Computer Organization and Architecture

Computer Evolution and Performance

Xiangzhong FANG
xzfang@sjtu.edu.cn

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

Straight ring/Overbeck counter

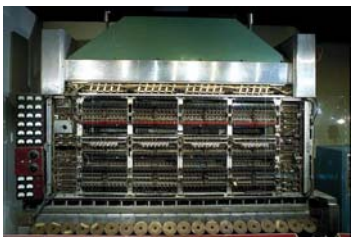
State	Q0	Q1	Q2	Q3
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
0	1	0	0	0

von Neumann/Turing

Father of computers
Von Neumann, 1903-1957

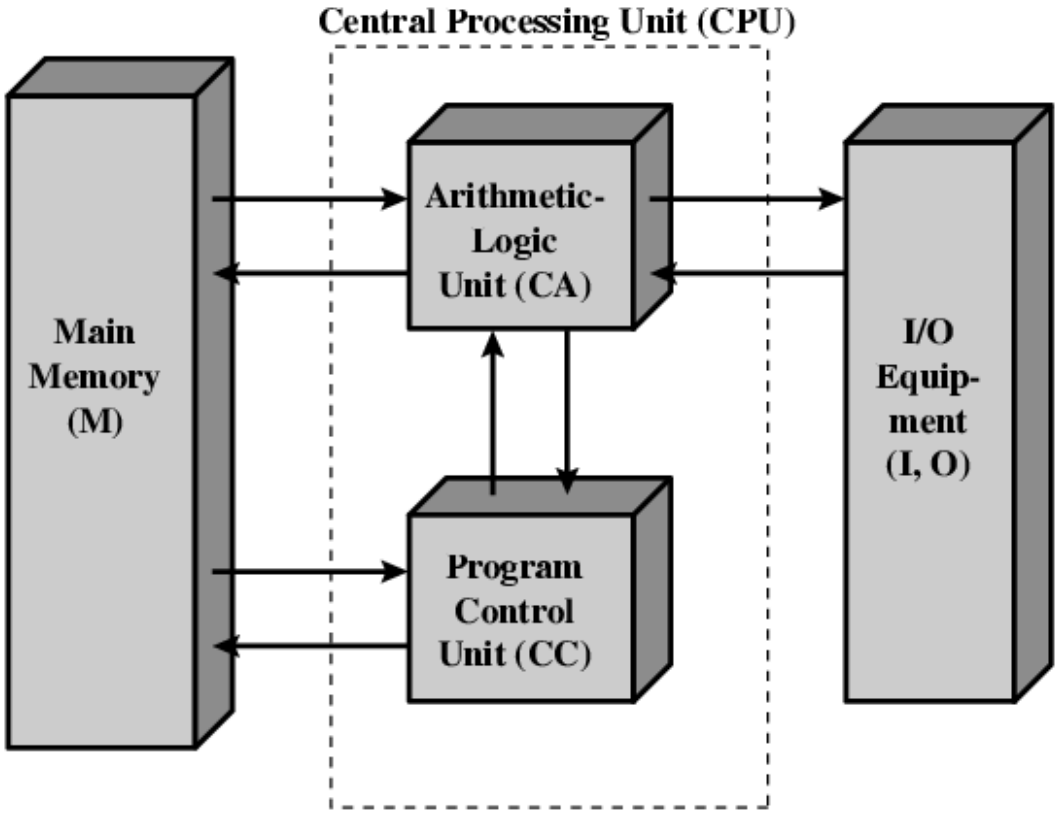
- Stored Program concept
- Main memory storing programs and data
- ALU operating on **binary data**
- Control unit interpreting instructions from memory and executing
- Input and output equipment operated by control unit
- Princeton Institute for Advanced Studies
  - —IAS computer
- Completed 1952

Alan Turing, 1912-1954



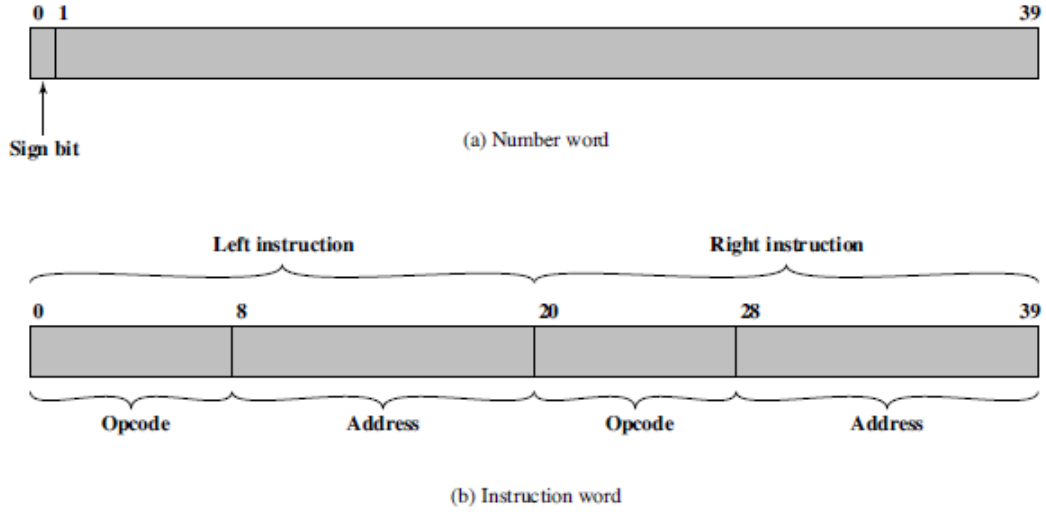
IAS computer

Structure of von Neumann machine

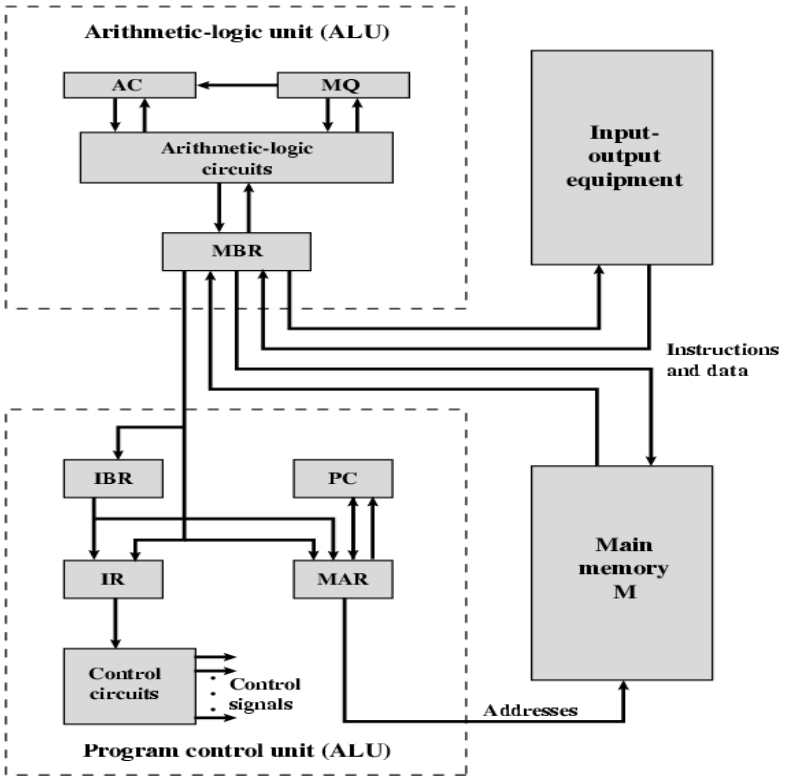


IAS - details

- 1000 x 40 bit words
  - —Binary number
  - —2 x 20 bit instructions



Structure of IAS – detail



- Memory Buffer Register
- Memory Address Register
- Instruction Register
- Instruction Buffer Register
- Program Counter
- Accumulator
- Multiplier Quotient

