Advanced Computer Architecture

Instructor

Dr. Dinh-Duc Anh-Vu

http://www.cse.hcmut.edu.vn/~anhvu

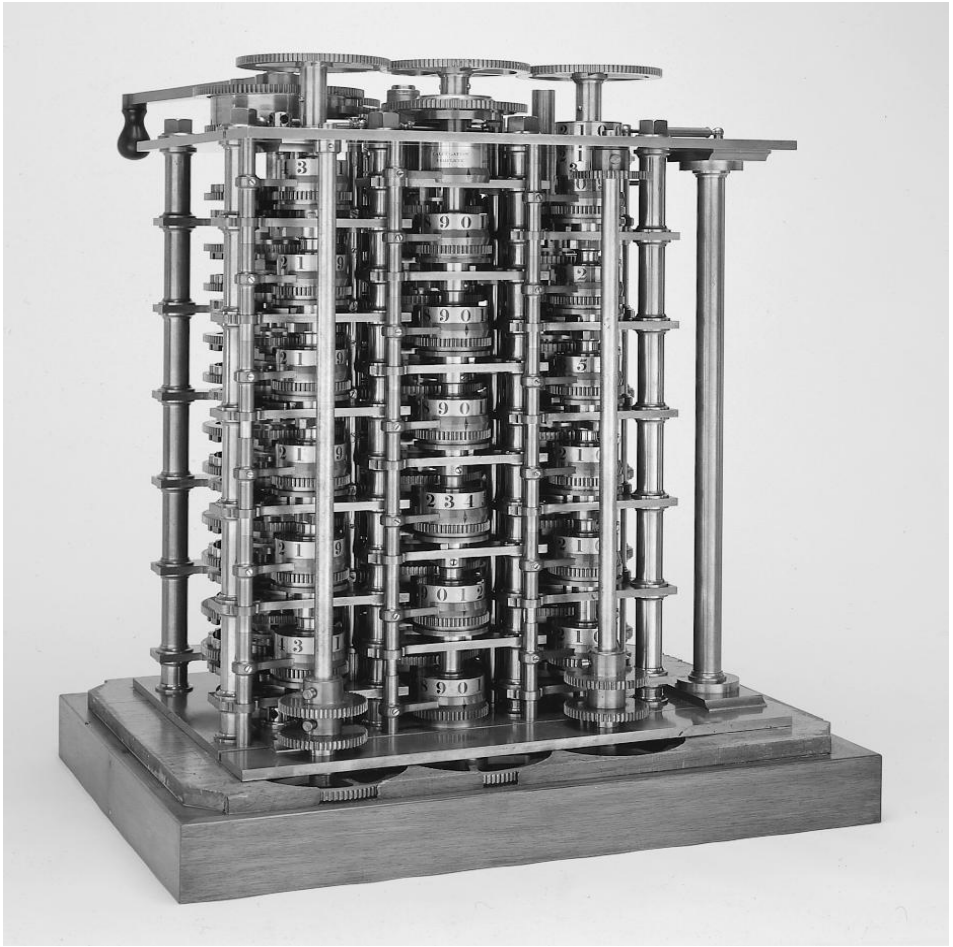
Chapter 1

COMPUTER ARCHITECTURE

Lecture 1 – Computer Architecture

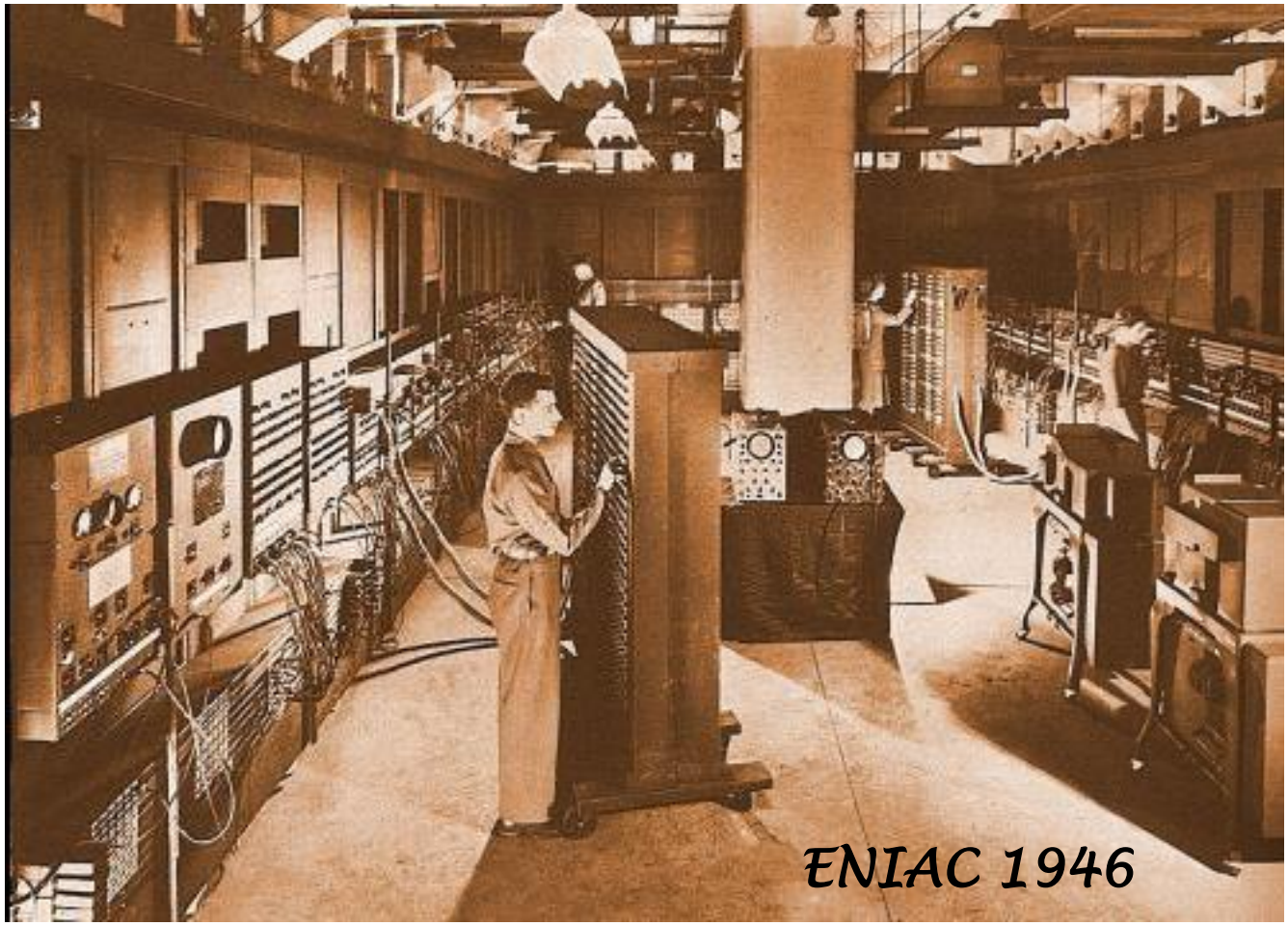
- History of Computer
- CPU and Instruction Execution
- Fundamental of Computer Design
- Designing for Performance

First Computer



The Babbage Difference Engine (1832)
25,000 parts
Cost: £17,470

1st Generation of Computers



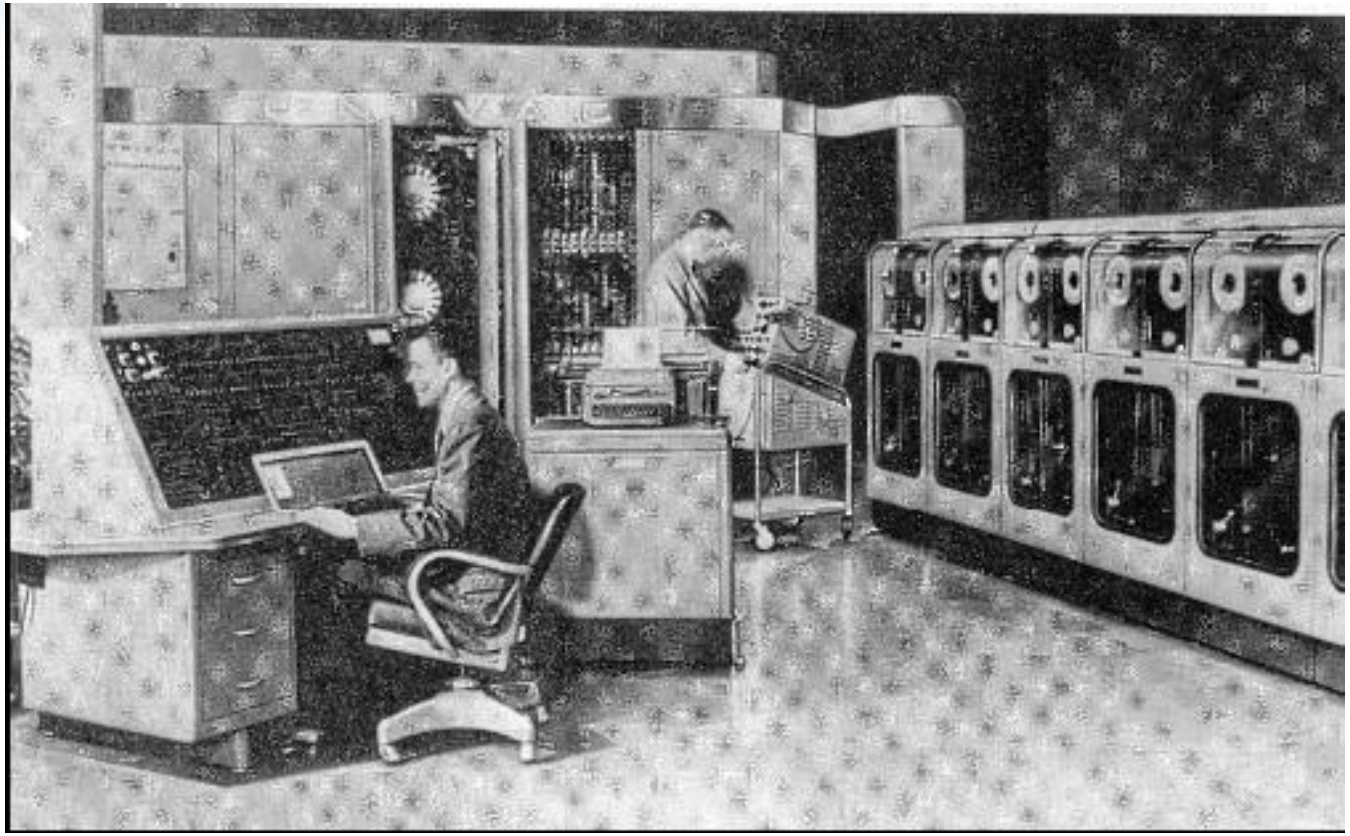
ENIAC – background

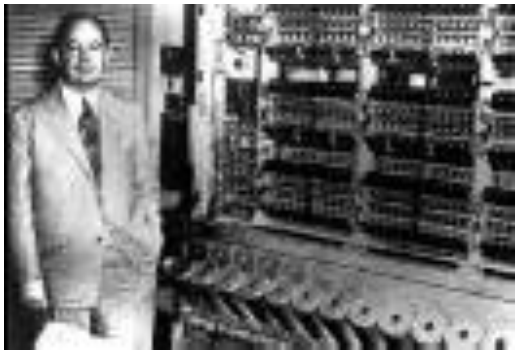
- Electronic Numerical Integrator And Computer
- John Eckert and John Mauchly
- University of Pennsylvania
- Trajectory tables for weapons
- Started 1943
- Finished 1946
  - Too late for war effort
  - Help determine the feasibility of the hydrogen bomb
- Used until 1955

ENIAC – details

- Decimal (not binary)
- 20 accumulators of 10 digits
- Programmed manually by switches
- 18,000 vacuum tubes
- 30 tons
- 15,000 square feet
- 140 kW power consumption
- 5,000 additions per second

von Neumann Machine



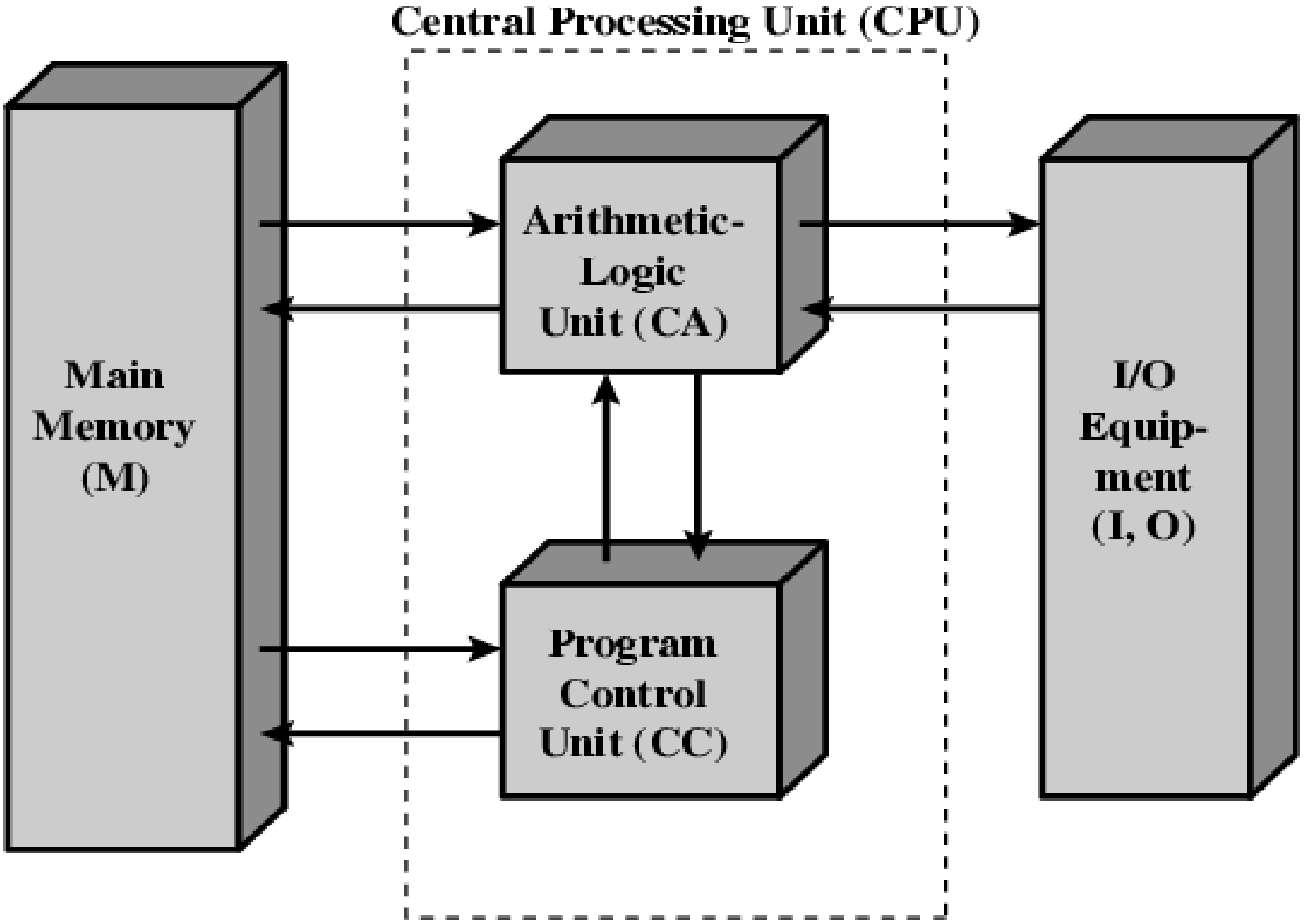


von Neumann/Turing

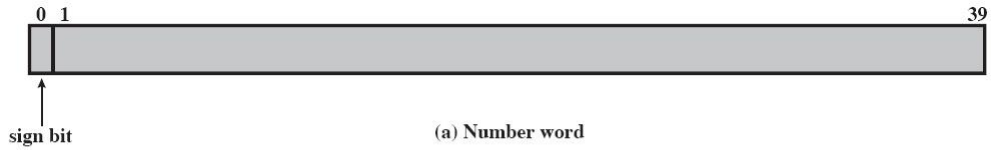
- Stored Program concept
- Princeton Institute for Advanced Studies
  - IAS
  - Began 1946 and completed 1952
- Main memory storing programs and data
- ALU operating on binary data
- Control unit interpreting instructions from memory and executing
- Input and output equipment operated by control unit

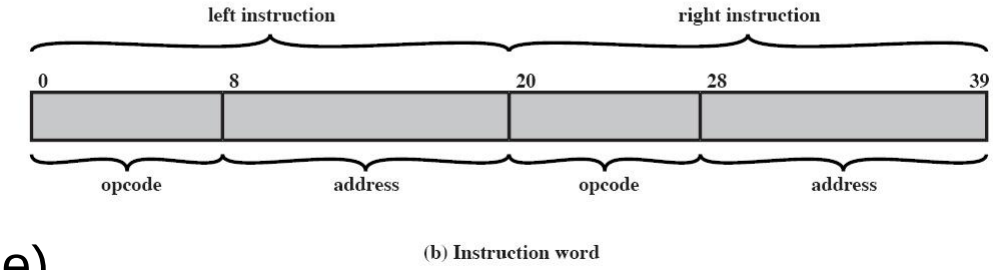
- Virtually all modern computer designs are based on the von Neumann architecture principles:
  - Data and instructions are stored in a single read/write memory.
  - The contents of this memory are addressable by location, without regard to what are stored there.
  - Instructions are executed sequentially (from one instruction to the next) unless the order is explicitly modified.
  - More detail later

Structure of von Neumann machine

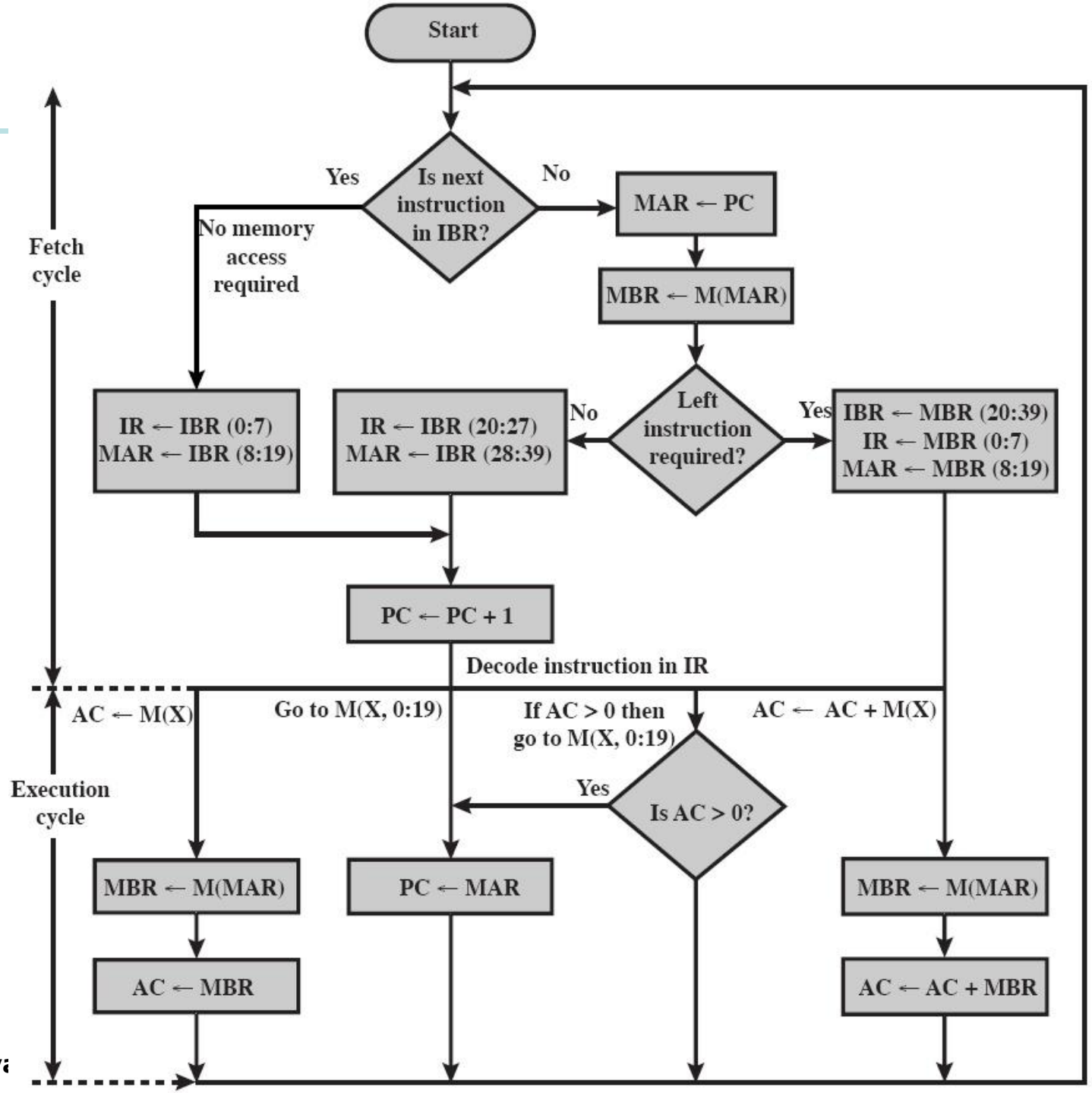


IAS – details (1)





- 1000 x 40 bit words
  - Binary number
  - 2 x 20 bit instructions
    - 8-bit operation code (opcode)
    - 12-bit address designating one word in memory (from 0 to 999)
- Set of registers (storage in CPU)
  - Memory Buffer Register (MBR)
  - Memory Address Register (MAR)
  - Instruction Register (IR)
  - Instruction Buffer Register (IBR)
  - Program Counter (PC)
  - Accumulator (AC)
  - Multiplier Quotient (MQ)


Start
Is next instruction in IBR?
Yes
No
No memory access required
MAR ← PC
MBR ← M(MAR)
Left instruction required?
No
Yes
IR ← IBR (0:7)
MAR ← IBR (8:19)
IR ← IBR (20:27)
MAR ← IBR (28:39)
IBR ← MBR (20:39)
IR ← MBR (0:7)
MAR ← MBR (8:19)
PC ← PC + 1
Decode instruction in IR
Fetch cycle
Execution cycle
AC ← M(X)
Go to M(X, 0:19)
If AC > 0 then go to M(X, 0:19)
AC ← AC + M(X)
Is AC > 0?
Yes
MBR ← M(MAR)
AC ← MBR
PC ← MAR
MBR ← M(MAR)
AC ← AC + MBR

IAS – details (2)

- 21 instructions can be grouped as follows
  - **Data transfer**: move data between memory and ALU registers or between two ALU registers
  - **Unconditional branch**: allows instructions to be executed repetitively
  - **Conditional branch**: the branch can be made dependent on a condition, thus allowing decision points
  - **Arithmetic**: operations performed by the ALU
  - **Address modify**: permits addresses to be computed in the ALU and then inserted into instructions stored in memory. This allows a program considerable addressing flexibility

Why von Neumann Architecture?

- General-purpose, programmable.
  - They can solve very different problems by executing different programs
- Instruction execution is done automatically.
- It can be built with very simple electronics components:
  - Data processing function is performed by gates.
  - Data storage function is provided by memory cells.
  - Data communication is achieved by electrical wires.

