: ρολόι 2GHz, χρόνος CPU 10s
- Σχεδίαση του υπολογιστή Β
  - Στόχος είναι χρόνος CPU 6s
  - Μπορεί το ρολόι να είναι ταχύτερο, αλλά προκαλεί αύξηση των κύκλων 1.2 × κύκλοι ρολογιού_Α
- Πόσο ταχύτερο μπορεί να είναι το ρολόι του Β;

$$\text{Ρυθμός ρολογιού}_B = \frac{\text{Κύκλοι ρολογιού}_B}{\text{Χρόνος CPU}_B} = \frac{1.2 \times \text{Κύκλοι ρολογιού}_A}{6\text{s}}$$

$$\text{Κύκλοι ρολογιού}_A = \text{Χρόνος CPU}_A \times \text{Ρυθμός ρολογιού}_A$$

$$= 10\text{s} \times 2\text{GHz} = 20 \times 10^9$$

$$\text{Ρυθμός ρολογιού}_B = \frac{1.2 \times 20 \times 10^9}{6\text{s}} = \frac{24 \times 10^9}{6\text{s}} = 4\text{GHz}$$

Απόδοση Υπολογιστών

- **Μια ρεαλιστική προσέγγιση είναι να ορίσουμε την απόδοση (performance) του υπολογιστή X σαν το αντίστροφο του χρόνου εκτέλεσης**
- **[Απόδοση]X = 1 / [Χρόνος εκτέλεσης]X**
- **Η απόδοση του υπολογιστή X είναι n φορές μεγαλύτερη σε σχέση με τον Y, όταν:**

$$\frac{\text{[Απόδοση]X}}{\text{[Απόδοση]Y}} = \frac{\text{[Χρόνος εκτέλεσης]Y}}{\text{[Χρόνος εκτέλεσης]X}} = n$$

Σχετική απόδοση

- **“ο Χ είναι η φορές ταχύτερος από τον Υ”**

- **Απόδοση_Χ/Απόδοση_Υ =Χρονος εκτέλεσης_Υ/Χρόνος εκτέλεσης_Χ**
- **παράδειγμα:**
  - **χρόνος εκτέλεσης προγρ/τος: 10s στον Α, 15s στον Β**
  - **ΧρόνοςεκτέλεσηςΒ / ΧρόνοςεκτέλεσηςΑ = 15s / 10s = 1.5**
  - **Άρα ο Α είναι 1.5 φορές ταχύτερος του Β**

Πως κρίνεις τα αποτελέσματα ?

System	Rate (Task 1)	Rate (Task 2)
A	10	20
B	20	10

Which system is faster?

- **Προσοχή: Το παράδειγμα είναι χαρακτηριστικό και ρεαλιστικό. Συγκεκριμένοι επεξεργαστές είναι βελτιστοποιημένοι για συγκεκριμένες κατηγορίες εφαρμογών**

Πως κρίνεις τα αποτελέσματα ?

System	Rate (Task 1)	Rate (Task 2)	Average
A	10	20	15
B	20	10	15

Average throughput

System	Rate (Task 1)	Rate (Task 2)	Average
A	0.50	2.00	1.25
B	1.00	1.00	1.00

Throughput relative to B

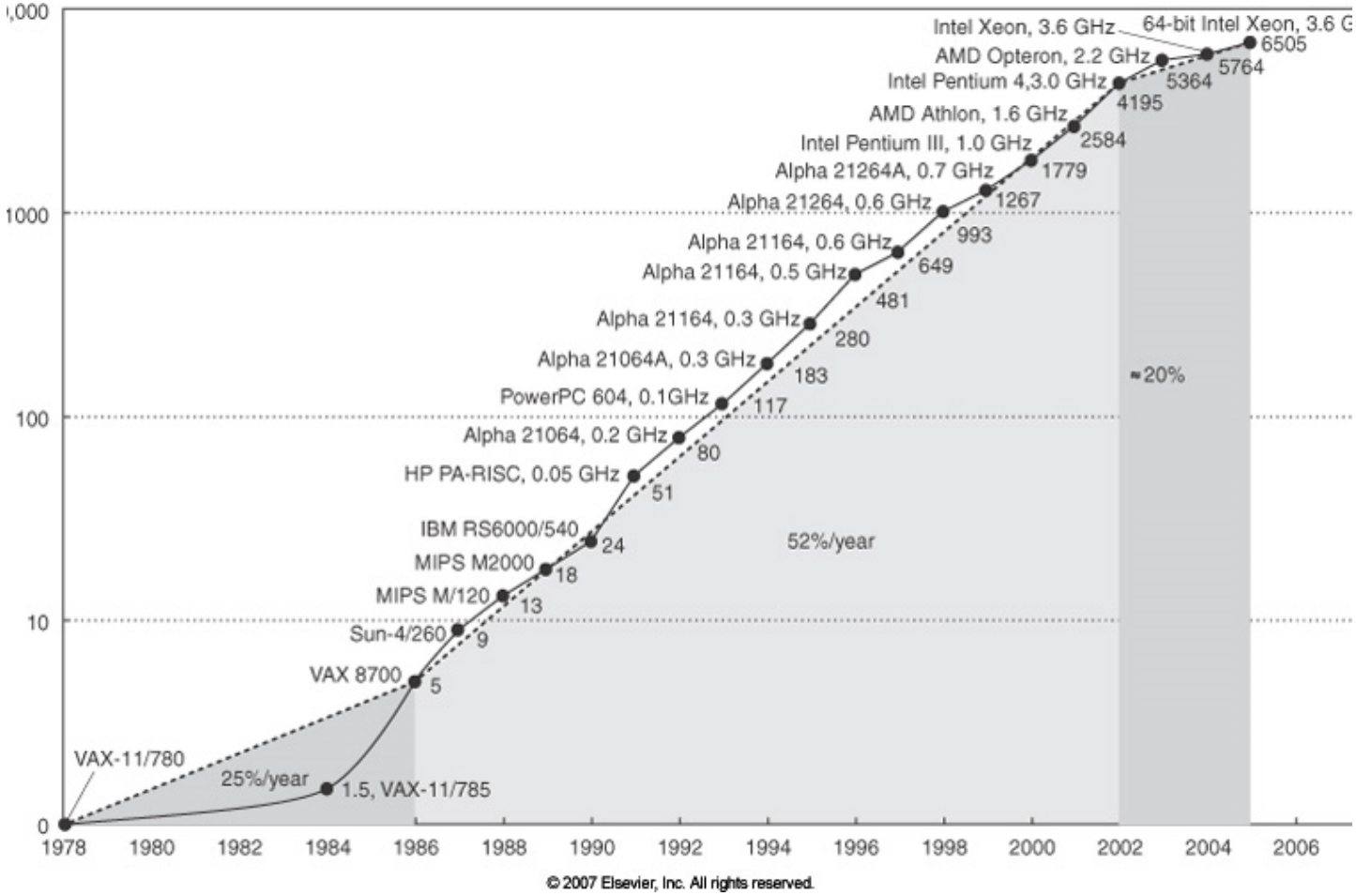
System	Rate (Task 1)	Rate (Task 2)	Average
A	1.00	1.00	1.00
B	2.00	0.50	1.25

Throughput relative to A

Πως κρίνεις τα αποτελέσματα ?

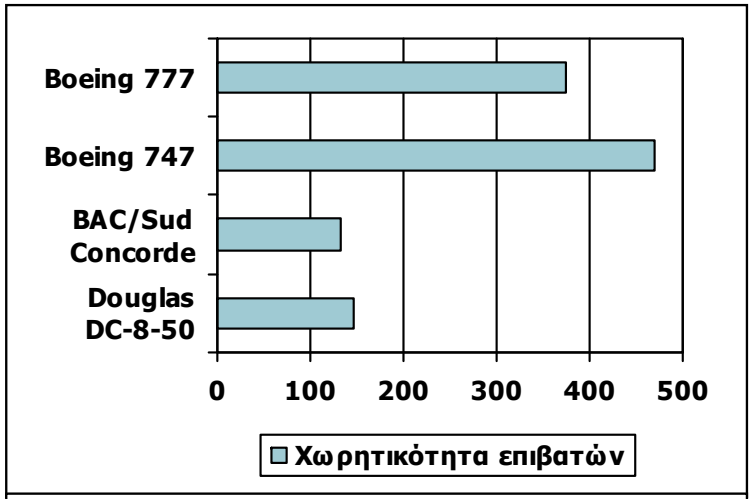
Program A (ή μισθός A)	200
Program B (ή μισθός B)	300
Program C (ή μισθός C)	1800
μέσος όρος	766,6667
γεωμετρικός μέσος	476,2203
αρμονικός μέσος	337,5