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

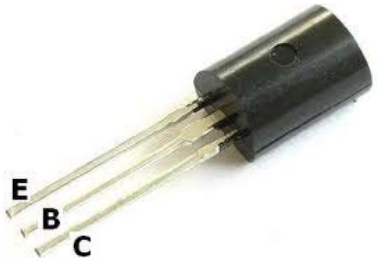
IBM

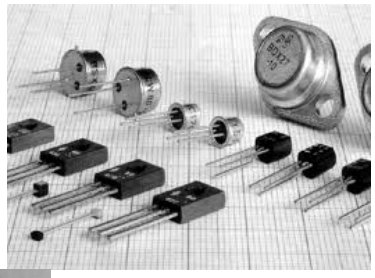
- Punched-card processing equipment
- 1953 - the 701
  - —IBM's first stored program computer
  - —Scientific calculations
- 1955 - the 702
  - —Business applications
- Lead to 700/7000 series

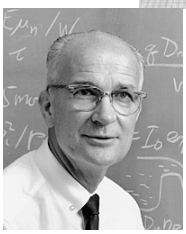
IBM 701 at Douglas Aircraft

Transistors – the 2nd generation

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







Transistor Based Computers

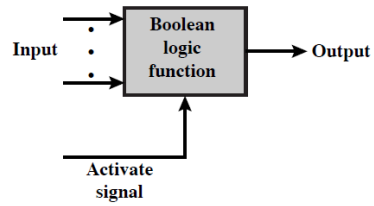
- Second generation machines
- NCR & RCA produced small transistor machines
- IBM
  - —Produced IBM 7000
- DEC - 1957
  - —Produced PDP-1

IBM 7000

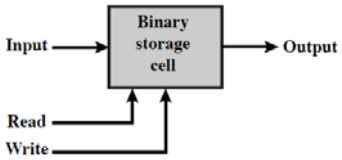
PDP-1

Microelectronics – 3rd Generation

- Literally - "small electronics"
- A computer is made up of gates, memory cells and interconnections
- These can be manufactured on a **semiconductor**
- e.g. silicon wafer

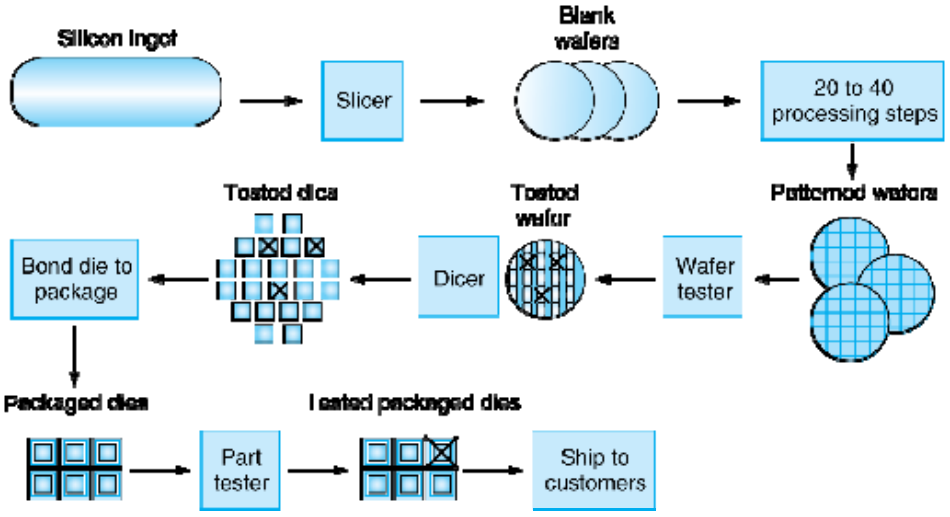


(a) Gate



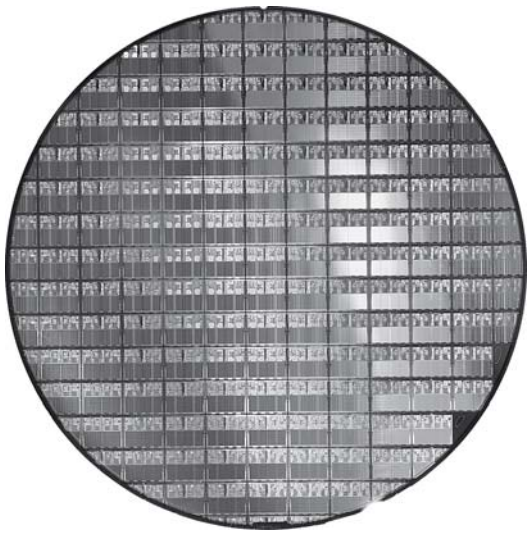
(b) Memory cell

Manufacturing ICs



- Yield: proportion of working dies per wafer

AMD Opteron X2 Wafer



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

Generations of Computer

- Vacuum tube - 1946-1957
- Transistor - 1958-1964
- Small scale integration - 1965 on
  - —Up to 100 devices on a chip
- Medium scale integration - to 1971
  - —100-3,000 devices on a chip
- Large scale integration - 1971-1977
  - —3,000 - 100,000 devices on a chip
- Very large scale integration - 1978 -1991
  - —100,000 - 100,000,000 devices on a chip
- Ultra large scale integration – 1991 -
  - —Over 100,000,000 devices on a chip

Generations of Computer

Generation	Approximate Dates	Technology	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	1,000,000
4	1972–1977	Large scale integration	10,000,000
5	1978–1991	Very large scale integration	100,000,000
6	1991–	Ultra large scale integration	1,000,000,000