Commercial Computers

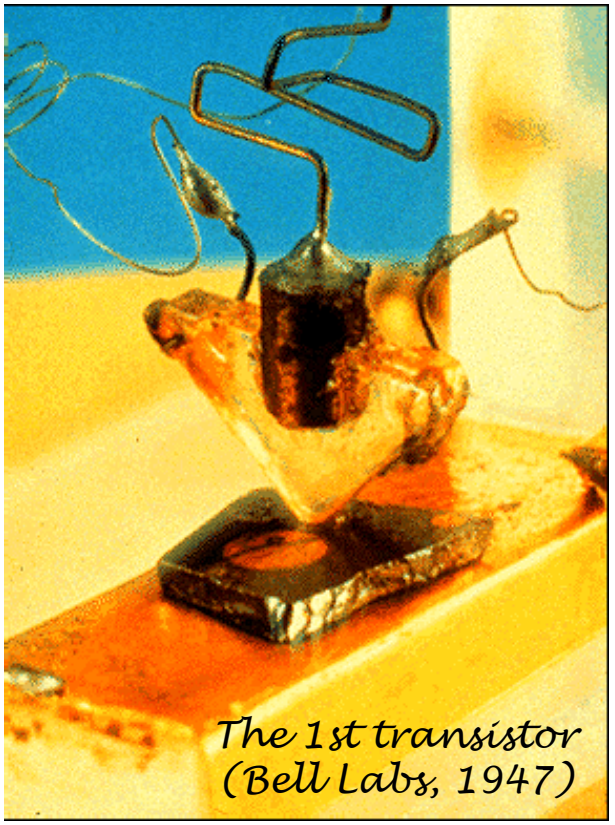
- 1947 – Eckert-Mauchly Computer Corporation
- UNIVAC I (Universal Automatic Computer)
- US Bureau of Census 1950 calculations
- Became part of Sperry-Rand Corporation
- Late 1950s – UNIVAC II
  - Faster
  - More memory
- Trends of the computer industry
  - Advances in technology allow to build larger, more powerful computers
  - Upward compatible with older machines
- Distinction in computer machines
  - Scientific app. oriented
  - Business app. oriented

IBM

- Punched-card processing equipment
- 1953 – the 701
  - IBM's first stored program computer
  - Scientific calculations
- 1955 – the 702
  - Business applications
- Lead to 700/7000 series

Transistors

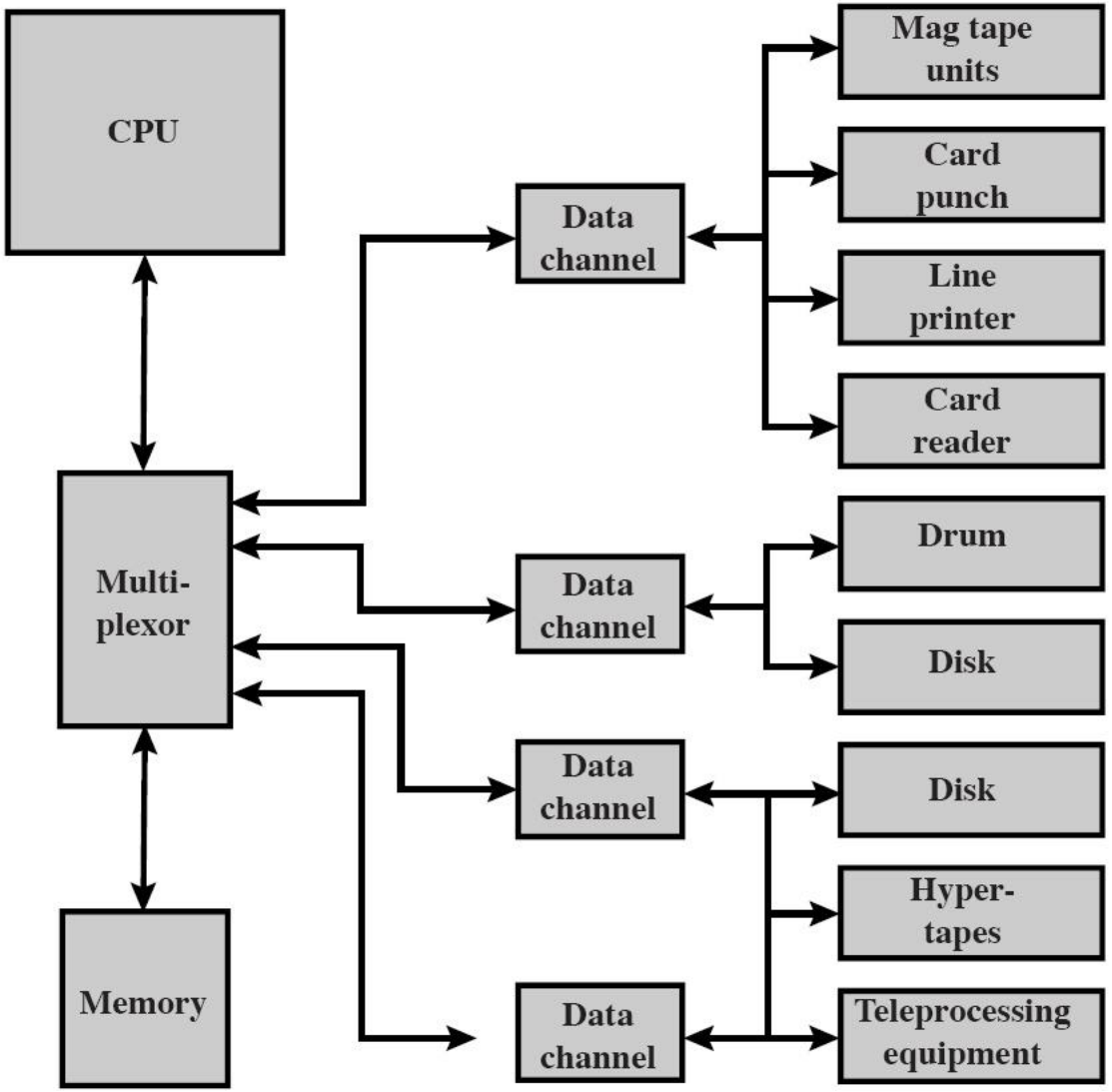
- Replaced vacuum tubes
- Smaller
- Cheaper
- Less heat dissipation
- Solid-state device
- Made from Silicon (sand)
- Invented 1947 at Bell Labs
- William Shockley et al.



2nd Generation of Computers

- Transistor-based Computers
- NCR & RCA produced small transistor machines
- IBM with 7000 series
- Introduction of more complex arithmetic and logic units and control units, the use of high-level programming languages and the provision of system software with the computer
- DEC – 1957
  - Produced PDP-1

IBM 7094 Configuration

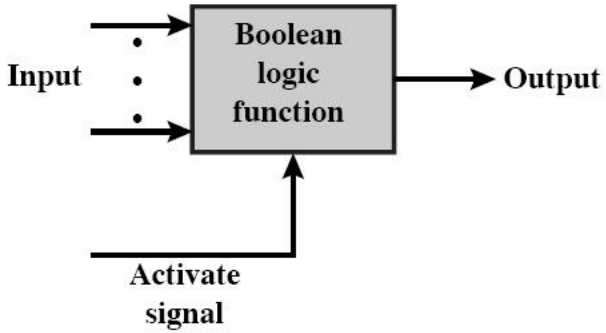


3rd Generation of Computers

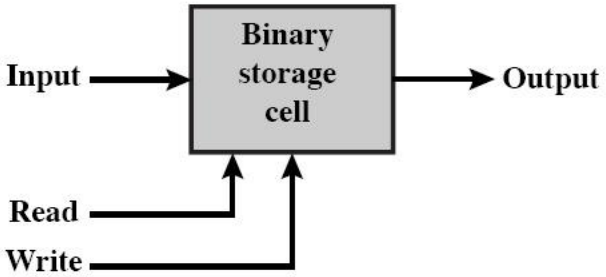
- Number of discrete components grew to the hundreds of thousands, making the manufacture of newer, more powerful machines increasingly difficult.
- 1958 – invention of the IC: era of microelectronics

Microelectronics

- Literally – “small electronics”
- Only 2 fundamental types of components are required to perform 4 basic functions of a digital computer: gates and memory cells
  - Data storage: provided by memory cells
  - Data processing: provided by gates
  - Data movements: the paths between components are used
  - Control: the paths between components can carry control signals
- A computer is made up of gates, memory cells and interconnections
- These can be manufactured on a semiconductor
  - e.g. silicon wafer

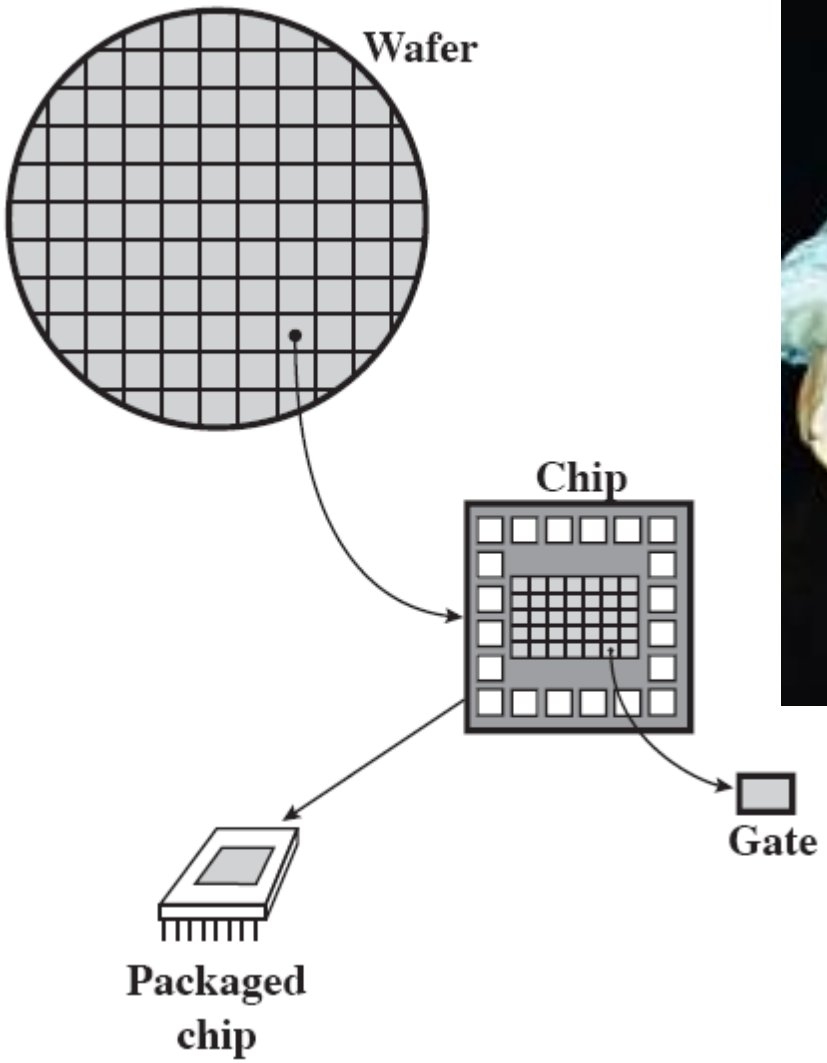


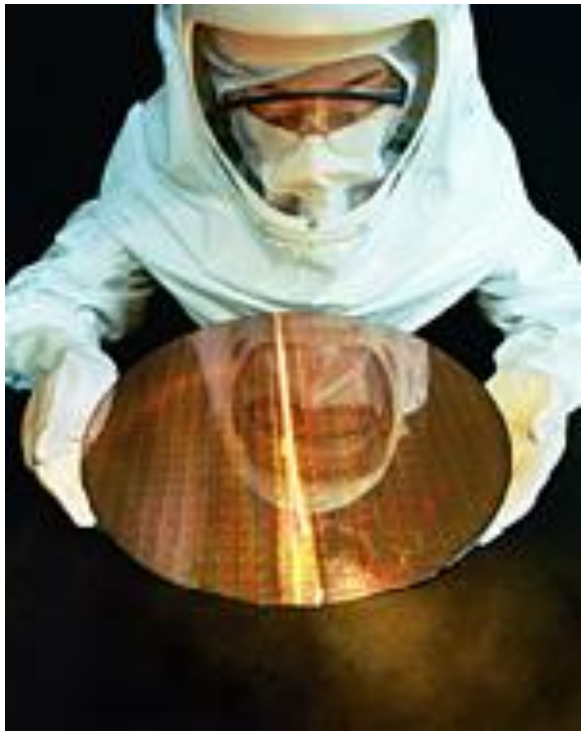
(a) Gate

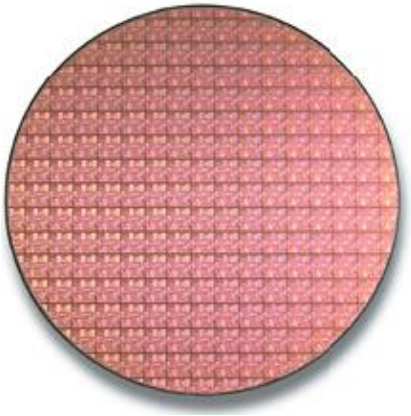


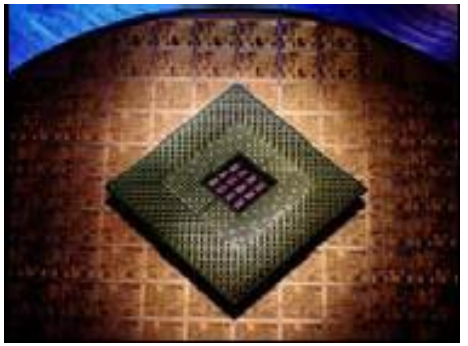
(b) Memory cell

Wafer, Chip and Gate

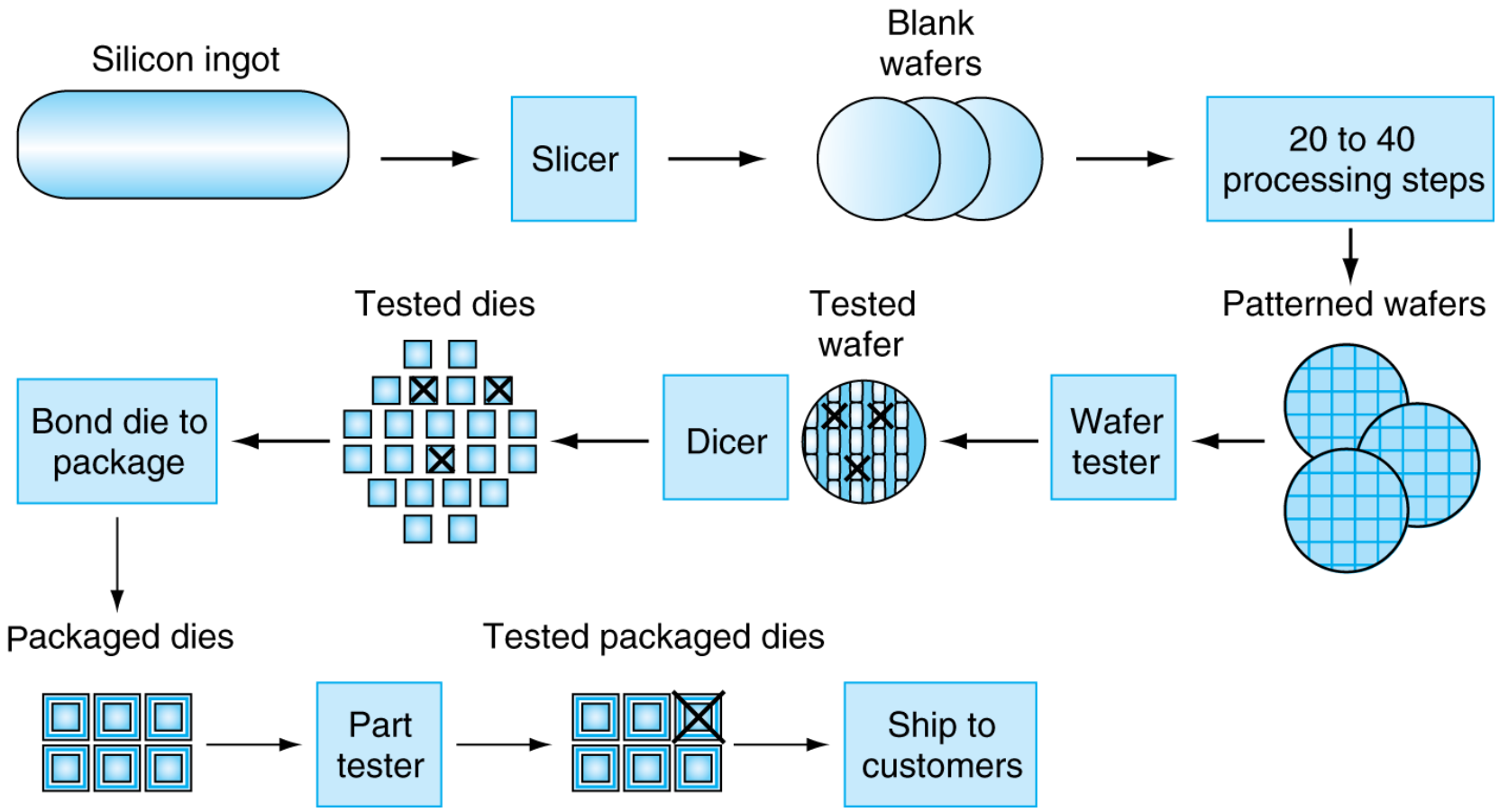




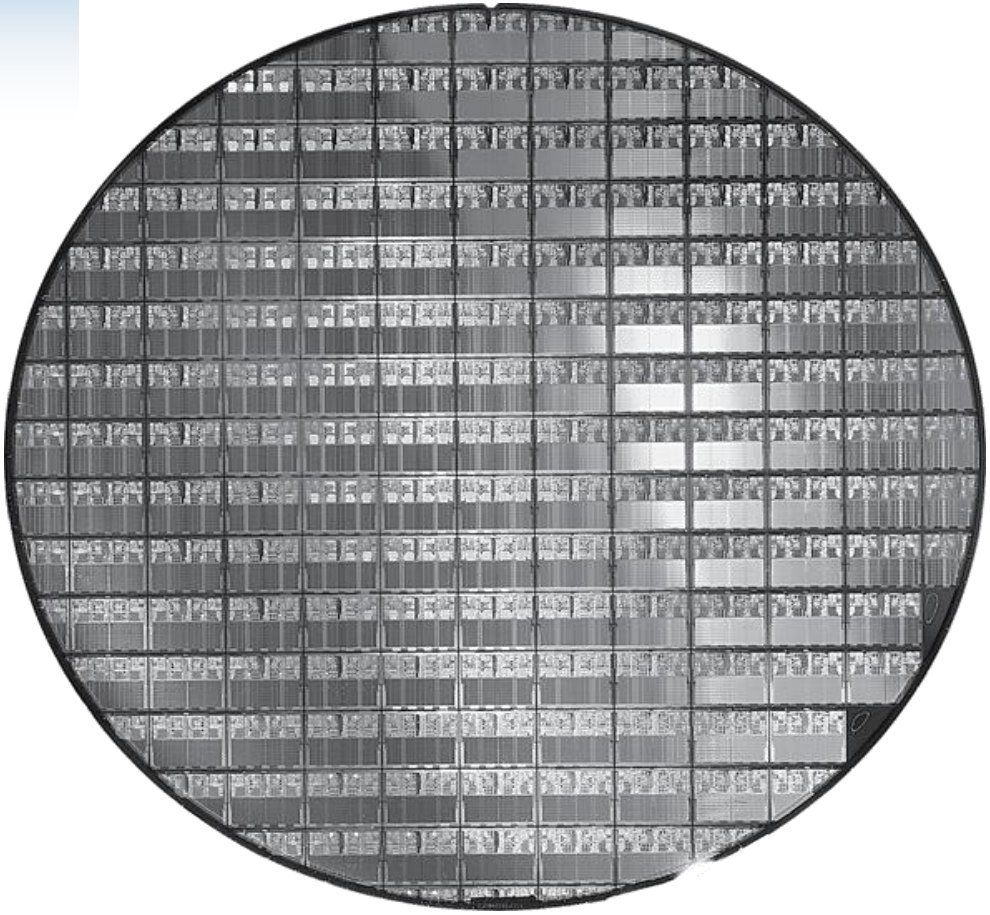


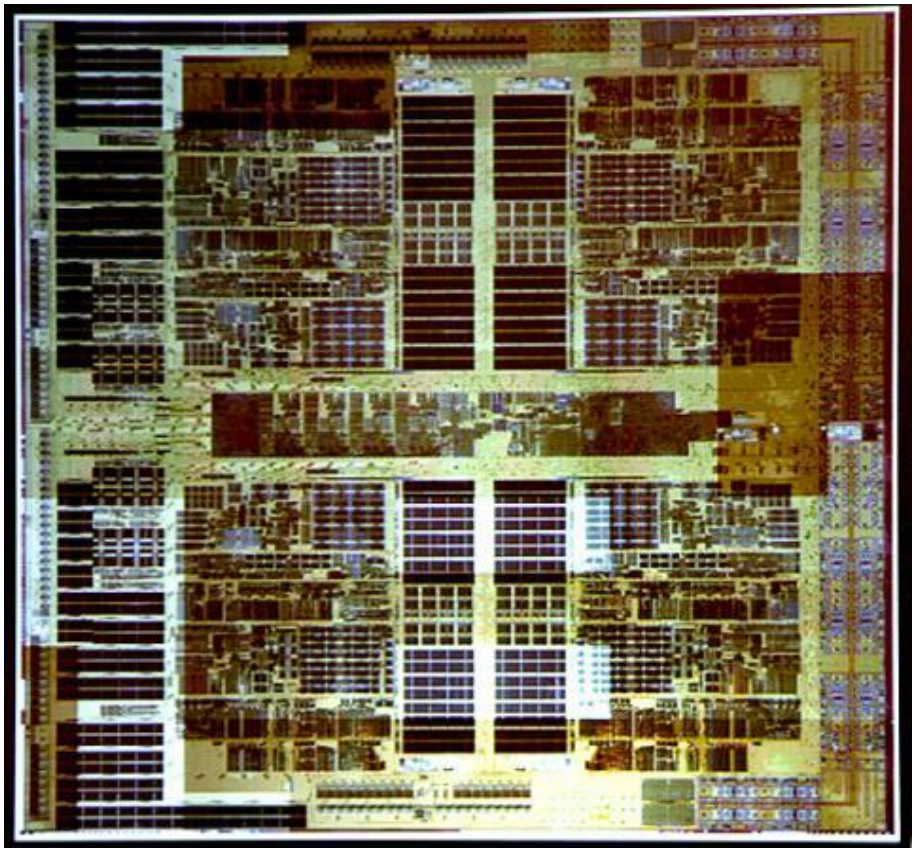


Manufacturing ICs



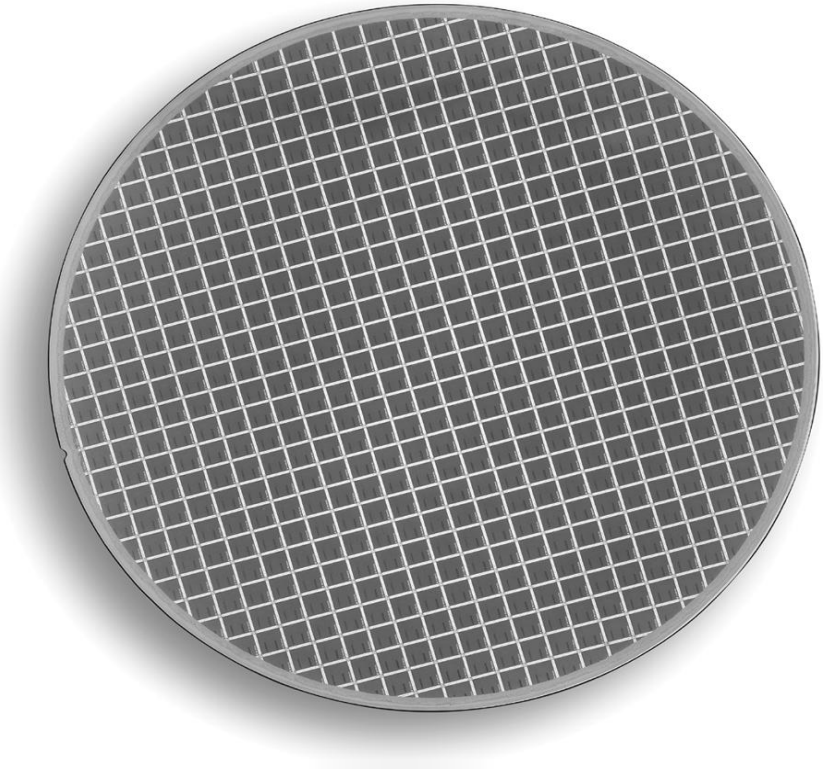
AMD Opteron X2 Wafer



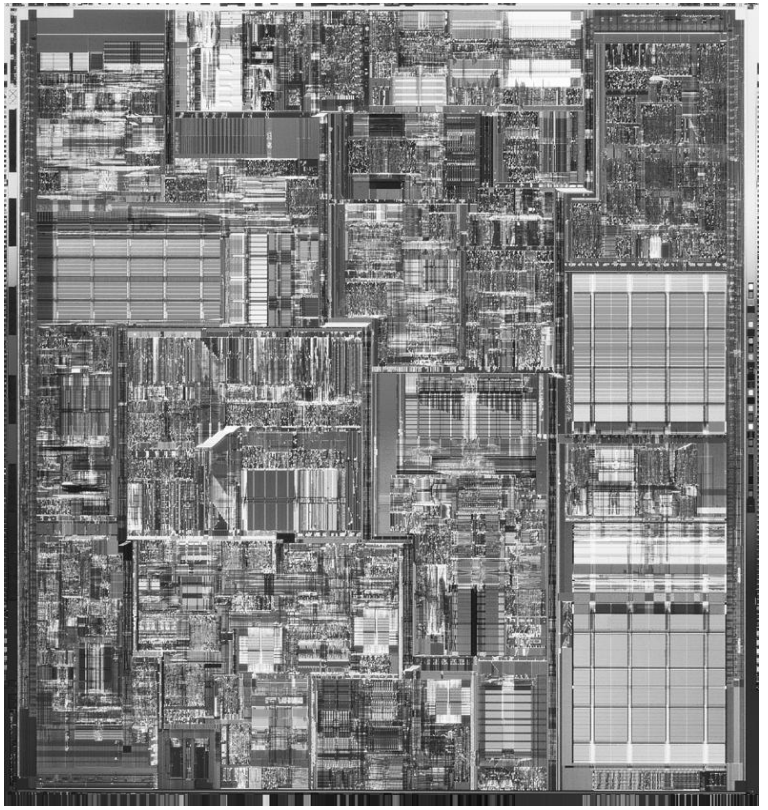


- X2: 300mm wafer, 117 chips/wafer, 90nm technology
- X4: 45nm technology

Silicon wafer and microprocessor die



This 8-inch wafer contains 564 MIPS64 R20K processors (0.18μ)