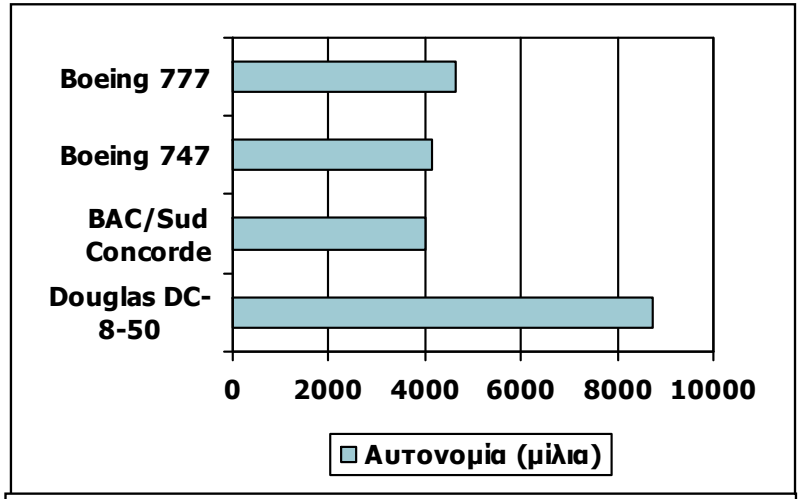
Processor Performance

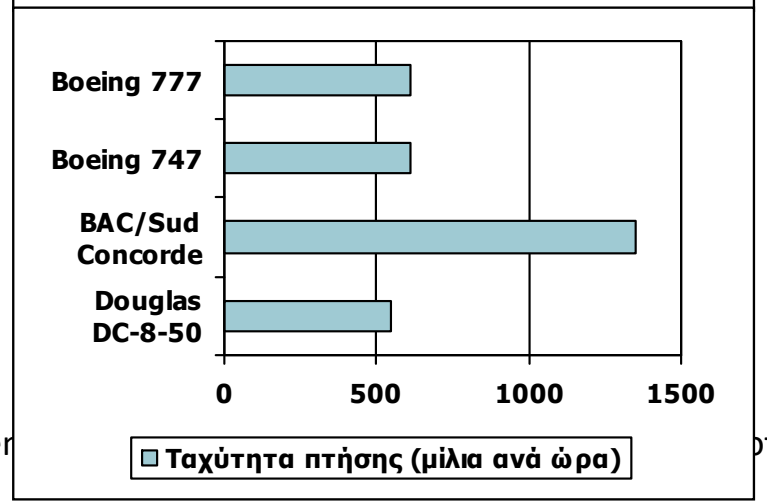


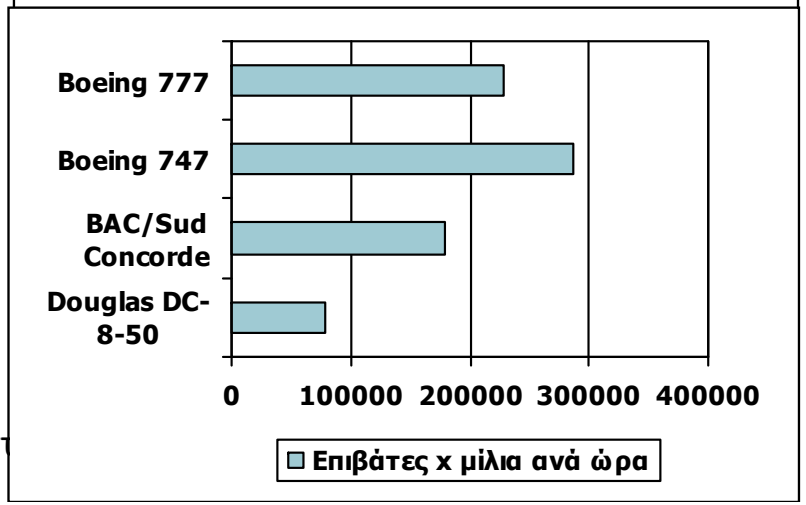
Ορισμός της απόδοσης

- **Ποιο αεροπλάνο έχει την καλύτερη απόδοση;**









Αξιολόγηση απόδοσης - ταχύτητα ρολογιού

- **Απόδοση Η/Υ: ένας υπολογιστής είναι αποδοτικός. ...σε τι ???**
- **Βασικές παράμετροι**
  - **απόδοση, κόστος, μέγεθος, ασφάλεια, αξιοπιστία, κατανάλωση ισχύος**

- **Άρα, η ταχύτητα ρολογιού δεν είναι το παν…**
  - **Τα σήματα στη CPU απαιτούν χρόνο για να καταλήξουν σε 1 ή 0**
  - **Εκτέλεση εντολής σε διακριτά βήματα**
    - **ανάκτηση, αποκωδικοποίηση, φόρτωση και αποθήκευση, αριθμητική ή λογική**
    - **Συνήθως απαιτούνται πολλαπλοί κύκλοι ρολογιού ανά εντολή**

Response time – Throughput

- **Χρόνος απόκρισης (response time) – αναφέρεται και ως latency (καθυστέρηση)**
  - **Πόσο διαρκεί η εκτέλεση μιας εργασίας**
- **Διεκπεραιωτική ικανότητα (throughput)**
  - **Συνολική δουλειά που γίνεται ανά μονάδα χρόνου**
    - **π.χ. εργασίες/συναλλαγές/… ανά ώρα**

Response time – Throughput

- **Ας σκεφτούμε το ακόλουθο σενάριο:**
  - **Σύστημα Α: ολοκληρώνει το task A σε 22 minutes**
  - **Σύστημα Β: ολοκληρώνει το task A σε 40 minutes**

  - **Σύστημα Α: ολοκληρώνει το task B σε 30 minutes**
  - **Σύστημα Β: ολοκληρώνει το task B σε 30 minutes**

  - **Σύστημα Α: ολοκληρώνει το task A + task B 60 minutes**
  - **Σύστημα Β: ολοκληρώνει το task A + task B σε 45 minutes**

Βελτιστοποιημένος ως προς το response time

Βελτιστοποιημένος ως προς το throughput

Response time – Throughput

Κάθε σύστημα είτε υλικού, είτε λογισμικού είναι βελτιστοποιημένο για μία συγκεκριμένη μετρική ή για ένα συγκεκριμένο τύπο εφαρμογών

"One-size-fits-all" → Δεν υπάρχει (ακόμα)

Πάμε να δούμε τι χρησιμοποιείται από τις εταιρίες κατασκευής επεξεργαστών

Ρυθμός εκτέλεσης εντολής

- **MIPS:** Millions of instructions per second (Αριθμός εντολών το δευτερόλεπτο)
- **MFLOPS:** Millions of floating point operations per second (Αριθμός λειτουργιών κινητής υποδιαστολής το δευτερόλεπτο)
- Ισχυρή εξάρτηση από:
  - το σύνολο εντολών
  - το σχεδιασμό του compiler
  - την υλοποίηση του επεξεργαστή
  - την ιεραρχία cache και κύριας μνήμης

MIPS

- **Μετράει ρυθμό δουλειάς που εκτελείται ανά δευτερόλεπτο (Million Instructions per Second)**

$$\text{MIPS} = \frac{\text{Instruction Count}}{\text{Execution Time}} 10^{-6}$$

- **Όμως, δεν είναι όλες οι εντολές ίδιες...**
- **...εξαρτάται από το σετ εντολών που υποστηρίζει ο κάθε επεξεργαστής**

MIPS

- **Υψηλού επιπέδου γλώσσα**
  - **Γραμμές κώδικα / εντολών**
- **Μεταγλώττιση σε γλώσσα μηχανής**
  - **Μεταφράζεται σε αριθμό εντολών (εκατομμύρια ?)**
  - **Διαφορετικοί τύποι εντολών**
- **π.χ.**

A=B+C σε γλώσσα μηχανής

```
LOAD R1 B
LOAD R2 C
ADD R2 R2 R1
STORE R1 A
```

Το MIPS μπορεί να είναι παραπλανητικό...