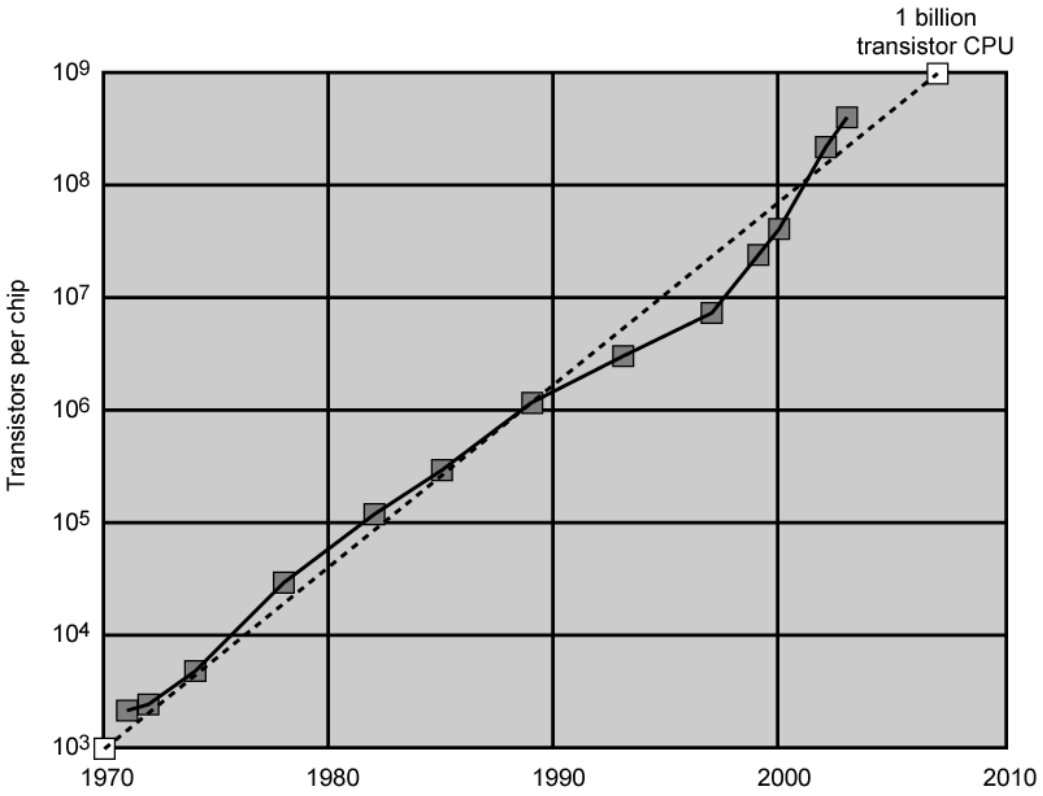
Moore's Law

- Increased density of components on chip
- Since 1970's development has slowed a little
  - Number of transistors doubles every 18 months
- Cost of a chip has remained almost unchanged
  - That means?

Gordon Moore – co-founder of Intel

Number of transistors on a chip will double every year

Growth in CPU Transistor Count



Consequences of Moore's Law

- Higher packing density means shorter electrical paths, giving higher performance
- Smaller size gives increased flexibility
- Reduced power and cooling requirements
- Fewer interconnections increases reliability

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
- Multiplexed switch structure

This series offered a standard interface and a wide choice of standard peripheral devices.
The standard interface made it easy to change the peripherals when the customer's needs changed.

Example members of System/360

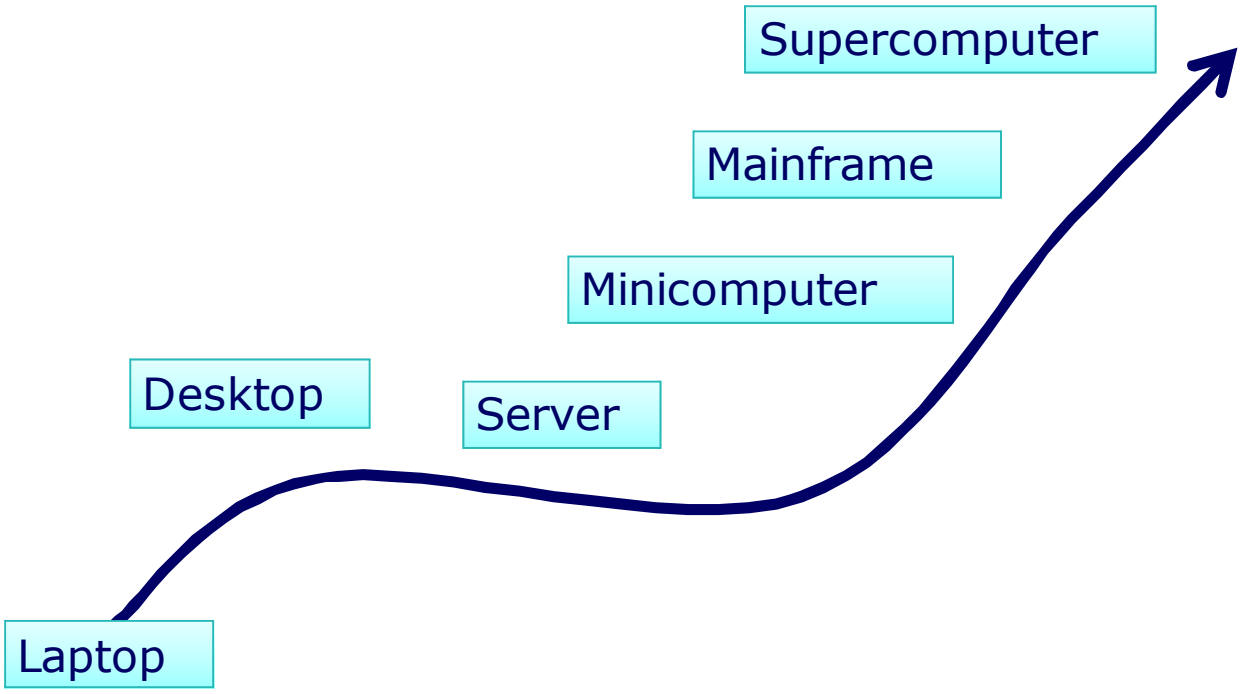
Characteristic	Model 30	Model 40	Model 50	Model 65	Model 75
Maximum memory size (bytes)	64K	256K	256K	512K	512K
Data rate from memory (Mbytes/sec)	0.5	0.8	2.0	8.0	16.0
Processor cycle time μs)	1.0	0.625	0.5	0.25	0.2
Relative speed	1	3.5	10	21	50
Maximum number of data channels	3	3	4	6	6
Maximum data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

DEC PDP-8

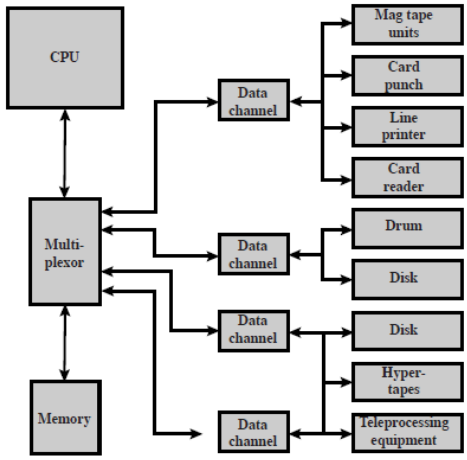
- 1964
- First minicomputer (after miniskirt!)
  - —Did not need air conditioned room
  - —Small enough to sit on a lab bench
- $16,000
  - —$100k+ for IBM 360