Intel Pentium 4 Microprocessor

Integrated Circuit Cost

$$\text{Cost per die} = \frac{\text{Cost per wafer}}{\text{Dies per wafer} \times \text{Yield}}$$

$$\text{Dies per wafer} \approx \frac{\text{Wafer area}}{\text{Die area}}$$

$$\text{Yield} = \frac{1}{(1 + (\text{Defects per area} \times \text{Die area}/2))^2}$$

- Nonlinear relation to area and defect rate
  - Wafer cost and area are fixed
  - Defect rate determined by manufacturing process
  - Die area determined by architecture and circuit design

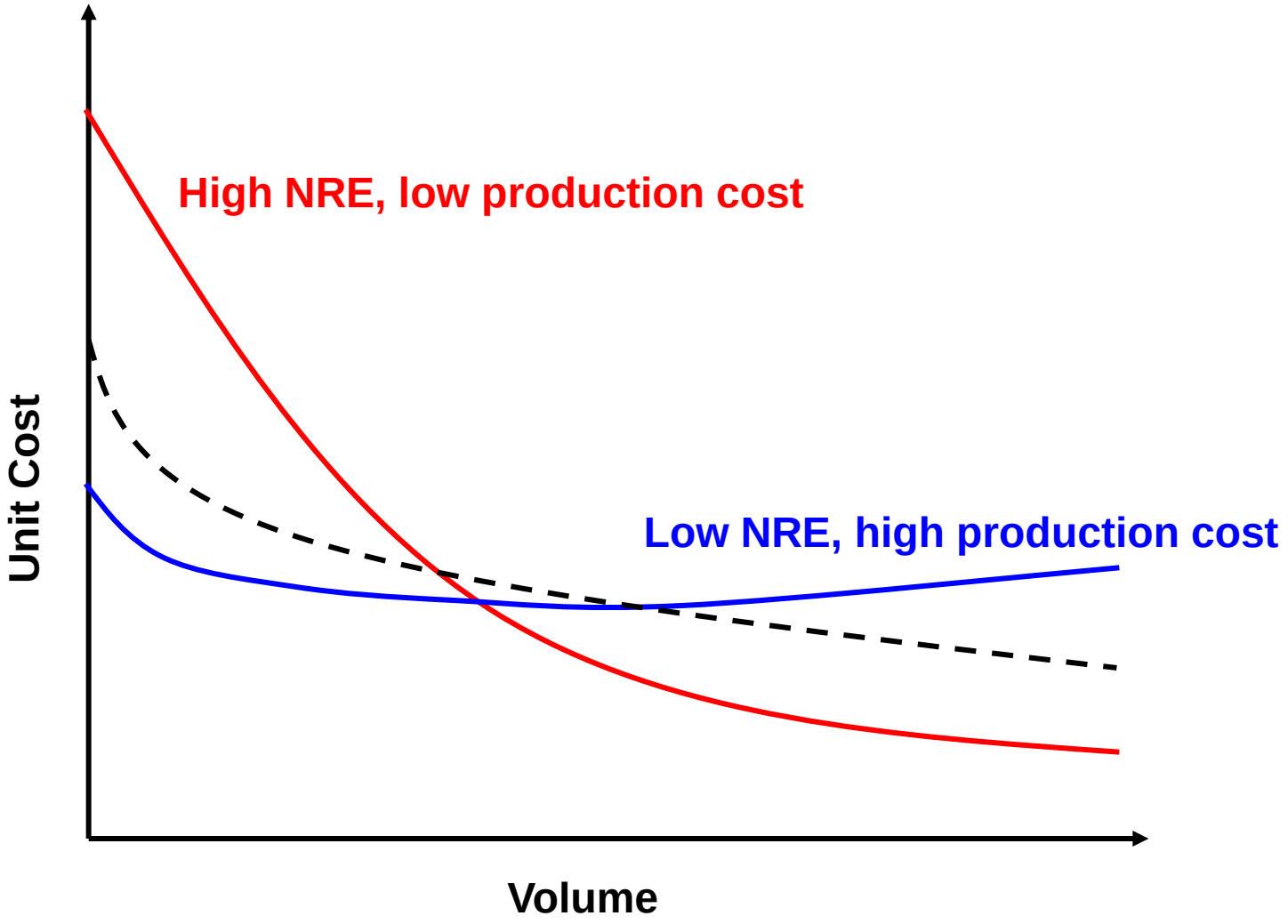
Cost

- Unit cost
  - Monetary cost of manufacturing one unit, excluding NRE cost
- NRE cost (Non-Recurring Engineering cost)
  - The one-time monetary cost of designing the system
- *Total cost*
  - *NRE cost + unit cost * # of unit*
- *Per-product cost*
  - *total cost / # of units = (NRE cost / # of units) + unit cost*

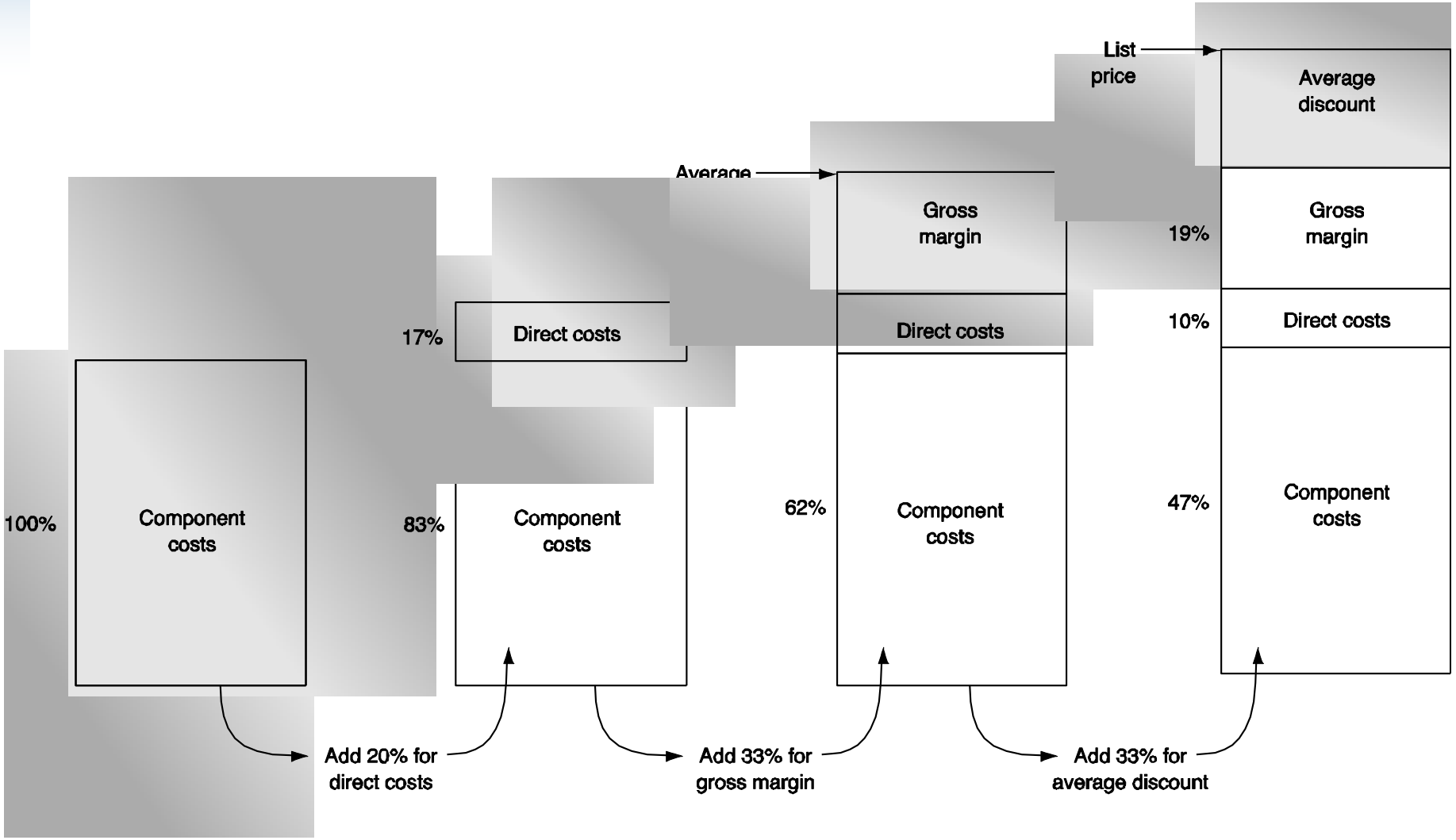
- Example
  - NRE=$2000, unit=$100
  - For 10 units
    - total cost = $2000 + 10*$100 = $3000
    - per-product cost = $2000/10 + $100 = $300

Amortizing NRE cost over the units results in an additional $200 per unit

NRE versus Unit Cost

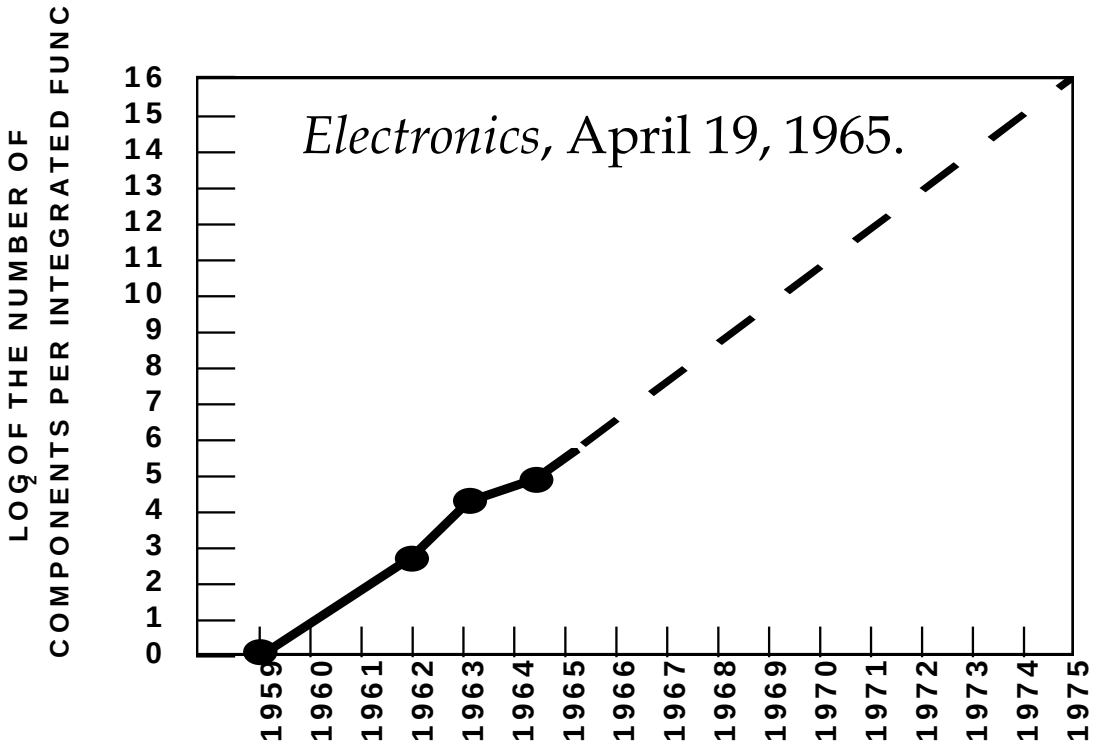


Cost versus Price



Moore's Law

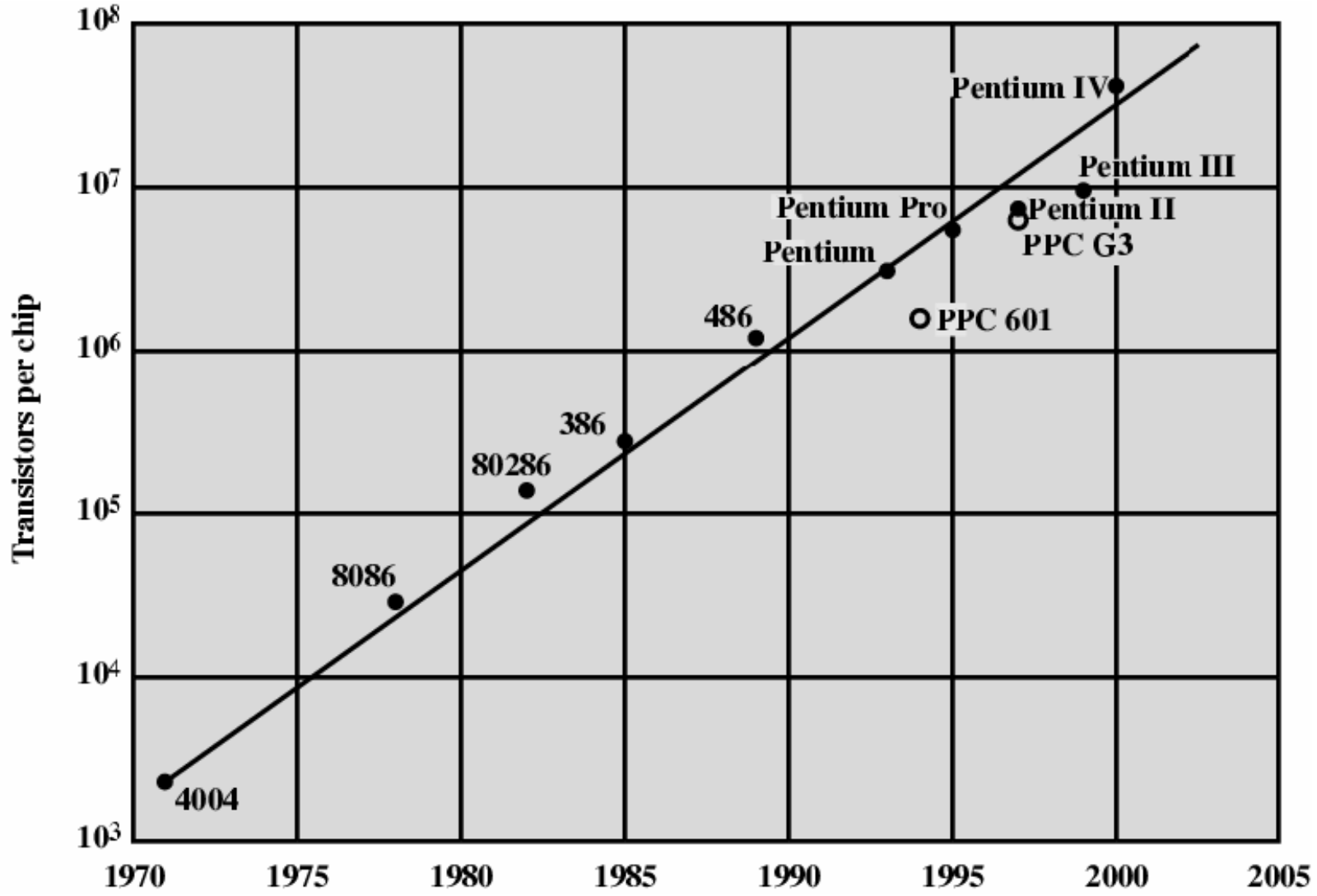
- Increased density of components on chip
- In 1965, Gordon Moore (co-founder of Intel) noted that the number of transistors on a chip doubled every 18 to 24 months.
- He made a prediction that semiconductor technology will double its effectiveness every 18 months



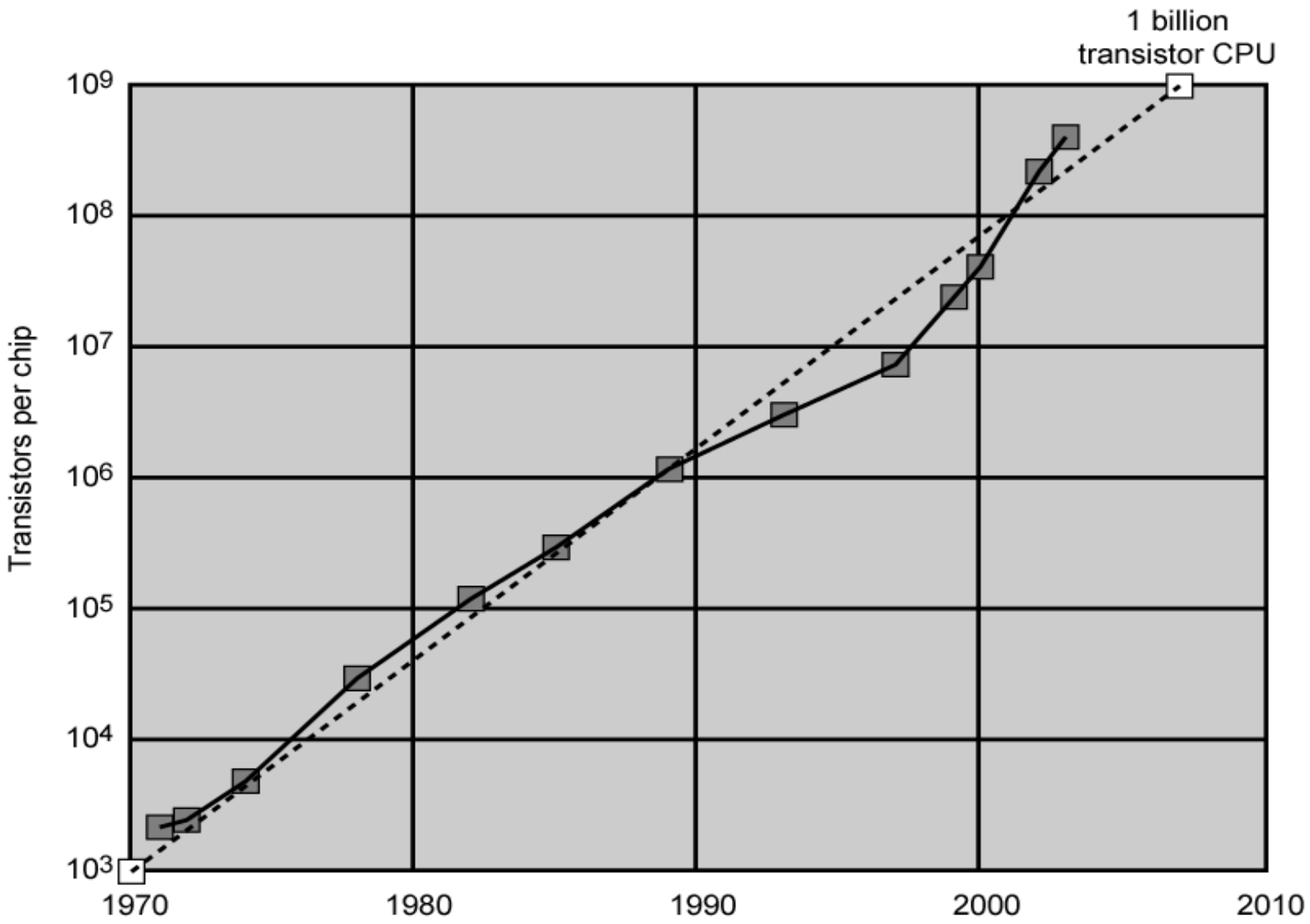
Who wants to be a Millionaire

- You double your investment everyday
  - Starting investment - one cent.
- How long it takes to become a millionaire
  a) 20 days One million cents
  b) 27 days Millionaire
  c) 37 days Billionaire
- Believe it or not
  - Each of us have more than a million ancestors in last 20 generations.
- Doubling transistors every 18 months
  - This growth rate is hard to imagine

