Il Calcolatore: astrazioni e tecnologia

Introduzione

- Il mondo dei sistemi di elaborazione e' in stretto contatto con realtà industriali importanti ed in crescita.

- Negli anni scorsi abbiamo assistito ad un progresso impressionante con innovazioni che hanno portato alla terza rivoluzione della civiltà (dopo quella agricola e quella industriale): quella dell'informazione.

- Se l'industria dei trasporti avesse visto lo stesso progresso oggi potremmo viaggiare in pochi secondi per migliaia di chilometri con 50 centesimi!!

Rivoluzione informatica

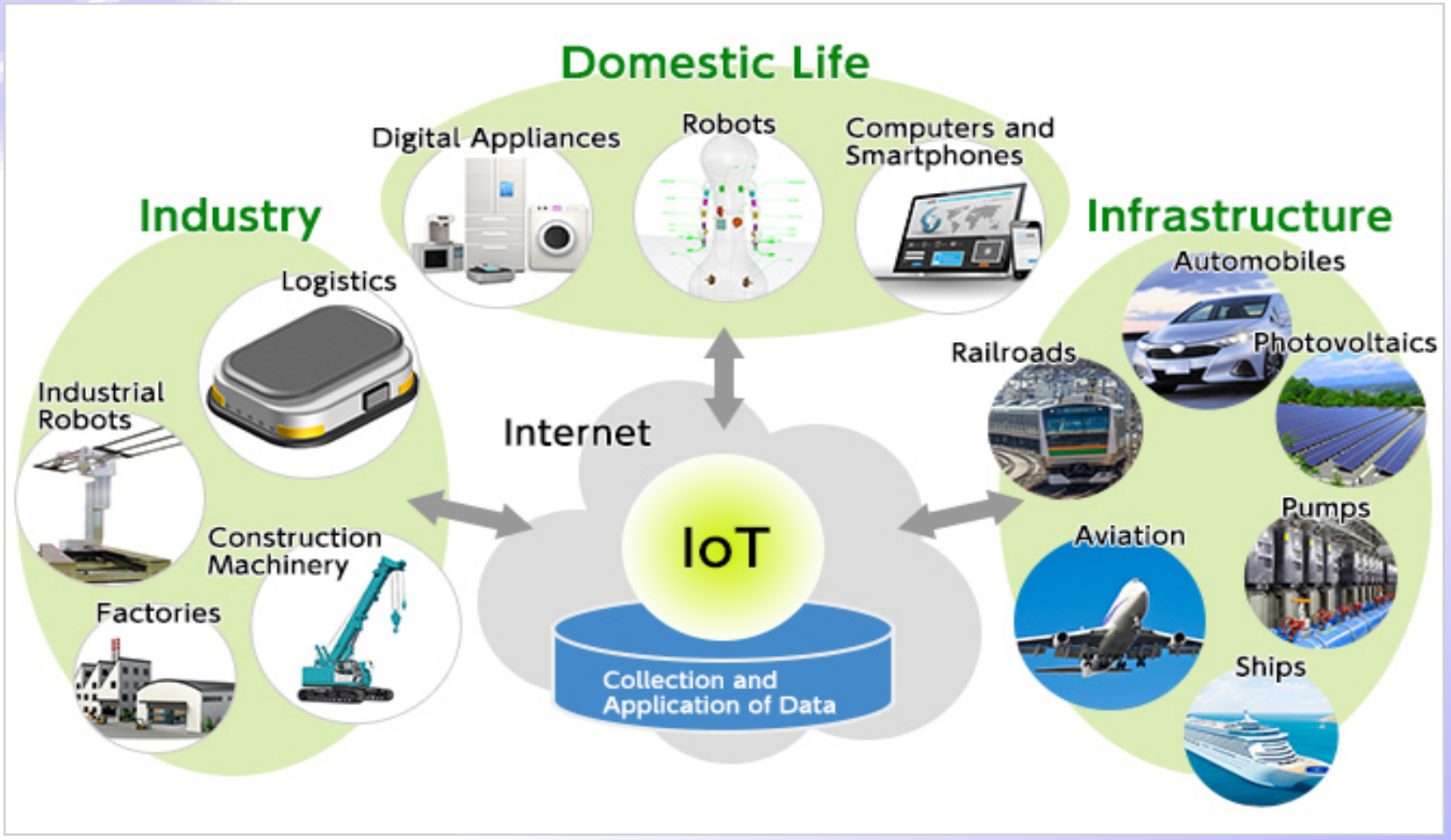
- I costi e le capacità (di elaborazione, memorizzazione, trasmissione) continuano a migliorare e rendono possibili applicazioni straordinarie.
- Ex 'fantascienza informatica':
  - sportelli bancari automatici
  - calcolatori nelle auto (e altri mezzi di trasporto)
  - calcolatori portatili
  - progetto Genoma Umano
  - rete mondiale (www)
  - Cloud computing
- Chissa' cosa succedera' prossimamente.....

Computing evolution

- Mainframe computing (60's-70's)
  - Large computers to execute big data processing applications
- Desktop computing & Internet (80's-90's)
  - One computer at every desk to do business/personal activities

- Ubiquitous computing (00's)
  - Numerous computing devices in every place/person
  - "Invisible" part of the environment
  - Millions for desktops vs. billions for embedded processors
- Cyber Physical Systems (10's)

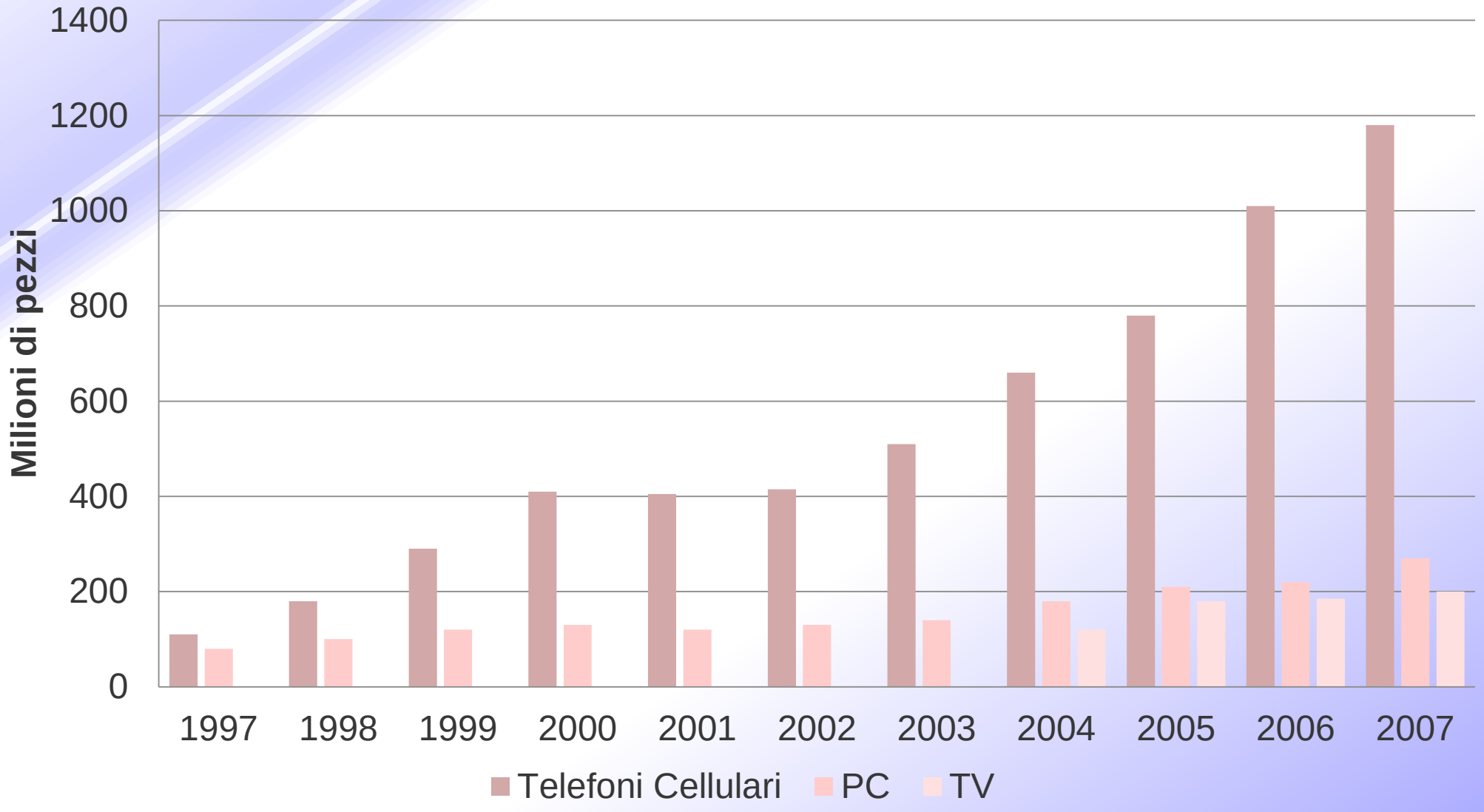
CPS... and related terms: *Internet of Things*



