Computer Organization and Architecture

Instructor: Dr. Rushdi Abu Zneit

Slide Sources: Patterson & Hennessy COD book.

Computer architecture vs Computer organization

Computer Architecture is concerned with the way hardware components are connected together to form a computer system.
Computer Organization is concerned with the structure and behaviour of a computer system as seen by the user.
It acts as the interface between hardware and software.
It deals with the components of a connection in a system.
Computer Architecture helps us to understand the functionalities of a system.
Computer Organization tells us how exactly all the units in the system are arranged and interconnected.
A programmer can view architecture in terms of instructions, addressing modes and registers.
Whereas Organization expresses the realization of architecture.

While designing a computer system architecture is considered first.
An organization is done on the basis of architecture.
Computer Architecture deals with high-level design issues.
Computer Organization deals with low-level design issues.
Architecture involves Logic (Instruction sets, Addressing modes, Data types, Cache optimization)
Organization involves Physical Components (Circuit design, Adders, Signals, Peripherals)

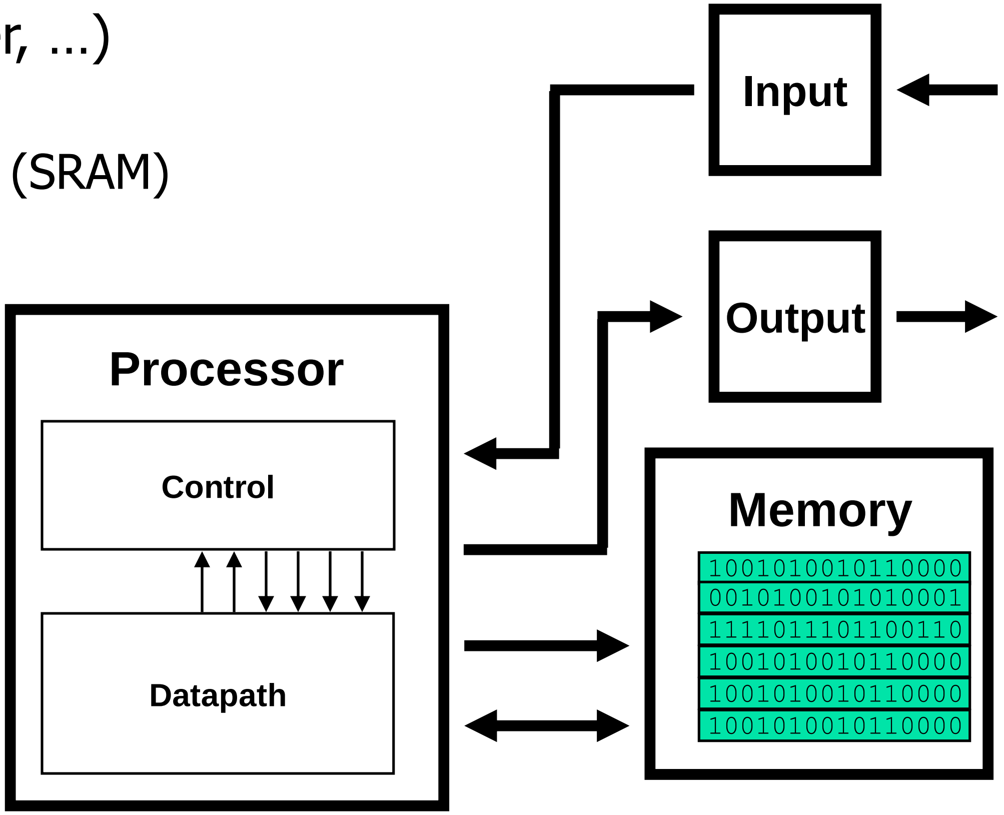
Introduction

- Rapidly changing field:
  - vacuum tube -> transistor -> IC -> VLSI
  - doubling every 1.5 years:
    - memory capacity
    - processor speed (due to advances in technology <u>and</u> hardware organization)
  - cute example: if Boeing had kept up with IBM we could *fly from Bangkok to HCM City in 10 minutes for 5 baht (2000 dong)* !!
- Things we'll be learning:
  - how computers work, what's a good design, what's not
  - how to make them – *yes, we will actually build working computers*!!
  - issues affecting modern processors (e.g., caches, pipelines)

The Five Classic Components of a Computer

- Input (mouse, keyboard, …)
- Output (display, printer, …)
- Memory
  - main (DRAM), cache (SRAM)
  - secondary (disk, CD, DVD, …)
- Datapath
- Control

} Processor (CPU)



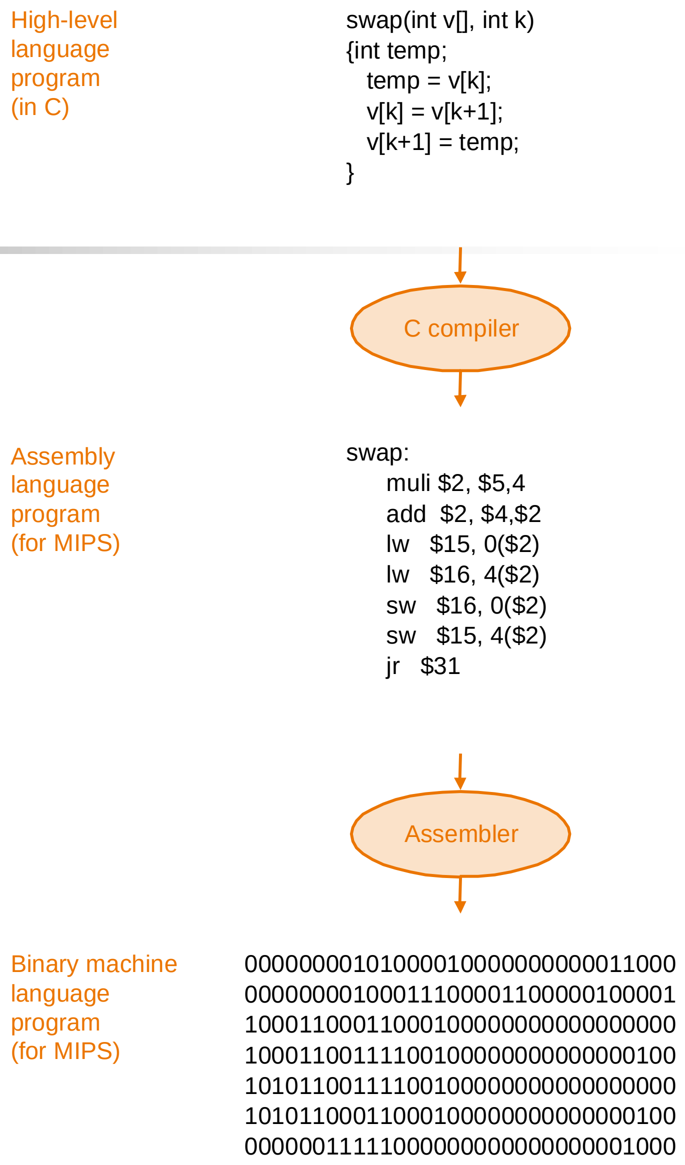
Our Primary Focus

- The processor (CPU)…
  - datapath
  - control
- …implemented using millions of transistors
- …impossible to understand by looking at individual transistors
- we need...