Intel Microprocessor Evolution



Growth in CPU Transistor Count

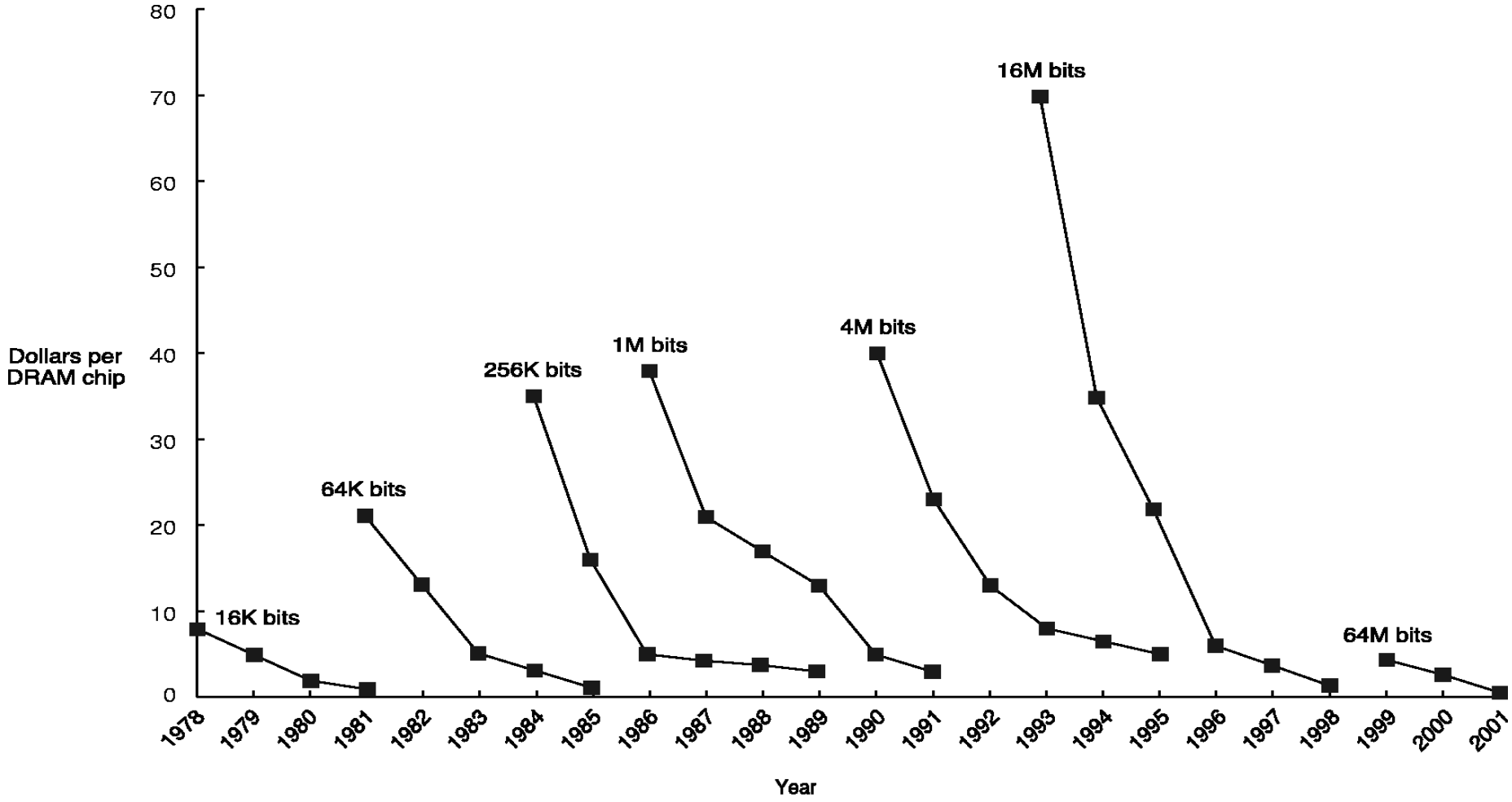


Transistor Counts

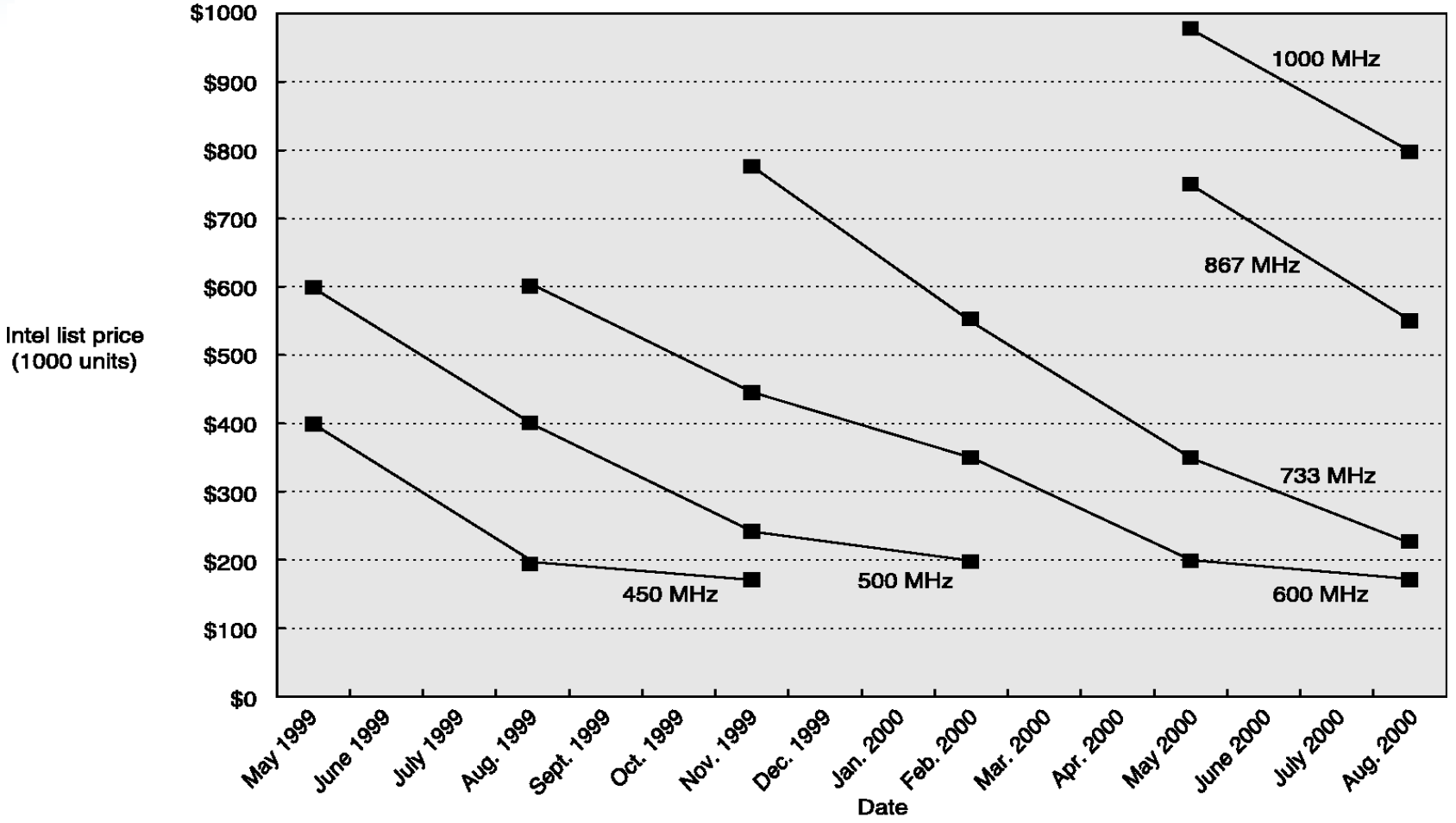
- Amazingly visionary – million transistor/chip barrier was crossed in the 1980's.
  - 2300 transistors, 1 MHz clock (Intel 4004) - 1971
  - 16 Million transistors (Ultra Sparc III)
  - 42 Million, 2 GHz clock (Intel P4) - 2001
  - 140 Million transistor (HP PA-8500)

- Cost of a chip has remained almost unchanged
- Higher packing density means shorter electrical paths, giving higher performance
- Smaller size gives increased flexibility
- Reduced power and cooling requirements
- Fewer interconnections increases reliability

DRAM Pricing



Processor Pricing (Intel Pentium III)

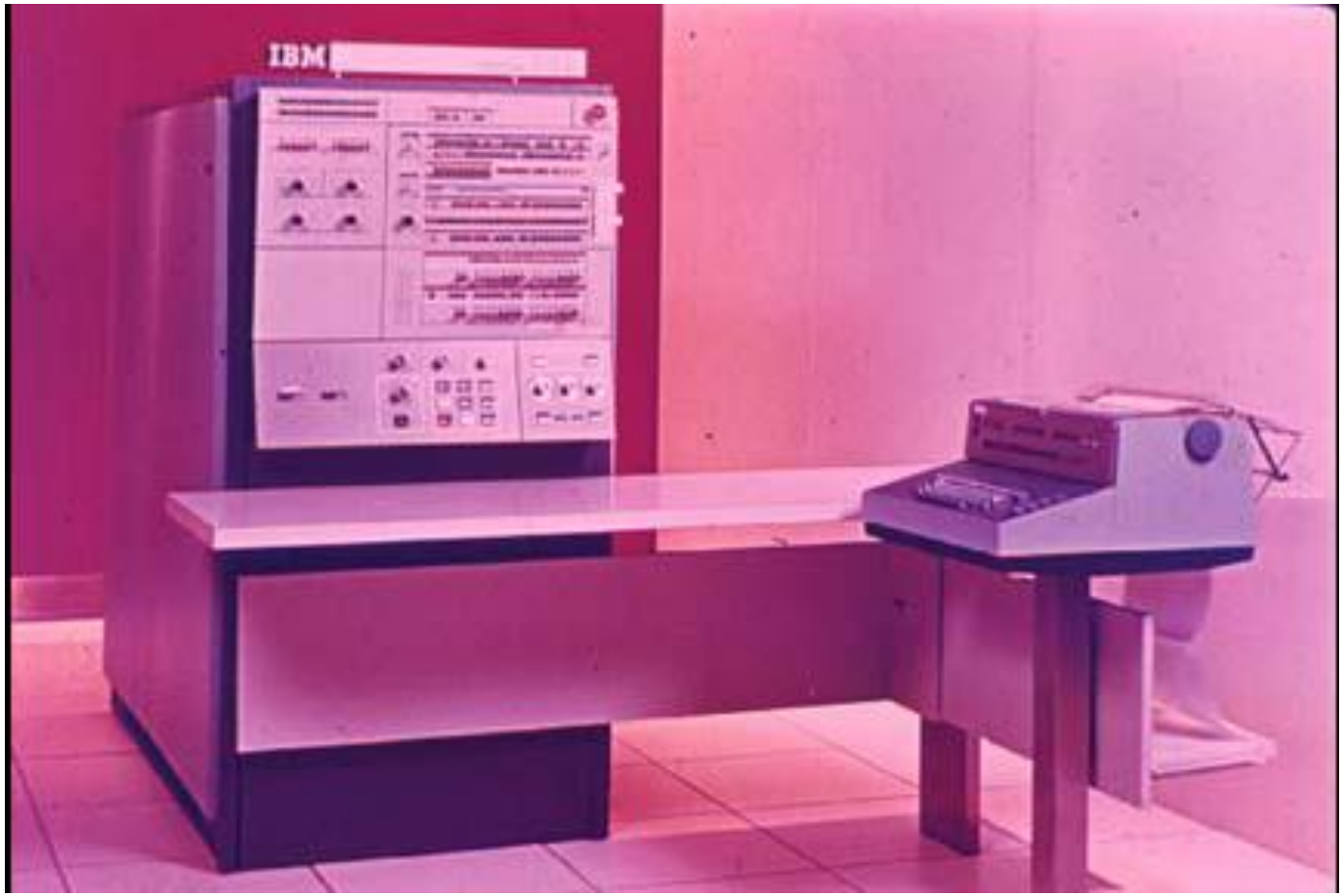


IBM 360 series

- 1964
- Replaced (& not compatible with) 7000 series
- First planned "family" of computers
  - Similar or identical instruction sets
  - Similar or identical O/S
  - Increasing speed
  - Increasing number of I/O ports (i.e. more terminals)
  - Increased memory size
  - Increased cost

Characteristic	Model 30	Model 40	Model 50	Model 65	Model 75
Maximum memory size (bytes)	64K	256K	256K	512K	512K
Data rate from memory (Mbytes/sec)	0.5	0.8	2.0	8.0	16.0
Processor cycle time μs)	1.0	0.625	0.5	0.25	0.2
Relative speed	1	3.5	10	21	50
Maximum number of data channels	3	3	4	6	6
Maximum data rate on one channel (Kbytes/s)	250	400	800	1250	1250

IBM System 360

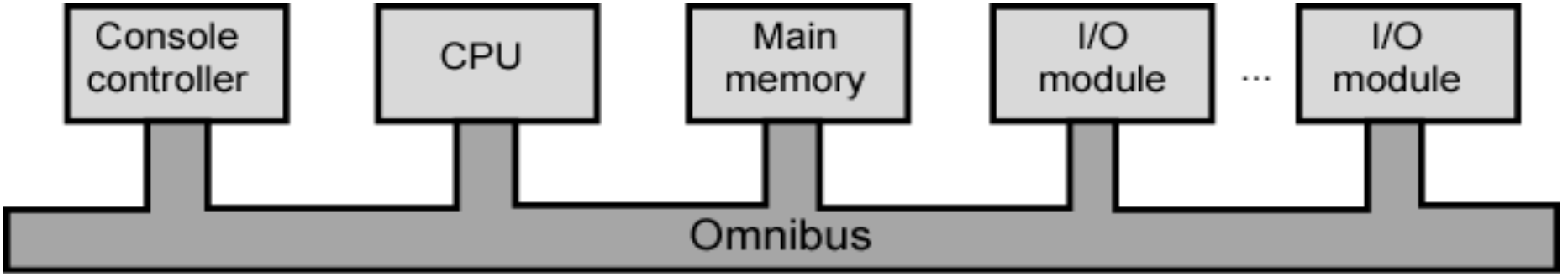


DEC PDP-8

DEC PDP-8

- 1964
- First minicomputer (after miniskirt!)
- Did not need air conditioned room
- Small enough to sit on a lab bench
- $16,000
  - $100k+ for IBM 360
- Embedded applications & OEM (Original Equipment Manufacturer)
- BUS STRUCTURE

DEC – PDP-8 Bus Structure



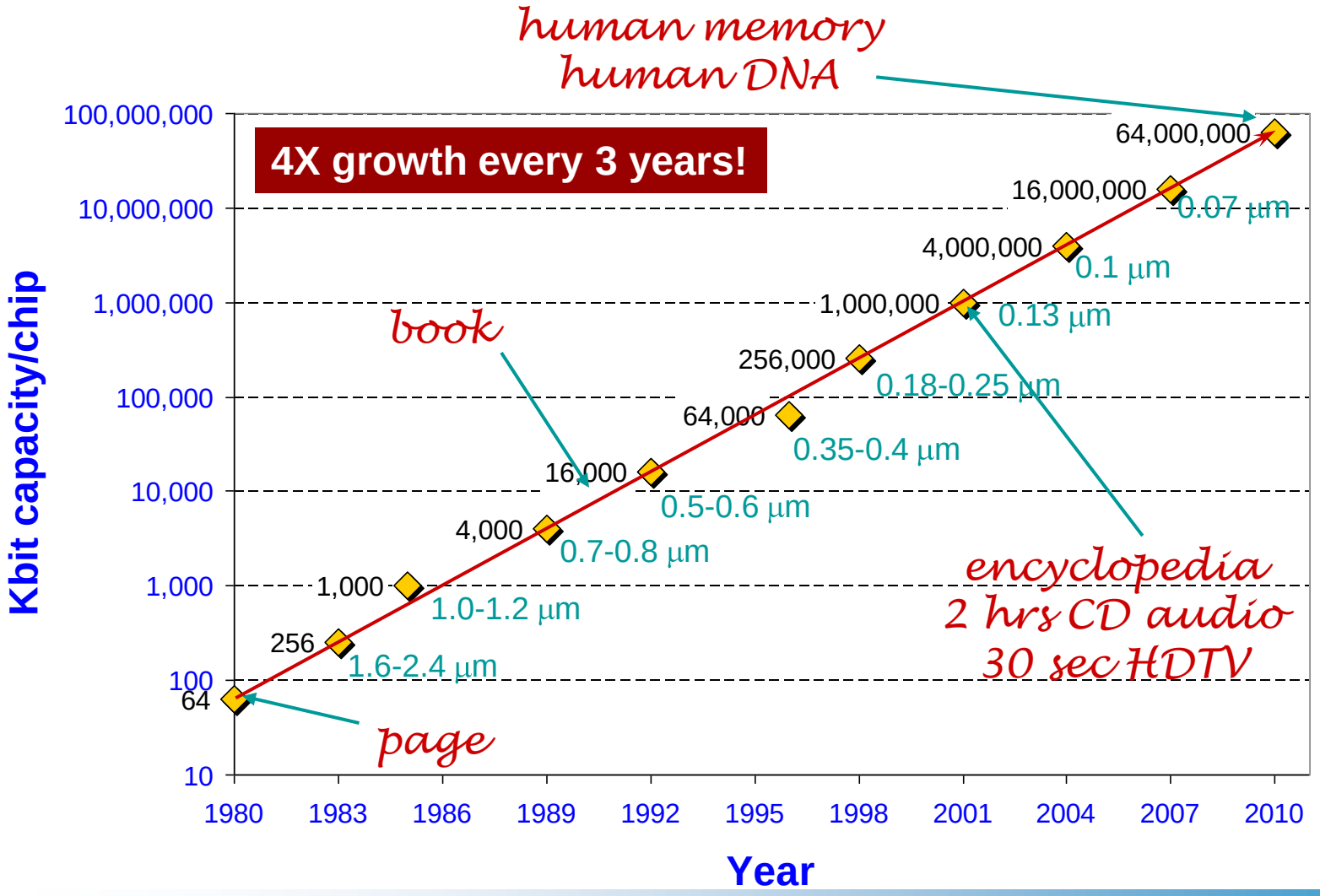
Generations of Computer

Generation	Approx. dates	Technology	Chip density	Typical speed (operations per second)
1	1946-1957	Vacuum tube		40,000
2	1958-1964	Transistor		200,000
3	1965-1971	Small and medium scale integration (SSI and MSI)	SSI: < 100 devices/chip MSI: 100 – 3,000 devices/chip	1,000,000
4	1972-1977	Large scale integration (LSI)	3,000 – 100,000 devices/chip	10,000,000
5	1978-1991	Very large scale integration (VLSI)	100,000 – 100,000,000 devices/chip	100,000,000
6	1991-????	Ultra large scale integration (ULSI)	> 100,000,000 devices/chip	1,000,000,000

Semiconductor Memory

- 1970
- Fairchild
- Size of a single core
  - i.e. 1 bit of magnetic core storage
- Holds 256 bits
- Non-destructive read
- Much faster than core
- Capacity approximately doubles each year

Evolution in DRAM Chip Capacity



Microprocessor

(a) 1970s Processors

	4004	8008	8080	8086	8088
Introduced	1971	1972	1974	1978	1979
Clock speeds	108 kHz	108 kHz	2 MHz	5 MHz, 8 MHz, 10 MHz	5 MHz, 8 MHz
Bus width	4 bits	8 bits	8 bits	16 bits	8 bits
Number of transistors	2,300	3,500	6,000	29,000	29,000
Feature size (μm)	10		6	3	6
Addressable memory	640 Bytes	16 KBytes	64 KBytes	1 MB	1 MB
Virtual memory	—	—	—	—	—

(b) 1980s Processors

	80286	386TM DX	386TM SX	486TM DX CPU
Introduced	1982	1985	1988	1989
Clock speeds	6 MHz - 12.5 MHz	16 MHz - 33 MHz	16 MHz - 33 MHz	25 MHz - 50 MHz
Bus width	16 bits	32 bits	16 bits	32 bits
Number of transistors	134,000	275,000	275,000	1.2 million
Feature size (μm)	1.5	1	1	0.8 - 1
Addressable memory	16 megabytes	4 gigabytes	16 megabytes	4 gigabytes
Virtual memory	1 gigabyte	64 terabytes	64 terabytes	64 terabytes

Microprocessor

(c) 1990s Processors

	486TM SX	Pentium	Pentium Pro	Pentium II
Introduced	1991	1993	1995	1997
Clock speeds	16 MHz - 33 MHz	60 MHz - 166 MHz,	150 MHz - 200 MHz	200 MHz - 300 MHz
Bus width	32 bits	32 bits	64 bits	64 bits
Number of transistors	1.185 million	3.1 million	5.5 million	7.5 million
Feature size (μm)	1	0.8	0.6	0.35
Addressable memory	4 gigabytes	4 gigabytes	64 gigabytes	64 gigabytes
Virtual memory	64 terabytes	64 terabytes	64 terabytes	64 terabytes

(d) Recent Processors

	Pentium III	Pentium 4	Itanium	Itanium 2
Introduced	1999	2000	2001	2002
Clock Speeds	450 - 660 MHz	1.3 - 1.8 GHz	733 - 800 MHz	900 MHz - 1 GHz
Bus Width	64 bits	64 bits	64 bits	64 bits
Number of Transistors	9.5 million	42 million	25 million	220 million
Feature size (μm)	0.25	0.18	0.18	0.18
Addressable Memory	64 gigaBytes	64 gigaBytes	64 gigaBytes	64 gigaBytes
Virtual Memory	64 teraBytes	64 teraBytes	64 teraBytes	64 teraBytes

Technology Trend

- Component
  - IC technology: transistor/chip increases 55% per year
  - DRAM: density increases 40-60% per year
  - Magnetic disk: density increases 100% per year
  - Network: Ethernet from 10 ➔ 100Mb took 10 years; 100Mb ➔ 1Gb in 5 years
- Scaling of performance, wires and power
  - Feature size: 10 micron in 1971; 0.18 in 2001, …
  - Microprocessor organization improvement
  - Wiring delay
  - Power issue: ~100 watts for 2GHz Pentium 4

Disk Comparison

- CDC Wren I, 1983
- 3600 RPM
- 0.03 GBytes capacity
- Tracks/Inch: 800
- Bits/Inch: 9550
- Three 5.25” platters
- Bandwidth: 0.6 MBytes/sec
- Latency: 48.3 ms
- Cache: none

- Seagate 373453, 2003
- 15000 RPM (4X)
- 73.4 GBytes (2500X)
- Tracks/Inch: 64000 (80X)
- Bits/Inch: 533,000 (60X)
- Four 2.5” platters (in 3.5” form factor)
- Bandwidth: 86 MBytes/sec (140X)
- Latency: 5.7 ms (8X)
- Cache: 8 MBytes

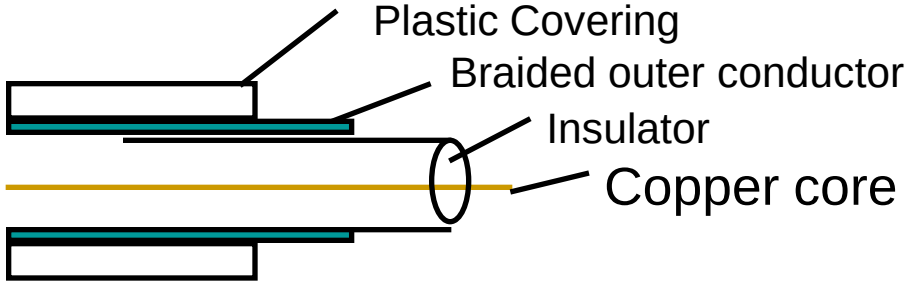
Memory Comparison

- 1980 DRAM (asynchronous)
- 0.06 Mbits/chip
- 64 K transistors, 35 mm^2
- 16-bit data bus per module
- 16 pins/chip
- 13 Mbytes/sec
- Latency: 225 ns
- (no block transfer)

- 2000 Double Data Rate Synchronous (clocked) DRAM
- 256.00 Mbits/chip (4000X)
- 256 M transistors, 204 mm2
- 64-bit data bus per DIMM, 66 pins/chip (4X)
- 1600 Mbytes/sec (120X)
- Latency: 52 ns (4X)
- Block transfers (page mode)

LAN Comparison

- Ethernet 802.3
- Year of Standard: 1978
- 10 Mbits/s link speed
- Latency: 3000 msec
- Shared media
- Coaxial cable



- Ethernet 802.3ae
- Year of Standard: 2003
- 10,000 Mbits/s (1000X) link speed
- Latency: 190 msec (15X)
- Switched media
- Category 5 copper wire

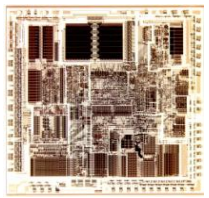
"Cat 5" is 4 twisted pairs in bundle

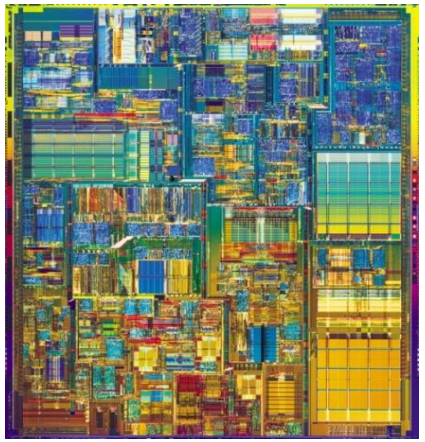
Twisted Pair:

Copper, 1mm thick,
twisted to avoid antenna effect

CPU Comparison

- 1982 Intel 80286
- 12.5 MHz
- 2 MIPS (peak)
- Latency 320 ns
- 134,000 xtors, 47 mm2
- 16-bit data bus, 68 pins
- Microcode interpreter, separate FPU chip
- (no caches)

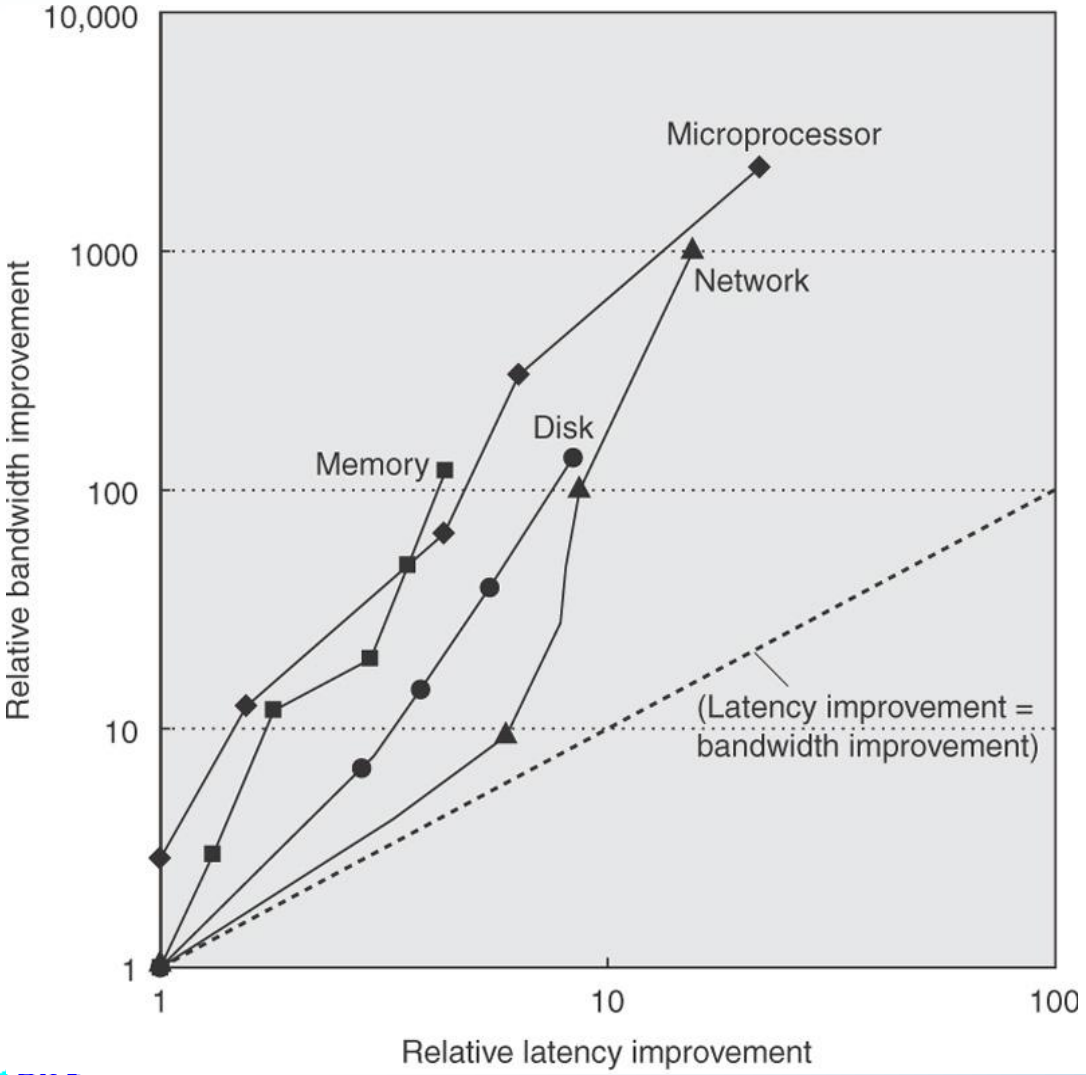




- 2001 Intel Pentium 4
- 1500 MHz (120X)
- 4500 MIPS (peak) (2250X)
- Latency 15 ns (20X)
- 42,000,000 xtors, 217 mm2
- 64-bit data bus, 423 pins
- 3-way superscalar, Dynamic translate to RISC, Superpipelined (22 stage), Out-of-Order execution
- On-chip 8KB Data caches, 96KB Instr. Trace cache, 256KB L2 cache

Bandwidth vs. Latency

Latency improvement is 10X while bandwidth improvement is 100X to 1000X.