Tipi di calcolatori e loro caratteristiche

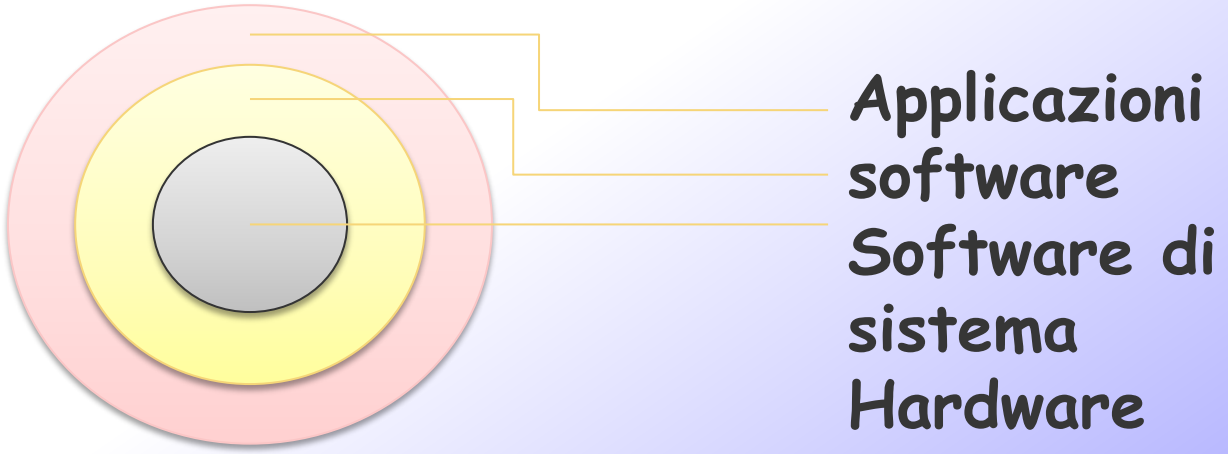
- Calcolatori diversi tra loro condividono la stessa tecnologia hardware (elettrodomestici, Cellulari, supercomputer…)
- Tuttavia adottano soluzioni architetturali differenti
  - Calcolatori Desktop
    - Progettato per essere utilizzato da una sola persona alla volta
    - Costo limitato
  - Calcolatori Server
    - Progettati per l'esecuzione di programmi di grosse dimensioni e per servire più utenti spesso simultaneamente
    - Tipicamente accesso solo via rete
  - Calcolatori Embedded
    - Posti all'interno di un altro dispositivo, usati esclusivamente per eseguire una predeterminata applicazione o insieme di programmi.

Numero di telefoni cellulari, pc e televisori prodotti dal 1997 al 2007



Cosa c'è dietro un programma

- Le comuni applicazioni sono formate da milioni di linee di codice e sofisticate librerie
- Il calcolatore può solo eseguire istruzioni a basso livello estremamente semplici
- Passare da istruzioni complesse (alto livello) ad istruzioni semplici (basso livello comprensibili dal calcolatore) coinvolge diversi strati di software organizzati in maniera gerarchica



Software di sistema

- Il SW di sistema ha alcune componenti principali
- Il Compilatore
  - Traduce le istruzioni scritte in linguaggio alto livello in istruzioni eseguibili dal calcolatore
- Il Sistema Operativo
  - Permette di interfacciare i programmi utente con l'hardware
  - Gestisce le operazioni di Input/Output
  - Alloca lo spazio nella memoria principale e nei vai dispositivi
  - Consente a più applicazioni di eseguire simultaneamente
- Il Middleware (o Application server)
  - Supporto per le applicazioni in ambiente virtuale

I calcolatori e gli 'umani'

- I calcolatori 'capiscono' impulsi elettrici on-off.
- Hanno quindi un alfabeto binario (0-1) BIT=binary digit
- Seguono ciecamente le istruzioni che ricevono.
- I primi programmatori comunicavano in binario ma presto furono inventate notazioni simboliche piu' vicine al linguaggio umano

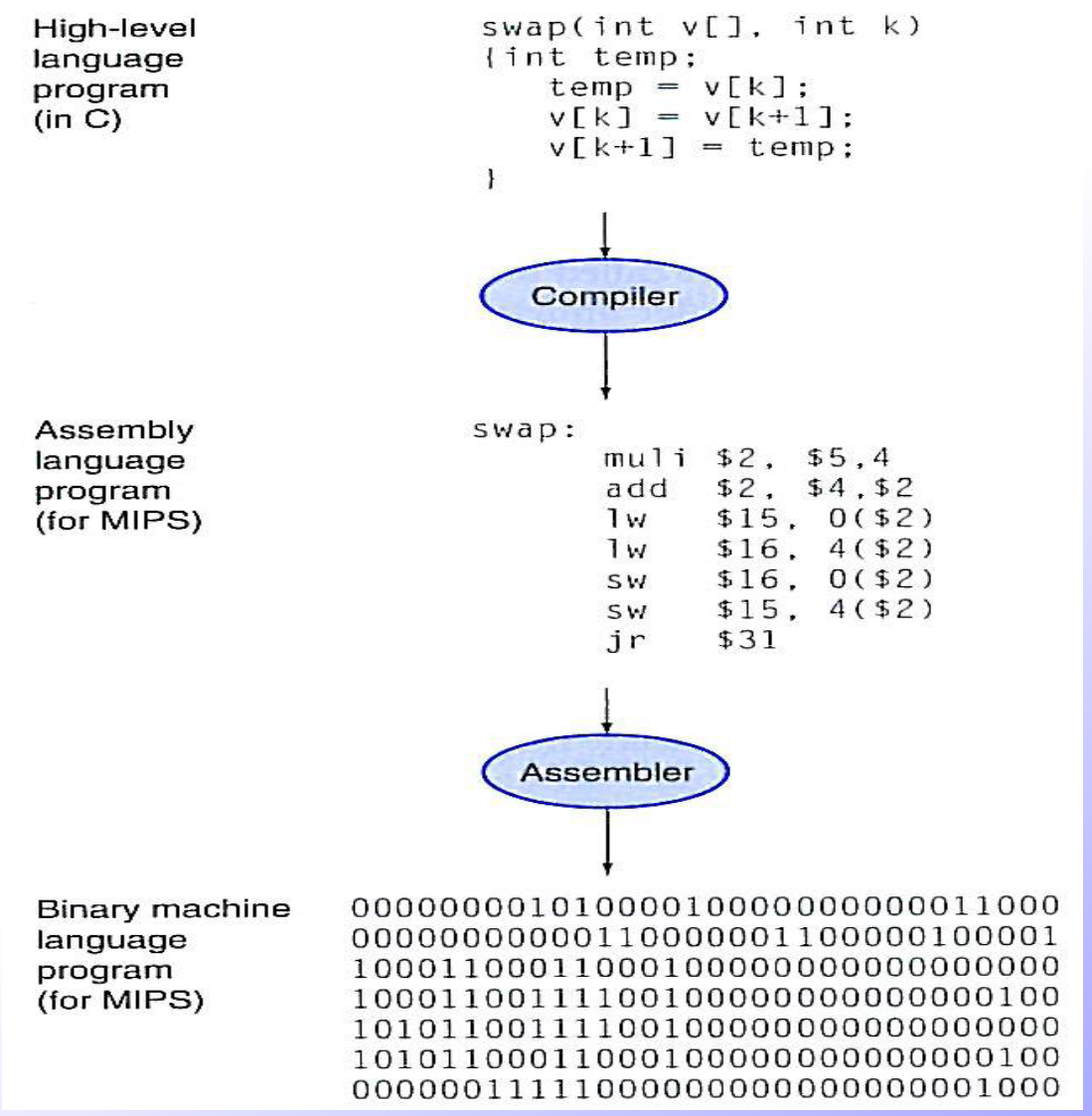
  →ASSEMBLER
- Successivamente furono inventati linguaggi di alto livello.

Vantaggi dei linguaggi di alto livello

- il programmatore puo pensare in modo piu' naturale
- I linguaggi possono essere progettati per l'uso che se ne prevede
- incremento di produttivita' dei programmatori: il tempo necessario per scrivere un programma e' proporzionale al numero di linee di codice.
- indipendenza dal calcolatore e portabilita': il programma puo essere tradotto in modo diverso per macchine diverse.

Traduzione dei programmi

- Dal linguaggio ad alto livello tramite il compilatore il programma viene tradotto in linguaggio assembler
- E poi tramite l'assemblatore in codice macchina: il linguaggio direttamente comprensibile dal calcolatore.