- Εξαρτάται από το σετ εντολών του επεξεργαστή
- Το ίδιο πρόγραμμα μπορεί να οδηγεί σε διαφορετικά σετ εντολών ανάλογα με τον μεταγλωττιστή (compiler)
- Εξαρτώμενο από το πρόγραμμα (μίγμα εντολών)

- ...όμως εξακολουθεί να ισχύει ως μετρικός δείκτης απόδοσης για οικογένειες επεξεργαστών με το ίδιο σετ εντολών

Η εξίσωση της CPU

- **Συνολικός χρόνος εκτέλεσης:**

$$instruction\ count \times \underbrace{\frac{number\ of\ clock\ cycles}{instruction\ count}}_{CPI} \times clock\ cycle$$

- **Αριθμός εντολών**
  - **Δυναμικός, περιλαμβάνει βρόχους (loops) και κλήσεις συναρτήσεων (procedure calls)**
- **CPI: Clocks per Instruction – εξαρτάται από την εντολή. Μερικές με CPI>1, μερικές με <1. Πως γίνεται αυτό?**

Clocks per Instruction: CPI

- **Η μέση τιμή CPI εμφανίζεται στην εξίσωση της CPU:**

$$CPI = \frac{CPU\ clock\ cycles\ for\ program}{Instruction\ count}$$

- **Εξαρτάται από την αρχιτεκτονική και το πρόγραμμα**
- **Ποιά εντολή?**
- **Περιεχόμενο, ποιες εντολές είναι πριν / μετά?**
- **Σχετίζεται με το MIPS:**

$$MIPS = \frac{CPU\ frequency}{CPI * 10^6}$$

Παράδειγμα CPI

- **Υπολογιστής Α: Cycle Time = 250ps, CPI = 2.0**
- **Υπολογιστής Β: Cycle Time = 500ps, CPI = 1.2**
- **Ίδια αρχιτεκτονική συνόλου εντολών (ISA)**
- **Ποιος είναι ταχύτερος, και κατά πόσο;**

$$\text{Χρόνος CPU}_A = \text{Πλήθος εντολών} \times \text{CPI}_A \times \text{Χρόνος κύκλου}_A$$
$$= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps}$$

Α ταχύτερος…

$$\text{Χρόνος CPU}_B = \text{Πλήθος εντολών} \times \text{CPI}_B \times \text{Χρόνος κύκλου}_B$$
$$= I \times 1.2 \times 500\text{ps} = I \times 600\text{ps}$$

$$\frac{\text{Χρόνος CPU}_B}{\text{Χρόνος CPU}_A} = \frac{I \times 600\text{ps}}{I \times 500\text{ps}} = 1.2$$

…κατά τόσο

Το CPI με λεπτομέρεια

- **Αν διαφορετικές κατηγορίες εντολών διαρκούν διαφορετικό αριθμό κύκλων**

$$\text{Κύκλοι ρολογιού} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{Πλήθος εντολών}_i)$$

  - Σταθμισμένο (weighted) μέσο CPI

$$\text{CPI} = \frac{\text{Κύκλοι ρολογιού}}{\text{Πλήθος εντολών}} = \sum_{i=1}^{n} \left(\text{CPI}_i \times \underbrace{\frac{\text{Πλήθος εντολών}_i}{\text{Πλήθος εντολών}}}_{\text{Σχετική συχνότητα εντολών}} \right)$$

Παράδειγμα CPI

- **Εναλλακτικές ακολουθίες μεταγλωττισμένου κώδικα με εντολές τριών κατηγοριών A, B, C**

Κατηγορία	A	B	C
CPI της κατηγορίας	1	2	3
Πλήθος εντολών (IC) ακολουθίας 1	2	1	2
Πλήθος εντολών (IC) ακολουθίας 2	4	1	1

- **Ακολουθία 1: IC = 5**
  - **Κύκλοι ρολογιού = 2×1 + 1×2 + 2×3 = 10 Μέσο CPI = 10/5 = 2.0**
- **Ακολουθία 2: IC = 6**
  - **Κύκλοι ρολογιού = 4×1 + 1×2 + 1×3 = 9 Μέσο CPI = 9/6 = 1.5**

Είναι τα FLOPS καλύτερα?

- **FLOPS=Floating Point Operations Per Second**
  - **...με στόχο να είναι πιο δίκαια από το MIPS (λειτουργίες αντί εντολές)**
  - **…οπότε μετρά ουσιαστική εργασία χωρίς να το απασχολεί το πώς ένας επεξεργαστής εκτελεί την εργασία**

- **Πάλι όμως εξαρτάται από το πρόγραμμα**
  - **Π.χ. Ο μεταγλωττιστής δεν έχει fp ops**

MFLOPS

Million Floating Point Operations Per Second

$$\text{MFLOPS} = \frac{\text{Number of fp ops}}{\text{Execution Time}} \, 10^{-6}$$

- **π.χ., μέτρα τον αριθμό των fp ops που χρειάζονται για τον πολλαπλασιασμό 2 πινάκων**
- **...και μη λαμβάνεις υπόψη τον ακριβή τρόπο υλοποίησης**

MFLOPS: παράδειγμα

```
double sum, a[];
for(i=0;i<30000;i++) {
  sum += a[i]*21; }
```

- **εκτελείται σε χρόνο 1,5msec. Η απόδοση είναι:**
  - A) 20 MFLOPS
  - B) 40 MFLOPS
  - C) 63 MFLOPS
  - D) 80 MFLOPS

Σύνολο fp ops: 2 fp ops (πολλ/σμός, πρόσθεση) σε Κάθε τιμή I επί 30000 τιμές i`

Εντολές για απλή πρόσθεση

- **πρόσθεση floating point μεταβλητών**

```
a = b + c
```

- **Μετά τη μεταγλώττιση σε γλώσσα μηχανής (Assembly):**

```
Load b to reg1
Load c to reg2
Fpadd reg1+reg2 in reg1
Store reg1 to a
```

- **Τελικά αριθμός εντολών ??**

2 Load
1 Store
1 fp op

Οι εντολές αυτές στο χρόνο...

- **υπολογισμός του πόσοι κύκλοι απαιτούνται για κάθε εντολή με βάση τις προδιαγραφές του Hardware**
- **Παραδειγμα (μη ρεαλιστικό):**

Load	1 cycle
Store	1 cycle
fp op	1 cycle

- **Η σειρά εκτέλεσης...**

L b	L c	FP	S a						

- **Το θεωρητικό μέγιστο θα ήταν για ένα fp κάθε κύκλο**
- **...οπότε σε MFLOPS, αυτός ο κώδικας εκτελείται στο ¼ του Peak.**

Μέτρηση εντολών σε πραγματικό κώδικα...

- **...πολύ δύσκολο να υλοποιηθεί**
- **Διαφορετικές εντολές απαιτούν διαφορετικό αριθμό κύκλων ρολογιού**
  - **Floating point απαιτεί πολλούς (ιδιαίτερα η διαίρεση)**
  - **Χρόνος φόρτωσης εξαρτάται από το αν τα δεδομένα είναι στην cache**
- **Διαφορετικού τύπου Διασωλήνωση "pipelining"**
  - **αν π.χ. υπάρχει μόνο ένα μονοπάτι προς τη μνήμη (δε μπορεί να εκτελέσει "load"-"store" ταυτόχρονα), αλλά μπορεί να εκτελέσει fp και μνήμη ταυτόχρονα (superscalar)**

L1x	L1y	L2x	L2y	S1	L3x	L3y	S2	L4x	L4y
		FP1			FP2			FP3	

12 [$Upper\ limit = (1/3) * (peak\ performance)$

MFLOPS όχι πάντα ίδια...

- **Διαφορετικές fp operations σε διαφορετικά μηχανήματα**

Cray: + , − , *

Pentium: + , − , * , / , sin , cos

- **..τα + και / δεν πρέπει να έχουν το ίδιο βάρος**
- **Σύστημα απόδοσης βαρών (Livermore loops), όχι το πραγματικό βάρος σε κάθε επεξεργαστή**

FP Operation	Weight
Add, Subtract, Multiply	1
Divide, Square root	4
Exp, Sin etc	8

Σύνοψη της απόδοσης

$$\text{Χρόνος CPU} = \frac{\text{Εντολές}}{\text{Πρόγραμμα}} \times \frac{\text{Κύκλοι ρολογιού}}{\text{Εντολή}} \times \frac{\text{Δευτερόλεπτα}}{\text{Κύκλος ρολογιού}}$$

- **Η απόδοση εξαρτάται από**
  - **Αλγόριθμο: επηρεάζει το πλήθος εντολών, πιθανόν και το CPI**
  - **Γλώσσα προγραμματισμού: επηρεάζει το πλήθος εντολών και το CPI**
  - **Μεταγλωττιστής: επηρεάζει το πλήθος εντολών και το CPI**
  - **Αρχιτεκτονική συνόλου εντολών (ISA): επηρεάζει το πλήθος εντολών, το CPI, και την περίοδο (συχνότητα) του ρολογιού**