- Processor: ‘286, ‘386, ‘486, Pentium, Pentium Pro, Pentium 4 (21x,2250x)
- Ethernet: 10Mb, 100Mb, 1000Mb, 10000 Mb/s (16x,1000x)
- Memory Module: 16bit plain DRAM, Page Mode DRAM, 32b, 64b, SDRAM, DDR SDRAM (4x,120x)
- Disk: 3600, 5400, 7200, 10000, 15000 RPM (8x, 143x)

Summary: Technology Trends

- For disk, LAN, memory, and microprocessor, bandwidth improves by square of latency improvement
  - In the time that bandwidth doubles, latency improves by no more than 1.2X to 1.4X
- Lag probably even larger in real systems, as bandwidth gains multiplied by replicated components
  - Multiple processors in a cluster or even in a chip
  - Multiple disks in a disk array
  - Multiple memory modules in a large memory
  - Simultaneous communication in switched LAN
- HW and SW developers should innovate assuming Latency Lags Bandwidth
  - If everything improves at the same rate, then nothing really changes
  - When rates vary, require real innovation

Technology Outlook

High Volume Manufacturing	2004	2006	2008	2010	2012	2014	2016	2018
Technology Node (nm)	90	65	45	32	22	16	11	8
Integration Capacity (BT)	2	4	8	16	32	64	128	256
Delay = CV/I scaling	0.7	~0.7	>0.7	Delay scaling will slow down				
Energy/Logic Op scaling	>0.35	>0.5	>0.5	Energy scaling will slow down				
Bulk Planar CMOS	High Probability						Low Probability	
Alternate, 3G etc	Low Probability						High Probability	
Variability	Medium			High			Very High	
ILD (K)	~3	<3	Reduce slowly towards 2 to 2.5					
RC Delay	1	1	1	1	1	1	1	1
Metal Layers	6-7	7-8	8-9	0.5 to 1 layer per generation				

Lecture 1 – Computer Architecture

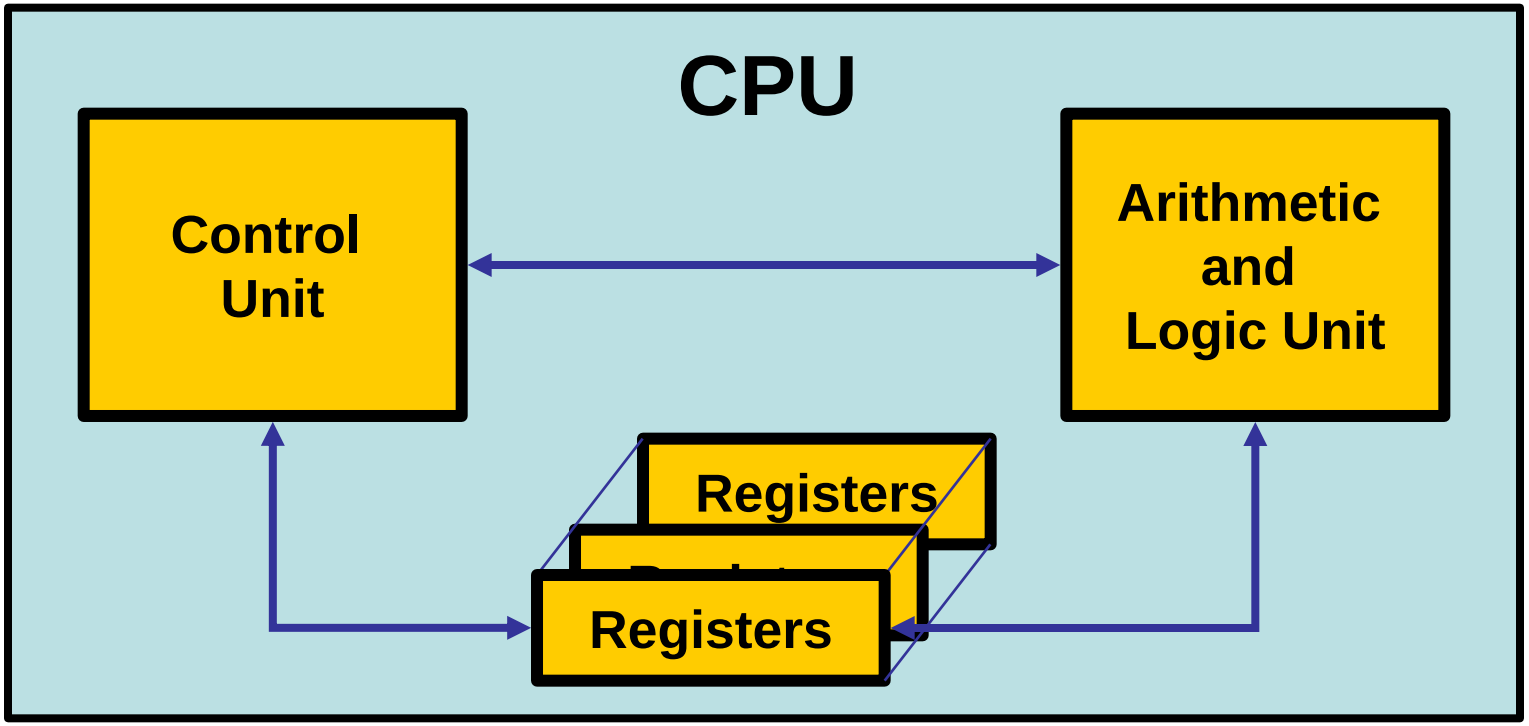
- History of Computer
- CPU and Instruction Execution
- Fundamental of Computer Design
- Designing for Performance

Central Processing Unit (CPU)

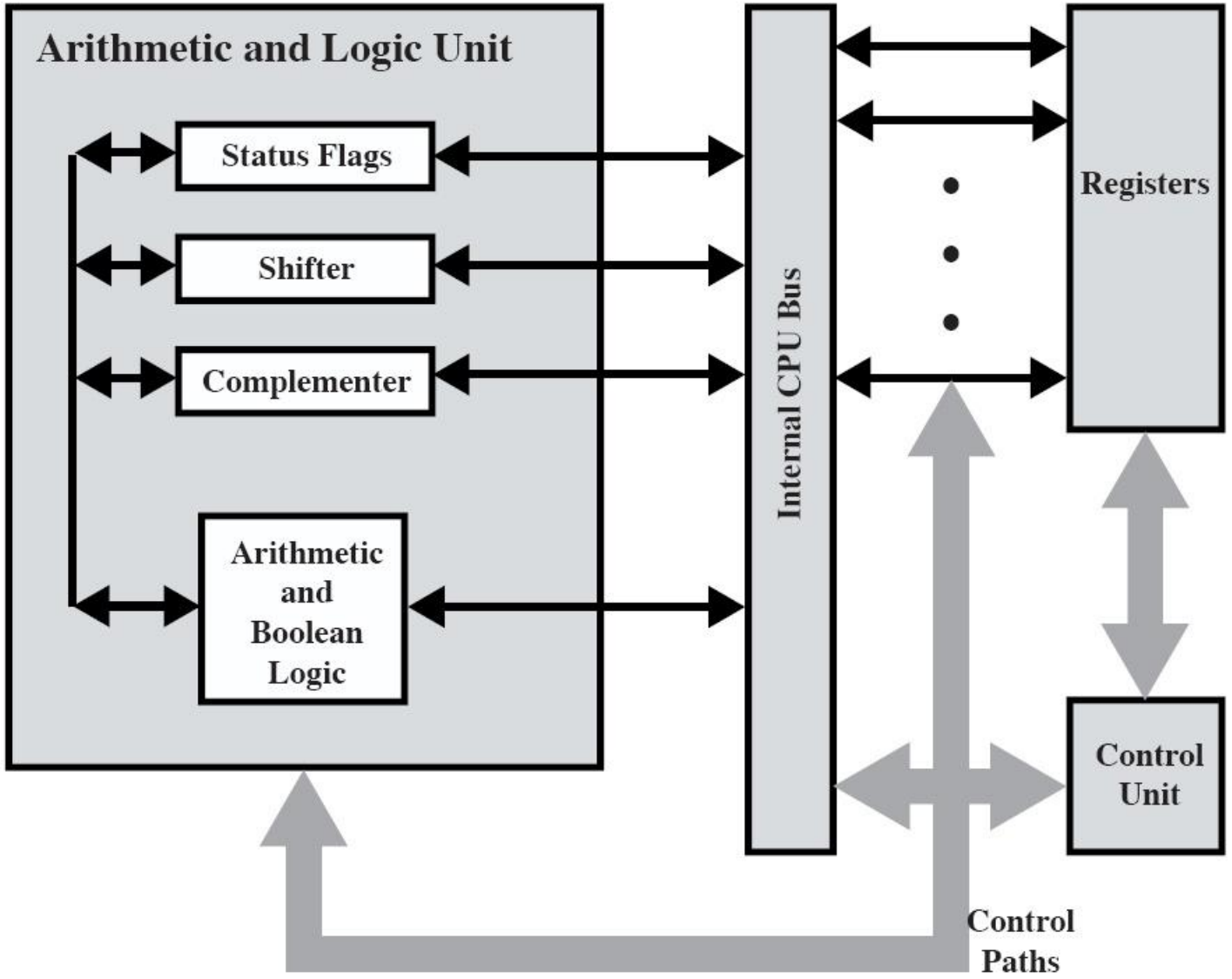
- The primary function of a CPU is to execute the instructions stored in the main memory.
- An instruction tells the CPU to perform one of its basic operations.
- The CPU includes also a set of registers, which are temporary storage devices used to hold control information, key data, and intermediate results.
- It includes also an internal bus infrastructure, which provides data movement paths among the control unit, ALU, and registers.
- The CU is the one which interprets (decodes) the instruction to be executed and "tells" the other components what to do.

CPU

- The CPU, also called processor, includes two main units:
  - A program control unit, and
  - An Arithmetic and Logic Unit (ALU)



CPU Internal Structure



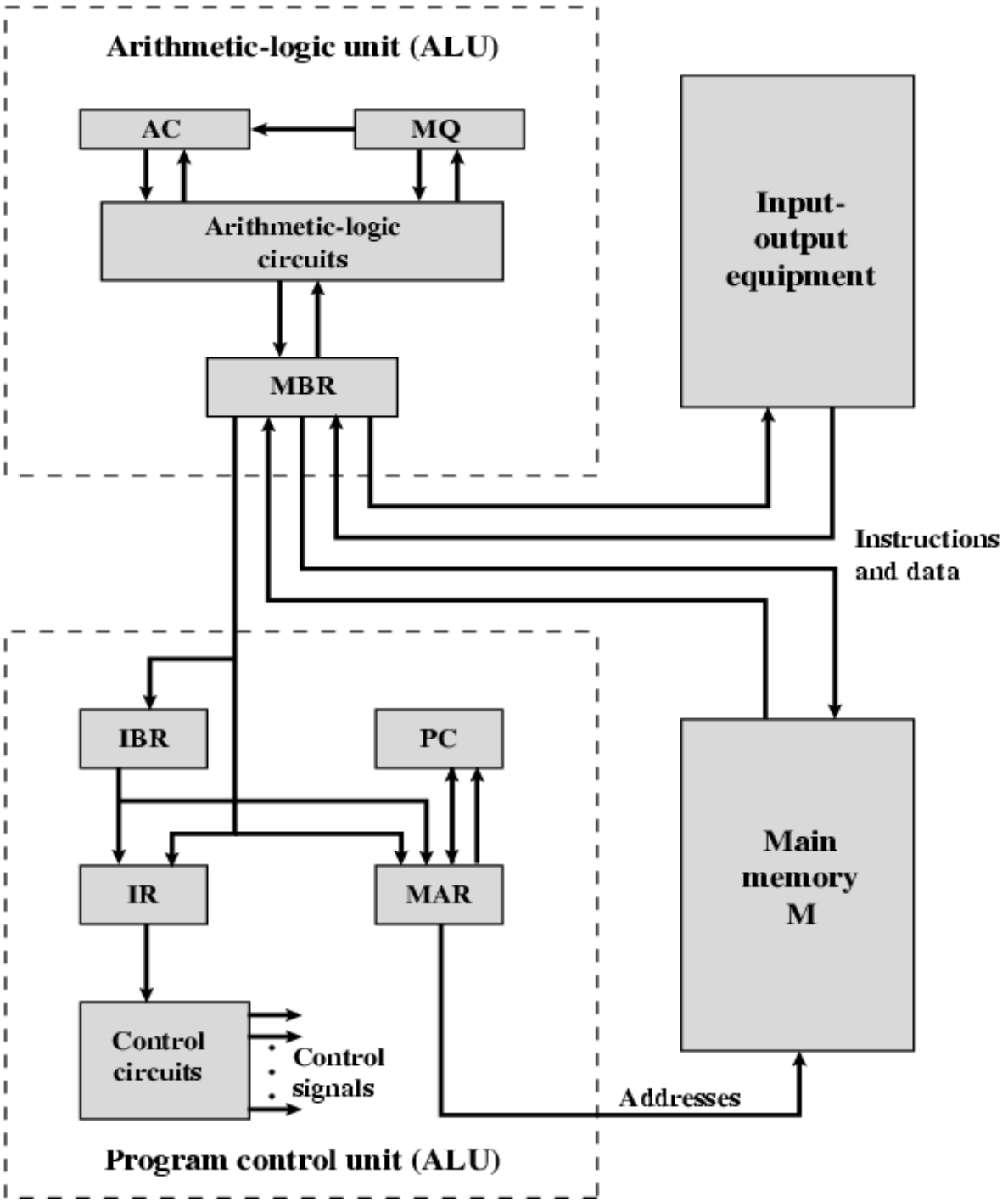
Registers

- CPU must have some working space (temporary storage).
- These storage units are called registers.
- They are the top level component in the memory hierarchy.
- Number and function of the registers vary between different computers.
- Registration organization is one of the major design decisions.

Register Organization

- The registers serve two main functions:
  - User-Visible Registers: used by machine or assembly language programmers to minimize memory access.
    - General-purpose registers
    - Data registers
    - Address registers
    - Condition code registers
  - Control and Status Registers: used by the control unit to control the operation of the CPU, and by the operating system to control the execution of programs.

Structure of IAS – detail



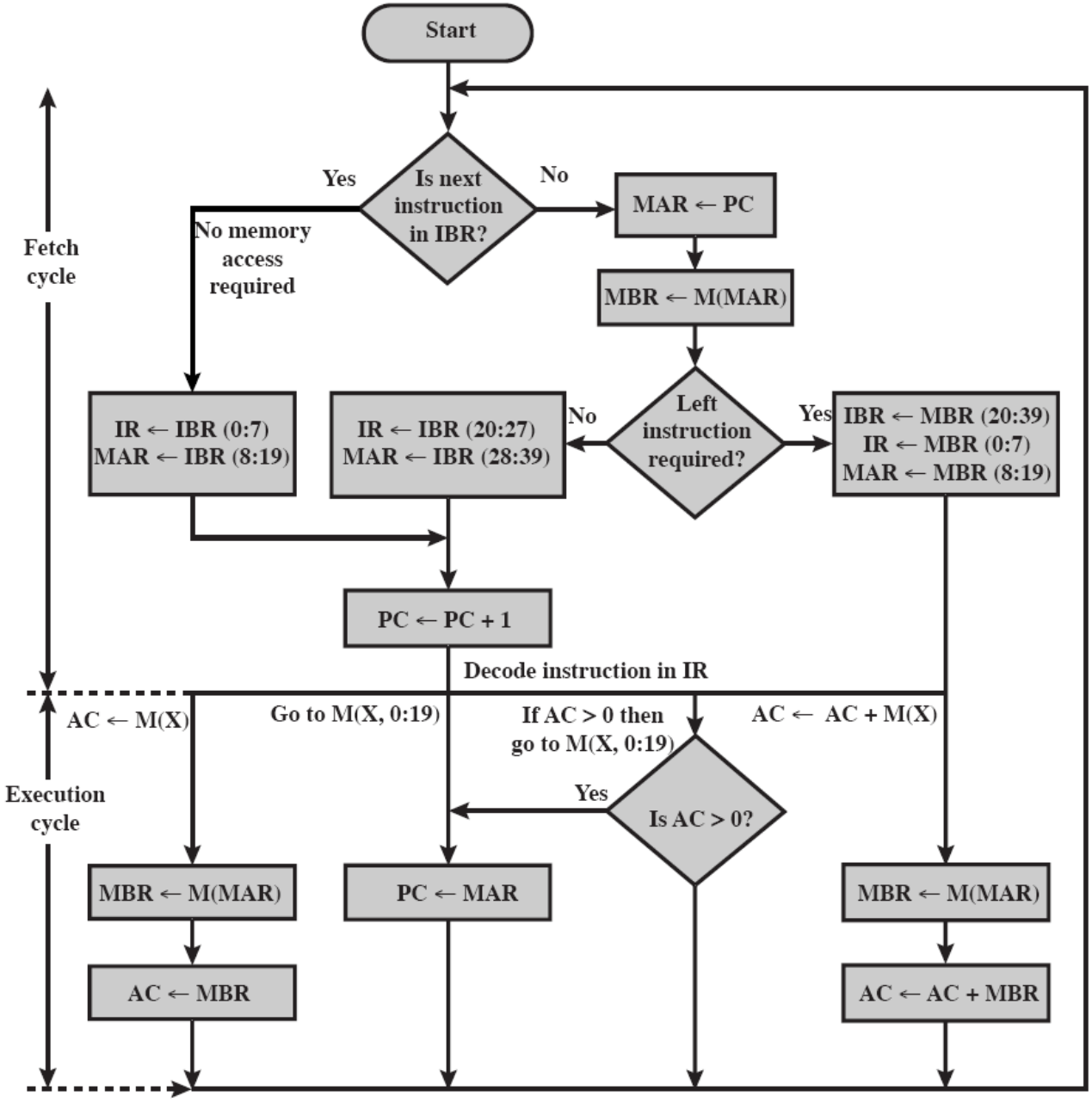
Machine Instructions

- The CPU can only execute machine code in binary format, called machine instructions.
- A machine instruction specifies the following information:
  - What has to be done (the operation code)
  - To whom the operation applies (source operands)
  - Where does the result go (destination operand)
  - How to continue after the operation is finished (next instruction address).
- Machine instructions are of four types:
  - Arithmetic and logic operations.
  - Data transfer between memory and CPU registers.
  - Program control (conditional branches, etc.).
  - I/O transfer.

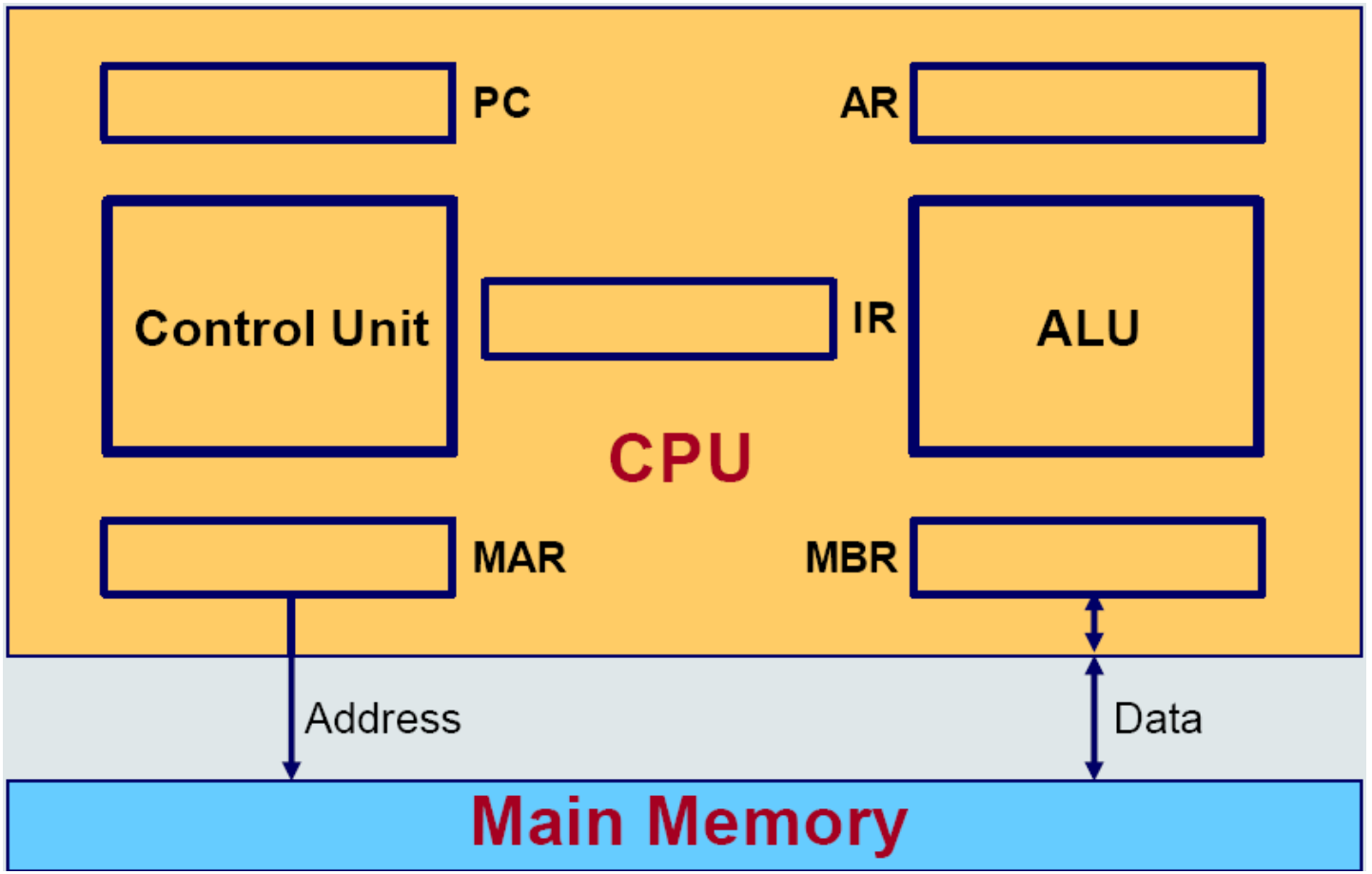
Instruction Set Design

- The design of an instruction set is critical to the operations of a computer system. The most important issues are:
  - Operation repertoire — How many and which operations to provide, and how complex operations should be.
  - Data types — Which data types to be supported.
  - Instruction format — Length, number of addresses, size of various fields, etc.
  - Registers — Number of CPU registers and their use.
  - Addressing — Which modes to be provided.
- The issues are highly interrelated and must be considered together.