I componenti di un calcolatore

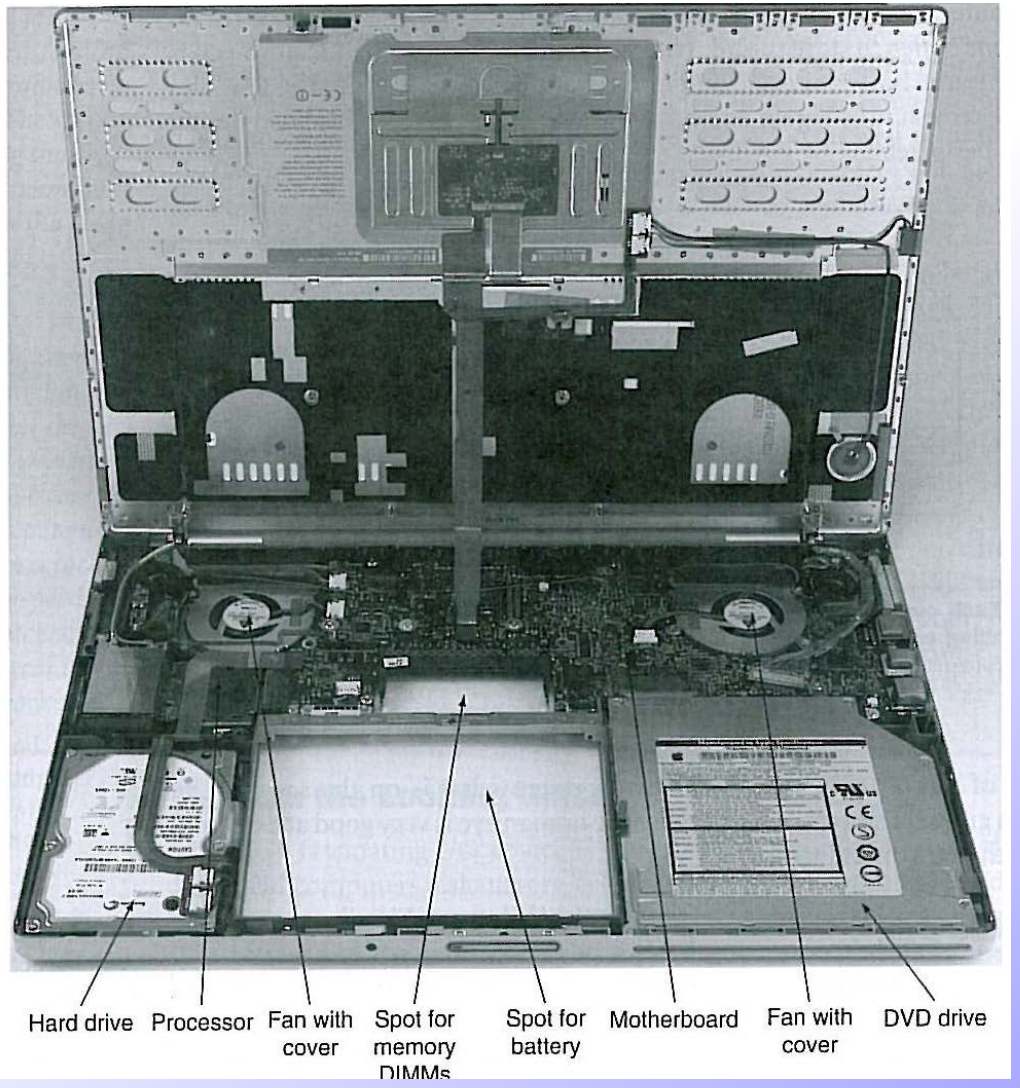
Componenti:

- Unita' di input (mouse, keyboard, reti)
- Unita' di output (display, printer, reti)
- Memoria (disk drives, DRAM, SRAM, CD)
- Unita' di elaborazione
- Unita' di controllo

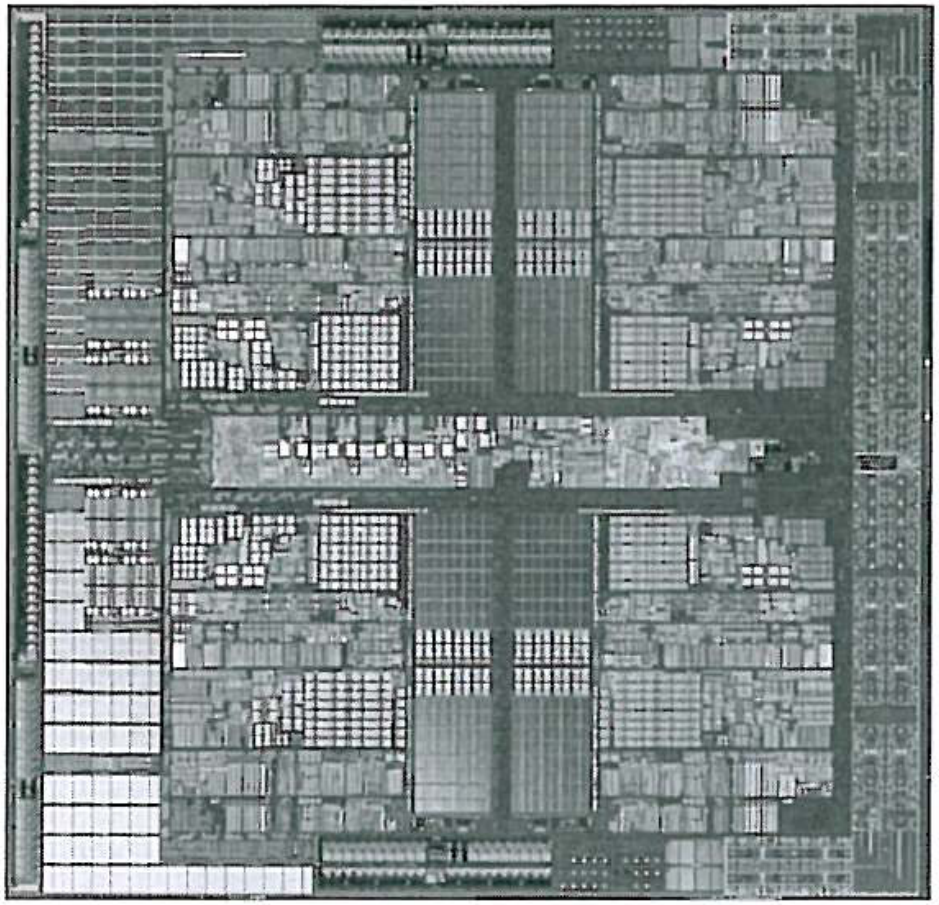
Il nostro scopo primario: il processore (elaborazione e controllo)

- realizzato usando milioni di transistors
- Impossibile da capire guardando ad ogni transistor
- Abbiamo bisogno di..

ASTRAZIONE



Il microprocessore AMD Barcelona



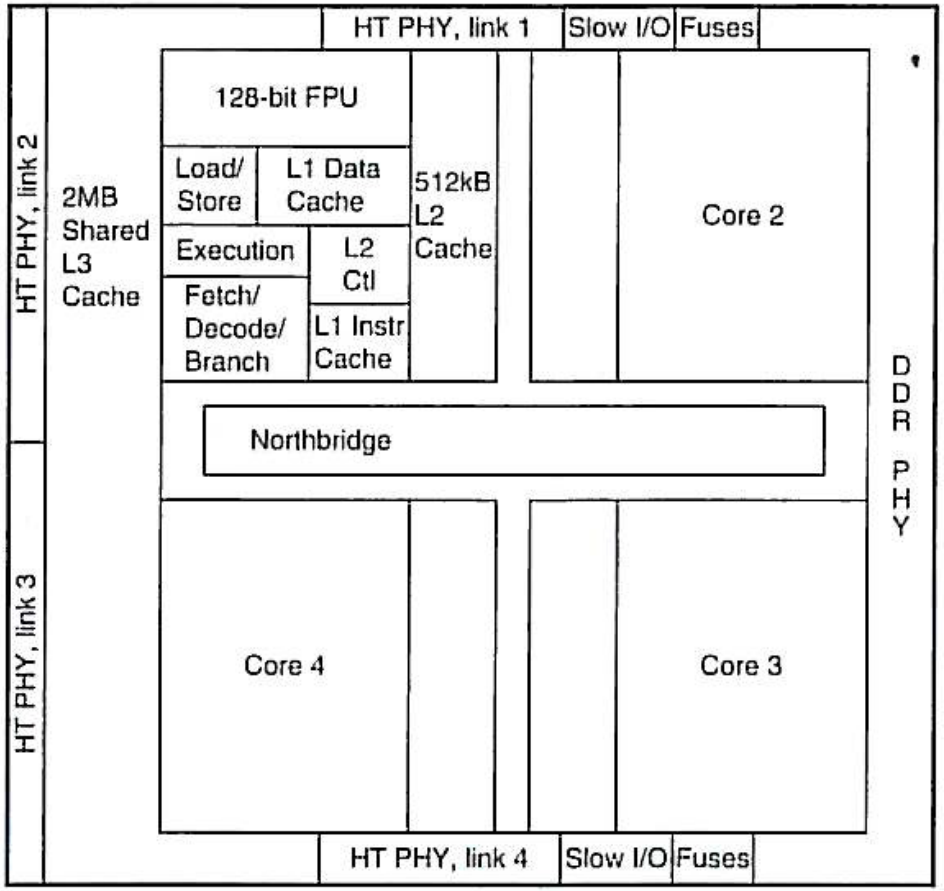


FIGURE 1.9 Inside the AMD Barcelona microprocessor. The left-hand side is a microphotograph of the AMD Barcelona processor chip, and the right-hand side shows the major blocks in the processor. This chip has four processors or "cores". The microprocessor in the laptop in Figure 1.7 has two cores per chip, called an Intel Core 2 Duo.