Abstraction

- Delving into the depths reveals more information, but…
- An abstraction omits "unneeded" detail, helps us cope with complexity

- *From the figure on the right, how does abstraction help the programmer and how does she avoid too much detail?*


High-level language program (in C)
C compiler
Assembly language program (for MIPS)
Assembler
Binary machine language program (for MIPS)

```
swap(int v[], int k)
{int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```

```
swap:
      muli $2, $5,4
      add  $2, $4,$2
      lw   $15, 0($2)
      lw   $16, 4($2)
      sw   $16, 0($2)
      sw   $15, 4($2)
      jr   $31
```

```
00000000101000010000000000011000
00000000100011100001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

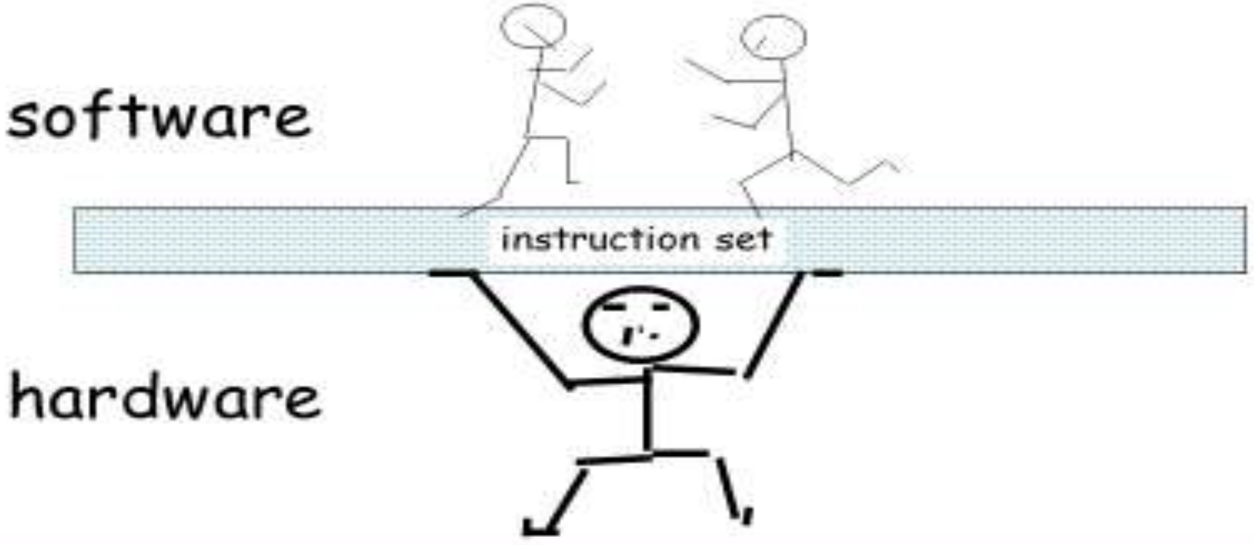
What is *Computer Architecture*

Computer Architecture =

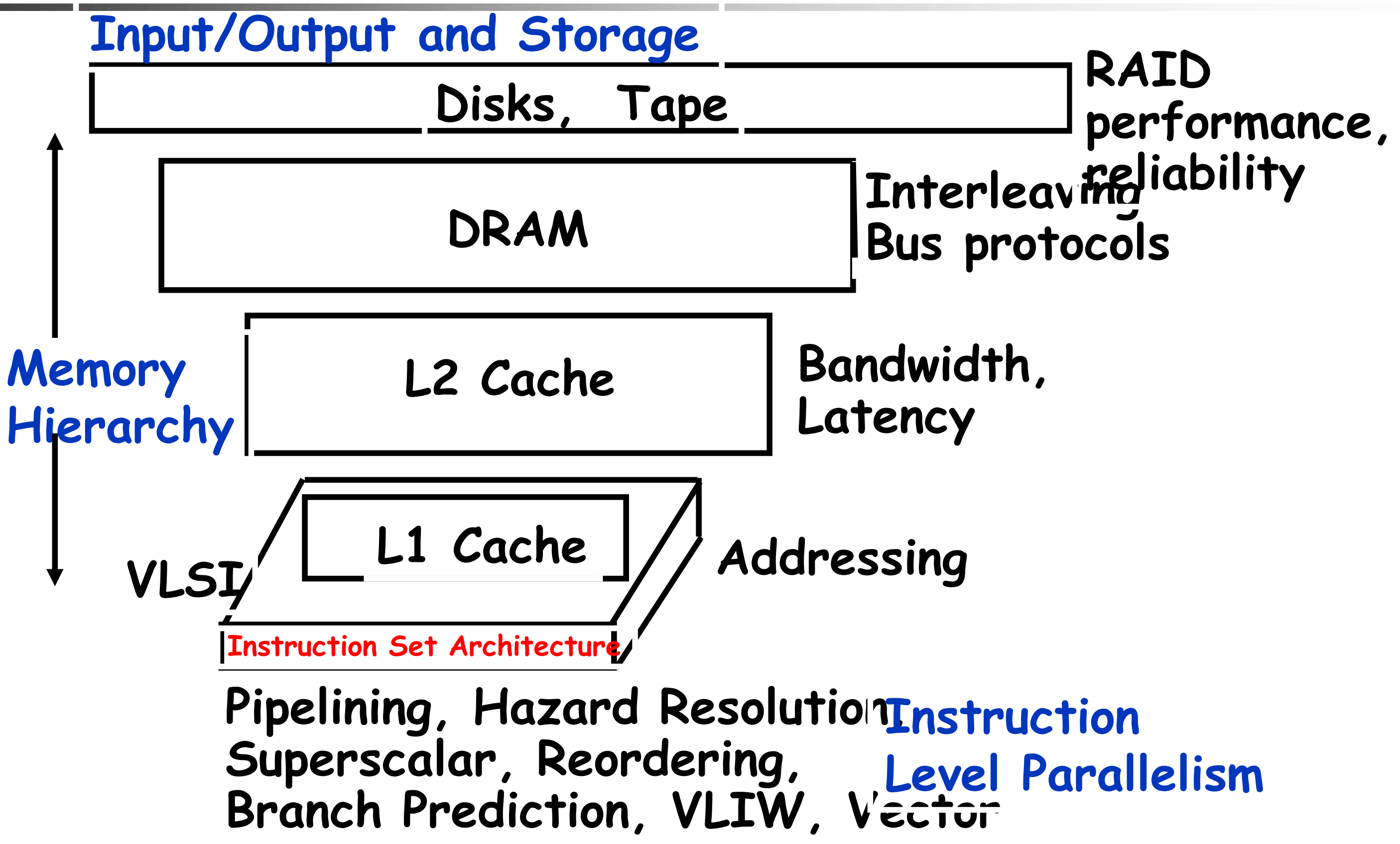
Instruction Set Architecture +

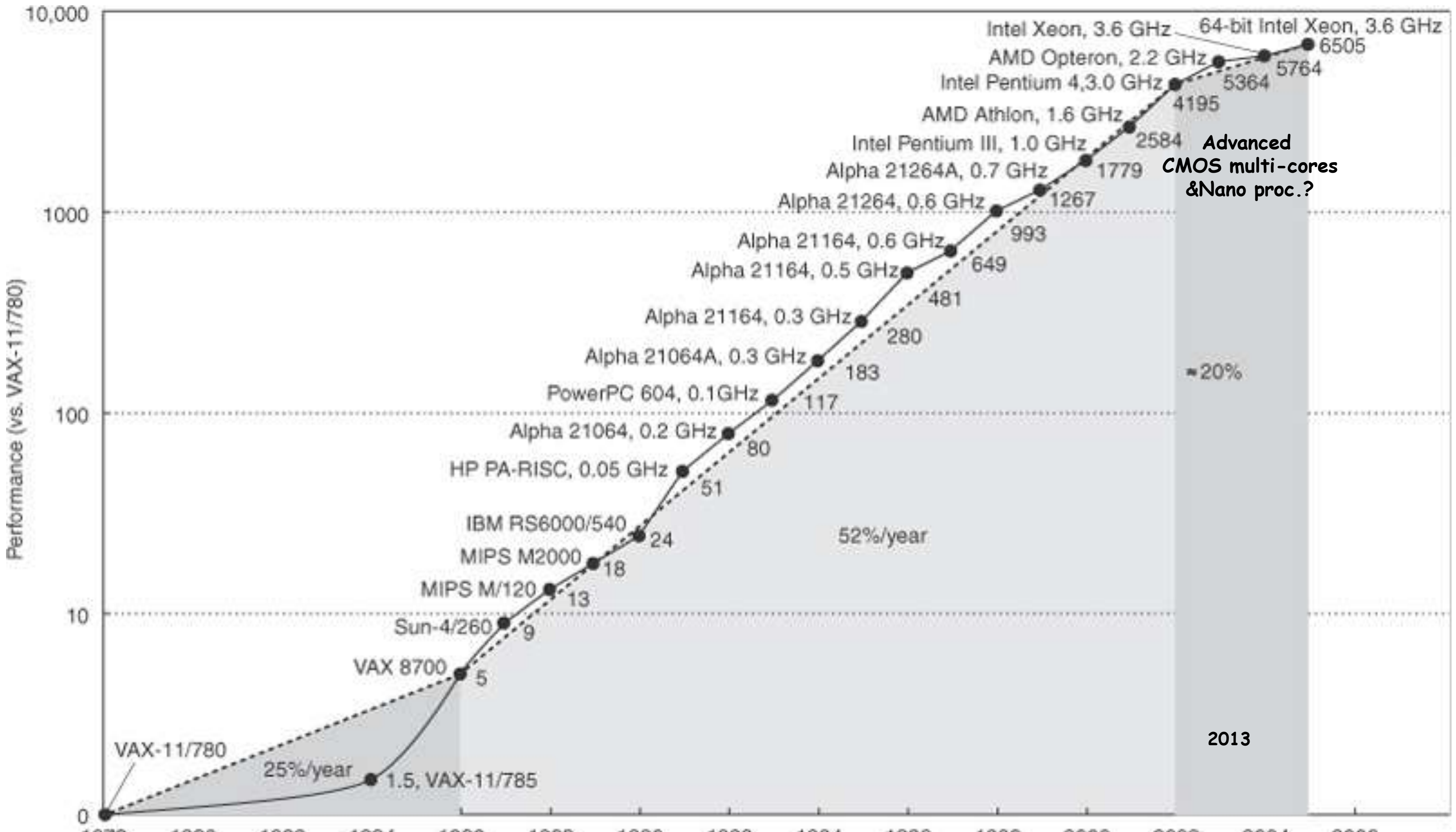
Organization + Hardware + ...