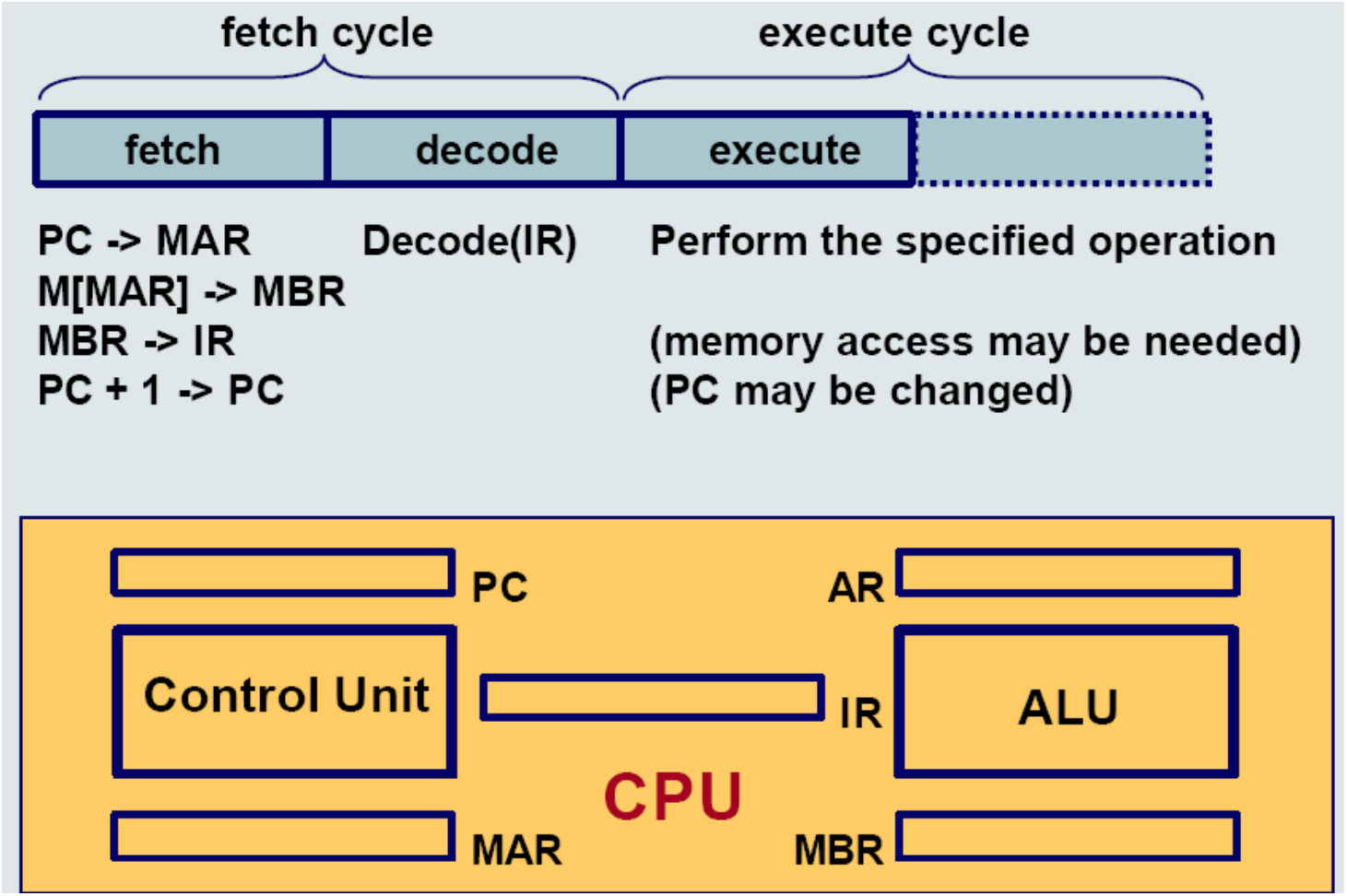
Partial flowchart of IAS operation



Instruction Execution Mechanism



Instruction Execution



Machine Cycles

- The execution of an instruction is carried out in a machine cycle (instruction cycle).
- The CPU executes one instruction after the other, cycle by cycle, repeatedly.
- The machine cycle time (or instruction execution time) of a computer gives an indication of its performance (speed).
  - Ex. a computer can have a performance of 733 MIPS (Millions of Instructions Per Second).
- Since different instructions need different time to execute, the average instruction execution time is used to calculate performance.

Summary

- A computer executes repeatedly a series of instructions (called programs) stored in its main memory:
  - It performs data processing operations specified by the programs.
  - It runs the programs automatically, with no need for human intervention.
  - It can perform the operations in extremely high speed.
  - It can store and manipulate a large amount of data.
  - It can communicate with each other and with users in an efficient way.
  - It represents program and data in the same way, which leads to flexibility.

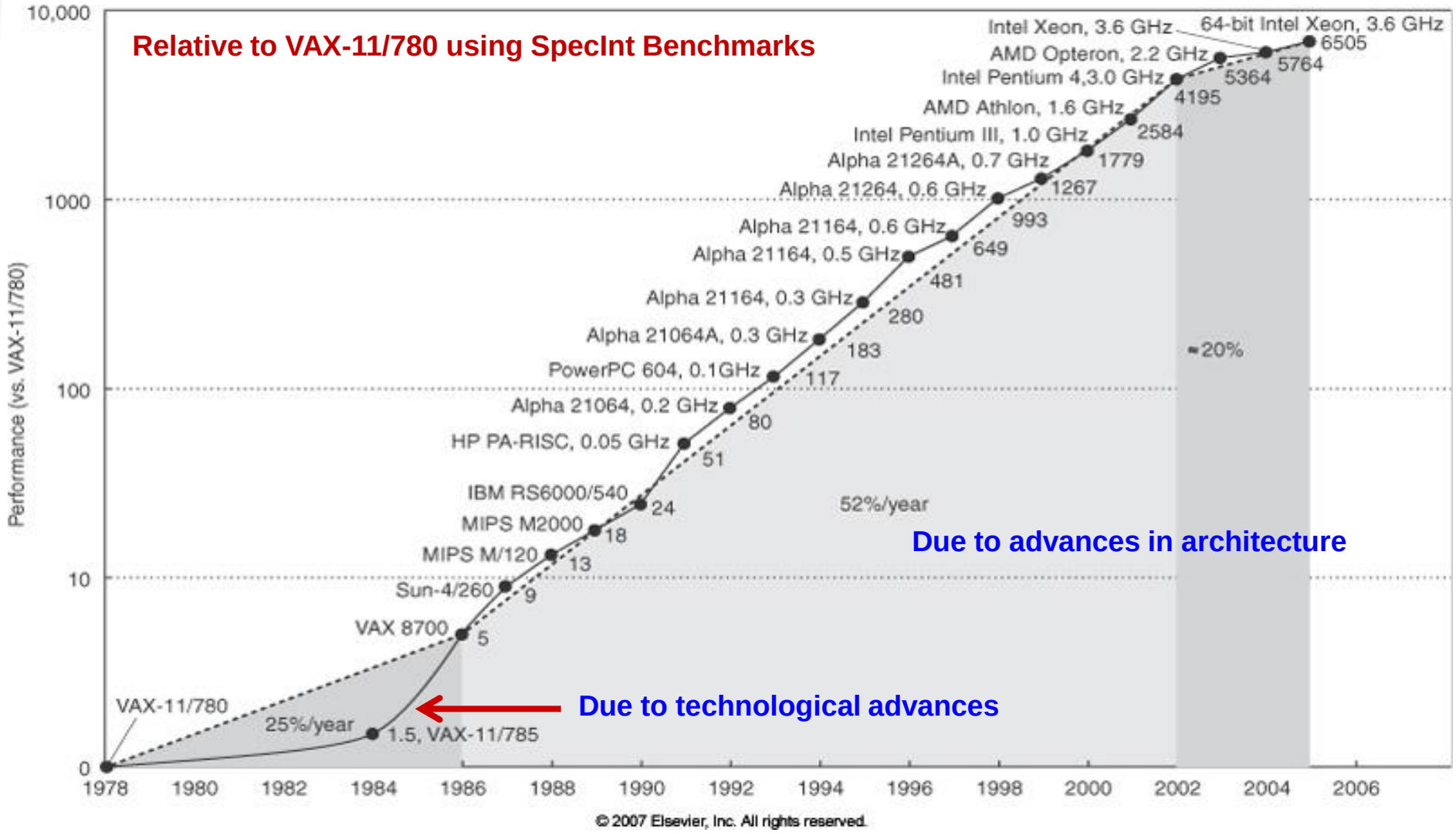
Lecture 1 – Computer Architecture

- History of Computer
- CPU and Instruction Execution
- Fundamental of Computer Design
- Designing for Performance

Fundamentals of Computer Design

- Trends in Technology
- Trends in Cost, Power and Performance
- Dependability
- Measuring and Reporting Performance
- Quantitative Principles of Computer Design

Microprocessor Performance Trends

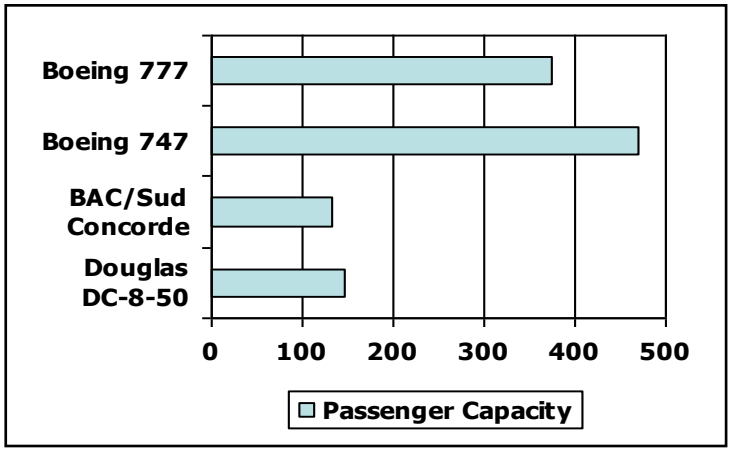


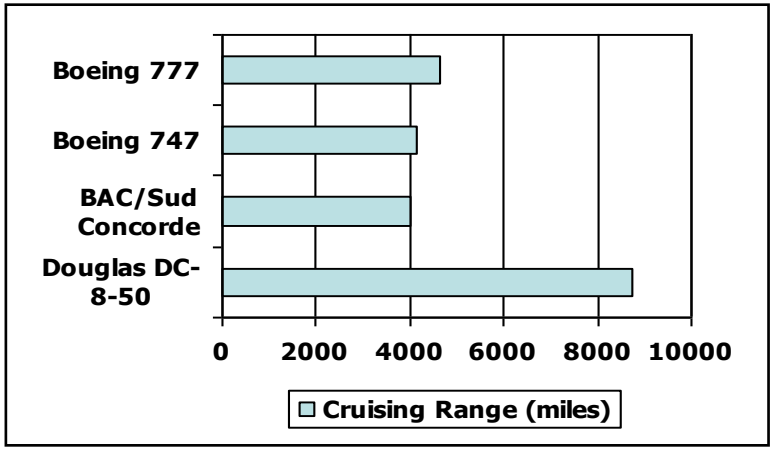
Performance Metrics

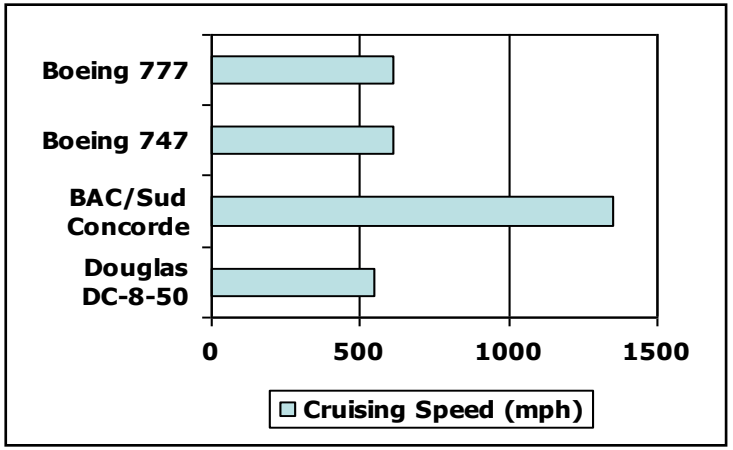
- Purchasing perspective
  - given a collection of machines, which has the
    - best performance ?
    - least cost ?
    - best cost/performance ?
- Design perspective
  - faced with design options, which has the
    - best performance improvement ?
    - least cost ?
    - best cost/performance ?
- Both require
  - basis for comparison
  - metric for evaluation
- Our goal is to understand what factors in the architecture contribute to overall system performance and the relative importance (and cost) of these factors

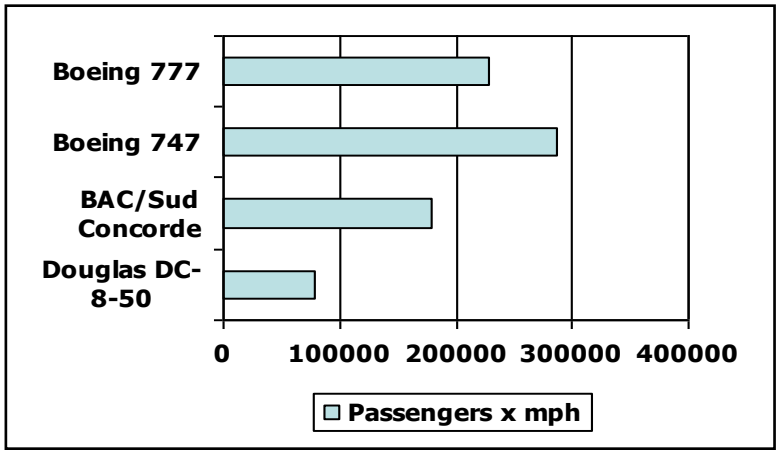
Defining Performance

- Which airplane has the best performance?









Throughput versus Response Time

- Response time (aka execution time) – the time between the start and the completion of a task
  - How long it takes to do a task (including disk accesses, IO activities, OS overhead, CPU execution time, …)
  - Important to individual users
- Throughput (bandwidth) – the total amount of work done in a given time
  - e.g., tasks/transactions/… per hour
  - Important to data center managers
- How are response time and throughput affected by
  - Replacing the processor with a faster version?
  - Adding more processors?
- Will need different performance metrics as well as a different set of applications to benchmark embedded and desktop computers, which are more focused on response time, versus servers, which are more focused on throughput

- We'll focus on response time for now…

Defining (Speed) Performance

- To maximize performance, need to minimize execution time

$$\text{performance}_X = \frac{1}{\text{execution\_time}_X}$$

If X is n times faster than Y, then

$$\frac{\text{performance}_X}{\text{performance}_Y} = \frac{\text{execution\_time}_Y}{\text{execution\_time}_X} = n$$

- Decreasing response time almost always improves throughput

Relative Performance Example

- If computer A runs a program in 10 seconds and computer B runs the same program in 15 seconds, how much faster is A than B?
  - We know that A is n times faster than B if

$$\frac{\text{performance}_A}{\text{performance}_B} = \frac{\text{execution\_time}_B}{\text{execution\_time}_A} = n$$

  - The performance ratio is $\frac{15}{10} = 1.5$
  - So A is 1.5 times faster than B

Measuring Performance

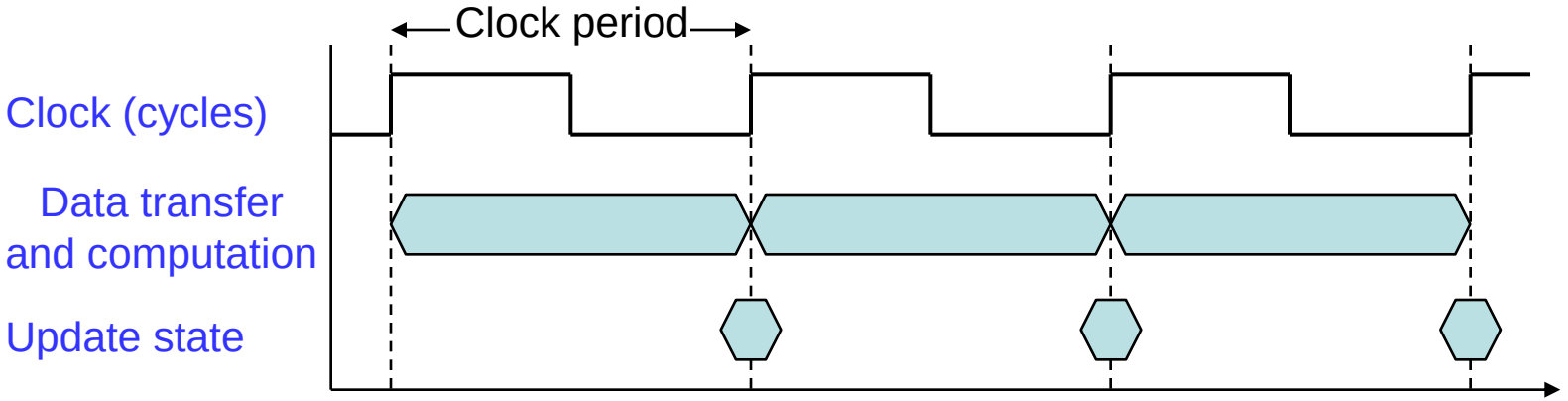
- Elapsed time (wall clock time, response time)
  - Total response time, including all aspects
    - Processing, I/O, OS overhead, idle time
  - Determines system performance
- CPU time
  - Time the CPU spends working on a task
    - Discounts I/O time, other jobs' shares
  - Comprises user CPU time and system CPU time
  - User CPU time determines CPU performance

- Different programs are affected differently by CPU and system performance

Review: CPU Clocking

- Operation of digital hardware governed by a constant-rate clock
- Clock period: duration of a clock cycle
  - e.g., 250ps = 0.25ns = 250×10^{-12}s
- Clock frequency (rate): cycles per second, is inverse of clock cycle time (clock period)
  - e.g., 4.0GHz = 4000MHz = 4.0×10^{9}Hz

CC = 1 / CR



10 nsec clock cycle => 100 MHz clock rate

1 nsec (10^{-9}) clock cycle => 1 GHz (10^{9}) clock rate

200 psec clock cycle => 5 GHz clock rate

Performance Factors

- CPU execution time (CPU time)

$$\text{CPU execution time for a program} = \text{\# CPU clock cycles for a program} \times \text{clock cycle time}$$

or

$$\text{CPU execution time for a program} = \frac{\text{\# CPU clock cycles for a program}}{\text{clock rate}}$$

  - Can improve performance by reducing either the number of clock cycles required for a program or the length of the clock cycle
    - Hardware designer must often trade off clock rate against cycle count

Improving Performance Example

- A program runs on computer A with a 2 GHz clock in 10 seconds. What clock rate must computer B run at to run this program in 6 seconds? Unfortunately, to accomplish this, computer B will require 1.2 times as many clock cycles as computer A to run the program.

To run the program in 6s, B must have twice the clock rate of A

$$\text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1}{}$$

$$\text{Clock Cycles}_A = \text{CPU Time}_A \times \text{Clock Rate}_A$$

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

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

Clock Cycles per Instruction

- Not all instructions take the same amount of time to execute
  - One way to think about execution time is that it equals the number of instructions executed multiplied by the average time per instruction

$$\text{CPU Clock Cycles for a program} = \text{Instruction Count for a program} \times \text{Clock Cycles per Instruction}$$

- Clock cycles per instruction (CPI) – the average number of clock cycles each instruction takes to execute
  - A way to compare two different implementations of the same ISA

	CPI for this instruction class		
	A	B	C
CPI	1	2	3

Using the Performance Equation

- Computers A and B implement the same ISA. Computer A has a clock cycle time of 250 ps and an effective CPI of 2.0 for some program and computer B has a clock cycle time of 500 ps and an effective CPI of 1.2 for the same program. Which computer is faster and by how much?

Each computer executes the same number of instructions, *I*, so

$$\text{CPU Time}_A = \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A$$
$$= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps}$$
$$\text{CPU Time}_B = \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B$$
$$= I \times 1.2 \times 500\text{ps} = I \times 600\text{ps}$$

Clearly, A is faster … by the ratio of execution times

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

The Performance Equation

- Our basic performance equation is then

$$\text{CPU Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

- These equations separate the three key factors that affect performance
  - Can measure the CPU execution time by running the program
  - The clock rate is usually given
  - Can measure overall instruction count by using profilers/ simulators without knowing all of the implementation details
  - CPI varies by instruction type and ISA implementation for which we must know the implementation details

CPI Example

- Alternative compiled code sequences using instructions in classes A, B, C

Class	A	B	C
CPI for class	1	2	3
IC in sequence 1	2	1	2
IC in sequence 2	4	1	1

- Sequence 1: IC = 5
  - Clock Cycles
    = 2×1 + 1×2 + 2×3
    = 10
  - Avg. CPI = 10/5 = 2.0
- Sequence 2: IC = 6
  - Clock Cycles
    = 4×1 + 1×2 + 1×3
    = 9
  - Avg. CPI = 9/6 = 1.5

CPI in More Detail

- If different instruction classes take different numbers of cycles
  - Computing the overall effective CPI is done by looking at the different types of instructions and their individual cycle counts and averaging

$$\text{Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{Instruction Count}_i)$$

Where

- IC_i is the count (percentage) of the number of instructions of class i executed
- CPI_i is the (average) number of clock cycles per instruction for that instruction class
- n is the number of instruction classes

- Weighted average CPI

$$\text{CPI} = \frac{\text{Clock Cycles}}{\text{Instruction Count}} = \sum_{i=1}^{n} \left(\text{CPI}_i \times \underbrace{\frac{\text{Instruction Count}_i}{\text{Instruction Count}}}_{\text{Relative frequency}} \right)$$

Determinates of CPU Performance

- The overall effective CPI varies by instruction mix – a measure of the dynamic frequency of instructions across one or many programs

CPU time = Instruction_count x CPI x clock_cycle

	Instruction_count	CPI	clock_cycle
Algorithm	X	X	
Programming language	X	X	
Compiler	X	X	
ISA	X	X	X
Core organization		X	X
Technology			X

A Simple Example

Op	Freq	CPI_i	Freq x CPI_i			
ALU	50%	1	.5	.5	.5	.25
Load	20%	5	1.0	.4	1.0	1.0
Store	10%	3	.3	.3	.3	.3
Branch	20%	2	.4	.4	.2	.4
			Σ = 2.2	1.6	2.0	1.95

- How much faster would the machine be if a better data cache reduced the average load time to 2 cycles?

  CPU time new = 1.6 x IC x CC so 2.2/1.6 means 37.5% faster

- How does this compare with using branch prediction to save a cycle off the branch time?