La memoria

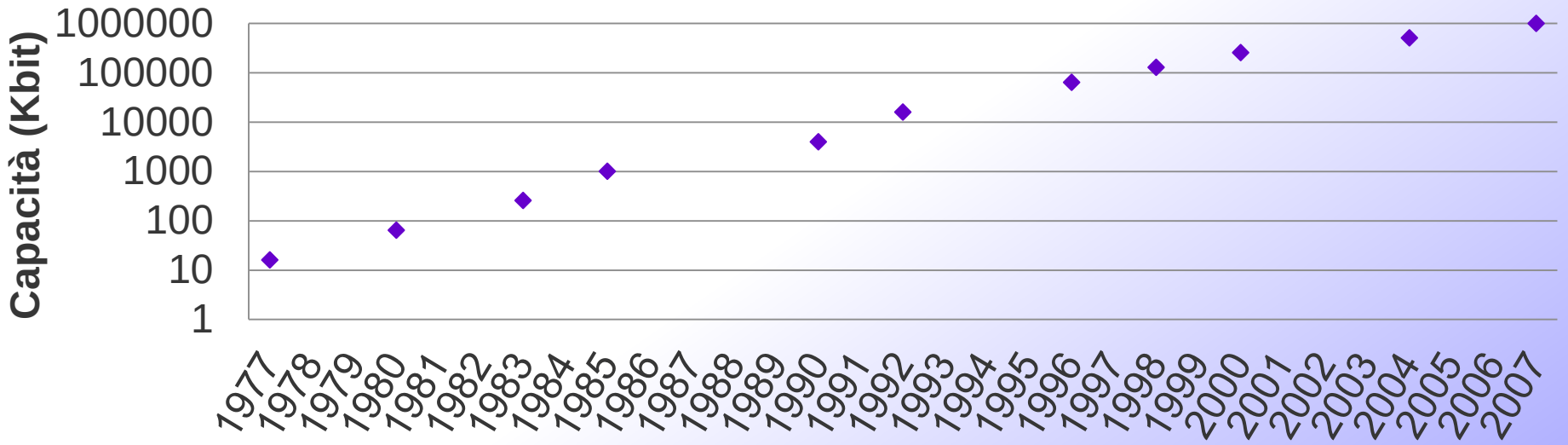
- Memoria Volatile
  - Capace di mantenere i dati solo se alimentata (SRAM, DRAM)
  - Viene usata per immagazzinare i programmi durante la loro esecuzione (memoria principale)
- Memoria non volatile
  - Conserva i dati anche senza alimentazione (Disco magnetico, Memoria a stato solido – 'chiavine usb')
  - Utilizzata per immagazzinare i programmi fra esecuzioni differenti (memoria di massa)

Comunicare con altri calcolatori

- Il collegamento alla rete collega tra loro i calcolatori consentendo agli utenti di estendere la capacità di elaborazione attraverso la comunicazione
- Vantaggi:
  - Comunicazione
  - Condivisione delle risorse
  - Accesso non locale
- Rete locale con filo ethernet (LAN) e senza Filo (Wireless LAN)
  - Distanza massima fra due nodi: 1km
  - Velocità massima di trasferimento dati: 10 Gb/s
- Rete geografica (wan, 3/4/5-G)
  - Attraversano i continenti
  - Sono l'ossatura di INTERNET

Tecnologia per la produzione di processori e memorie

Anno	Tecnologia utilizzata nei calcolatori	Prestazioni relative per unità di costo
1951	**Tubo a vuoto** (valvola)	1
1965	Transistor	35
1975	Circuito integrato	900
1995	Circuito integrato a grandissima scala di integrazione	2 400 000
2005	Circuito integrato a scala di integrazione estremamente grande	6 200 000 000



Il ruolo delle prestazioni - qui

- L'importanza delle prestazioni e di saperle misurare.
- Definire le prestazioni.
- La misura delle prestazioni.
- Relazione fra le metriche.
- La scelta dei benchmarks.

Performance

- Measure, Report, and Summarize
- Make intelligent choices
- See through the marketing advertising
- Key to understanding underlying organizational motivation
- *Why is some hardware better than others for different programs?*
- *What factors of system performance are hardware related?*
  - *(e.g., Do we need a new machine, or a new operating system?)*
- *How does the machine's instruction set affect performance?*

Which of these airplanes has the best performance?

Airplane	Passengers	Range (mi)	Speed (mph)
Boeing 737-100	101	630	598
Boeing 747	470	4150	610
BAC/Sud Concorde	132	4000	1350
Douglas DC-8-50	146	8720	544

- How much faster is the Concorde compared to the 747?
- How much bigger is the 747 than the Douglas DC-8?

- The problem is to DEFINE Performance (what really matters) (different for different artifacts)

Computer Performance: TIME, TIME, TIME

- Response Time (latency)
  - How long does it take for my job to run?
  - How long does it take to execute a job?
  - How long must I wait for the database query?
- Throughput
  - How many jobs can the machine run at once?
  - What is the average execution rate?
  - How much work is getting done?
- *If we upgrade a machine with a new faster processor what do we increase?*
- *If we add a new machine to the lab what do we increase?*

Risposta

- *If we upgrade a machine with a new faster processor what do we increase?*
  - In questo caso migliorano le prestazioni sia perche' diminuisce il tempo di risposta ma anche il throughput

- *If we add a new machine to the lab what do we increase?*
  - Migliorano anche cosi le prestazioni
  - Principalmente il throughput perche' possono essere eseguiti piu' lavori, anche se nessuno piu' velocemente.
  - Pero' se nel sistema abbiamo una coda di lavori in attesa ecco che anche il tempo di risposta migliora.

Execution Time

- Elapsed Time
  - counts everything (disk and memory accesses, I/O , etc.)
  - a useful number, but often not good for comparison purposes
- CPU time
  - doesn't count I/O or time spent running other programs
  - can be broken up into system time, and user time
- Our focus: user CPU time
  - time spent executing the lines of code that are "in" our program

Book's Definition of Performance

- For some program running on machine X

$$Performance(X) = \frac{1}{Execution\,Time\,(X)}$$

- X is n times faster than Y

$$n = \frac{Performance(X)}{Performance\,(Y)} = \frac{Execution\,Time(Y)}{Execution\,Time\,(Y)}$$

- Problem:
  - machine A runs a program in 20 seconds
  - machine B runs the same program in 25 seconds

Risposta

$$Performance(X) = \frac{1}{Execution\ Time\ (X)}$$

- X is n times faster than Y $\quad n = \frac{Performance(X)}{Performance\ (Y)} = \frac{Execution\ Time(Y)}{Execution\ Time\ (Y)}$

➢ Problem:

- A runs a program in 20 seconds — EtA = 20 seconds
- B runs the same program in 25 seconds — EtB = 25 seconds

$$n = \frac{25\ seconds}{20\ seconds} = \frac{5}{4} = 1.25$$

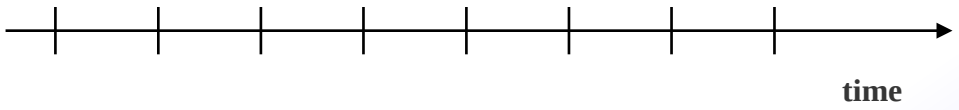
➢ A is 1.25 times faster than B

Clock Cycles

- Instead of reporting execution time in seconds, we often use cycles

$$\frac{seconds}{program} = \frac{cycles}{program} \times \frac{seconds}{cycles}$$

- Clock "ticks" indicate when to start activities (one abstraction):



- cycle time = time between ticks = seconds per cycle
- clock rate (frequency) = cycles per second (1 Hz. = 1 cycle/sec)