The Instruction Set: a Critical Interface

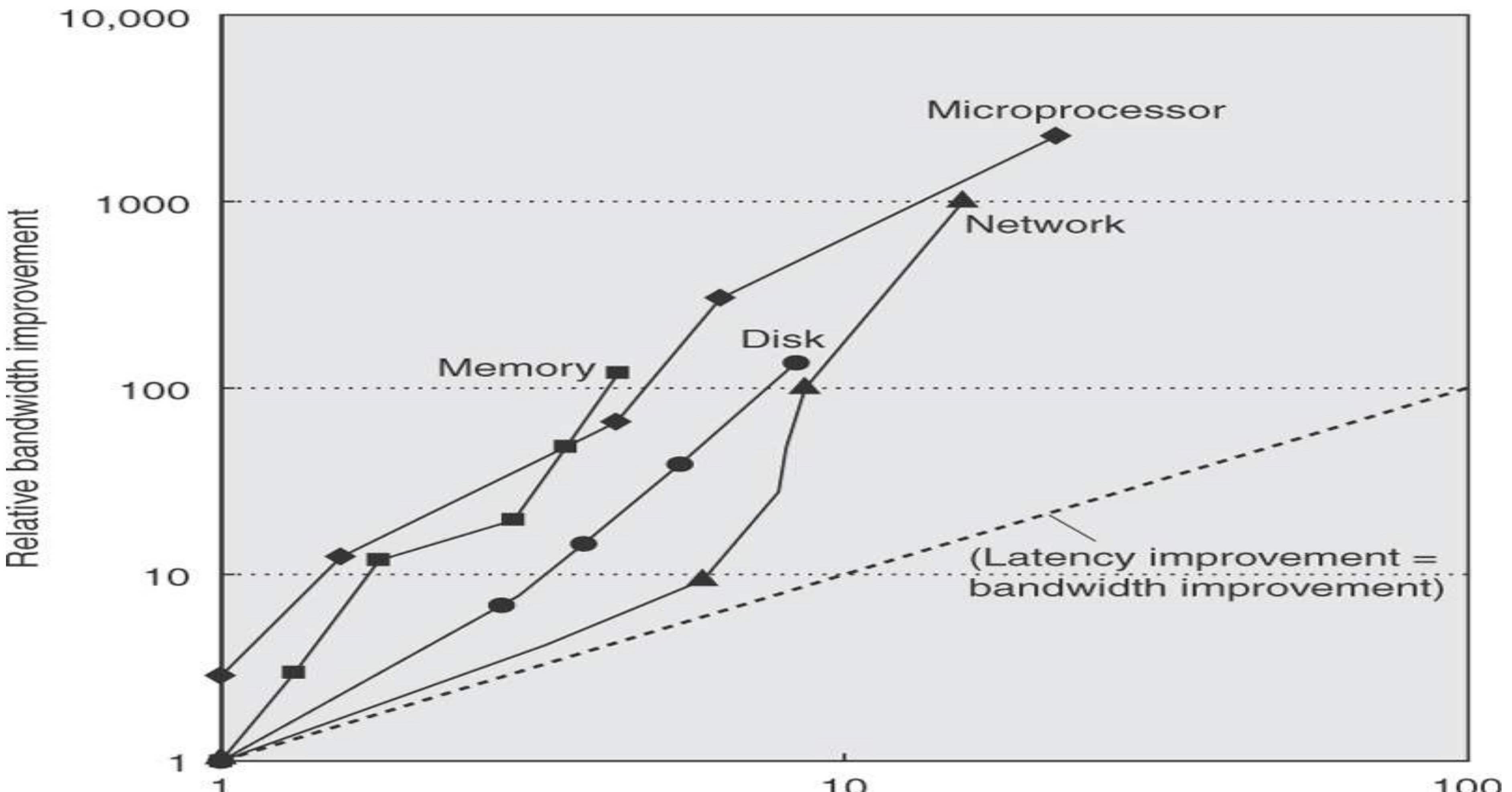


Computer Architecture Topics - Processors


Input/Output and Storage
Disks, Tape
RAID performance, reliability
DRAM
Interleaving
Bus protocols
Memory Hierarchy
L2 Cache
Bandwidth, Latency
L1 Cache
Addressing
VLSI
Instruction Set Architecture
Pipelining, Hazard Resolution, Superscalar, Reordering, Branch Prediction, VLIW, Vector
Instruction Level Parallelism



© 2007 Elsevier, Inc. All rights reserved.


10,000
1000
100
10
1
Relative bandwidth improvement
Microprocessor
Network
Disk
Memory
(Latency improvement =
bandwidth improvement)
1
10
100

Instruction Set Architecture

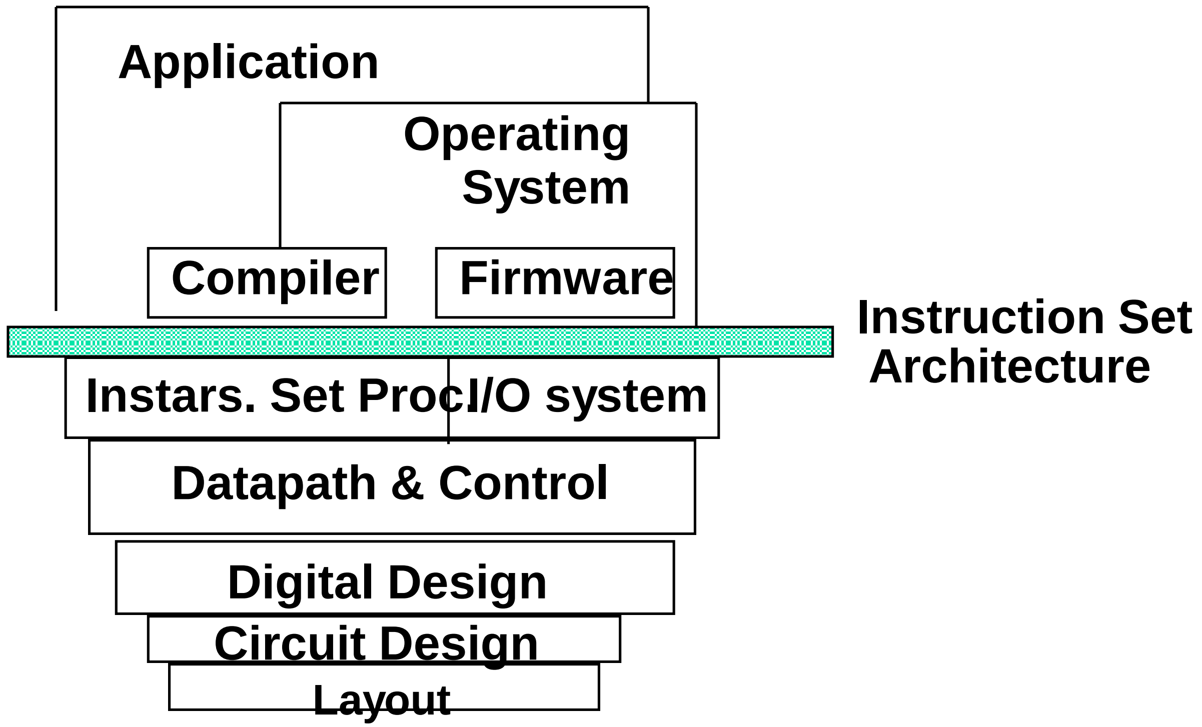
- A very important abstraction:
  - *interface* between hardware and low-level software
  - *standardizes* instructions, machine language bit patterns, etc.
  - advantage: *allows different implementations of the same architecture*
  - disadvantage: *sometimes prevents adding new innovations*

- Modern instruction set architectures:
  - 80x86/Pentium/K6, PowerPC, DEC Alpha, MIPS, SPARC, HP