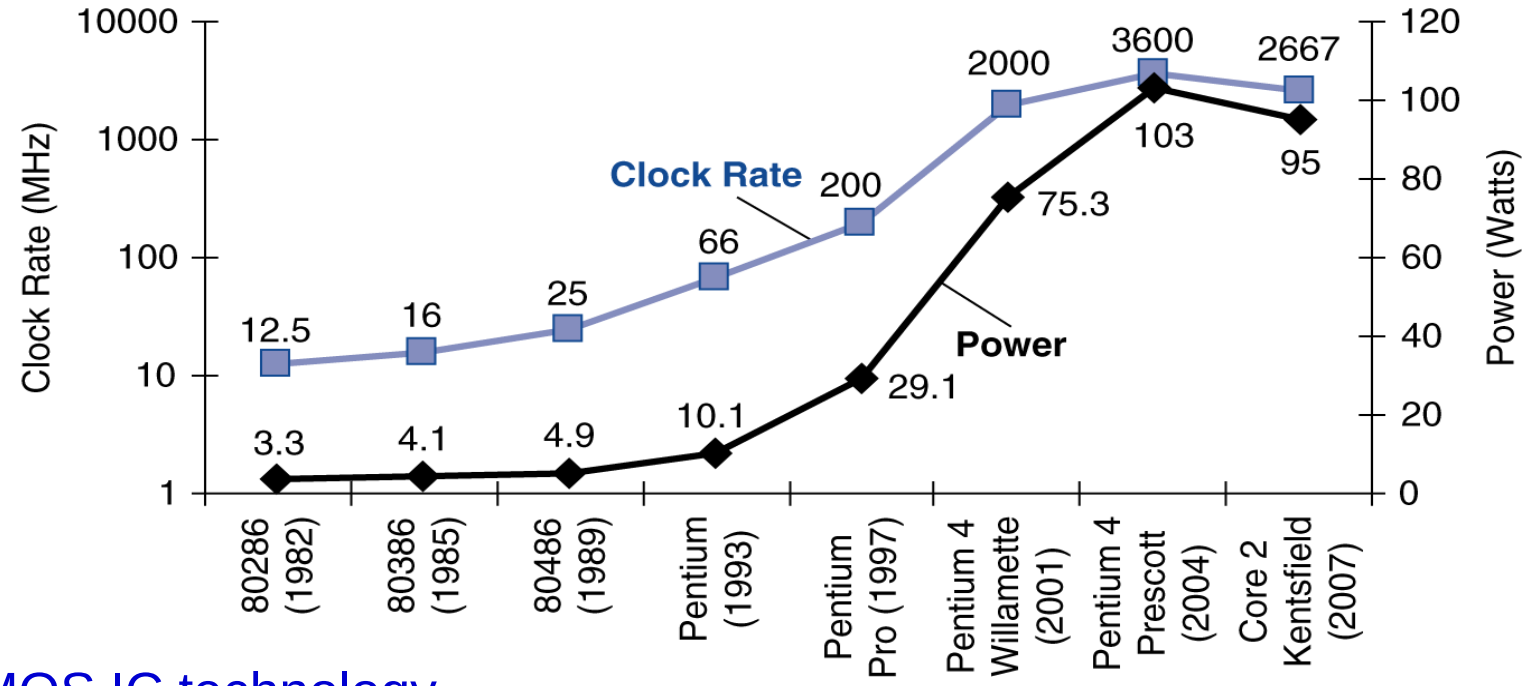
  CPU time new = 2.0 x IC x CC so 2.2/2.0 means 10% faster

- What if two ALU instructions could be executed at once?

  CPU time new = 1.95 x IC x CC so 2.2/1.95 means 12.8% faster

Other Performance Metrics

- Power consumption – especially in the embedded market where battery life is important
  - For power-limited applications, the most important metric is energy efficiency



In CMOS IC technology

$$\text{Power} = \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency}$$

×30 | 5V → 1V | ×1000

Power and Energy

- For CMOS, traditional dominant energy consumption has been in switching transistors, called *dynamic power*

$$Power_{dynamic} = 1/2 \times CapacitiveLoad \times Voltage^2 \times FrequencySwitched$$

- For mobile devices, energy better metric

$$Energy_{dynamic} = CapacitiveLoad \times Voltage^2$$

- For fixed task, slowing clock rate (frequency switched) reduces power, but not energy
- Capacitive load, a function of number of transistors connected to output and technology, which determines capacitance of wires and transistors
- Dropping voltage helps both, dropped from 5V to 1V
- Turn off clock to save energy & dynamic power

Reducing Power

- Suppose a new CPU has
  - 85% of capacitive load of old CPU
  - 15% voltage and 15% frequency reduction

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

- The power wall
  - We can't reduce voltage further
  - We can't remove more heat
- How else can we improve performance?

Static Power

- Because leakage current flows even when a transistor is off, now *static power* important too

$$Power_{static} = Current_{static} \times Voltage$$

- Leakage current increases in processors with smaller transistor sizes
- Increasing the number of transistors increases power even if they are turned off
- In 2006, goal for leakage is 25% of total power consumption; high performance designs at 40%
- Very low power systems even gate voltage to inactive modules to control loss due to leakage

Workloads and Benchmarks

- Workload – a set of programs run on a computer that is either the actual collection of applications run by a user or constructed from a real programs to approximate such a mix
- Benchmarks – a set of programs that form a "workload" specifically chosen to measure performance
- SPEC (System Performance Evaluation Cooperative) creates standard sets of benchmarks for CPU, I/O, Web, … starting with SPEC89.
  - www.spec.org
- The latest is SPEC CPU2006 which consists of 12 integer benchmarks (CINT2006) and 17 floating-point benchmarks (CFP2006)
  - Elapsed time to execute a selection of programs
  - Negligible I/O, so focuses on CPU performance
  - Normalize relative to reference machine

- There are also benchmark collections for power workloads (SPECpower_ssj2008), for mail workloads (SPECmail2008), for multimedia workloads (mediabench), …

SPEC CINT2006 on Barcelona (CC = 0.4 x 10^9)

Name	IC x 10^9	CPI	ExTime	RefTime	SPEC ratio
perl	2,1118	0.75	637	9,770	15.3
bzip2	2,389	0.85	817	9,650	11.8
gcc	1,050	1.72	724	8,050	11.1
mcf	336	10.00	1,345	9,120	6.8
go	1,658	1.09	721	10,490	14.6
hmmer	2,783	0.80	890	9,330	10.5
sjeng	2,176	0.96	837	12,100	14.5
libquantum	1,623	1.61	1,047	20,720	19.8
h264avc	3,102	0.80	993	22,130	22.3
omnetpp	587	2.94	690	6,250	9.1
astar	1,082	1.79	773	7,020	9.1
xalancbmk	1,058	2.70	1,143	6,900	6.0
Geometric Mean					11.7

Comparing and Summarizing Performance

- How do we summarize the performance for benchmark set with a single number?
  - Old method: the average of execution times that is directly proportional to total execution time is the arithmetic mean (AM)

$$AM = \frac{1}{n}\sum_{i=1}^{n} Time_i$$

  - Where $Time_i$ is the execution time for the ith program of a total of n programs in the workload
  - A smaller mean indicates a smaller average execution time and thus improved performance

  - First the execution times are normalized giving the "SPEC ratio" (bigger is faster, i.e., SPEC ratio is the inverse of execution time)
  - The SPEC ratios are then "averaged" using the geometric mean (GM)

$$GM = \sqrt[n]{\prod_{i=1}^{n} \text{Execution time ratio}_i}$$

- Guiding principle in reporting performance measurements is reproducibility – list everything another experimenter would need to duplicate the experiment (version of the operating system, compiler settings, input set used, specific computer configuration (clock rate, cache sizes and speed, memory size and speed, etc.))

CINT2006 for Opteron X4 2356

Name	Description	IC×10^9	CPI	Tc (ns)	Exec time	Ref time	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

High cache miss rates

SPEC Power Benchmark

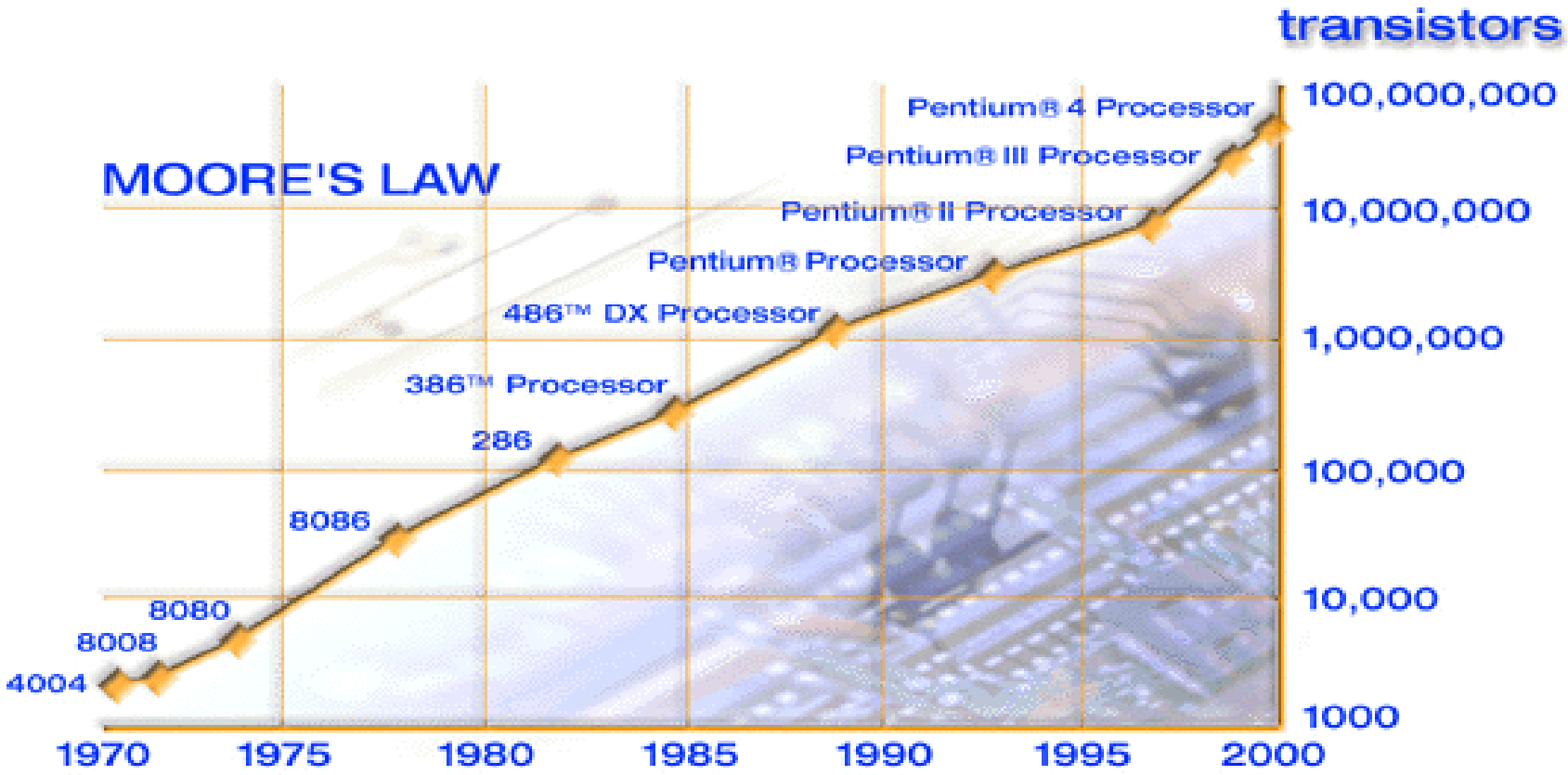
- Power consumption of server at different workload levels
  - Performance: ssj_ops/sec
  - Power: Watts (Joules/sec)

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

SPECpower_ssj2008 for X4

Target Load %	Performance (ssj_ops/sec)	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%	920,350	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**

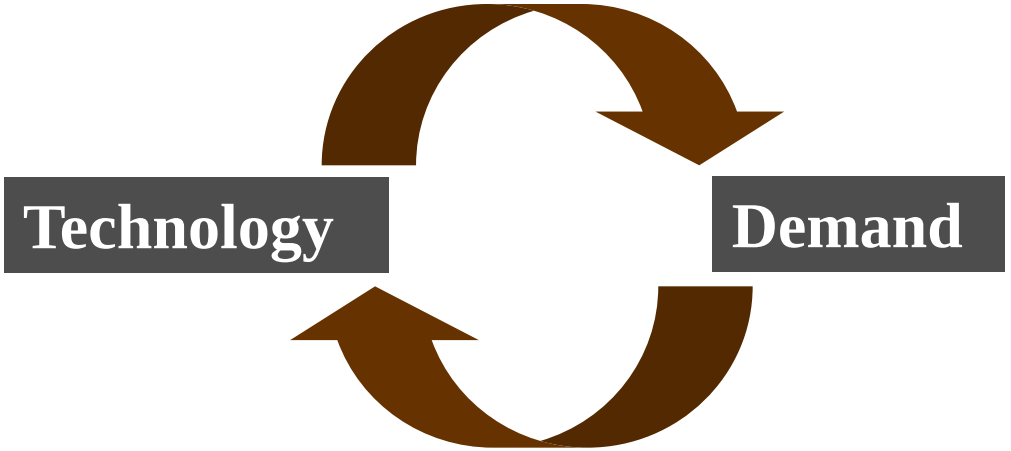
Design Complexity



Exponential Growth – doubling of transistors every couple of years

Technology and Demand

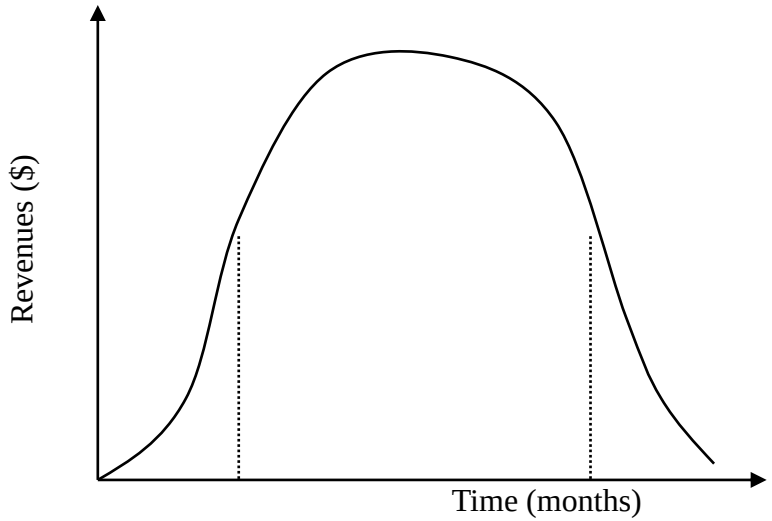
# of transistors are doubling every 2 years



Communication, multimedia, entertainment, networking

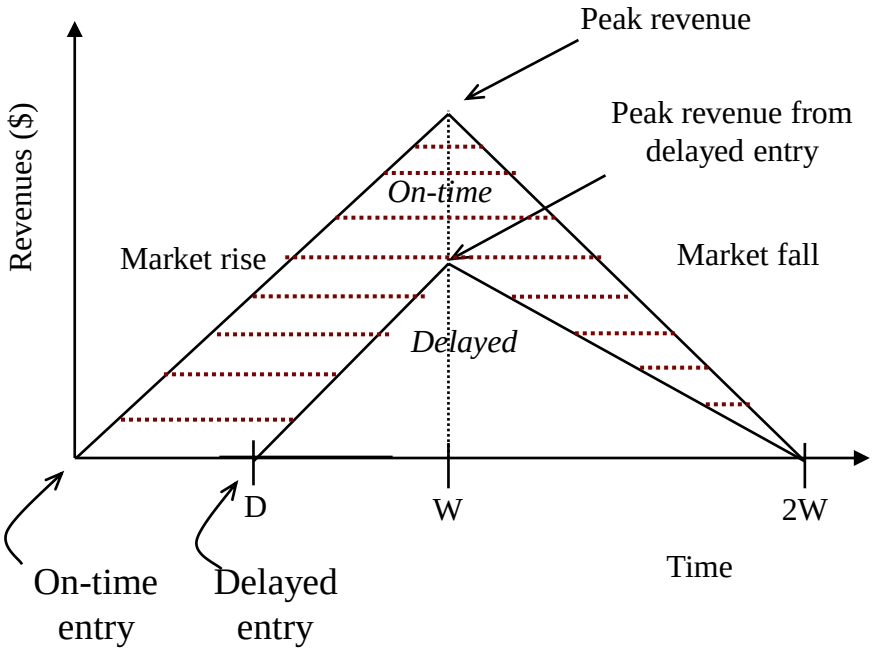
Exponential growth of design complexity ➡ verification complexity

Time-to-Market



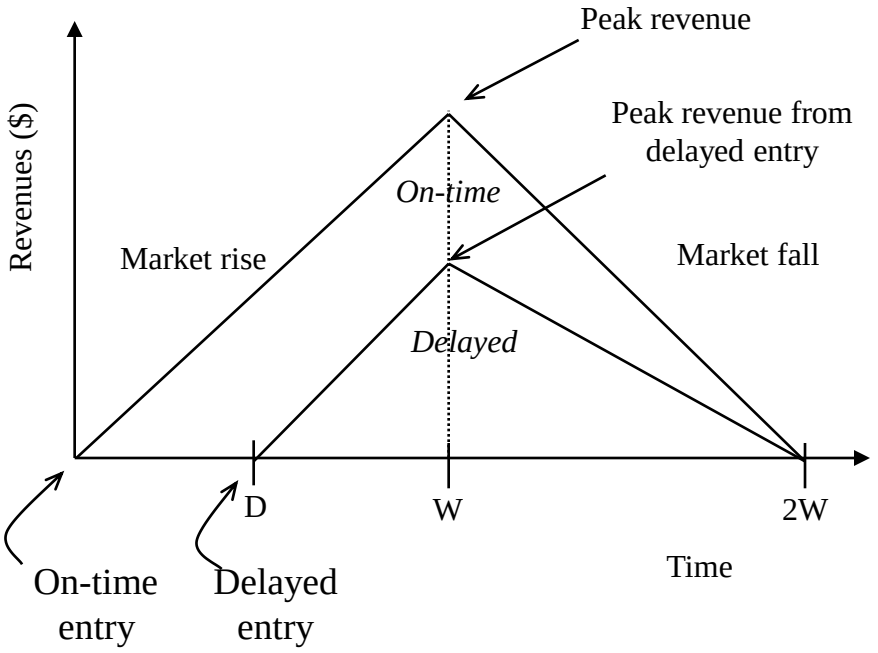
- Time required to develop a product to the point it can be sold to customers
- Market window
  - Period during which the product would have highest sales
- Average time-to-market constraint is about 8 months
- Delays can be costly

Losses due to Delayed Market Entry



- Simplified revenue model
  - Product life = 2W, peak at W
  - Time of market entry defines a triangle, representing market penetration
  - Triangle area equals revenue
- Loss
  - The difference between the on-time and delayed triangle areas (shaded region)

Delayed Market Entry (cont.)



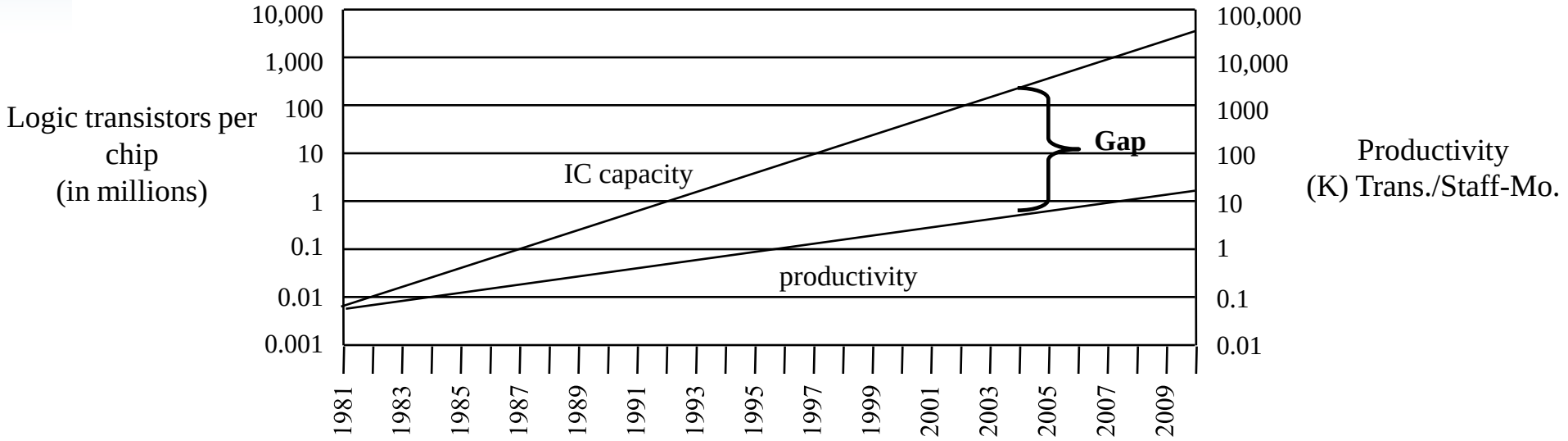
- Area = 1/2 * base * height
  - On-time = 1/2 * 2W * W
  - Delayed = 1/2 * (W-D+W)*(W-D)
- Percentage revenue loss = $(D(3W-D)/2W^2)*100\%$
- Try some examples

1. Lifetime 2W=52 wks, delay D=4 wks Loss = $(4*(3*26 -4)/2*26^2) = 22\%$

2. Lifetime 2W=52 wks, delay D=10 wks Loss = $(10*(3*26 -10)/2*26^2) = 50\%$

- **Delays are costly!**

Design productivity gap

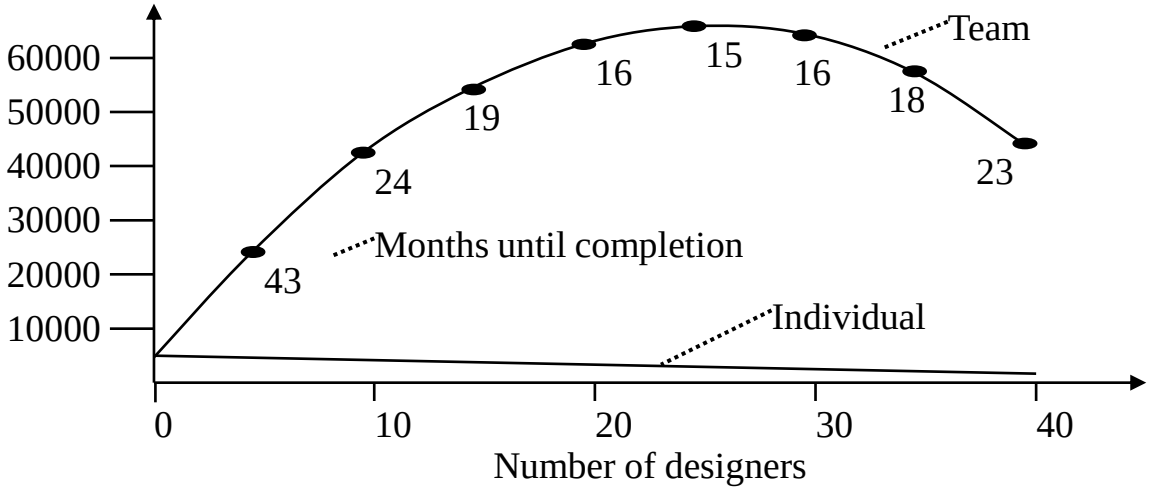


- 1981 leading edge chip required 100 man-months
  - 10,000 transistors / 100 transistors/month
- 2002 leading edge chip requires 30K man-months
  - 150,000,000 / 5000 transistors/month
- Designer cost increase from $1M to $300M

The mythical man-month

- In theory, adding designers reduces project completion time
- In reality, productivity per designer decreases due to complexities of team management and communication overhead
  - At some point, can actually lengthen project completion time!

