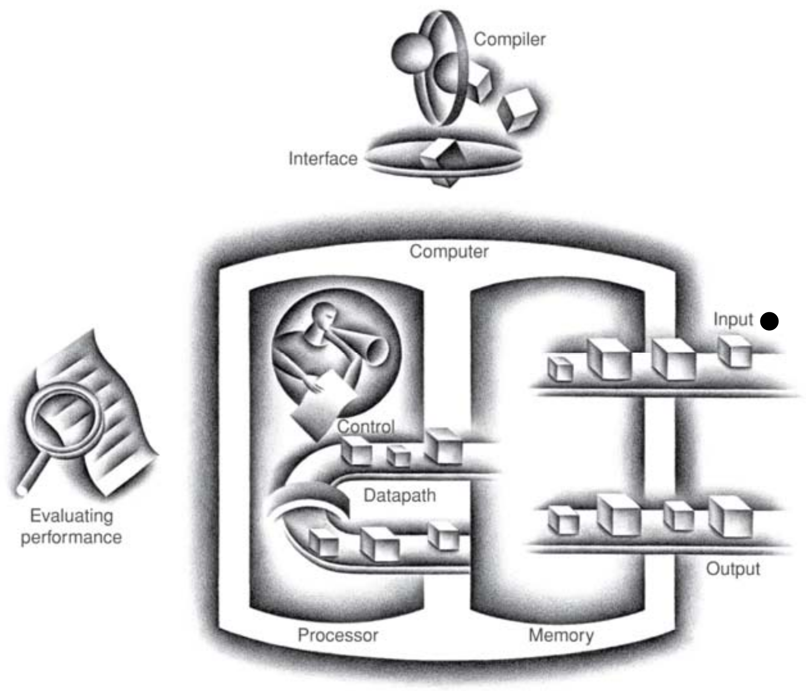
Components of a Computer

The BIG Picture



- Same components for all kinds of computer
  - Desktop, server, embedded
- Input/output includes
  - User-interface devices
    - Display, keyboard, mouse
  - Storage devices
    - Hard disk, CD/DVD, flash
  - Network adapters
    - For communicating with other computers

Level 5 Problem-oriented language level

↓ Translation (compiler)

Level 4 Assembly language level

↓ Translation (assembler)

Level 3 Operating system machine level
(set of new instruction memory organization ability to be multiprogramming)
(some of instruction carried out by OS and some by microprogram)

↓ Partial interpretation (OS)

Level 2 Instruction set architecture level (machine language)

↓ Interpretation (microprogram) or direct execute

Level 1 Microarchitecture level→ local memory, ALU, register connected to ALU→Datapath controlled by microprogram

↓ Hardware

Level 0 Digital logics level →transistor, gates, register

History

- First generation : Vacuum tube
  - ENIAC (Electronic Numerical Integrator and Calculator/Computer)
    - 1943 completed 1946
    - John W Mauchly and John Presper Eckert at U. of Pennsylvania
    - 80 feet long by 8.5 feet high and several feet wide → 30 tons occupy 15,000 square feet of floor space and containing > 18,000 vacuum tubes → 140 KW → decimal not binary
    - Memory: 20 accumulator → each hold 10 digit decimal →represented by a ring of 10 vacuum tubes only one on

      1000000000 = 0, 0100000000=1
    - Drawback→ programmed manually by setting switches and plugging and unplugging cables