- Embedded applications & OEM
- BUS STRUCTURE

Spectrum of Computers

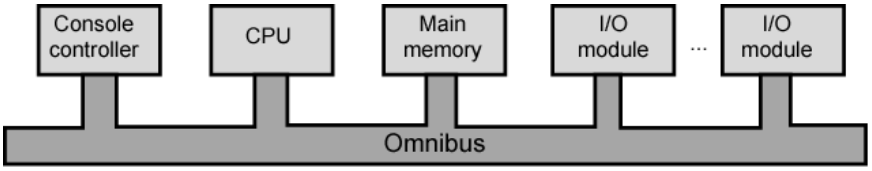


Central Control vs Bus



Multiplexed switch structure

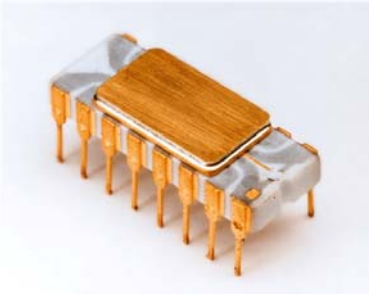
BUS STRUCTURE



Intel

Later generations

- 1971 - 4004
  - —First **microprocessor** (All CPU components on a single chip)
  - —4 bit
- Followed in 1972 by 8008
  - —8 bit
  - —Both designed for specific applications
- 1974 - 8080
  - —Intel's first general purpose microprocessor



Intel processors in 1970s and 1980s

(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 KB	64 KB	1 MB	1 MB

(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 MB	4 GB	16 MB	4 GB
Virtual memory	1 GB	64 TB	64 TB	64 TB
Cache	—	—	—	8 kB

Intel processors in 1990s and beyond

(c) 1990s Processors

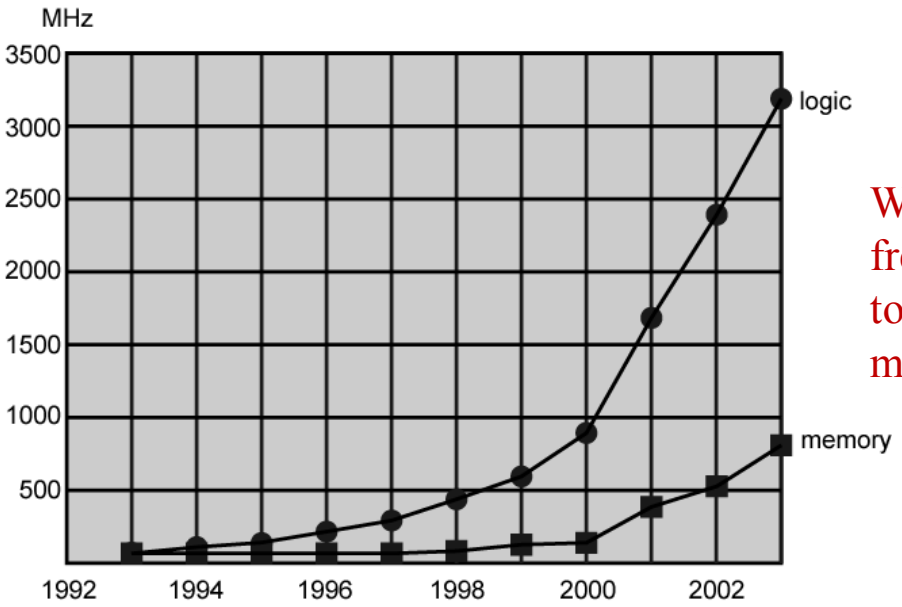
	486TM SX	Pentium	Pentium Pro	Pentium II
Introduced	1991	1993	1995	1997
Clock speeds	16 MHz–33 MHz	60 MHz–166 MHz,	150 MHz–200 MHz	200 MHz–300 MHz
Bus width	32 bits	32 bits	64 bits	64 bits
Number of transistors	1.185 million	3.1 million	5.5 million	7.5 million
Feature size (μm)	1	0.8	0.6	0.35
Addressable memory	4 GB	4 GB	64 GB	64 GB
Virtual memory	64 TB	64 TB	64 TB	64 TB
Cache	8 kB	8 kB	512 kB L1 and 1 MB L2	512 kB L2

(d) Recent Processors

	Pentium III	Pentium 4	Core 2 Duo	Core 2 Quad
Introduced	1999	2000	2006	2008
Clock speeds	450–660 MHz	1.3–1.8 GHz	1.06–1.2 GHz	3 GHz
Bus sidth	64 bits	64 bits	64 bits	64 bits
Number of transistors	9.5 million	42 million	167 million	820 million
Feature size (nm)	250	180	65	45
Addressable memory	64 GB	64 GB	64 GB	64 GB
Virtual memory	64 TB	64 TB	64 TB	64 TB
Cache	512 kB L2	256 kB L2	2 MB L2	6 MB L2

Performance Gap between Processor & Memory

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



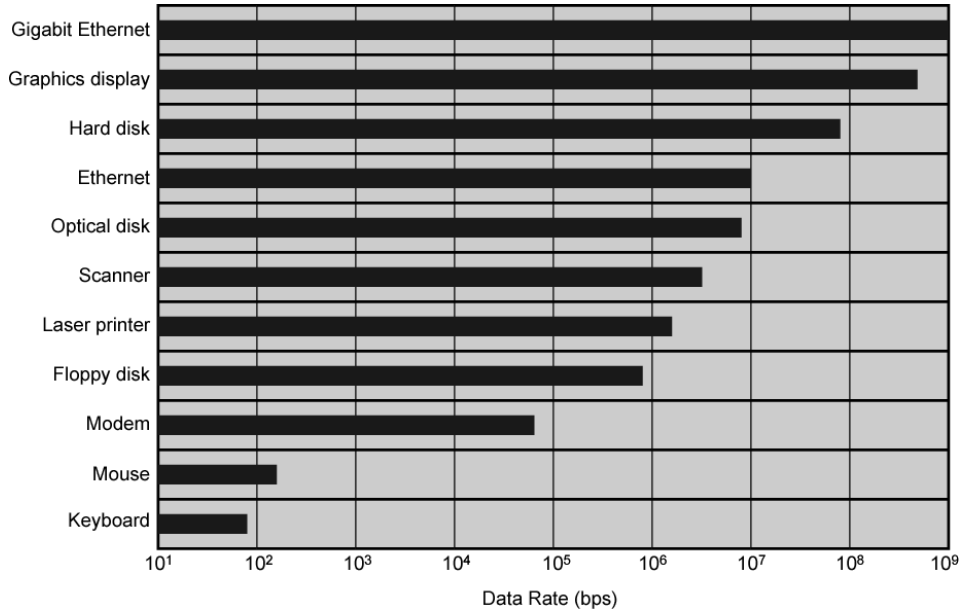
What is the frequency of today's memory?

Solutions

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

I/O Devices

- Peripherals with intensive I/O demands
  - —Large data throughput demands



Problem and Solutions

- **Problem** moving data between processor and peripherals

- **Solutions:**
  - —Caching
  - —Buffering
  - —Higher-speed interconnection buses
  - —More elaborate bus structures
  - —Multiple-processor configurations

Key is Balance

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

Balance the throughput and processing demands of processor, memory, IO, etc.

Speeding Microprocessor Up

How to make faster processors?