Δείκτες απόδοσης: Σύνοψη

- **Χρόνος: το απόλυτο κριτήριο, αλλά δε χρησιμοποιείται αυτούσιος**
- **MIPS: όχι πλέον κατάλληλο, αλλά το CPI συχνά χρησιμοποιείται για ποσοτικοποίηση αρχιτεκτονικής**
- **MFLOPS: γνωστά μειονεκτήματα, όμως ευρύτατα χρησιμοποιούμενο**
- **δεν απαλλασσόμαστε από την εξάρτηση από το πρόγραμμα**

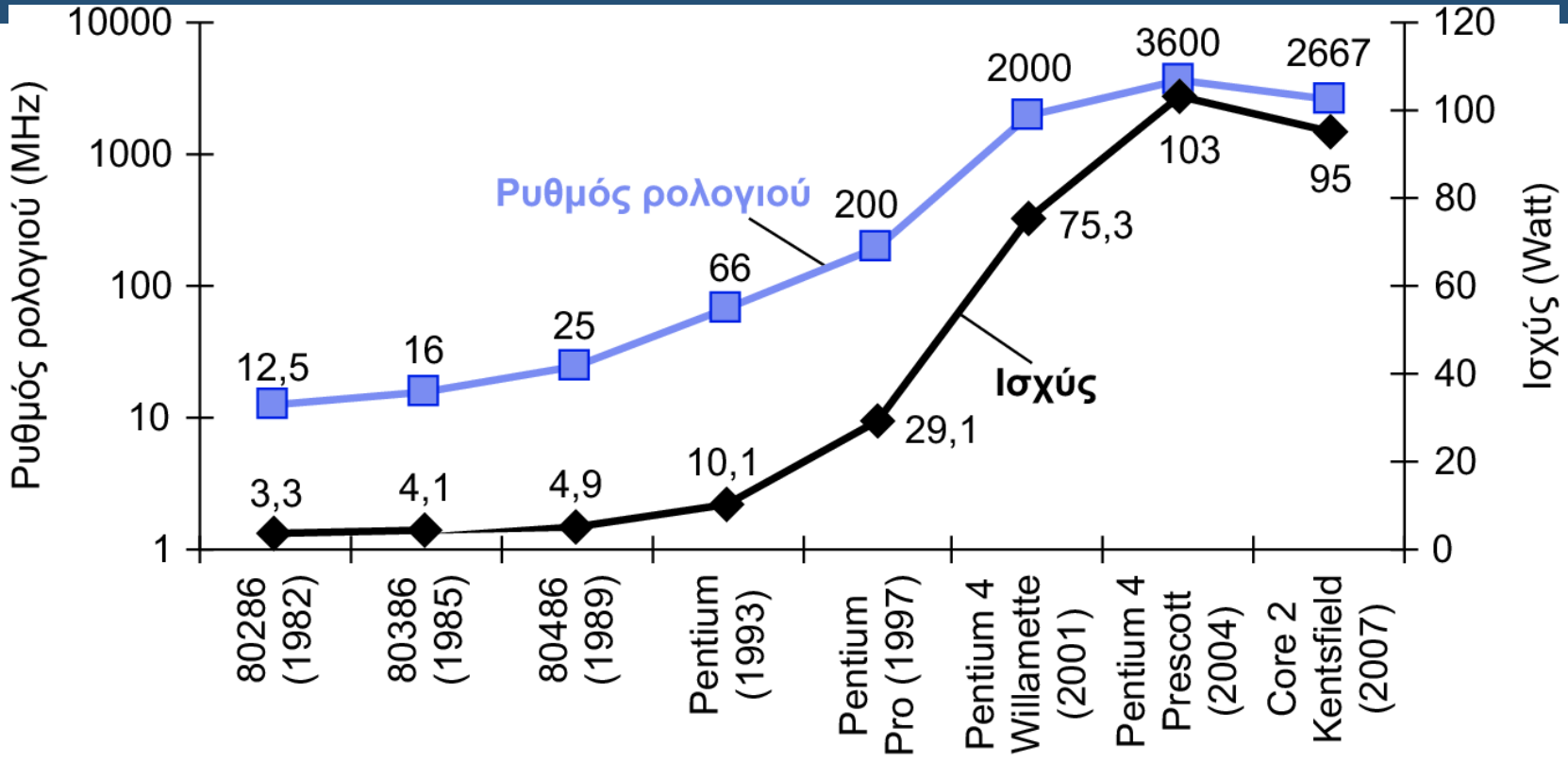
Μετρώντας την απόδοση

- **Μέσω αξιολόγησης του Υλικού (Hardware): Μέγιστη Απόδοση**
  - **Ο μέγιστος αριθμός λειτουργιών που μπορεί να εκτελέσει ο επεξεργαστής ανά δευτερόλεπτο**
  - **..εξαρτάται μόνο από τα τεχνικά χαρακτηριστικά του Υλικού**
  - **Στενά συνδεδεμένο με τον κύκλο ρολογιού**
  - **Δεν εξαρτάται από το πρόγραμμα...**
  - **...αλλά μπορεί να μην είναι εφικτό για κάθε πρόγραμμα**
- **...είναι η απόδοση που το μηχάνημα «εγγυάται ότι δε θα ξεπεράσει ποτέ».**

Μέγιστη Απόδοση (Peak Performance)

- **Εξαρτάται από χαρακτηριστικά CPU, όχι από τον κώδικα**
  - **(αριθμός fp εντολών που εκτελούνται ανά κύκλο)*(ρυθμό ρολογιού)**
  - **Π.χ. Αν ένας επεξεργαστής έχει 2 fp/cycle, 500MHz clock-rate: τότε peak= 1 GFLOP**
- **...αλλά στην πράξη ποτέ δε θα πιάσει αυτή την τιμή.**
  - **λόγω ακαταλληλότητας κώδικα**
  - **λόγω καθυστερήσεων στην πρόσβαση στη μνήμη**
- **Για συγκεκριμένο πολύ απλό κώδικα, ίσως μερικές φορές αγγίξει το άνω όριο της απόδοσης (ποσοστό της θεωρητικά μέγιστης)**

Οι τάσεις στην ηλεκτρική ισχύ



- **Στη τεχνολογία ολοκληρωμένων κυκλωμάτων CMOS**

$$\text{Ισχύς} = \text{Φορτίο χωρητικότητας} \times \text{Τάση}^2 \times \text{Συχνότητα}$$

Μείωση της ισχύος

- **Υποθέστε ότι μια νέα CPU έχει**
  - **85% του φορτίου χωρητικότητας (capacitive load) της παλιάς CPU**
  - **μειωμένη τάση κατά 15% και συχνότητα κατά 15%**

$$\frac{P_{new}}{P_{old}} = \frac{C_{old} \times 0.85 \times (V_{old} \times 0.85)^2 \times F_{old} \times 0.85}{C_{old} \times V_{old}^{\ 2} \times F_{old}} = 0.85^4 = 0.52$$

- **Το τείχος της ισχύος (power wall)**
  - **Δεν μπορούμε να μειώσουμε άλλο την τάση**
  - **Δεν μπορούμε να απαγάγουμε τη θερμότητα**
- **Πώς αλλιώς μπορούμε να βελτιώσουμε την απόδοση;**

Ο ρόλος του Σχεδιαστή Υπολογιστών

- **Καθορίζει ποια χαρακτηριστικά είναι σημαντικά για ένα νέο μηχάνημα. Στη συνέχεια σχεδιάζει ένα μηχάνημα που να μεγιστοποιεί την επίδοση και παράλληλα να μην υπερβαίνει τους περιορισμούς κόστους**
- **Επιμέρους χαρακτηριστικά**
  - **Σχεδιασμός του instruction set**
  - **Οργάνωση των λειτουργιών**
  - **Λογικός σχεδιασμός και υλοποίηση (IC design, packaging, power, cooling …)**