- A 2.0 Ghz. clock has a $\frac{1}{2\times10^9} \times 10^9 = 0{,}5\ nanoseconds$ cycle time

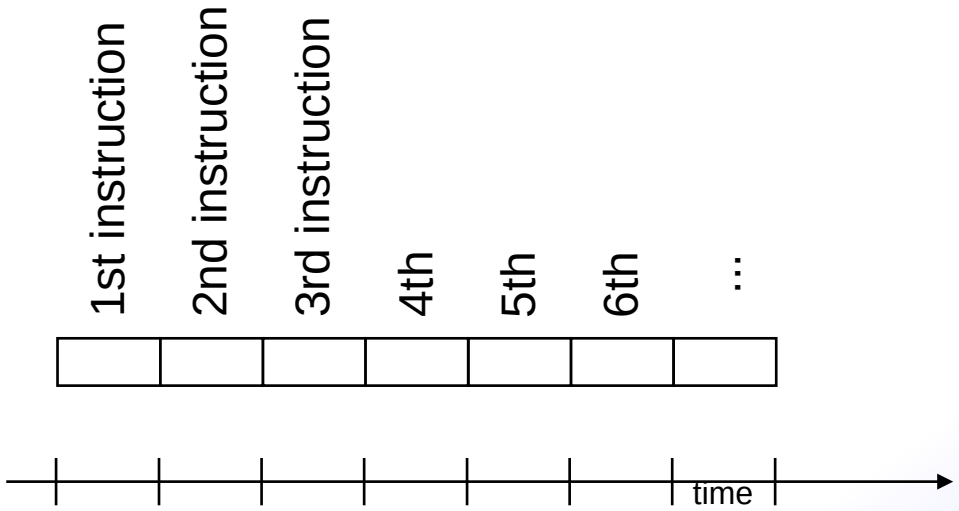
How to Improve Performance

$$\frac{seconds}{program} = \frac{cycles}{program} \times \frac{seconds}{cycles}$$

- So, to improve performance (everything else being equal) you can either
- _____ the # of required cycles for a program, or
- _____ the clock cycle time or, said another way,
- _____ the clock rate.

How many cycles are required for a program?

- Could assume that # of cycles = # of instructions???

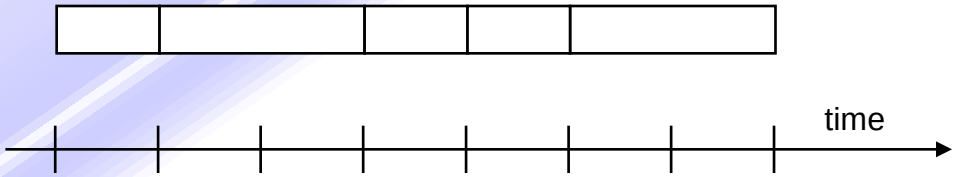


This assumption is incorrect,

different instructions take different amounts of time on different machines.

Why? remember that these are machine instructions, not lines of C code

Different numbers of cycles for different instructions



- Multiplication takes more time than addition
- Floating point operations take longer than integer ones
- Accessing memory takes more time than accessing registers

- Important point: changing the cycle time often changes the number of cycles required for various instructions (more later)

Performance considering cycles per instruction

- Cycles per Instruction = CPI

$$\frac{seconds}{program} = \frac{instructions}{program} \times \frac{cycles}{instructions} \times \frac{seconds}{cycles}$$

Example

- Our favorite program runs in 10 seconds on computer A, which has a 400 Mhz. clock.
- We are trying to help a computer designer build a new machine B, that will run this program in 6 seconds.
- The designer can use new (or perhaps more expensive) technology to substantially increase the clock rate, but has informed us that this increase will affect the rest of the CPU design, causing machine B to require 1.2 times as many clock cycles as machine A for the same program.
- What clock rate should we tell the designer to target?"
- Don't Panic, can easily work this out from basic principles

Example -- risposta

- 10 seconds on computer A, which has a 400 Mhz. clock.
- B will run this program in 6 seconds.
- B requires 1.2 times as many clock cycles as machine A.
- What clock rate should we tell the designer to target?"
- A:
  - $10\,\text{sec} = \frac{\#cycles}{400\,Mhz}$
  - $\#cycles = 4000 \times 10^6$
- B:
  - $6\,\text{sec} = \frac{1.2 \times 4000 \times 10^6}{???}$
  - $??? = \frac{4800 \times 10^6}{6} = 800\,Mhz$

Now that we understand cycles

- A given program will require
  - some number of instructions (machine instructions)
  - some number of cycles
  - some number of seconds
- We have a vocabulary that relates these quantities:
  - cycle time (seconds per cycle)
  - clock rate (cycles per second)
  - CPI (cycles per instruction)
    - *a floating point intensive application might have a higher CPI*
  - MIPS (millions of instructions per second)
    - *this would be higher for a program using simple instructions*

Performance

- Performance is determined by execution time!

- Do any of the other variables equal performance?
  - # of cycles to execute program?
  - # of instructions in program?
  - # of cycles per second?
  - average # of cycles per instruction? CPI
  - average # of instructions per second? MIPS

- Common pitfall: thinking one of the variables is indicative of performance when it really isn't.

CPI Example

- Suppose we have two implementations of the same instruction set architecture (ISA).
- For some program,
  - Machine A has a clock cycle time of 10 ns. and a CPI of 2.0
  - Machine B has a clock cycle time of 20 ns. and a CPI of 1.2
- What machine is faster for this program, and by how much?

  - *If two machines have the same ISA which of our quantities (e.g., clock rate, CPI, execution time, # of instructions, MIPS) will always be identical?*

CPI Example -- risposta

- Same instruction set architecture (ISA) and program.
  - A clock cycle time = 10 ns. and CPI = 2.0
  - B clock cycle time = 20 ns. and CPI = 1.2

- What machine is faster for this program, and by how much?
  - A --> 20 ns. Per instruction
  - B --> 24 ns. Per instruction
  - A is 1.2 times faster than B

  - *If two machines have the same ISA which of our quantities (e.g., clock rate, CPI, execution time, # of instructions, MIPS) will always be identical?*
- # of instructions

of Instructions Example

- A compiler designer is trying to decide between two code sequences for a particular machine. Based on the hardware implementation, there are three different classes of instructions:
  - Class A, Class B, and Class C, and they require one, two, and three cycles (respectively).
  - The first code sequence has 5 instructions:
    - 2 of A, 1 of B and 2 of C
  - The second sequence has 6 instructions:
    - 4 of A, 1 of B, and 1 of C.

- Which sequence will be faster? How much?
- What is the CPI for each sequence?

of Instructions Example -- risposta

$$CPI(A) = 1\ CPI(B) = 2\ CPI(C) = 3$$

- The first code sequence has 5 instructions: 2 of A, 1 of B, and 2 of C
- The second sequence has 6 instructions: 4 of A, 1 of B, and 1 of C.
- Which sequence will be faster? How much?

$$Cycles\ Seq1 \rightarrow 2 + 2 + 6 = 10$$
$$Cycles\ Seq2 \rightarrow 4 + 2 + 3 = 8$$

- Seq2 will be 1.25 times faster than Seq1

- What is the CPI for each sequence?

$$CPI(seq1) = \frac{10\ cycles}{5\ instructions} = 2 \qquad CPI(seq2) = \frac{8\ cycles}{6\ instructions} = 1.33$$

MIPS example

- Two different compilers are being tested for a 100 MHz. machine with three different classes of instructions:
  - Class A, Class B, and Class C,
- which require one, two, and three cycles (respectively).
- Both compilers are used to produce code for a large piece of software.
- The first compiler's code uses
  - 5 million Class A instructions,
  - 1 million Class B instructions,
  - 1 million Class C instructions.
- The second compiler's code uses
  - 10 million Class A instructions,
  - 1 million Class B instructions,
  - 1 million Class C instructions.

- Which sequence will be faster according to MIPS?
- Which sequence will be faster according to execution time?