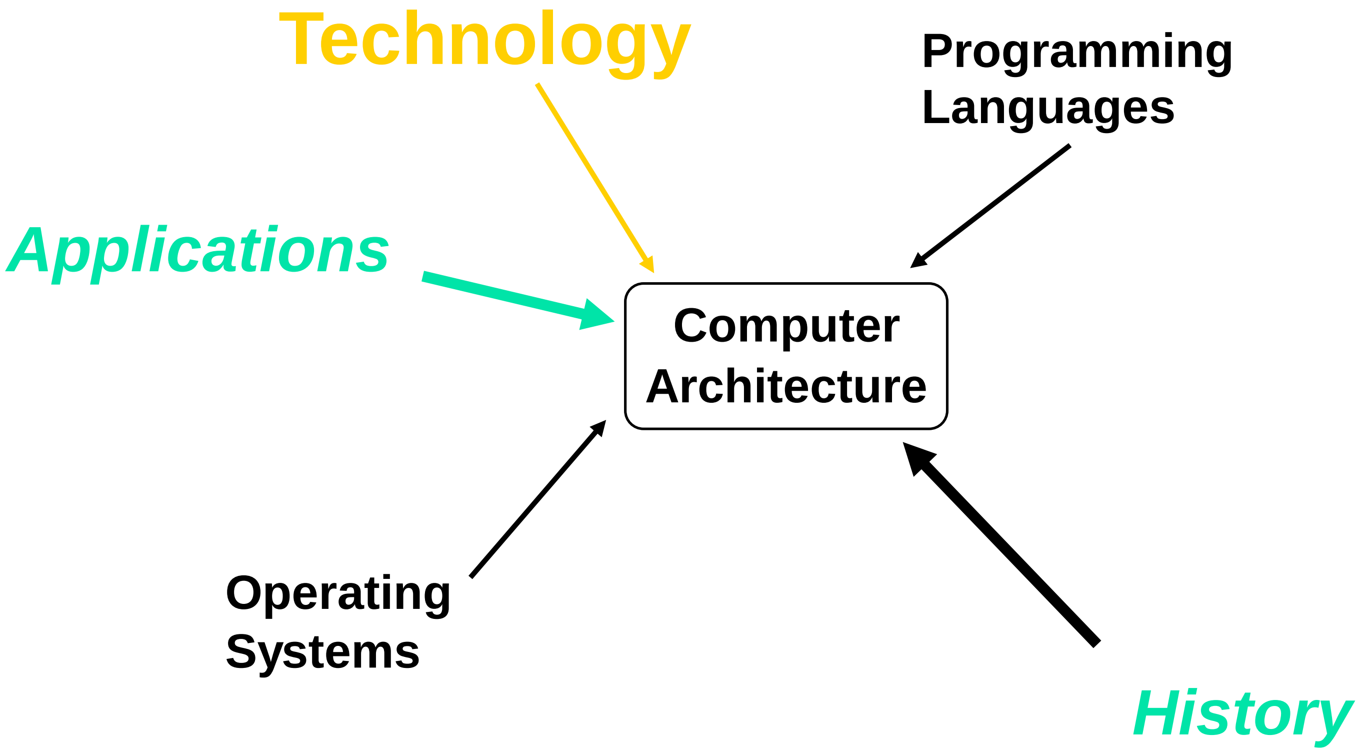
What is Computer Architecture? Easy Answer

Computer Architecture = Instruction Set Architecture + Machine Organization

What is Computer Architecture? Better (More Detailed) Answer



Forces on Computer Architecture



Introduction and Design Principals

-

new set of architecture called RISC "Reduced Instruction Set Computer" in 1980s.

- RISC focused the designers on two performance techniques:

1- Appling of instruction level parallelism using:

a) Pipelining.

b) Multiple instruction issue.

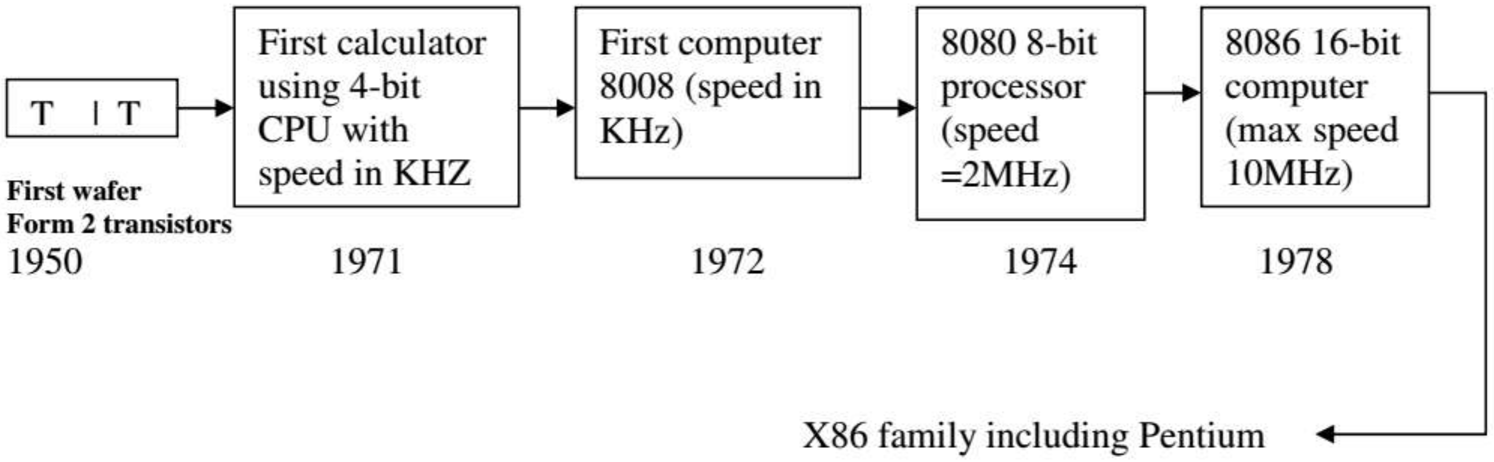
2- Use of cache "simple then complex optimizations"

- This growth led to the dominance of microprocessor based computers:
- Workstations and PCs.
- Minicomputers "traditionally made from off-the shelf logic or from gate arrays" have been replaced by servers made using microprocessors.
- Mainframes replaced with multiprocessors "small number".
- Supercomputers being built with collections of microprocessors.

- These improvements made modern X86 processors consist of a unit to decode X86 instructions and maps them to be executed on a RISC style pipelined processors.

Computer History:

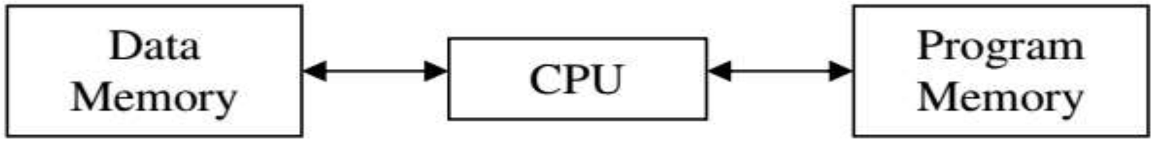
- In 1960 large mainframes stored in computer rooms.
- In 1970 minicomputer birth for scientific use.
- In 1980 rise of desktop computers.
- In 1990 saw the emergence of the internet and the World Wide Web.


T | T
First wafer
Form 2 transistors
1950
First calculator using 4-bit CPU with speed in KHZ
1971
First computer 8008 (speed in KHz)
1972
8080 8-bit processor (speed =2MHz)
1974
8086 16-bit computer (max speed 10MHz)
1978
X86 family including Pentium

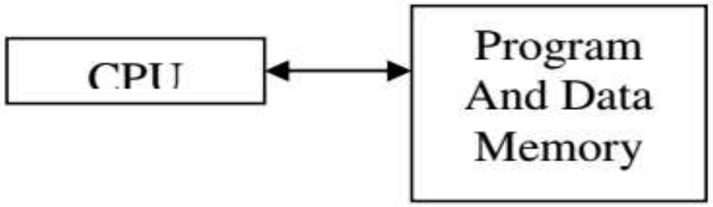
Name	Signification	Transistor count	Number of gates
SSI	small-scale integration	1 to 10	about 10 gates
MSI	medium-scale integration	10 to 500	less than 100 gates
LSI	large-scale integration	500 to 20 000	more than 100gates
VLSI	very large-scale integration	20 000 to 1 000 000	more than 1000 gates

Two types of architecture:

- **Harvard:** used for control circuits (as PLC and PIC microcontroller).