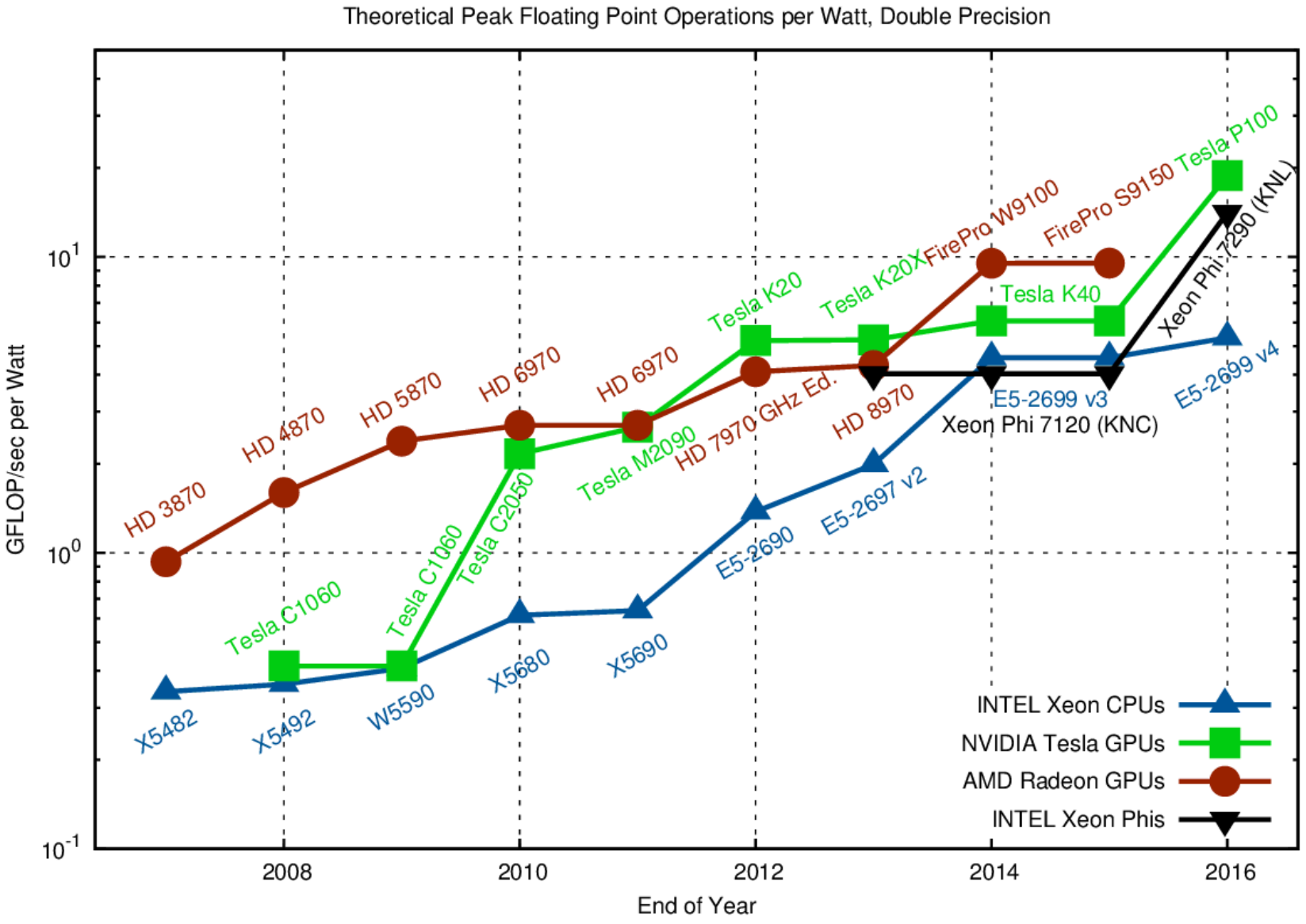
Supercomputing TOP 500 / Nov 2007

Top 500 June 2018

Name	Development	Hardware	Cores	Performance TFLOPS	Power (KW)
1	DOE/SC/Oak Ridge National Laboratory United States	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband IBM	2,282,544	122,300.0	8,806
2	National Supercomputing Center in Wuxi China	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93,014.6	15,371
3	DOE/NNSA/LLNL United States	**Sierra** - IBM Power System S922LC, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband IBM	1,572,480	71,610.0	
4	National Super Computer Center in Guangzhou China	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000 NUDT	4,981,760	61,444.5	18,482
5	National Institute of Advanced Industrial Science and Technology (AIST) Japan	**AI Bridging Cloud Infrastructure (ABCI)** - PRIMERGY CX2550 M4, Xeon Gold 6148 20C 2.4GHz, NVIDIA Tesla V100 SXM2, Infiniband EDR Fujitsu	391,680	19,880.0	1,649

Top Green 500 June 2018

Name	Rank in top 500	Hardware	Cores	Performance TFLOPS	Power (kW)	Power Efficiency (GFlops/watts)
1	359	**Shoubu system B** - ZettaScaler-2.2, Xeon D-1571 16C 1.3GHz, Infiniband EDR, PEZY-SC2 , PEZY Computing / Exascaler Inc. Advanced Center for Computing and Communication, RIKEN Japan	794,400	857.6	47	18.404
2	419	**Suiren2** - ZettaScaler-2.2, Xeon D-1571 16C 1.3GHz, Infiniband EDR, PEZY-SC2 , PEZY Computing / Exascaler Inc. High Energy Accelerator Research Organization /KEK Japan	762,624	798.0	47	16.835
3	385	**Sakura** - ZettaScaler-2.2, Xeon E5-2618Lv3 8C 2.3GHz, Infiniband EDR, PEZY-SC2 , PEZY Computing / Exascaler Inc. PEZY Computing K.K. Japan	794,400	824.7	50	16.657
4	227	**DGX SaturnV Volta** - NVIDIA DGX-1 Volta36, Xeon E5-2698v4 20C 2.2GHz, Infiniband EDR, NVIDIA Tesla V100 , Nvidia NVIDIA Corporation United States	22,440	1,070.0	97	15.113
5	1	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM DOE/SC/Oak Ridge National Laboratory United States	2,282,544	122,300.0	8,806	13.889


Theoretical Peak Floating Point Operations per Watt, Double Precision
GFLOP/sec per Watt
10^1
10^0
10^{-1}
2008
2010
2012
2014
2016
End of Year
HD 3870
HD 4870
HD 5870
HD 6970
HD 6970
HD 7970 GHz Ed.
HD 8970
FirePro W9100
FirePro S9150
Tesla K20
Tesla K20X
Tesla K40
Tesla P100
Xeon Phi 7290 (KNL)
Tesla C1060
Tesla C1060
Tesla C2050
Tesla M2090
E5-2699 v3
Xeon Phi 7120 (KNC)
E5-2699 v4
E5-2697 v2
E5-2690
X5482
X5492
W5590
X5680
X5690
INTEL Xeon CPUs
NVIDIA Tesla GPUs
AMD Radeon GPUs
INTEL Xeon Phis

Μετρώντας την απόδοση…

- **…σημεία αναφοράς επιδόσεων (Benchmarks)**

- **Είναι προγράμματα που αποσκοπούν στη δοκιμή ή σύγκριση μηχανών**
- **Μετρούν πόσο χρόνο εκτελείται**
- **Συχνά μετρούν-αναφέρονται σε MFLOPS**
- **…κάθε πρόγραμμα έχει ένα χαρακτηριστικό μίγμα εντολών: υπάρχει κατ'αντιστοιχία ένα τυπικό μίγμα – τυπικό Benchmark?**

Μετρώντας την απόδοση

- **...επιλογή Benchmark ?**

- **Δεν υπάρχει ένα τυπικός φόρτος εργασίας για κάθε υπολογιστικό σύστημα**
  - **...άρα δεν υπάρχει κοινό κατάλληλο Benchmark, ή κατάλληλο μίγμα από Benchmarks**
  - **…και αντίστοιχα, δεν υπάρχει τυπικό κατάλληλο μίγμα εντολών προς βελτιστοποίηση από σχεδιαστές CPU**
- **Αποτέλεσμα: ο καθένας γράφει ή επιλέγει το κατάλληλο μίγμα Benchmarks για το σύστημά του**
- **Εύκολο για ένα κέντρο πρόβλεψης καιρού – πολύ δύσκολο για γενικών εφαρμογών Data Centers**

SPEC

Standard Performance Evaluation Cooperative

www.spec.org

- **Ανεξάρτητο – χαίρει εμπιστοσύνης και από χρήστες και από εταιρείες**
- **Ανανεώνεται τακτικά**
- **Βασίζεται σε γεωμετρικούς μέσους**
- **..περιέχει και εξειδικευμένα benchmarks**
- **Μετρικές ταχύτητας και ρυθμού:**
  - **Ταχύτητα εκτέλεσης μοναδικού καθήκοντος (single task) και ρυθμοαπόδοση (throughput)**

Μετροπρογράμματα SPEC CPU

- Χρησιμοποιούνται προγράμματα για τη μέτρηση της απόδοσης
  - Υποτίθεται τυπικά για ένα πραγματικό φορτίο εργασίας (workload)
- Standard Performance Evaluation Corp (SPEC)
  - Αναπτύσσει μετροπρογράμματα (benchmarks) για CPU, είσοδο/έξοδο, Ιστό, …
- SPEC CPU2006
  - Χρόνος για την εκτέλεση μιας συλλογής προγραμμάτων
    - Αμελητέα είσοδος/έξοδος, άρα εστιάζουν στην απόδοση της CPU
  - Κανονικοποίηση σε σχέση με μια μηχανή αναφοράς (reference machine)
  - Σύνοψη ως γεωμετρικός μέσος (geometric mean) των λόγων απόδοσης (performance ratios)
    - CINT2006 (integer) and CFP2006 (floating-point)