- Clock rate
- Logic density
- Pipelining
- On board L1 & L2 cache
- Branch prediction
- Data flow analysis
- Speculative execution

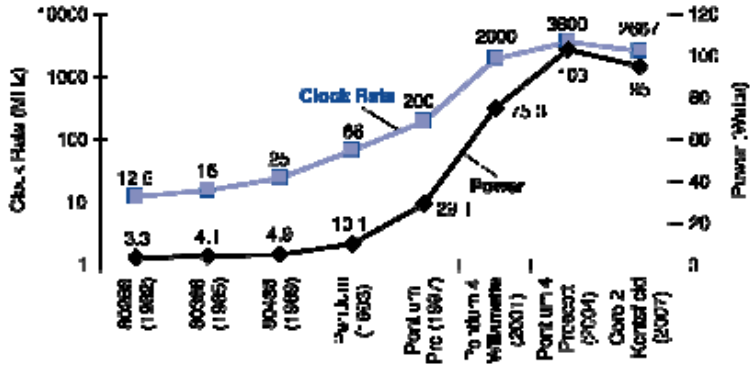
Increase hardware speed of processor

- Fundamentally due to shrinking logic gate size
  - More gates, packed more tightly, and increasing clock rate
  - Propagation time for signals reduced

Problem with Clock Speed

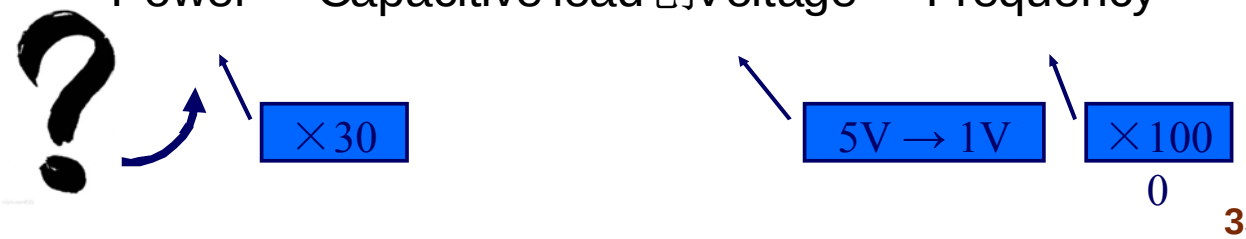
- **Power**
  - Power density increases with density of logic and clock speed
  - Dissipating heat
  - Some fundamental physical limits are being reached

Power Trends



In CMOS IC technology

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



Reducing Power

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

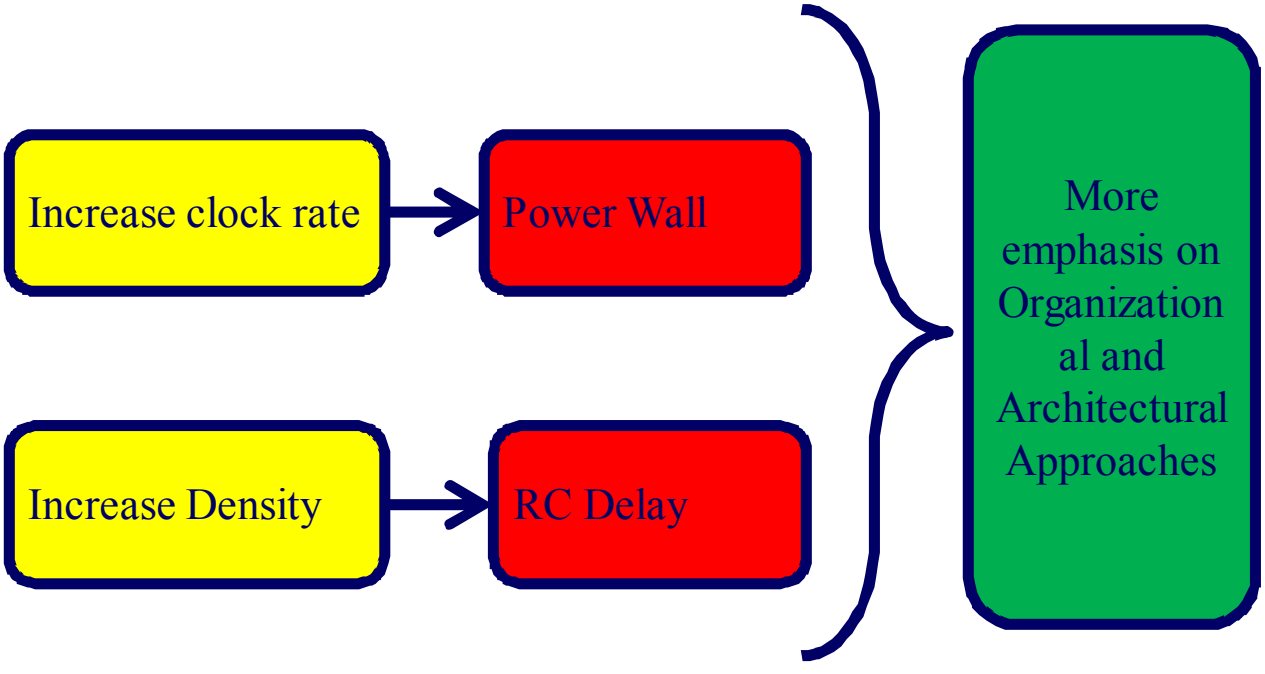
$$\frac{P_{new}}{P_{old}} = \frac{C_{old} \times 0.85 \times (V_{old} \times 0.85)^2 \times F_{old} \times 0.85}{C_{old} \times V_{old}^{\ 2} \times F_{old}} = 0.85^4 = 0.52$$

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

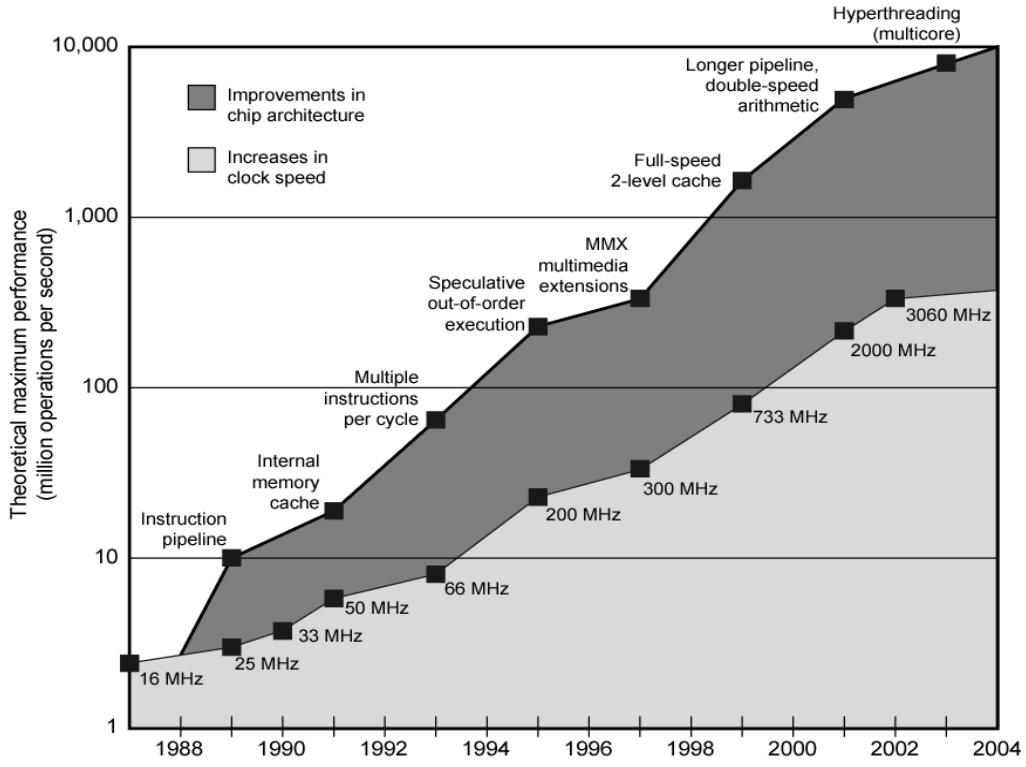
Problems with Logic Density

- **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

Solution?