- **Von-Neumann:** used in computers.



- Changes in computers use have led to three different computing markets depending on "applications, requirements, and computing technologies":

1- Desktop computers:
- First largest market and still.
- Optimize price-performance.

Performance:

a) Compute performance.
b) Graphics performance.

- PCs focused on clock rate to measure the performance, which lead to poor decisions.

2- Servers:
- Provide larger scale and computing services.
- Internet growth leads to the need of accelerated improvements in server's performance, which replaced the traditional mainframes.

Servers characteristics:

1- Availability:
The system effectively provide a service by maintain availability using redundancy "availability not mean that the system never fails"

2- Scalability:
Scale up the computing capacity, the memory, the storage, and the I/O bandwidth of a server.

3- Efficient throughput:
The overall performance, which determined by how many requests can be handled in a unit time.

3- Embedded computers:
In: microwaves, washing machines, printers, cell phones (mobiles), network switches, cars, smart cards, video games, ect….

Embedded systems requirements:

- Real time performance.
- Minimize memory (code size).
- Minimize power.

- In the past computer architecture often referred to instruction set design. Now a day it is referred to:

1- Organization:
    - Memory.
    - Bus structure
    - Design of the internal CPU.

2- Hardware:
    - Detailed logic design.
    - Packaging technology.

3- Instruction set design.

- Computer architects must consider:
  1- Functional requirements.
  2- Price
  3- Power
  4- Performance

- Price is what you sell a finished good for. Cost is the amount spent to produce it including overhead.

Lecture 2:
Wafers and Dies and Computer Performance

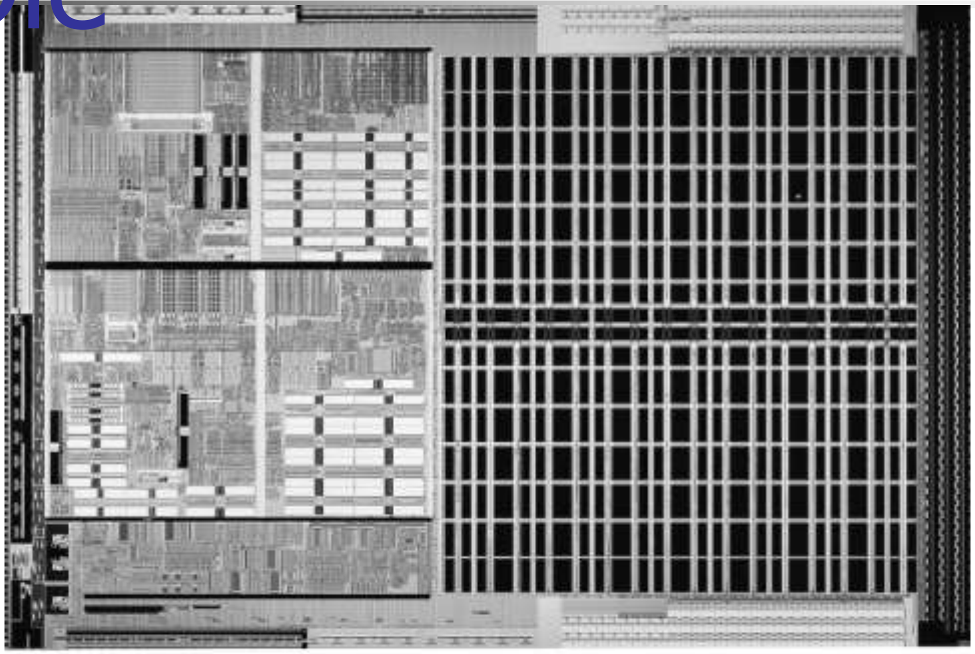
Performance and Cost

- Purchasing perspective
  - given a collection of machines, which has the
    - best performance ?
    - least cost ?
    - best performance / cost ?
- Design perspective
  - faced with design options, which has the
    - best performance improvement ?
    - least cost ?
    - best performance / cost ?
- Both require
  - basis for comparison
  - metric for evaluation

Building Computer Chips : Cost & Timing

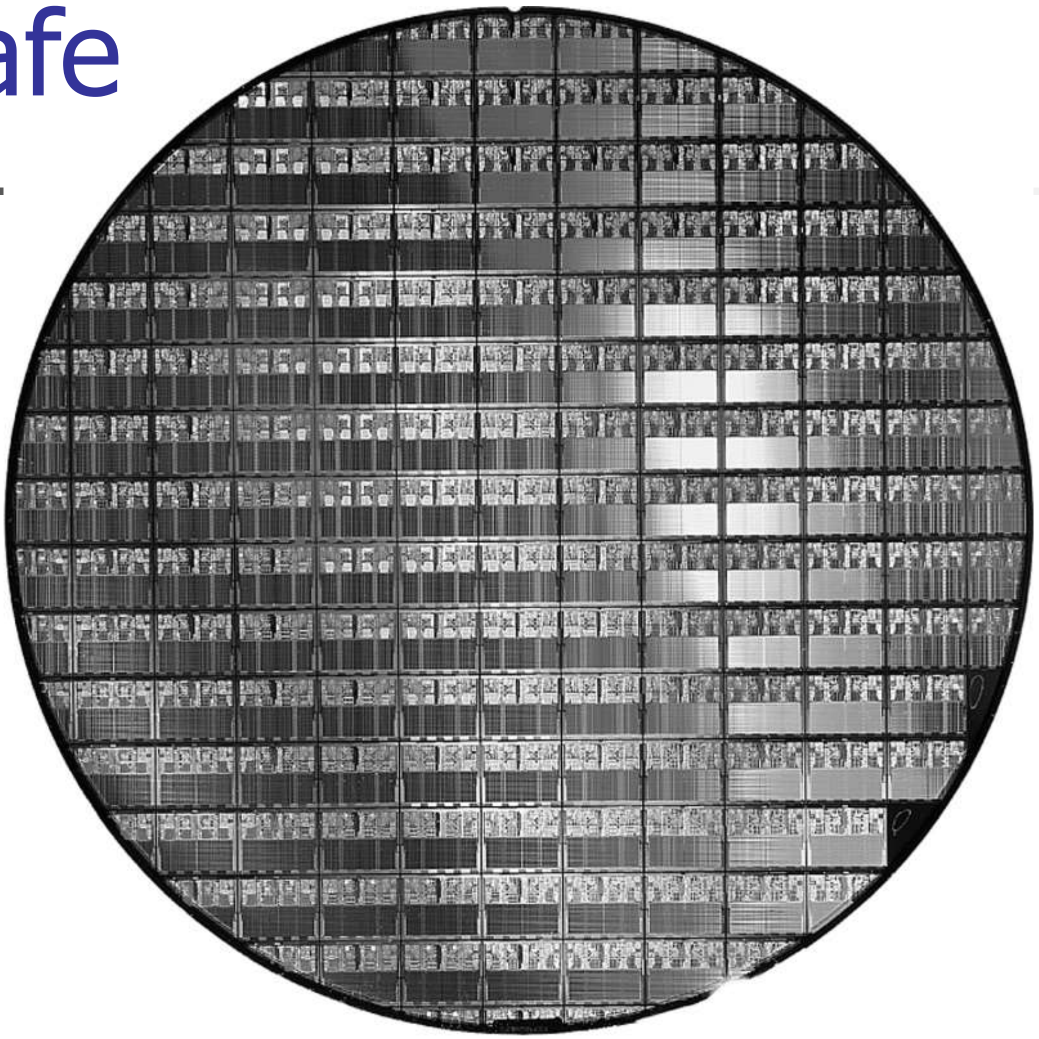
- Complex multi-step process
  - Slice ingots into wafers
  - Process wafers into patterned wafers
  - Dice patterned wafers into dies
  - Test dies, select good dies
  - Bond to package
  - Test parts
  - Ship to customers and make money