SPEC CPU2006

- **Υπολογίζει την απόδοση κάθε κώδικα συγκριτικά με ένα συμφωνημένο μηχάνημα αναφοράς (2006: Ultrasparc 296MHz)**
  - **17 προγράμματα κινητής υποδιαστολής σε C, C++, Fortran**
  - **12 προγράμματα ακεραίων σε C, C++**
  - **3 εκατ. γραμμές κώδικα**
- **Κανονικοποιημένοι λόγοι – όχι σκέτα MFLOPS**
  - **Τα αποτελέσματα αναφέρονται ως ο λόγος του χρόνου αναφοράς προς το χρόνο εκτέλεσης του συστήματος**
    - **Trefi χρόνος εκτέλεσης του διαγνωστικού i στη μηχανή αναφοράς**
    - **Tsuti χρόνος εκτέλεσης του διαγνωστικού i στο σύστημα υπό δοκιμή**

$$r_i = \frac{Tref_i}{Tsut_i}$$

SPEC CPU2006

- **Μέτρηση ταχύτητας: Σύνδεση των αποτελεσμάτων σε ένα μοναδικό δείκτη απόδοσης**
  - **Η συνολική απόδοση υπολογίζεται με το μέσο όρο των λόγων και από τα 12 διαγνωστικά ακεραίων**
  - **Χρήση γεωμετρικού μέσου (geometric mean)**
  - **Κατάλληλος για κανονικοποιημένους αριθμούς, όπως οι λόγοι**

$$r_G = \left(\prod_{i=1}^{n} r_i \right)^{1/n}$$

SPEC CPU2006

- **Μέτρηση ρυθμοαπόδοσης (throughput) μιας μηχανής που εκτελεί κάποια tasks**
  - **Πολλαπλά αντίγραφα διαγνωστικών εκτελούνται ταυτόχρονα: Συνήθως, ίδιος αριθμός με τον αριθμό επεξεργαστών**
  - **Ο λόγος υπολογίζεται ακολούθως:**

$$r_i = \frac{N \times Tref_i}{Tsut_i}$$

    - **Trefi χρόνος εκτέλεσης αναφοράς του διαγνωστικού i**
    - **N αριθμός αντιγράφων που εκτελούνται ταυτόχρονα**
    - **Tsuti χρόνος που πέρασε από την αρχή εκτέλεσης του προγράμματος σε όλους τους επεξεργαστές μέχρι την ολοκλήρωση όλων των αντιγράφων του προγράμματος**
  - **Και πάλι, υπολογίζεται ο γεωμετρικός μέσος**

CINT2006 για Opteron X4 2356

Όνομα	Περιγραφή	IC×10^9	CPI	Tc (ns)	Χρόν. εκτ.	Χρον. ανφ.	SPECratio
perl	Interpreted string processing	2,118	0.75	0.40	637	9,777	15.3
bzip2	Block-sorting compression	2,389	0.85	0.40	817	9,650	11.8
gcc	GNU C Compiler	1,050	1.72	0.47	24	8,050	11.1
mcf	Combinatorial optimization	336	10.00	0.40	1,345	9,120	6.8
go	Go game (AI)	1,658	1.09	0.40	721	10,490	14.6
hmmer	Search gene sequence	2,783	0.80	0.40	890	9,330	10.5
sjeng	Chess game (AI)	2,176	0.96	0.48	37	12,100	14.5
libquantum	Quantum computer simulation	1,623	1.61	0.40	1,047	20,720	19.8
h264avc	Video compression	3,102	0.80	0.40	993	22,130	22.3
omnetpp	Discrete event simulation	587	2.94	0.40	690	6,250	9.1
astar	Games/path finding	1,082	1.79	0.40	773	7,020	9.1
xalancbmk	XML parsing	1,058	2.70	0.40	1,143	6,900	6.0
Geometric mean							11.7

Υψηλοί ρυθμοί αστοχίας κρυφής μνήμης (cache misses)

Μετροπρογράμματα SPEC Power

- **Κατανάλωση ισχύος διακομιστή (server) σε διαφορετικά επίπεδα φορτίου εργασίας**
  - **Απόδοση: ssj_ops/sec**
  - **Ισχύς: Watts (Joules/sec)**

$$\text{Overall ssj\_ops per Watt} = \left(\sum_{i=0}^{10} \text{ssj\_ops}_i \right) \Big/ \left(\sum_{i=0}^{10} \text{power}_i \right)$$

SPECpower_ssj2008 για X4

Φορτίο στόχου %	Απόδοση (ssj_ops/sec)	Μέση ισχύς (Watts)
100%	231,867	295
90%	211,282	286
80%	185,803	275
70%	163,427	265
60%	140,160	256
50%	118,324	246
40%	920,35	233
30%	70,500	222
20%	47,126	206
10%	23,066	180
0%	0	141
Overall sum	1,283,590	2,605
∑ssj_ops/ ∑power		493

Πλάνη: Χαμηλή ισχύς αδράνειας

- Δείτε το μετροπρόγραμμα ισχύος στον X4
  - Στο 100% του φορτίου: 295W
  - Στο 50% του φορτίου: 246W (83%)
  - Στο 10% του φορτίου: 180W (61%)
- Κέντρο δεδομένων Google
  - Κυρίως λειτουργεί στο 10% – 50% του φορτίου
  - Με φορτίο 100% σε λιγότερο από 1% του χρόνου

- Θα θέλαμε επεξεργαστές με κατανάλωση ισχύος ανάλογη με το φορτίο!

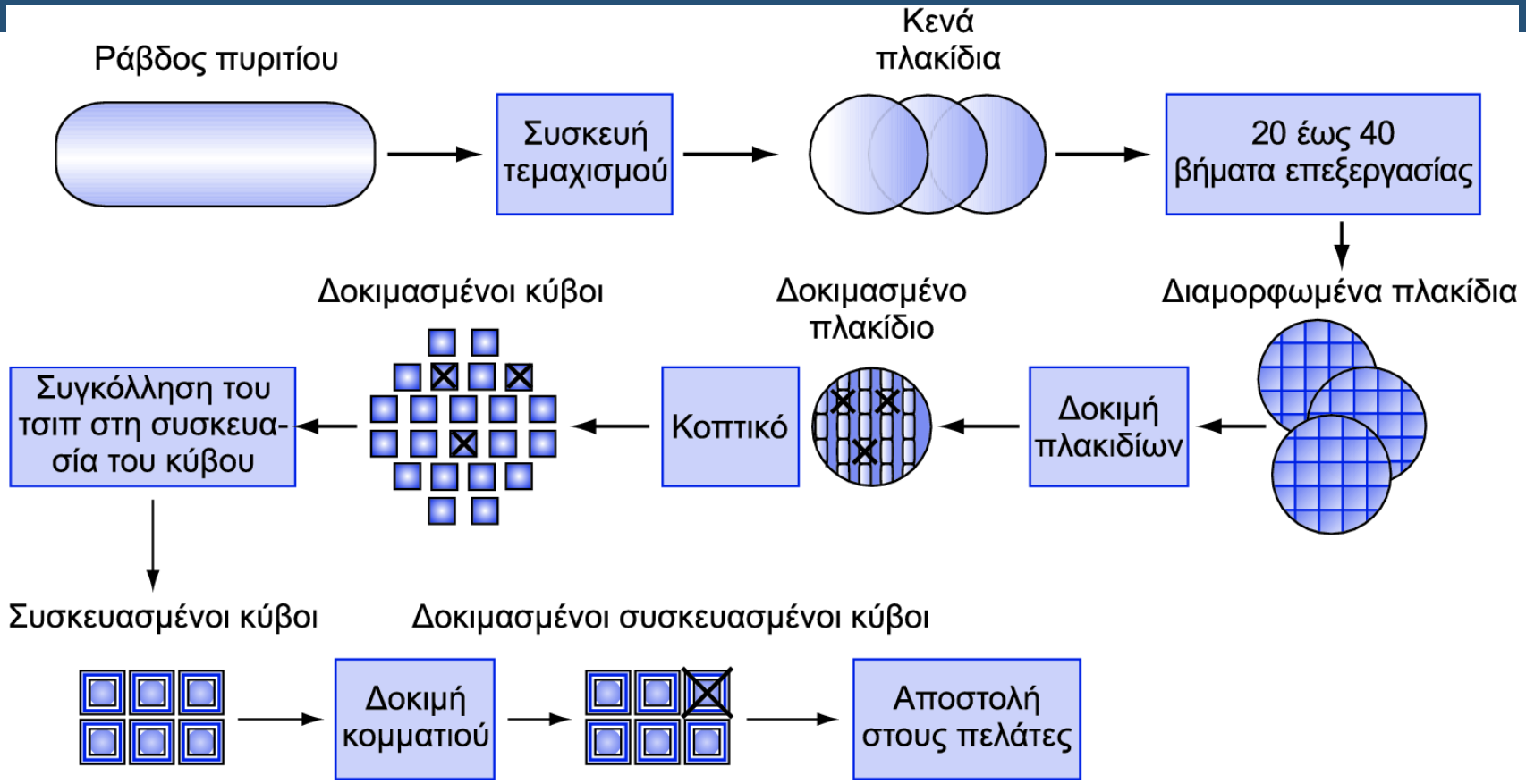
Μερικά “επιστημονικά” Benchmarks”