MIPS example -- risposta

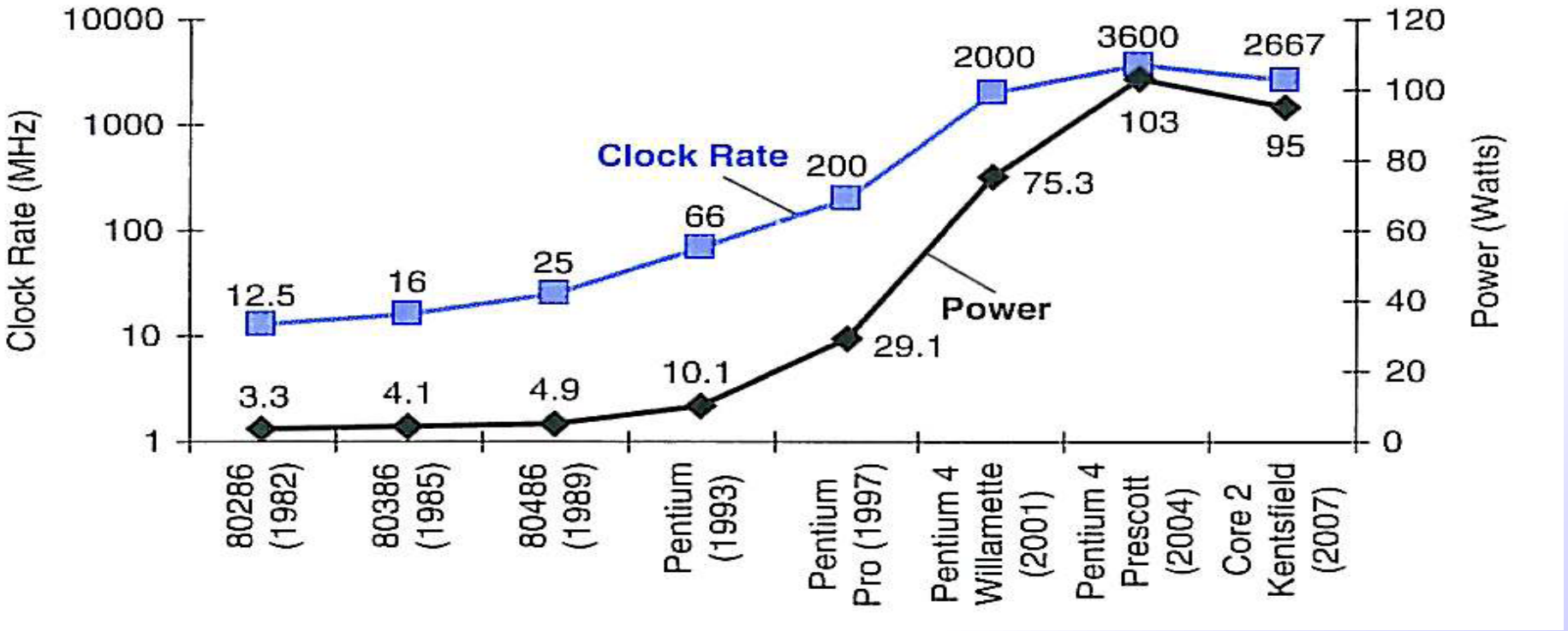
- 100 MHz. machine ; CPI(A)=1, CPI(B)=2, CPI(C)=3.
- C1 --> 5 million A, 1 million B, and 1 million C.
- C2 --> 10 million A, 1 million B, and 1 million C.

- Which sequence will be faster according to MIPS?
- $\#instructions(C1) = 7mil$ $\#instructions(C2) = 12mil$
- $\#cycles(C1) = 10mil$ $\#cycles(C2) = 15mil$
- $CPI(C1) = \frac{10}{7}$ $CPI(C2) = \frac{5}{4}$
- $MIPS(C1) = \frac{100Mhz}{\frac{10}{7}\times 10^6} = 70$ $MIPS(C2) = \frac{100Mhz}{\frac{5}{4}\times 10^6} = 80$

- Which sequence will be faster according to execution time?

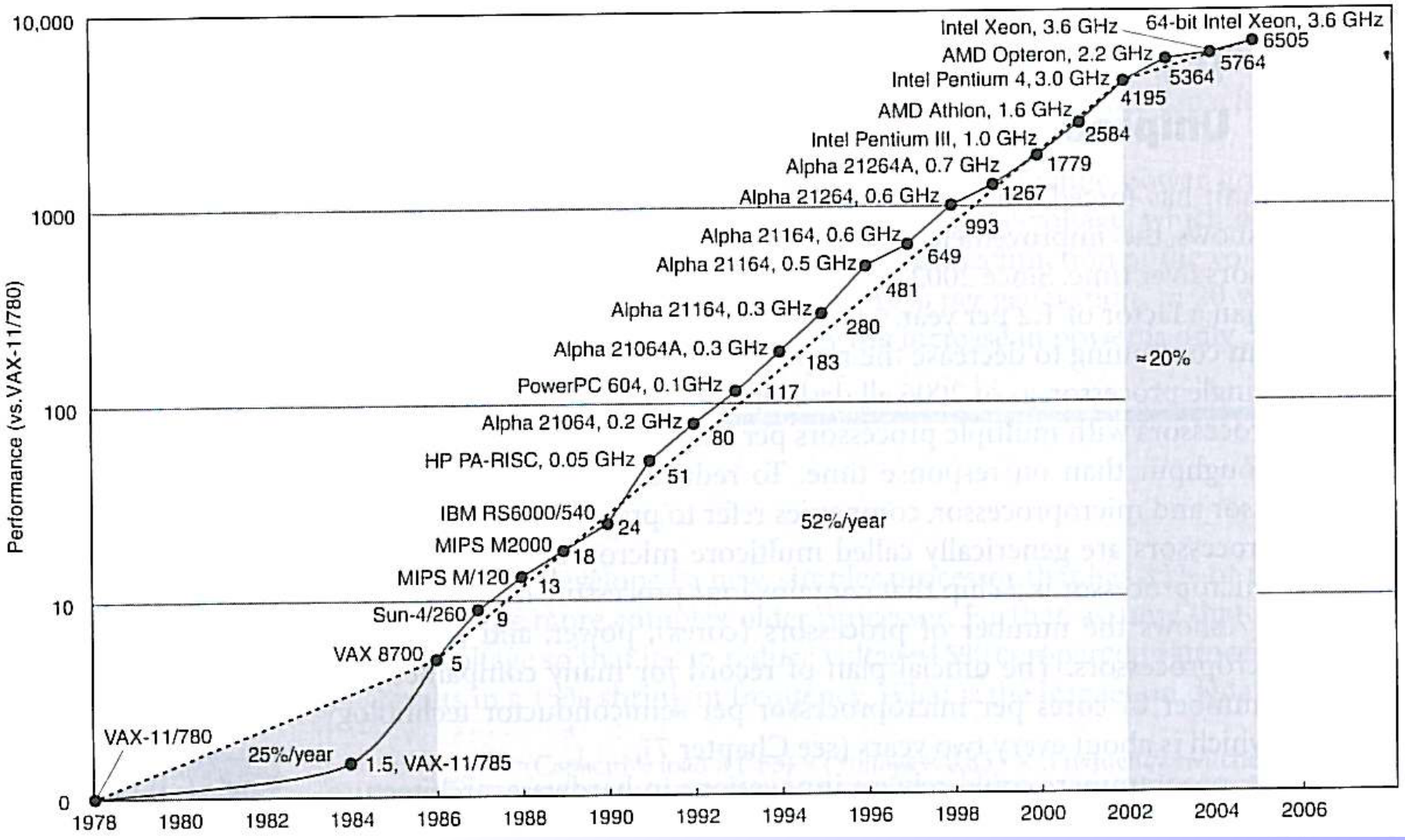
- $ExeTime(C1) = \frac{7\times 106}{70\times 10^6} = 0.1\, sec.$ $ExeTime(C2) = \frac{12\times 106}{80\times 10^6} = 0.15\, sec.$

Clock Rate Vs Power



- La frequenza dei clock e la potenza elettrica richiesta dai microprocessori sono cresciute rapidamente negli anni ma recentemente si sono fermate
- È stata raggiunta la massima potenza dissipabile dai sistemi di raffreddamento montati nei microprocessori

Crescita nelle prestazioni dei processori



I sistemi multicore

- Il limite dell'assorbimento di potenza ha cambiato il modo di progettare i microprocessori
- Dal 2006 quasi tutti i desktop e server hanno microprocessori contenenti più processori sul singolo chip
  - Si migliora più il throughput che il tempo di esecuzione
- Un microprocessore quadcore è un chip che contiene al suo interno quattro core (quattro processori)