- 1M transistors, one designer=5000 trans/month
- Each additional designer reduces for 100 trans/month



Lecture 1 – Computer Architecture

- History of Computer
- CPU and Instruction Execution
- Fundamental of Computer Design
- Designing for Performance

Computer design

- Computer design must meet
  - Functional requirements
  - Area/performance/cost/power goals
    - Optimize, evaluate, find best possible architecture
  - Consider other factors
    - Time-to-market, technology trend, …

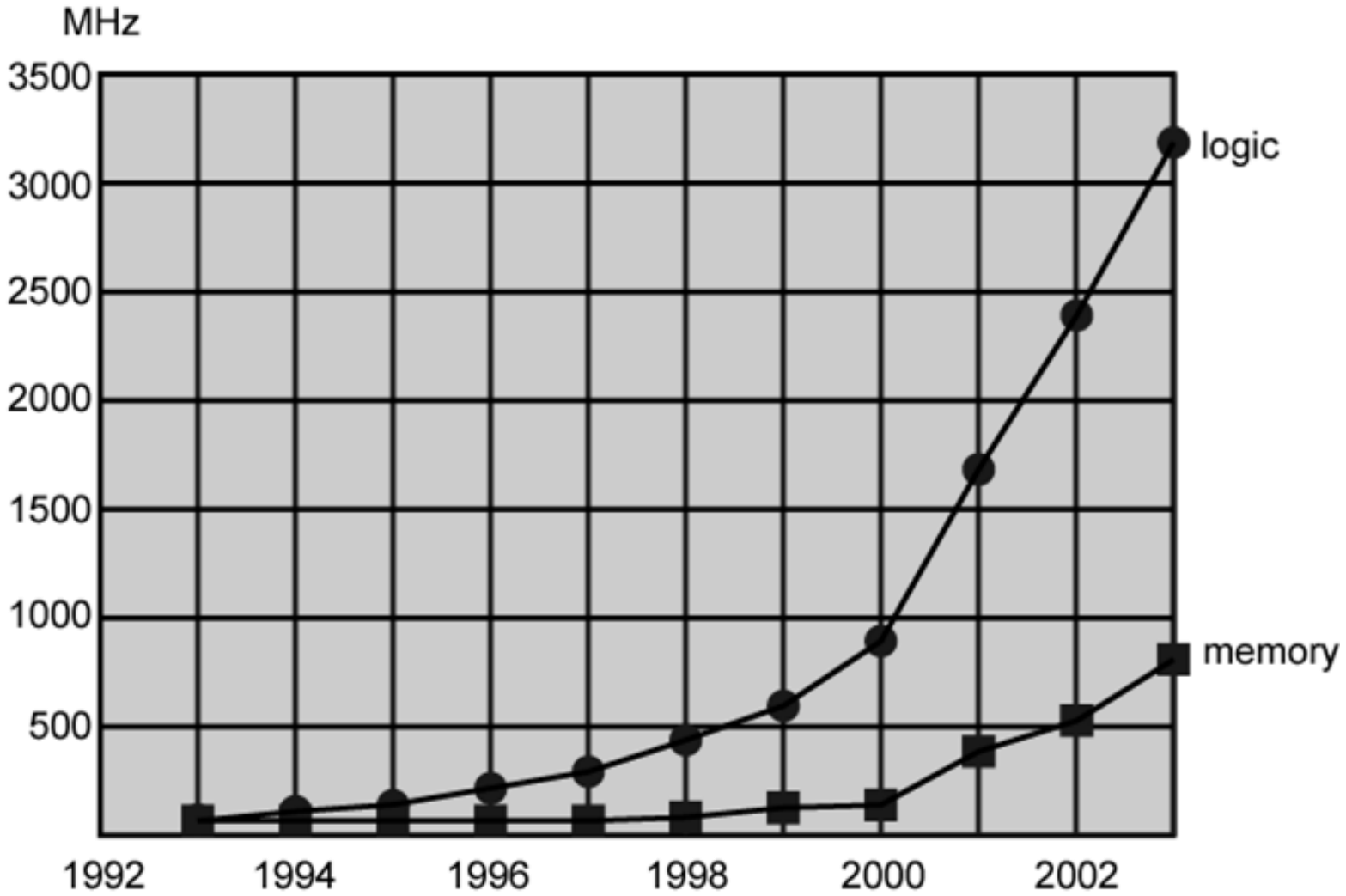
Speeding it up

- Pipelining
- On board cache
- On board L1 & L2 cache
- Branch prediction
- Data flow analysis
- Speculative execution

Performance Balance

- Processor speed increased
- Memory capacity increased
- Memory speed lags behind processor speed

Logic and Memory Performance Gap



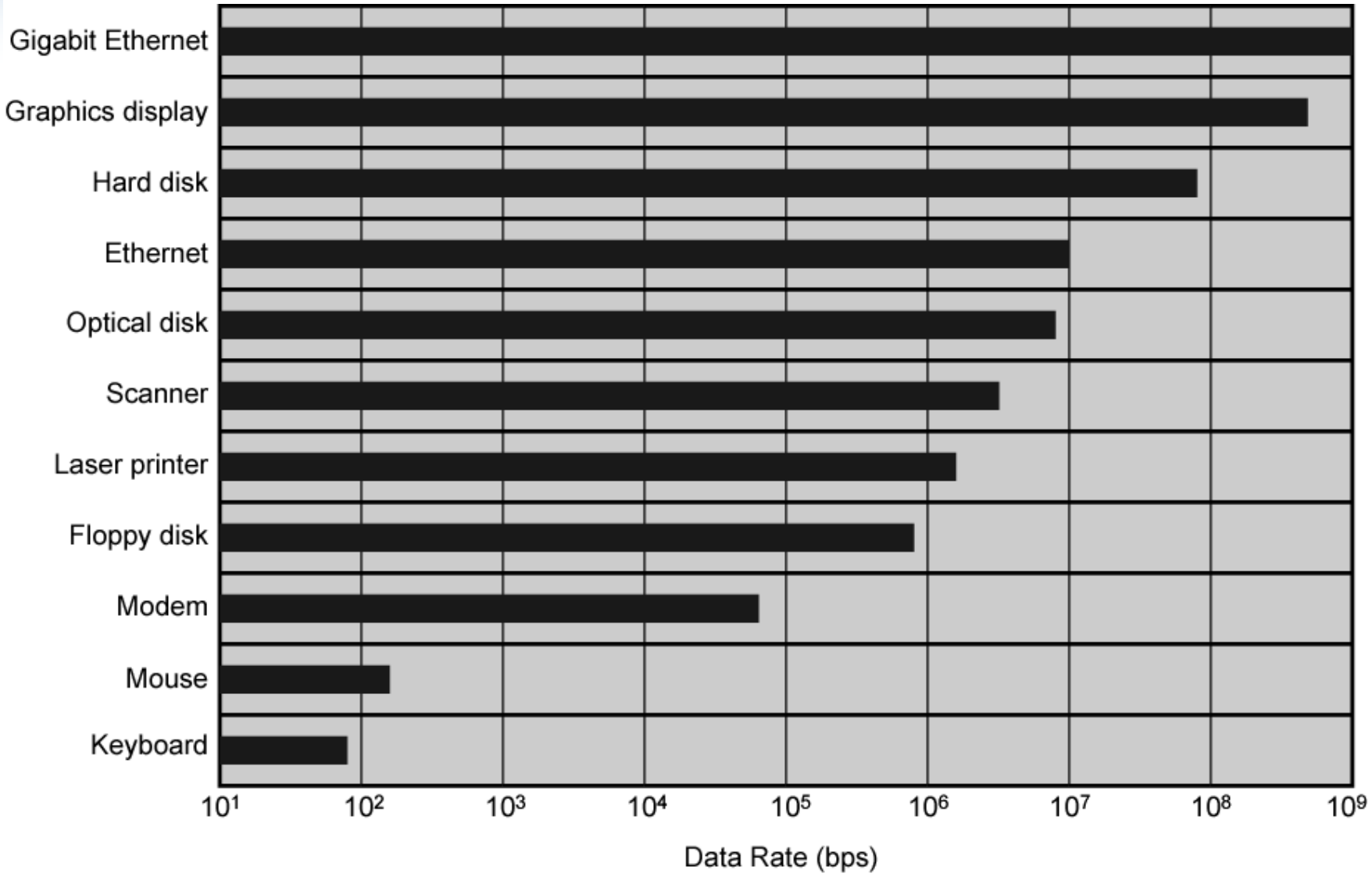
Solutions

- Increase number of bits retrieved at one time
  - Make DRAM “wider” rather than “deeper”
- Change DRAM interface
  - Cache
- Reduce frequency of memory access
  - More complex cache and cache on chip
- Increase interconnection bandwidth
  - High speed buses
  - Hierarchy of buses

I/O Devices

- Peripherals with intensive I/O demands
- Large data throughput demands
- Processors can handle this
- Problem moving data
- Solutions:
  - Caching
  - Buffering
  - Higher-speed interconnection buses
  - More elaborate bus structures
  - Multiple-processor configurations

Typical I/O Device Data Rates



Key is Balance

- Processor components
- Main memory
- I/O devices
- Interconnection structures

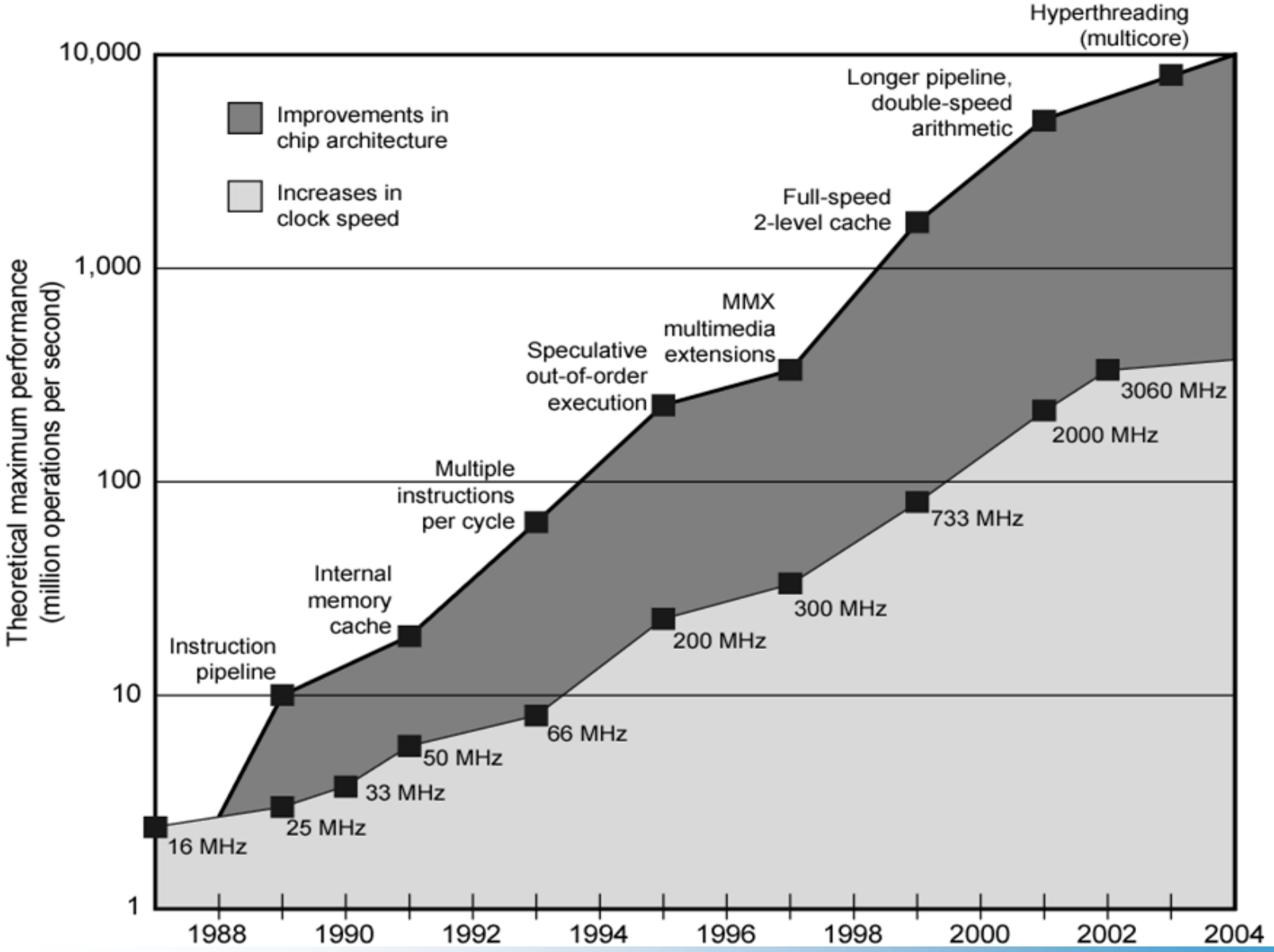
Improvements in Chip Organization and Architecture

- Increase hardware speed of processor
  - Fundamentally due to shrinking logic gate size
    - More gates, packed more tightly, increasing clock rate
    - Propagation time for signals reduced
- Increase size and speed of caches
  - Dedicating part of processor chip
    - Cache access times drop significantly
- Change processor organization and architecture
  - Increase effective speed of execution
  - Parallelism

Problems with Clock Speed and Logic Density

- Power
  - Power density increases with density of logic and clock speed
  - Dissipating heat
- RC delay
  - Speed at which electrons flow limited by resistance and capacitance of metal wires connecting them
  - Delay increases as RC product increases
  - Wire interconnects thinner, increasing resistance
  - Wires closer together, increasing capacitance
- Memory latency
  - Memory speeds lag processor speeds
- Solution:
  - More emphasis on organizational and architectural approaches

Intel Microprocessor Performance



Increased Cache Capacity

- Typically two or three levels of cache between processor and main memory
- Chip density increased
  - More cache memory on chip
    - Faster cache access
- Pentium chip devoted about 10% of chip area to cache
- Pentium 4 devotes about 50%

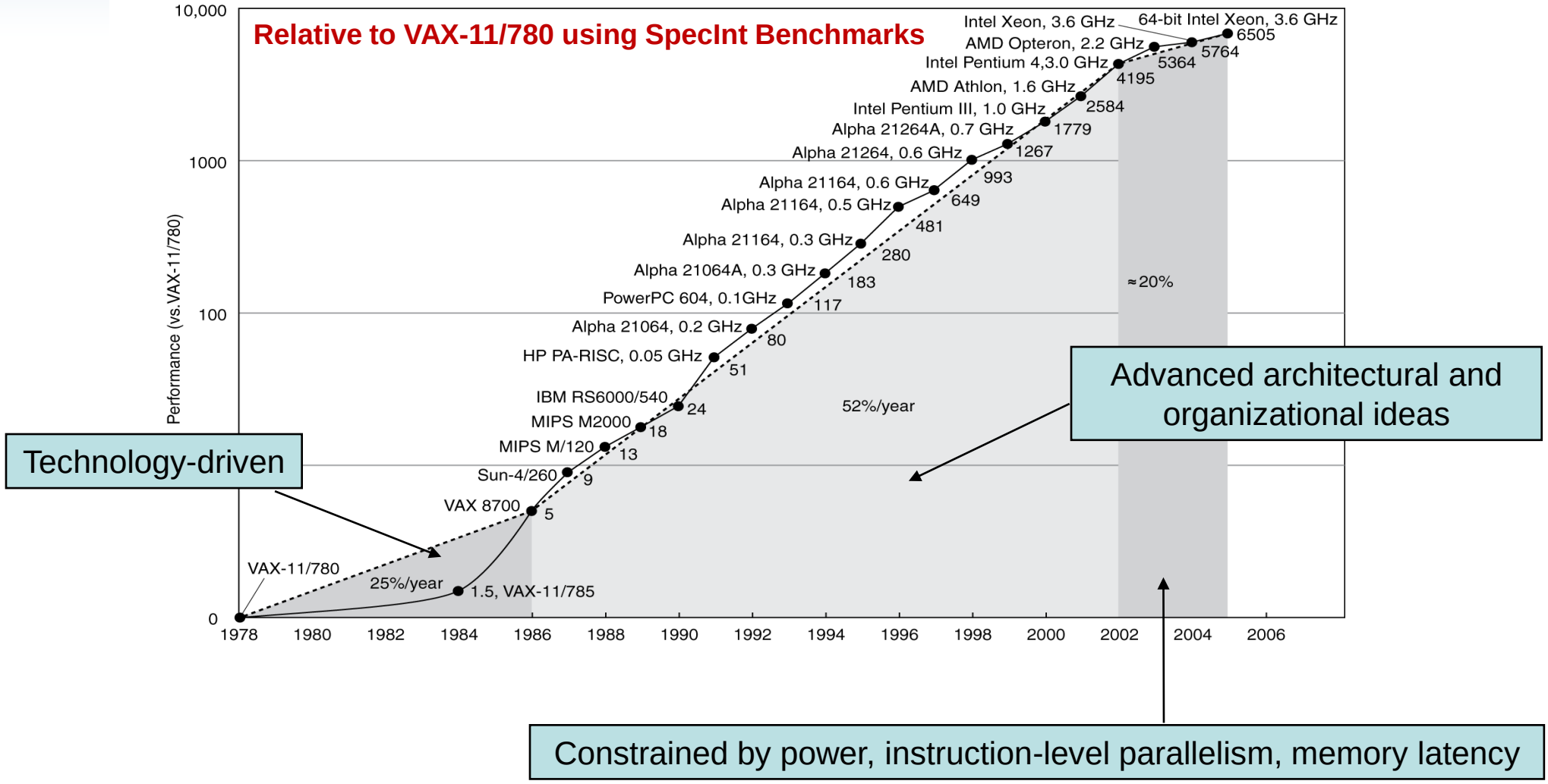
More Complex Execution Logic

- Enable parallel execution of instructions
- Pipeline works like assembly line
  - Different stages of execution of different instructions at same time along pipeline
- Superscalar allows multiple pipelines within single processor
  - Instructions that do not depend on one another can be executed in parallel

Diminishing Returns

- Internal organization of processors complex
  - Can get a great deal of parallelism
  - Further significant increases likely to be relatively modest
- Benefits from cache are reaching limit
- Increasing clock rate runs into power dissipation problem
  - Some fundamental physical limits are being reached

Uniprocessor Performance



New Approach – Multiple Cores

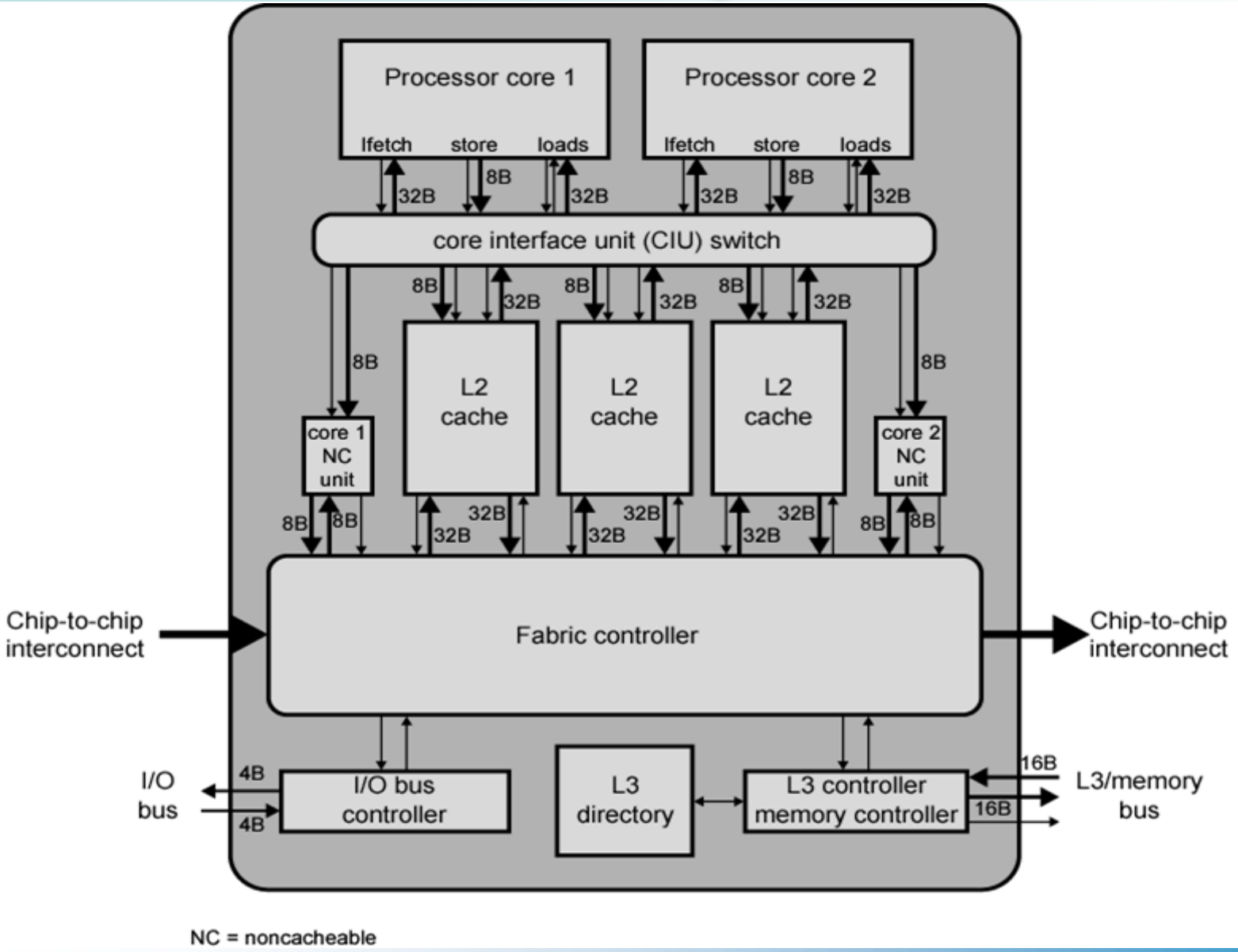
- Multiple processors on single chip
  - Large shared cache
- Within a processor, increase in performance proportional to square root of increase in complexity
- If software can use multiple processors, doubling number of processors almost doubles performance
- So, use two simpler processors on the chip rather than one more complex processor
- With two processors, larger caches are justified
  - Power consumption of memory logic less than processing logic
- Example: IBM POWER4
  - Two cores based on PowerPC

Multiprocessors

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

- Requires explicitly parallel programming
  - Compare with instruction level parallelism
    - Hardware executes multiple instructions at once
    - Hidden from the programmer
  - Hard to do
    - Programming for performance
    - Load balancing
    - Optimizing communication and synchronization

POWER4 Chip Organization



Pentium Evolution (1)

- 8080
  - first general purpose microprocessor
  - 8 bit data path
  - Used in first personal computer – Altair
- 8086
  - much more powerful
  - 16 bit
  - instruction cache, pre-fetch few instructions
  - 8088 (8 bit external bus) used in first IBM PC
- 80286
  - 16 Mbyte memory addressable
  - up from 1Mb
- 80386
  - 32 bit
  - Support for multitasking

Pentium Evolution (2)

- 80486
  - sophisticated powerful cache and instruction pipelining
  - built in maths co-processor
- Pentium
  - Superscalar
  - Multiple instructions executed in parallel
- Pentium Pro
  - Increased superscalar organization
  - Aggressive register renaming
  - branch prediction
  - data flow analysis
  - speculative execution

Pentium Evolution (3)

- Pentium II
  - MMX technology
  - graphics, video & audio processing
- Pentium III
  - Additional floating point instructions for 3D graphics
- Pentium 4
  - Note Arabic rather than Roman numerals
  - Further floating point and multimedia enhancements
- Itanium
  - 64-bit organization
  - IA-64 architecture (see later)
- Itanium 2
  - Hardware enhancements to increase speed
- See Intel web pages for detailed information on processors

PowerPC

- 1975, 801 minicomputer project (IBM) RISC
- Berkeley RISC I processor
- 1986, IBM commercial RISC workstation product, RT PC.
  - Not commercial success
  - Many rivals with comparable or better performance
- 1990, IBM RISC System/6000
  - RISC-like superscalar machine
  - POWER architecture
- IBM alliance with Motorola (68000 microprocessors), and Apple, (used 68000 in Macintosh)
- Result is PowerPC architecture
  - Derived from the POWER architecture
  - Superscalar RISC
  - Apple Macintosh
  - Embedded chip applications

PowerPC Family (1)

- 601:
  - Quickly to market.
  - 32-bit machine
- 603:
  - Low-end desktop and portable
  - 32-bit
  - Comparable performance with 601
  - Lower cost and more efficient implementation
- 604:
  - Desktop and low-end servers
  - 32-bit machine
  - Much more advanced superscalar design
  - Greater performance
- 620:
  - High-end servers
  - Full 64-bit architecture, including 64-bit registers and data paths

PowerPC Family (2)

- 740/750:
  - Also known as G3
  - Two levels of cache on chip
- G4:
  - Increases parallelism and internal speed
- G5:
  - Improvements in parallelism and internal speed
  - 64-bit organization

PowerPC Processors

	First ship date	Clock speeds	L1 cache	L2 cache	Number of transistors (10^6)
601	1993	50 - 120 MHz	—	—	2.8
603/603e	1994	100 - 300 MHz	16 KByte instr 16 KByte data	—	1.6 - 2.6
604/604e	1994	166 - 350 MHz	32 KByte inst 32 KByte data	—	3.6 - 5.1
740/750 (G3)	1997	200 - 366 MHz	32 KByte instr 32 KByte data	256 KByte - 1 MByte	6.35
G4	1999	500 MHz	32 KByte instr 32 KByte data	256 KByte - 1 MByte	
G5	2003	2.5 GHz	64 KByte instr 32 KByte data	512 KByte	58