Die

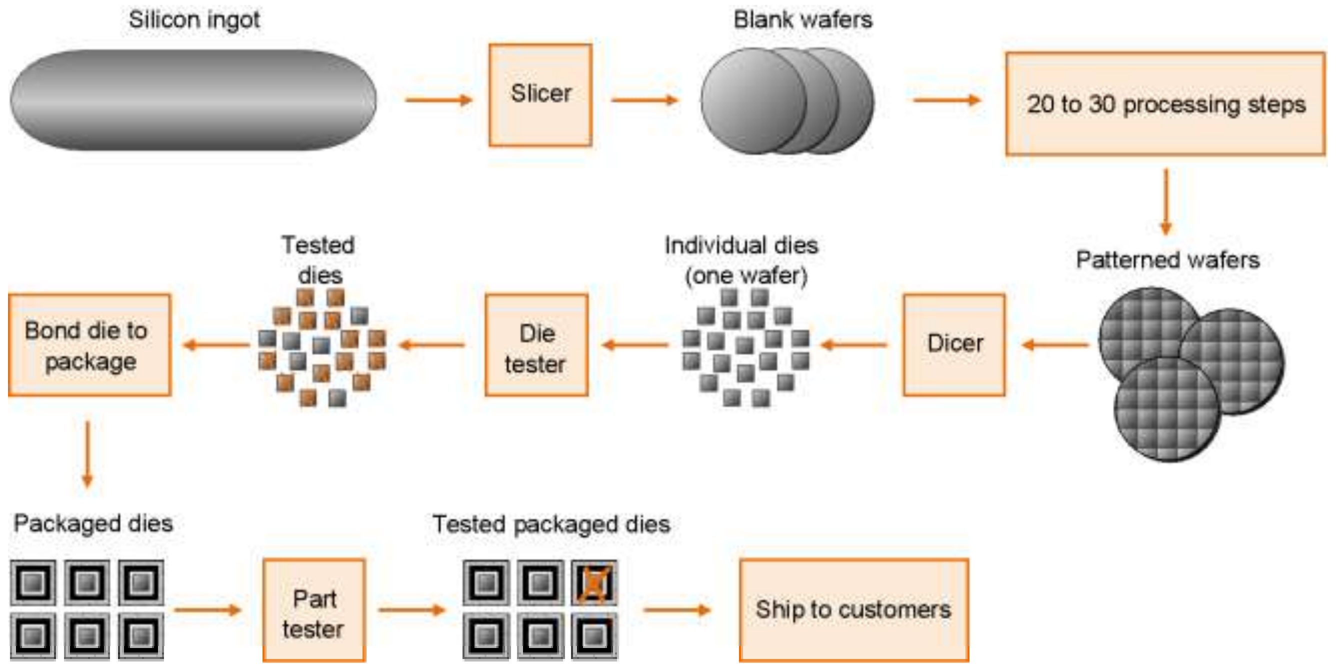


© 2007 Elsevier, Inc. All rights reserved.

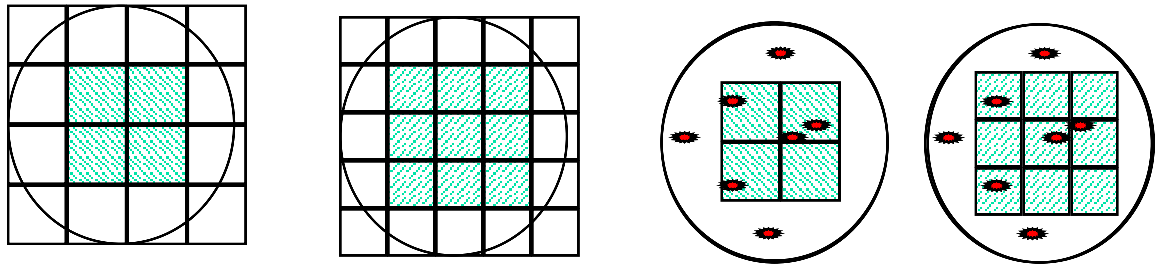
Wafe r


© 2007 Elsevier, Inc. All rights reserved.

Building Computer Chips : Cost



Integrated Circuits Costs



Wafer and IC's

- Cost of integrated circuit depends on its **Volume.**

- IC made using a wafer which tested and chopped into dies that are packaged.

- Cost of IC $= \dfrac{\text{cost of the die} + \text{cost of test die} + \text{cost of packaging}}{\text{Final test yield}}$

- $\text{Cost of Die} = \dfrac{\text{Cost of wafer}}{\text{Dies per wafer} \times \text{Die yield}}$

- Dies per wafer $= \dfrac{\pi \times (\text{Wafer dimeter}/2)^2}{\text{Die area}} - \dfrac{\pi \times \text{Wafer diameter}}{\sqrt{2 \times \text{Die area}}}$

- Note:

1. The first part of the previous equation is **the ratio of wafer area to die area.**

2. The second part of the previous equation is **to eliminate circuit boundary.**

Example:

Find the number of dies per 30cm wafer for a die that is 0.7cm on a side.

$$\text{Die area} = 0.7 \times 0.7 = 0.49\ \text{cm}^2$$

$$\text{Die per wafer} = \frac{\pi\,(30/2)^2}{0.49} - \frac{\pi \times 30}{\sqrt{2 \times 0.49}} = 1347$$

- Last example gives the maximum number of dies per wafer.

- So we need die yield, which is the percentage of good dies on a wafer.

- Assuming defects are randomly distributed:

$$\text{Die yield} = \text{Wafer yield}\left(1 + \frac{\text{Defectes per unit area} \times \text{Die area}}{\alpha}\right)^{-\alpha}$$

- Where wafer yield is 100 %.