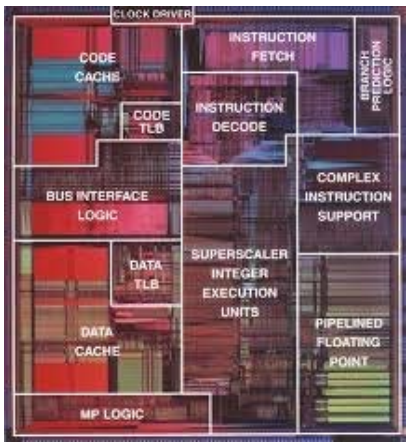
Techniques developed



Intel Microprocessor History

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

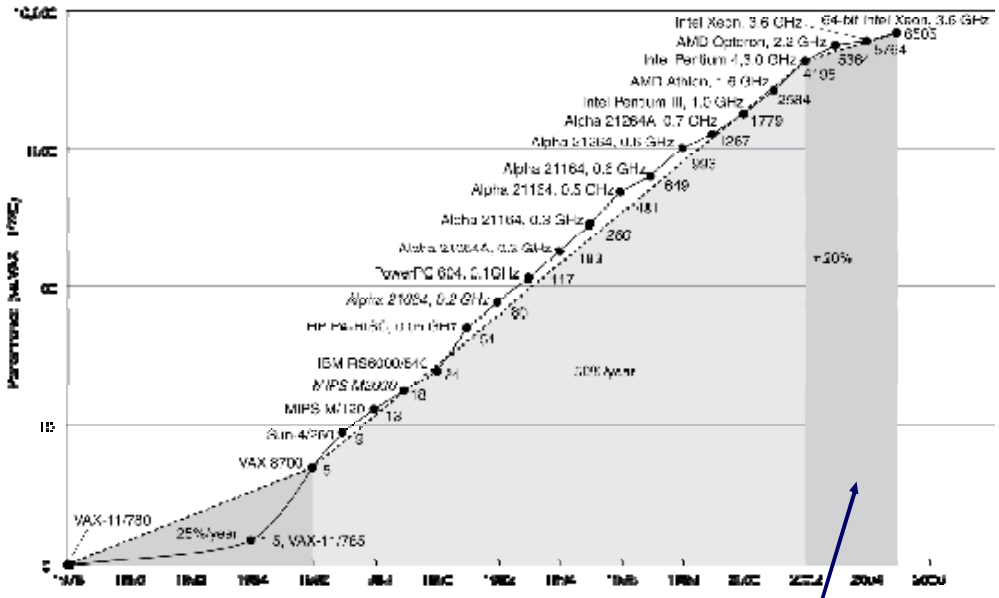
- Enable parallel execution of instructions
  - —Pipelining
  - —Superscalar
- **Pipeline** works like assembly line
  - —Different stages of execution of different instructions at same time along pipeline
- **Superscalar** allows multiple pipelines within single processor
  - —Instructions that do not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

Example in your daily life?

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

Uniprocessor Performance



Constrained by power, instruction-level parallelism, memory latency

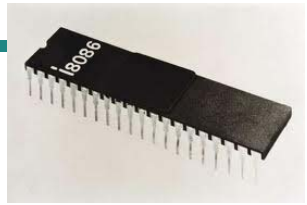
New Approach – Multiple Cores

- **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

Multicore Challenges

- 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

x86 Evolution (1)



- 8080
  - first general purpose microprocessor
  - 8 bit data path
  - Used in first personal computer – Altair
- 8086 – 5MHz – 29,000 transistors
  - much more powerful
  - **16 bit**
  - instruction cache, prefetch 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
- 80486
  - sophisticated powerful cache and instruction pipelir
  - built in **maths co-processor**

x86 Evolution (2)

- Pentium
  - **Superscalar**
  - Multiple instructions executed in parallel
- Pentium Pro
  - Increased superscalar organization
  - Aggressive register renaming
  - branch prediction
  - data flow analysis
  - **speculative execution**

- Pentium II
  - **MMX technology**
  - graphics, video & audio processing
- Pentium III
  - Additional floating point instructions for 3D graphics

x86 Evolution (3)

- Pentium 4
  - Note Arabic rather than Roman numerals
  - Further **floating point and multimedia enhancements**
- Core
  - First x86 with **dual core**

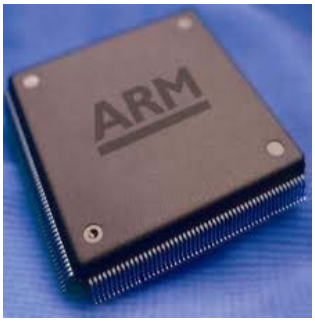
- Core 2
  - **64 bit architecture**
- Core 2 Quad – 3GHz – 820 million transistors
  - **Four processors on chip**

x86 Evolution (4)

- x86 architecture dominant outside embedded systems
- Organization and technology changed dramatically
- Instruction set architecture evolved with backwards compatibility
- ~1 instruction per month added
- **500 instructions** available

Embedded Systems & ARM

- ARM evolved from RISC design
- Used mainly in embedded systems
  - —Used within product
  - —Not general purpose computer
  - —Dedicated function
  - —E.g. Anti-lock brakes in car (ABS)