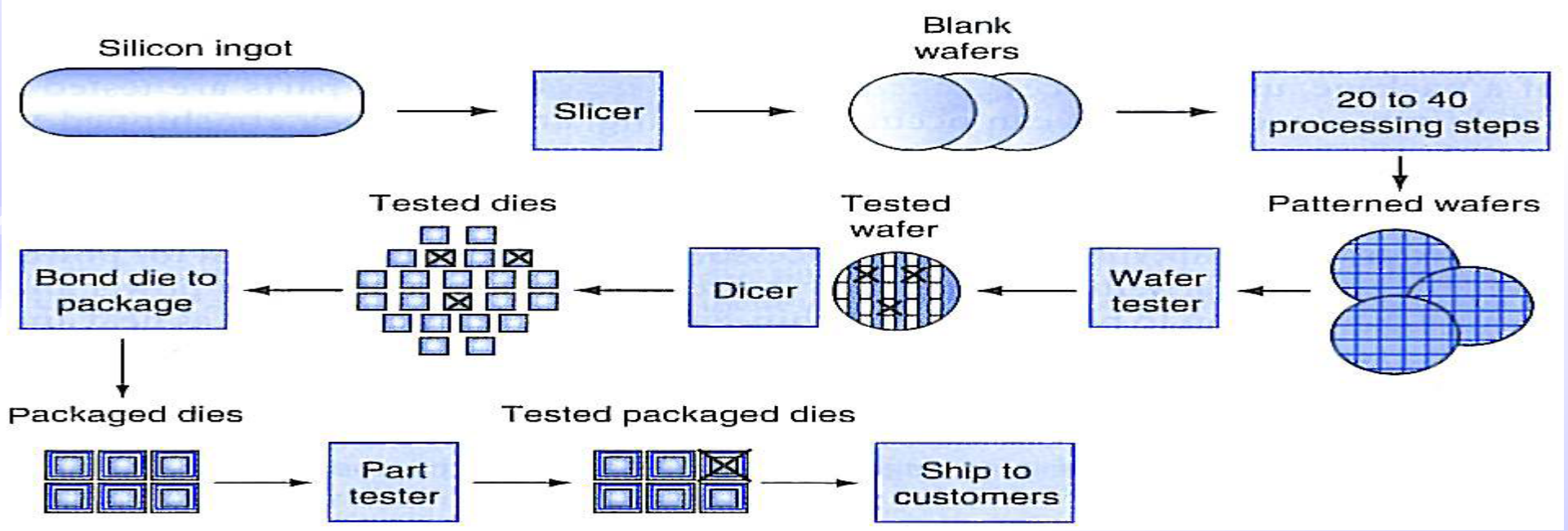
Product	AMD Opteron X4 (Barcelona)	Intel Nehalem	IBM Power 6	Sun Ultra SPARC T2 (Niagara 2)
Cores per chip	4	4	2	8
Clock rate	2.5 GHz	~ 2.5 GHz ?	4.7 GHz	1.4 GHz
Microprocessor power	120 W	~ 100 W ?	~ 100 W ?	94 W

FIGURE 1.17 Number of cores per chip, clock rate, and power for 2008 multicore microprocessors.

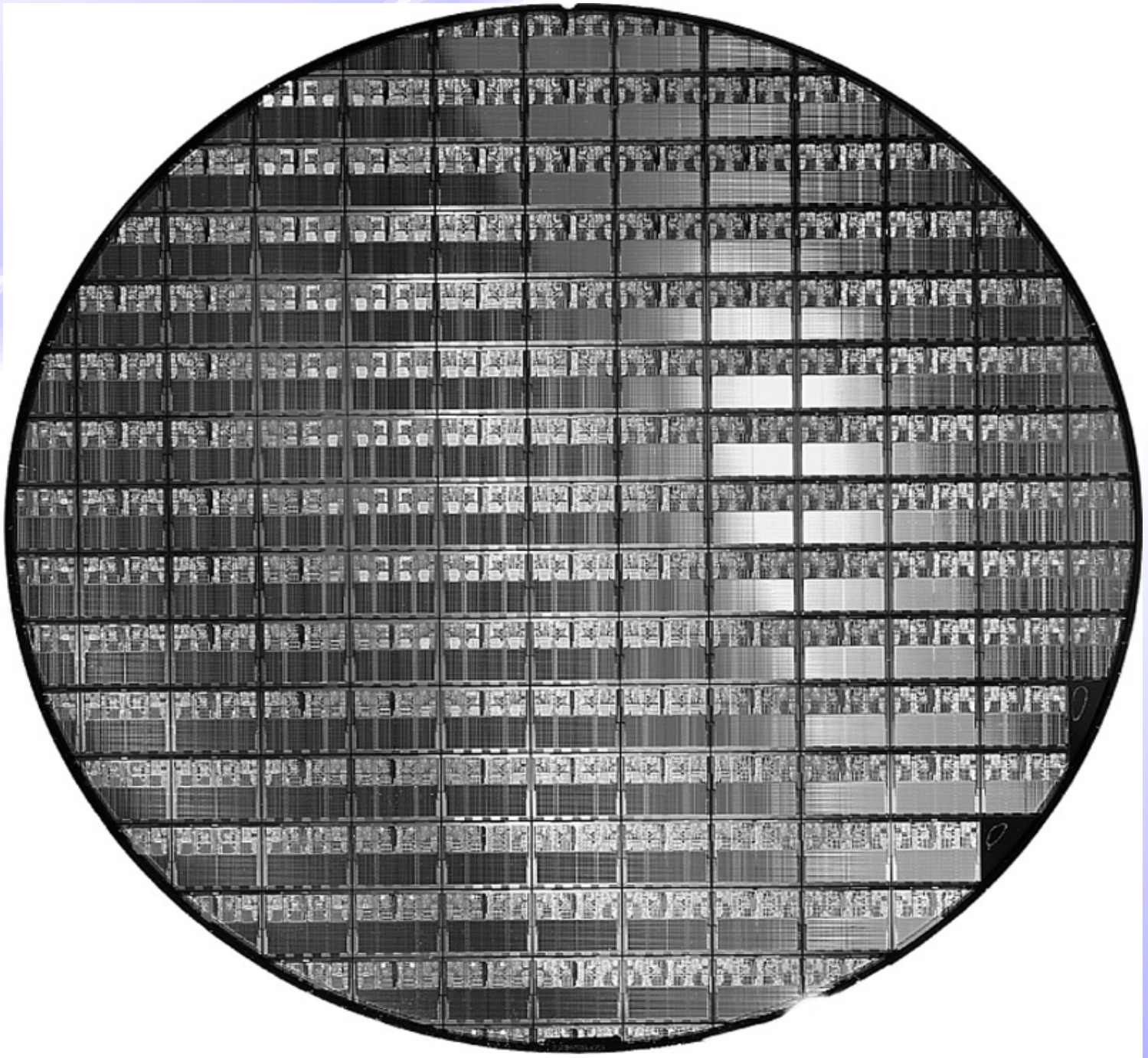
I circuiti integrati

- Un transistor si comporta come un interruttore: aperto/chiuso.
- La tecnologia VLSI (very large-scale integrated circuit) consente di integrare in una superficie minuscola milioni di transistor.
- Si parte dal silicio un semiconduttore, e, con processi chimici si aggiungono materiali che permettono a piccole aree di silicio di comportarsi
  - come un conduttore,
  - come un isolante
  - come un interruttore (transistor)

Il processo produttivo di un chip



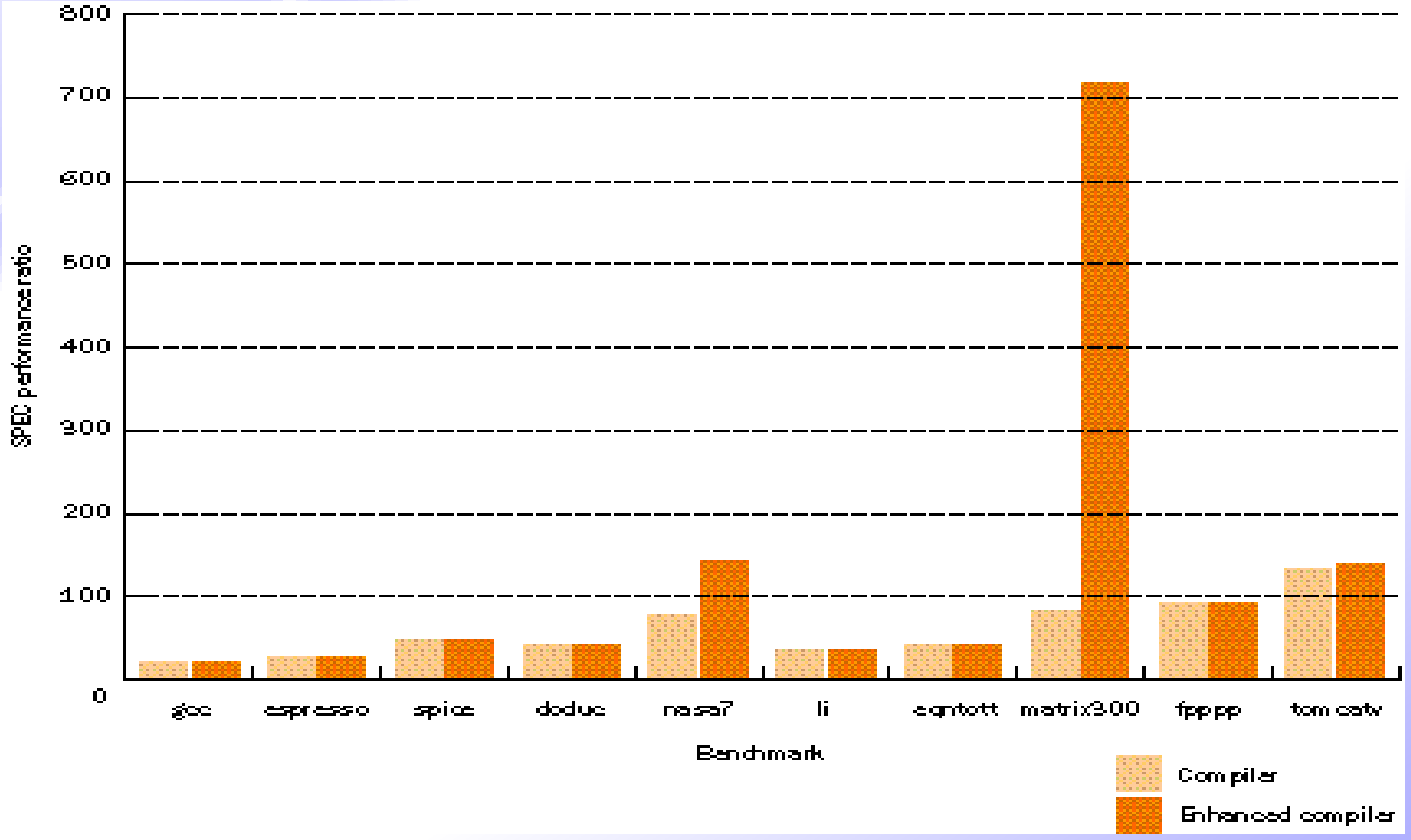
- Dal lingotto di silicio si tagliano fette di 2 mm detti wafer.
- Si attua il processo chimico (soggetto a difetti) e si separano I singoli die.
- Quelli buoni vengono messi in packages e vengono collegati gli ingressi e le uscite. Dopo altri test quelli buoni vengono consegnati