- In 2001 defects per unit where between 0.4 and 0.8 per cm^2.

- α is a measure of manufacturing complexity. For today's CMOS processes, a good estimated is α =4.

- Number of good dies = Dies per wafer * Die yield

Example:

Find the die yield for dies that are 1cm on a side and 0.7cm on a side, assuming a defect density of 0.6 per cm².

Total dies areas are 1 cm² and 0.49 cm²

For 1 cm² die:

$$\text{Die yield} = \left(1 + \frac{0.6 \times 1}{4}\right)^{-4} = 0.57$$

For 0.49 cm² die:

$$\text{Die yield} = \left(1 + \frac{0.6 \times 0.49}{4}\right)^{-4} = 0.75$$

**Research and development increase performance

Real World Examples

Chip	Metal	Line	Wafer	Defect	Area	Dies/	Yield	Die Cost
	layers	width	cost	/cm^2	mm^2	wafer		
386DX	2	0.90	$900	1.0	43	360	71%	$4
486DX2 3	0.80	$1200	1.0	81	181	54%	$12	
PowerPC 601	4	0.80	$1700	1.3	121	115	28%	$53
HP PA 7100	3	0.80	$1300	1.0	196	66	27%	$73
DEC Alpha	3	0.70	$1500	1.2	234	53	19%	$149
SuperSPARC	3	0.70	$1700	1.6	256	48	13%	$272
Pentium3	0.80	$1500	1.5	296	40	9%	$417	

- From "Estimating IC Manufacturing Costs," by Linley Gwennap, *Microprocessor Report*, August 2, 1993, p. 15

Case 1: Wafer 1

Diameter of the wafer = 15 cm
Area of a wafer = $3.14 \times 7.5 \times 7.5 = 176.625$

Number of dies per wafer = 84

Hence, area of 1 die = $\frac{176.625}{84} = 2.10 \text{ cm}^2$ (2 decimal places)

$$\text{Yield} = \frac{1}{\left(1 + \text{defects per area} \times \frac{\text{die area}}{2}\right)^2}$$

$$= \frac{1}{\left(1 + 0.02 \times \frac{2.10}{2}\right)^2}$$

$$= \frac{1}{1.021}$$

$$= 0.9794$$

Hence, yield for first wafer = 0.9794

Case 2: Wafer 2

Diameter of the wafer = 20 cm
Area of a wafer = $3.14 \times 10 \times 10 = 314$

Number of dies per wafer = 100

Hence, area of 1 die = $\frac{314}{100}$ =3.14 cm^2 (2 decimal places)

$$\text{Yield} = \frac{1}{\left(1 + \text{defects per area} \times \frac{\text{die area}}{2}\right)^2}$$

$$= \frac{1}{\left(1 + 0.031 \times \frac{3.14}{2}\right)^2}$$

$$= \frac{1}{1.04867}$$

$$= 0.9535$$

Die Area and Cost

- **Processor Area:**

$$\text{Die yield} = \text{Wafer yield} \times \left(1 + \frac{\text{Defects per unit area} \times \text{Die area}}{\alpha}\right)^{-\alpha}$$

- **Example:**

  Find the die yield for dies that are 1.5 cm on a side and 1.0 cm on a side, assuming a defect density of 0.4 per cm ^{2} and α is 4.

- ***Answer:***

  The total die areas are 2.25 cm ^{2} and 1.00 cm ^{2} . For the larger die, the yield is

$$\text{Die yield} = \left(1 + \frac{0.4 \times 2.25}{4.0}\right)^{-4} = 0.44$$

For the smaller die, it is $\text{Die yield} = \left(1 + \frac{0.4 \times 1.00}{4.0}\right)^{-4} = 0.68$

That is, less than half of all the large die are good but more than two-thirds of the small die are good.

Define and quantity dependability

- *Module reliability* = measure of continuous service accomplishment (or time to failure).
  2 metrics
  - *Mean Time To Failure* (*MTTF*) measures Reliability
  - *Failures In Time* (*FIT*) = 1/MTTF, the rate of failures
    - Traditionally reported as failures per billion hours of operation

- *Mean Time To Repair* (*MTTR*) measures Service Interruption
  - *Mean Time Between Failures* (*MTBF*) = MTTF+MTTR
- *Module availability* (MA) measures service as alternate between the 2 states of accomplishment and interruption (number between 0 and 1, e.g. 0.9)
  - *Module availability MA = MTTF / (MTTF + MTTR)*

Example calculating reliability

- If modules have *exponentially distributed lifetimes* (age of module does not affect probability of failure), overall failure rate is the sum of failure rates of the modules
- Calculate FIT and MTTF for 10 disks (1M hour MTTF per disk), 1 disk controller (0.5M hour MTTF), and 1 power supply (0.2M hour MTTF):

$$FailureRate =$$

$$MTTF =$$

Example calculating reliability

- If modules have *exponentially distributed lifetimes* (age of module does not affect probability of failure), overall failure rate is the sum of failure rates of the modules
- Calculate FIT and MTTF for 10 disks (1M hour MTTF per disk), 1 disk controller (0.5M hour MTTF), and 1 power supply (0.2M hour MTTF):

$$
\begin{aligned}
FailureRate &= 10 \times (1/1{,}000{,}000) + 1/500{,}000 + 1/200{,}000 \\
&= 10 + 2 + 5/1{,}000{,}000 \\
&= 17/1{,}000{,}000 \\
&= 17{,}000 FIT \\
MTTF &= 1{,}000{,}000{,}000/17{,}000 \\
&\approx 59{,}000 hours
\end{aligned}
$$

PERFORMANCE:

-A desktop is faster when a program runs in less time. ((Execution time: time between start and complete event))

-A large server is faster when it completes more jobs in an hour. ((Through put))

-If X faster than Y by N.
→Execution time Y / Execution time X=N

→N=(1/performance Y)/(1/performance X)
=performance X/performance Y

* Increases performance decreases execution time.

_ Execution time (response time, elapsed time): this is the latency to complete a task, including disk access, memory access, input\output activities, operating system over head

* CPU time can be divided into:
_User CPU time: CPU time spent in the program
_System CPU time: CPU time spent in the operating system tasks requested by the program.

* System performance: the elapsed time on an unloaded system.

* CPU performance: user CPU time an unloaded system.

Performance evaluation:
_Using programs benchmarks to evaluate the performance

_SPEC (Standard Performance Evaluation Corporation)
Create standardized benchmark application suites. To deliver better benchmarks for workstations.
1- Desktop benchmarks.
2- Server benchmarks.
3- Embedded benchmarks.

Timing is also Important: Performance vs. Design Time

- Time to market is critically important
- E.g., a new design may take 3 years
  - It will be 3 times faster
  - But if technology improves 50%/year
  - In 3 years $1.5^3 = 3.38$
  - So the new design is worse!

    (unless it also employs new technology)