Embedded System - Definition

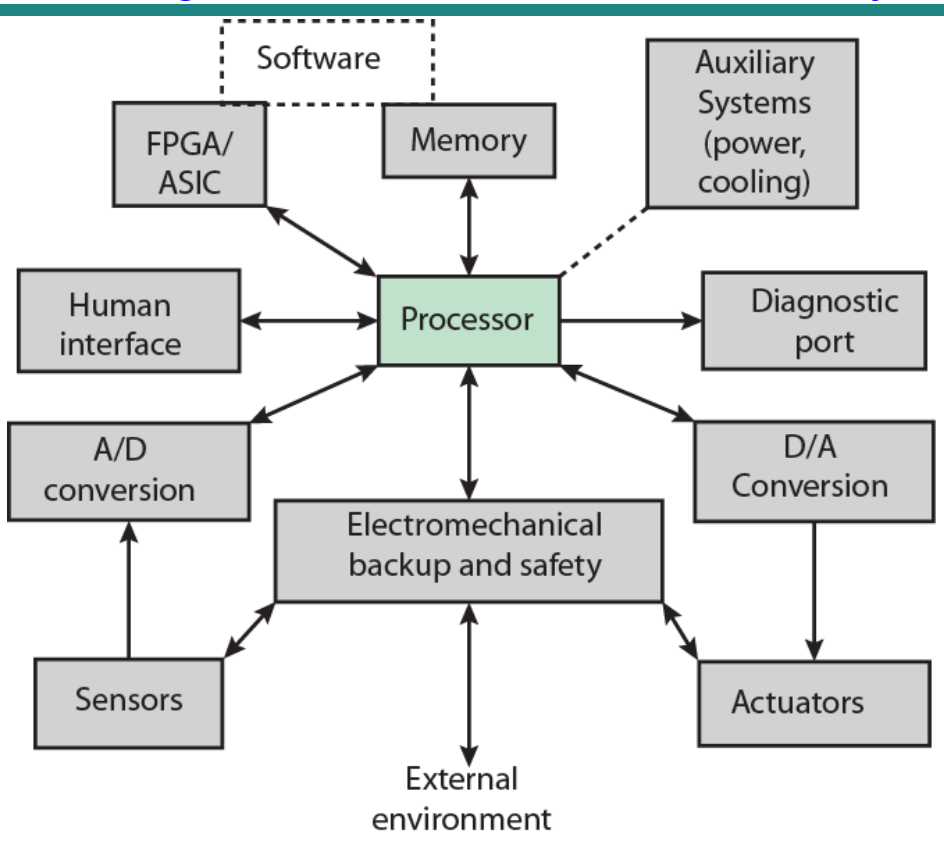
A combination of computer hardware and software, and perhaps additional mechanical or other parts, designed to perform a dedicated function. In many cases, embedded systems are part of a larger system or product, as in the case of an antilock braking system in a car

Market	Embedded Device
Automotive	Ignition system Engine control Brake system
Consumer electronics	Digital and analog televisions Set-top boxes (DVDs, VCRs, Cable boxes) Personal digital assistants (PDAs) Kitchen appliances (refrigerators, toasters, microwave ovens) Automobiles Toys/games Telephones/cell phones/pagers Cameras Global positioning systems
Industrial control	Robotics and controls systems for manufacturing Sensors
Medical	Infusion pumps Dialysis machines Prosthetic devices Cardiac monitors
Office automation	Fax machine Photocopier Printers Monitors Scanners

Embedded Systems Requirements

- Different sizes
  - —Different constraints, optimization, reuse
- Different requirements
  - —Safety, reliability, real-time, flexibility, legislation
  - —Lifespan
  - —Environmental conditions
  - —Static v dynamic loads
  - —Slow to fast speeds
  - —Computation v I/O intensive
  - —Descrete event v continuous dynamics

Possible Organization of an Embedded System



ARM Evolution

- Designed by ARM Inc., Cambridge, England
- Licensed to manufacturers
- High speed, small die, low power consumption
- PDAs, hand held games, phones
  - —E.g. iPod, iPhone, HTC
- **Acorn** produced ARM1 & ARM2 in 1985 and ARM3 in 1989
- Acorn, VLSI and Apple Computer founded ARM Ltd.

ARM Systems Categories

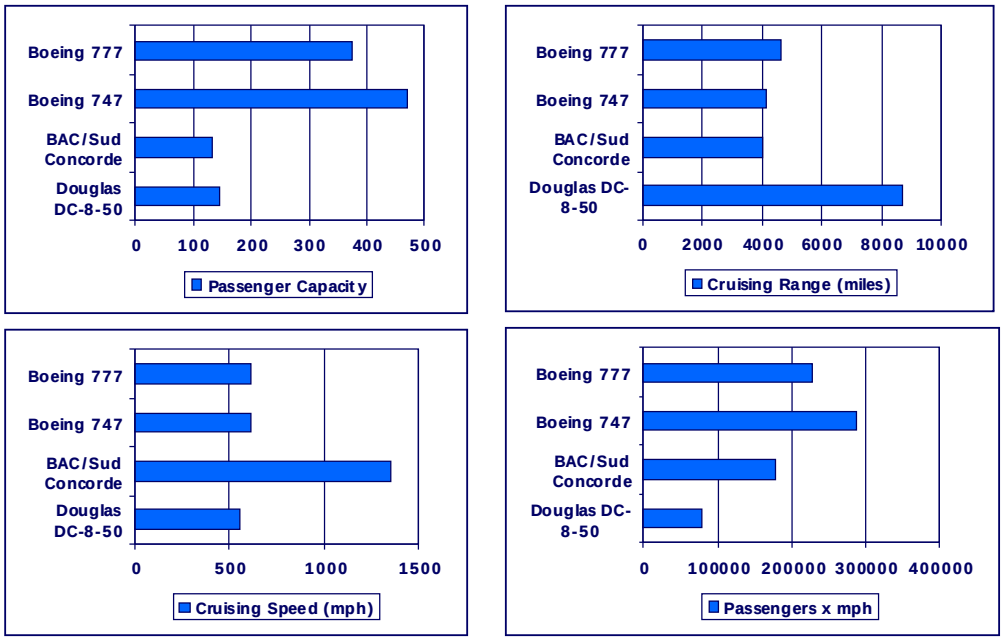
- Embedded real time
- Application platform
  - —Linux, Palm OS, Symbian OS, Windows mobile
- Secure applications

Computer Assessment

- **Performance**
- Cost
- Size
- Security
- Reliability
- Power consumption
- Cool look

Defining Performance

- Which airplane has the best performance?



Response Time and Throughput

- **Response time**
  - —How long it takes to do a task
- **Throughput**
  - —Total work done per unit time
    - e.g., tasks/transactions/… per hour

- How are response time and throughput affected by
  - —Replacing the processor with a faster version?
  - —Adding more processors?

Relative Performance – response time

- Define Performance = 1/Execution Time
- "X is n times faster than Y"

$$\text{Performance}_X / \text{Performance}_Y = \text{Execution time}_Y / \text{Execution time}_X = n$$

- Example: time taken to run a program
  - 10s on A, 15s on B
  - Execution Time$_B$ / Execution Time$_A$ = 15s / 10s = 1.5
  - So A is 1.5 times faster than B

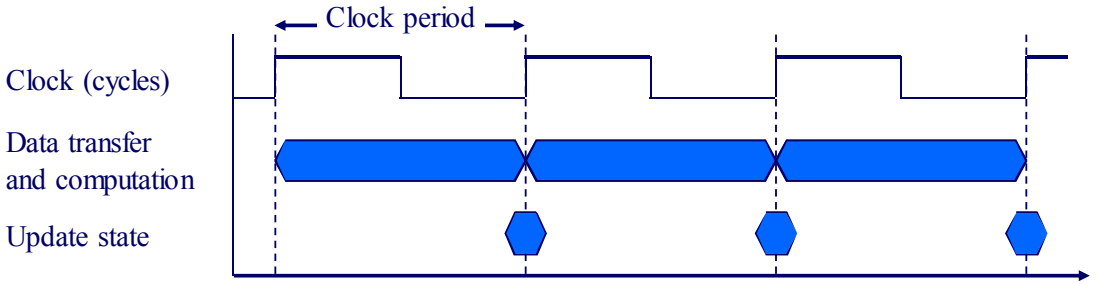
Measuring Execution Time

- **Elapsed time**
  - —Total response time, including all aspects
    - Processing, I/O, OS overhead, idle time
  - —Determines system performance