Benchmarks qui

- Performance best determined by running a real application
  - Use programs typical of expected workload
  - Or, typical of expected class of applications e.g., compilers/editors, scientific applications, graphics, etc.
- Small benchmarks
  - nice for architects and designers
  - easy to standardize
  - can be abused
- SPEC (System Performance Evaluation Cooperative)
  - companies have agreed on a set of real program and inputs
  - can still be abused (Intel's "other" bug)
  - valuable indicator of performance (and compiler technology)

SPEC '89 Compiler "enhancements" and performance



SPEC '95

Benchmark	Description
go	Artificial intelligence; plays the game of Go
m88ksim	Motorola 88k chip simulator; runs test program
gcc	The Gnu C compiler generating SPARC code
compress	Compresses and decompresses file in memory
li	Lisp interpreter
ijpeg	Graphic compression and decompression
perl	Manipulates strings and prime numbers in the special-purpose programming language Perl
vortex	A database program
tomcatv	A mesh generation program
swim	Shallow water model with 513 x 513 grid
su2cor	quantum physics; Monte Carlo simulation
hydro2d	Astrophysics; Hydrodynamic Naiver Stokes equations
mgrid	Multigrid solver in 3-D potential field
applu	Parabolic/elliptic partial differential equations
trub3d	Simulates isotropic, homogeneous turbulence in a cube
apsi	Solves problems regarding temperature, wind velocity, and distribution of pollutant
fpppp	Quantum chemistry
wave5	Plasma physics; electromagnetic particle simulation

SPECINTC2006 su AMD Opteron X4

Description	Name	Instruction Count × 10^9	CPI	Clock cycle time (seconds × 10^9)	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2,118	0.75	0.4	637	9,770	15.3
Block-sorting compression	bzip2	2,389	0.85	0.4	817	9,650	11.8
GNU C compiler	gcc	1,050	1.72	0.4	724	8,050	11.1
Combinatorial optimization	mcf	336	10.00	0.4	1,345	9,120	6.8
Go game (AI)	go	1,658	1.09	0.4	721	10,490	14.6
Search gene sequence	hmmer	2,783	0.80	0.4	890	9,330	10.5
Chess game (AI)	sjeng	2,176	0.96	0.4	837	12,100	14.5
Quantum computer simulation	libquantum	1,623	1.61	0.4	1,047	20,720	19.8
Video compression	h264avc	3,102	0.80	0.4	993	22,130	22.3
Discrete event simulation library	omnetpp	587	2.94	0.4	690	6,250	9.1
Games/path finding	astar	1,082	1.79	0.4	773	7,020	9.1
XML parsing	xalancbmk	1,058	2.70	0.4	1,143	6,900	6.0
Geometric Mean							11.7

FIGURE 1.20 SPECINTC2006 benchmarks running on AMD Opteron X4 model 2356 (Barcelona). As the equation on page 35 explains, execution time is the product of the three factors in this table: instruction count in billions, clocks per instruction (CPI), and clock cycle time in nanoseconds. SPECratio is simply the reference time, which is supplied by SPEC, divided by the measured execution time. The single number quoted as SPECINTC2006 is the geometric mean of the SPECratios. Figure 5.40 on page 542 shows that mcf, libquantum, omnetpp, and xalancbmk have relatively high CPIs because they have high cache miss rates.

SPECpower su consumo di potenza

Target Load %	Performance (ssj_ops)	Average Power (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%	92,035	233
30%	70,500	222
20%	47,126	206
10%	23,066	180
0%	0	141
Overall Sum	1,283,590	2,605
Σssj_ops / Σpower =		493

FIGURE 1.21 SPECpower_ssj2008 running on dual socket 2.3 GHz AMD Opteron X4 2356 (Barcelona) with 16 GB Of DDR2-667 DRAM and one 500 GB disk.

SPECpower su consumo di potenza - 2

- L'utilizzo di server va dal 10% al 50%
- Questi test mostrano che un server utilizzato al 10% assorbe circa due terzi della potenza di picco
- Se si riprogettasse l'HW in modo che il lavoro delle macchine fosse proporzionale all'energia consumata si potrebbe diminuire drasticamente il consumo energetico

Server Manufacturer	Micro-processor	Total Cores/ Sockets	Clock Rate	Peak Performance (ssj_ops)	100% Load Power	50% Load Power	50% Load/ 100% Power	10% Load Power	10% Load/ 100% Power	Active Idle Power	Active Idle/ 100% Power
HP	Xeon E5440	8/2	3.0 GHz	308,022	269 W	227 W	84%	174 W	65%	160 W	59%
Dell	Xeon E5440	8/2	2.8 GHz	305,413	276 W	230 W	83%	173 W	63%	157 W	57%
Fujitsu Seimens	Xeon X3220	4/1	2.4 GHz	143,742	132 W	110 W	83%	85 W	65%	80 W	60%

FIGURE 1.22 SPECPower results for three servers with the best overall ssj_ops per watt in the fourth quarter of 2007. The overall ssj_ops per watt of the three servers are 698, 682, and 667, respectively. The memory of the top two servers is 16 GB and the bottom is 8 GB.

Amdahl's Law

$$ExeTime\ After\ Improvement =$$

$$= ExeTimeUnaffected + \frac{ExeTimeAffected}{Amount\ of\ Improvement}$$

- Example:

  "Suppose a program runs in 100 seconds on a machine, with multiply responsible for 80 seconds of this time. How much do we have to improve the speed of multiplication if we want the program to run 4 times faster?"

- How about making it 5 times faster?

- *Principle: Make the common case fast*

Example

- Suppose we enhance a machine making all floating-point instructions run five times faster. If the execution time of some benchmark before the floating-point enhancement is 10 seconds, what will the speedup be if half of the 10 seconds is spent executing floating-point instructions?

- We are looking for a benchmark to show off the new floating-point unit described above, and want the overall benchmark to show a speedup of 3. One benchmark we are considering runs for 100 seconds with the old floating-point hardware. How much of the execution time would floating-point instructions have to account for in this program in order to yield our desired speedup on this benchmark?

Example -- risposta

- Floating-point instructions run five times faster. execution time before the floating-point enhancement is 10 seconds, what will the speedup be if half of the 10 seconds is spent executing floating-point instructions?
  - 6 seconds
- We want to show a speedup of 3. One benchmark we are considering runs for 100 seconds with the old floating-point hardware. How much of the execution time would floating-point instructions have to account for in this program in order to yield our desired speedup on this benchmark?

$$\frac{100}{3} = (100 - x) + \frac{x}{5}$$

$$\frac{4}{5}x = 100 - \frac{100}{3}$$

$$x = \frac{200}{3} \times \frac{5}{4} = \frac{1000}{12} = 83.333\ldots$$

Remember

- Performance is specific to a particular program/s
  - Total execution time is a consistent summary of performance
- For a given architecture performance increases come from:
  - increases in clock rate (without adverse CPI affects)
  - improvements in processor organization that lower CPI
  - compiler enhancements that lower CPI and/or instruction count
- Pitfall: expecting improvement in one aspect of a machine's performance to affect the total performance
- You should not always believe everything you read! Read carefully! (see newspaper articles)