- **(...ιστορία) Whetstone: Πρώιμη (1972) προσπάθεια προσομοίωσης επιστημονικού φόρτου εργασίας. Σύνθετο. Σειριακό.**
- **(...ιστορία) Livermore Loops: Kernels 24 loops από λειτουργικούς κώδικες στο LLNL. Ανακοίνωσε MFLOPS. Kernel. Σειριακό**
- **Linpack: βασισμένο σε πακέτο γραμμικής άλγεβρας. Kernel. Σειριακό/Παράλληλο.**
- **SPEC: βιομηχανικός οργανισμός. Μεγάλη ποικιλία εξυπηρέτησης κώδικα.**

Συμπερασματικές παρατηρήσεις

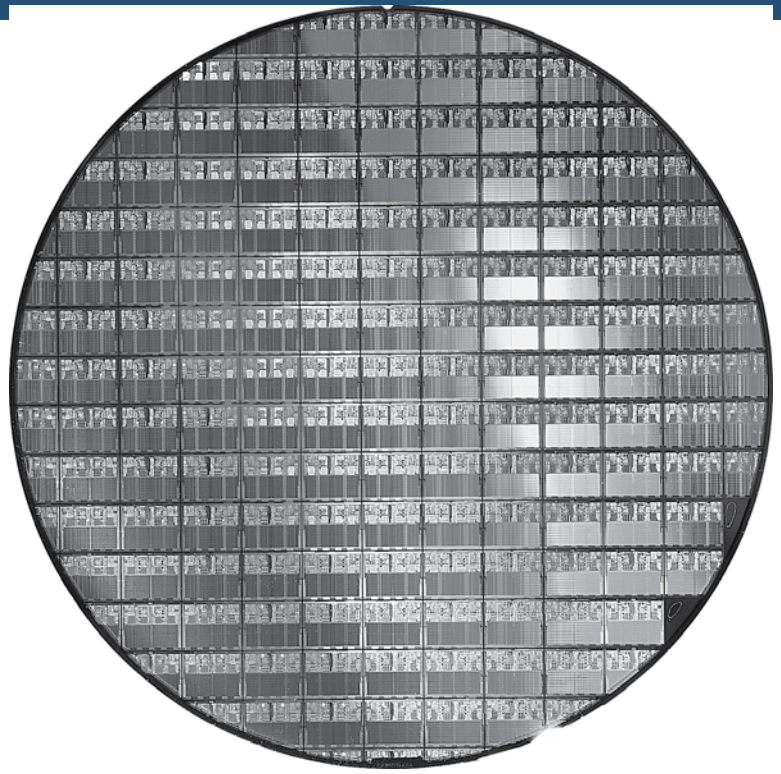
- Ο λόγος κόστος/απόδοση βελτιώνεται
  - Λόγω της εξέλιξης της τεχνολογίας
- Ιεραρχικά επίπεδα αφαίρεσης
  - Στο υλικό και στο λογισμικό
- Αρχιτεκτονική συνόλου εντολών (Instruction set architecture – ISA)
  - Η διασύνδεση υλικού και λογισμικού
- Χρόνος εκτέλεσης: το καλύτερο μέτρο απόδοσης
- Η ισχύς είναι περιοριστικός παράγοντας
  - Χρήση παραλληλίας για βελτίωση της απόδοσης

Κατασκευή ολοκληρωμένων



- **Εσοδεία (yield): ποσοστό τσιπ ανά πλακίδιο (wafer) που λειτουργούν σωστά**

AMD Opteron X2 Wafer



- **X2: 300mm πλακίδιο, 117 τσιπ, τεχνολογία 90nm**
- **X4: τεχνολογία 45nm**

Κόστος Ολοκληρωμένων

$$\text{Κόστος ανά κύβο} = \frac{\text{Κόστος ανά πλακίδιο}}{\text{Κύβοι ανά πλακίδιο} \times \text{Εσοδεία}}$$

$$\text{Κύβοι ανά πλακίδιο} \approx \text{Επιφάνεια πλακιδίου} / \text{Επιφάνεια κύβου}$$

$$\text{Εσοδεία} = \frac{1}{(1 + \text{Ατέλειες ανά μονάδα επιφανείας} \times \text{Επιφάνεια κύβου}/2))^2}$$

- **Μη γραμμική εξάρτηση από την επιφάνεια (area) και το ρυθμό ατελειών (defect rate)**
  - **Κόστος και επιφάνεια πλακιδίου (wafer): σταθερά**
  - **Ρυθμός ατελειών (defect rate) εξαρτάται από τη διαδικασία κατασκευής**
  - **Επιφάνεια τσιπ (die/chip area): εξαρτάται από την αρχιτεκτονική και τη σχεδίαση του κυκλώματος**

Κανόνας του Amdahl (Amdahl's law)

- **…όταν περισσότεροι του ενός παράγοντες συμβάλλουν στη συνολική απόδοση**
- **Αν βελτιωθεί ένας συγκεκριμένος παράγοντας, η βελτίωση της συνολικής απόδοσης περιορίζεται από τη συχνότητα χρήσης αυτού του παράγοντας**
- **…οπότε: "make the common case fast" – εντονότερη προσπάθεια βελτίωσης των συχνότερων**

$$Execution\ time\ after\ improvement = \frac{Execution\ time\ affected\ by\ improvement}{Amount\ of\ improvement} + Execution\ time\ unaffected$$

Compute Speedup – Amdahl's Law: Make the common case fast

- **Το Speedup λόγω του enhancement:**

$$\text{Speedup (E)} = \frac{\text{Execution time w/o E (Before)}}{\text{Execution time w E (After)}}$$

Επιτάχυνση (Speedup)

- **Είναι ο λόγος της ταχύτητας πριν την εφαρμογή της βελτίωσης προς την ταχύτητα μετά τη βελτίωση**
- **Πόσο γρηγορότερα θα εκτελεστεί μία εργασία μετά τη βελτίωση σε σχέση με πριν**
- **Είναι λόγος – δεν έχει μονάδες**

Amdahl:

- $f_{\texttt{enhanceable}}$ the fraction of the original computation time that can be converted to take advantage of the enhancement
- `Speedup of enhancement` the improvement gained by the enhancement. The speedup if the whole computation was enhancable

Κανόνας του Amdahl (Amdahl's law)

$$\text{Speedup} = \frac{\text{Execution Time (performance) without enhancement}}{\text{Execution Time (performance) with enhancement}}$$

$$= \frac{T}{(1 - f_{enhanceable})T + T\, f_{enhanceable} / \text{Speedup}_{enhancement}} =$$

$$= \frac{1}{(1 - f_{enhanceable}) + f_{enhanceable} / \text{Speedup}_{enhancement}}$$

- **...π.χ., αν μία βελτίωση μπορεί να επιταχύνει το 30% των υπολογισμών, $f_{enhanceable}=0.3$**
- **...κι αν η βελτίωση διπλασιάζει την ταχύτητα εκτέλεσης του ποσοστού χρόνου f, τότε $\text{Speedup}_{enhancement}=2$**

Παγίδα: νόμος του Amdahl

- **Η βελτίωση μιας πλευράς ενός υπολογιστή και η αναμονή ανάλογης βελτίωσης της συνολικής απόδοσης**

$$T_{\text{μετά τη βελτίωση}} = \frac{T_{\text{που επηρεάζεται}}}{\text{συντελεστής βελτίωσης}} + T_{\text{που δεν επηρεάζεται}}$$

  - Παράδειγμα: ο πολ/σμός είναι τα 80s/100s
    - Πόση βελτίωση της απόδοσης του πολ/σμού ώστε η συνολική απόδοση να 5-πλασιαστεί;

$$20 = \frac{80}{n} + 20$$

    - Δεν γίνεται!