- **CPU time**
  - —Time spent processing a given job
    - Discounts I/O time, other jobs' shares
  - —Comprises user CPU time and system CPU time
  - —Different programs are affected differently by CPU and system performance

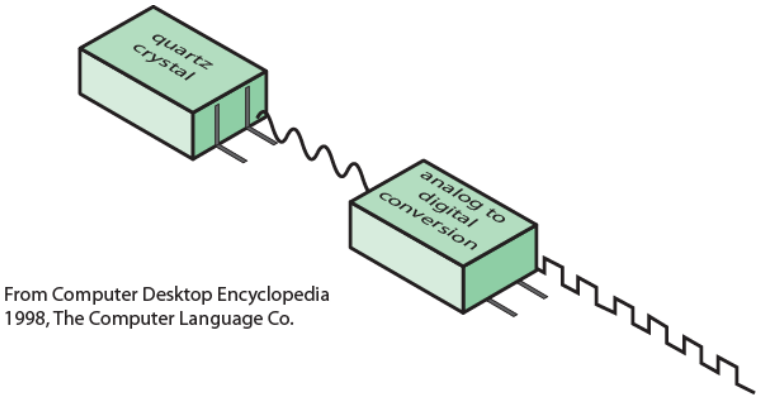
CPU Clocking

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
  - e.g., 4.0GHz = 4000MHz = 4.0×10^{9}Hz

System Clock



System clock cannot be arbitrarily fast

- Signals in CPU take time to settle down to 1 or 0
- Signals may change at different speeds
- Operations need to be synchronised

CPU Time

$$\text{CPU Time} = \text{CPU Clock Cycles} \times \text{Clock Cycle Time}$$

$$= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}$$

- **Performance improved by**
  - Reducing number of clock cycles
  - Increasing clock rate
  - Hardware designer must often **trade off** clock rate against cycle count

CPU Time Example

- Computer A: 2GHz clock, 10s CPU time
- Designing Computer B
  - Aim for 6s CPU time
  - Can do faster clock, but causes 1.2 × clock cycles
- How fast must Computer B clock be?

$$\text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \times \text{Clock Cycles}_A}{6\text{s}}$$

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

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

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

Instruction Count and CPI

CPI= consumer price index??

$$\text{Clock Cycles} = \text{Instruction Count} \times \text{Cycles per Instruction}$$

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

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

- Instruction Count for a program
  - Determined by program, ISA and compiler
- Average cycles per instruction
  - Determined by CPU hardware
  - Different instructions have different CPI
    - Average CPI affected by instruction mix

CPI Example

- Computer A: Cycle Time = 250ps, CPI = 2.0
- Computer B: Cycle Time = 500ps, CPI = 1.2
- Same ISA
- Which is faster, and by how much?

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

$$= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps}$$ ← A is faster…

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

$$= I \times 1.2 \times 500\text{ps} = I \times 600\text{ps}$$

$$\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{I \times 600\text{ps}}{I \times 500\text{ps}} = 1.2$$ ← …by this much

CPI in More Detail

- If different instruction classes take different numbers of cycles

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

  - Weighted average CPI

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

Relative frequency

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

CPU Time Summary

The BIG Picture

$$\text{CPU Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

- Performance depends on
  - —**Algorithm**: affects IC, possibly CPI
  - —**Programming language**: affects IC, CPI
  - —**Compiler**: affects IC, CPI
  - —**Instruction set architecture**: affects IC, CPI, T_c

Instruction Execution Rate

Instructions executed per second!

- **MIPS**: Millions of instructions per second
- **MFLOPS**: Millions of floating point instructions per second

Heavily dependent on
- instruction set
- compiler design
- processor implementation
- cache & memory hierarchy

Benchmarks

- **Benchmark: Programs designed to test performance**
  - Written in high level language: Portable
  - Represents style of task: Systems, numerical, commercial
  - Easily measured
  - Widely distributed
- E.g. System Performance Evaluation Corporation (**SPEC**)
  - CPU2006 for computation bound
    - 17 floating point programs in C, C++, Fortran
    - 12 integer programs in C, C++
    - 3 million lines of code
  - **Speed** and **rate metrics**
    - Single task and throughput

SPEC Speed Metric -Single task

- Base runtime defined for each benchmark using reference machine
- Results are reported as ratio of reference time to system run time
  - $Tref_i$ execution time for benchmark i on reference machine
  - $Tsut_i$ execution time of benchmark i on test system

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

- Overall performance calculated by averaging ratios for all 12 integer benchmarks
  - Use geometric mean: Appropriate for normalized numbers such as ratios

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

Amdahl's Law

- Gene Amdahl [AMDA67]
- Potential speed up of program using multiple processors

- Concluded that:
  - —Code needs to be parallelizable
  - —Speed up is bound, giving diminishing returns for more processors

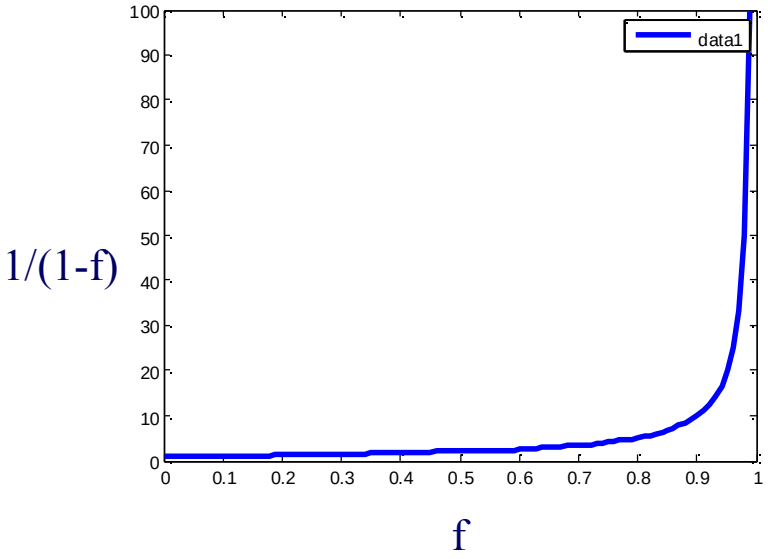
Amdahl's Law Formula

- For program running on single processor
  - — Fraction *f* of code infinitely parallelizable with no scheduling overhead
  - — Fraction (1-*f*) of code inherently serial
  - — T is total execution time for program on single processor
  - — N is number of processors that fully exploit parallel portions of code

$$Speedup = \frac{\text{time to execute program on a single processor}}{\text{time to execute program on } N \text{ parallel processors}}$$

$$= \frac{T(1-f)+Tf}{T(1-f)+\frac{Tf}{N}} = \frac{1}{(1-f)+\frac{f}{N}}$$

Amdahl's Law Formula



- *f* small, parallel processors has little effect
- *N* ->∞, speedup bound by 1/(1 – *f*)
  - —Diminishing returns for using more processors

Acknowledgements

- These slides contain material developed and copyright by:
  - —Arvind (MIT)
  - —Krste Asanovic (MIT/UCB)
  - —John Kubiatowicz (UCB)
  - —David Patterson (UCB)
  - —Geng Wang (SJTU)
  - —Yanmin Zhu (SJTU)