- **Πόρισμα: κάνε τη συνηθισμένη περίπτωση γρήγορη**

Amdahl: Παράδειγμα Ferrari

- **Σε ένα ταξίδι πρέπει κάποιος να διασχίσει 80 μίλια μέσα από βουνά και 200 μίλια μέσα από έρημο. Για τα 80 μίλια βουνών μπορεί μόνο πεζός να κινηθεί, αλλά για τα 200 μίλια έρημο μπορεί να χρησιμοποιήσει διάφορους τρόπους μετακίνησης/ οχήματα:**
  - **Βάδην με ταχύτητα 4 μίλια την ώρα**
  - **Ποδήλατο με μέση ταχύτητα 10 μίλια την ώρα**
  - **Αυτοκίνητο Hyundai με μέση ταχύτητα 50 μίλια την ώρα**
  - **Αυτοκίνητο Ferrari με μέση ταχύτητα 120 μίλια την ώρα**
  - **Αεροπλάνο με μέση ταχύτητα 600 μίλια την ώρα**
- **Πόση ώρα θα διαρκέσει το ταξίδι, και ποια είναι η επιτάχυνση σε σχέση με το να κινούνταν πεζός καθόλη τη διαδρομή?**

Λύση

- **Συνολικός χρόνος ταξιδιού:**

$$T=80/4 + 200/x$$

όπου x η ταχύτητα του εκάστοτε μέσου

- **Το ποσοστό του χρόνου fenhanceable που επιδέχεται βελτίωσης είναι:**

$$f_{enhanceable} = \frac{200/4}{(80+200)\ /4} \qquad Speedup_{enhanceable} = \frac{x}{4}$$

$$Speedup = \frac{(80+200)\ /4}{80/4 + 200/x} = \frac{1}{1 - \frac{200/4}{(80+200)/4} + \frac{200\ /4}{(80+200)/4} \cdot \frac{4}{x}}$$

Amdahl: Παράδειγμα Cache

- Έστω μνήμη cache που είναι 10 φορές ταχύτερη από την κυρίως μνήμη, και έστω ότι χρησιμοποιείται για το 90% του χρόνου

- Πόσο speedup κερδίζουμε χρησιμοποιώντας αυτή την cache?

Ο Amdahl σε παράλληλες μηχανές

- **Μερικές φορές ο κανόνας του Amdahl αναφέρεται σε παράλληλες μηχανές, όπου τα προγράμματα έχουν συνήθως ένα σειριακό κομμάτι (που δε μπορεί να παραλληλιστεί), κι ένα κομμάτι που μπορεί να παραλληλιστεί**

- **Το σειριακό κομμάτι περιορίζει το speedup που μπορεί να επιτευχθεί**
- **...θα επιστρέψουμε σε αυτό αργότερα**

Ο Amdahl σε παράλληλες μηχανές

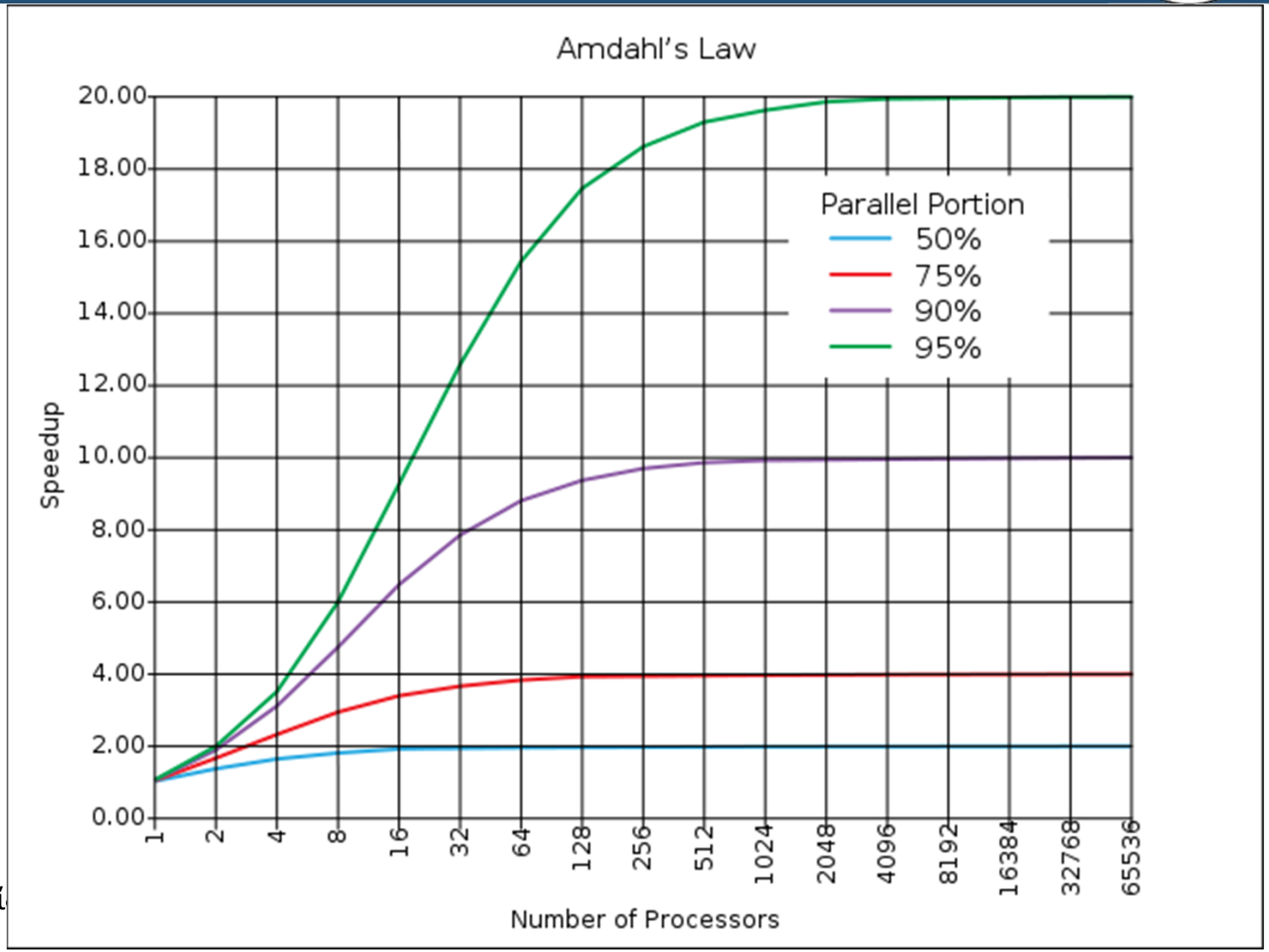
- **Για ένα πρόγραμμα που τρέχει σε ένα μοναδικό επεξεργαστή:**
  - **ποσοστό f του κώδικα είναι άπειρα παραλληλοποιήσιμο χωρίς κόστος δρομολόγησης**
  - **ποσοστό (1-f) του κώδικα είναι αμιγώς σειριακό**
  - **Τ ο συνολικός χρόνος εκτέλεσης για ένα πρόγραμμα σε ένα μοναδικό επεξεργαστή**
  - **Ν είναι ο αριθμός των επεξεργαστών που εκμεταλλεύονται πλήρως ένα παράλληλο τμήμα του κώδικα**

$$Speedup = \frac{\text{time to execute program on a single processor}}{\text{time to execute program on } N \text{ parallel processors}} = \frac{T(1-f)+Tf}{T(1-f)+\frac{Tf}{N}} = \frac{1}{(1-f)+\frac{f}{N}}$$

Ο Amdahl σε παράλληλες μηχανές

Συμπεράσματα

- **Μικρό f, οι παράλληλοι επεξεργαστές έχουν μικρή επίδραση**
- **N ->∞, η βελτίωση απόδοσης έχει όριο 1/(1 – f)**
- **Η βελτίωση μειώνεται καθώς αυξάνουν οι επεξεργαστές**



Γενικός – Χοντρικός Κανόνας

- **Το 10-20% του κώδικα ενός προγράμματος είναι υπεύθυνο για το 80-90% του χρόνου εκτέλεσης του προγράμματος**
- **Δεν είναι πάντα εύκολο να προσδιοριστεί αυτό το κομμάτι του κώδικα**
  - **Πχ. Τι γίνεται όταν αλλάζει το input ?**

- **Πως βρίσκω αυτά τα κομμάτια του κώδικα???**