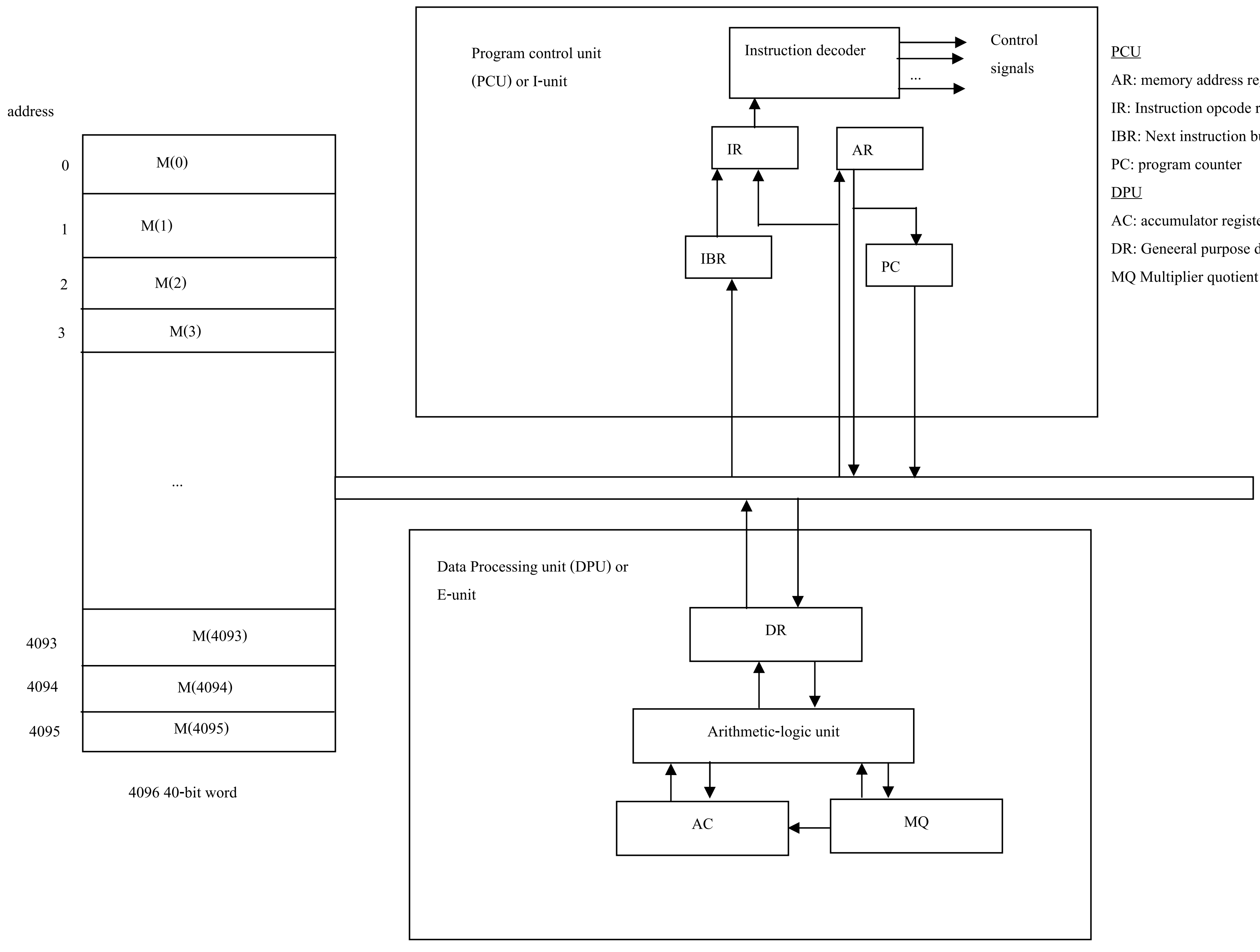
History

- EDVAC (Electronic Discrete variable automatic computers)
  - 1944 – 1952
  - John von Neumann $\rightarrow$want to improve the way program were entered and stored as numbers
  - Eckert and Mauchly’s work but left before it is completed
  - Program and data $\rightarrow$ stored in same high-speed memory
  - Use true binary or base 2 form
  - 1K word (1st memory), 20 K word (2nd memory)
  - Data processed serially$\rightarrow$ instruction forming a program was placed in EDVAC’s main memory one at a time from main memory to CPU for execution

History

- EDSAC (Electronic Deay Storage Automatic Calculator)
  - 1946
  - Maurice Wilkes at Cambridge U.
  - First full-scale operational, stored program computer
  - Mark-I (prototype) built at U. of Manchester in 1948 might be the first operational stored-program machine)
- IAS
  - 1947
  - Goldstine of Arthur Burks and von Neumann at Institute for Advanced Study→ memo about architectural concepts
  - Built by Julian Begelow
  - 8-bit opcode
  - General structure, PCU, main memory for storing active program secondary memory for backup, I/O
  - Data processed in parallel

Organization of CPU and main memory of the IAS computer



<u>PCU</u>

AR: memory address register (12 bit)

IR: Instruction opcode register

IBR: Next instruction buffer register

PC: program counter

<u>DPU</u>

AC: accumulator register

DR: Geneeral purpose data register

MQ Multiplier quotient register

History

- Whirlwind
  - Begin 1947
  - MIT-introduced the ferrite core memory stored in magnetic form until 1970
  - 2048 16-bit word
- BINAC
  - Commercial machine
  - Then UNIVAC I (Universal Automatic computer) a general-purpose computer
  - 1 million dollars
- IBM 701
  - 1953
  - Use machine language : add x1 x2 → might be 0001000…
  - Assembly language used in 1950

History

- Second generation : transistor
  - Transistor replacing ferrite core in 1970
  - Magnetic disk for secondary memory
  - Transistor serve as a high-speed electronic switch for binary signals, smaller, cheaper, sturdier
  - Need less power than vacuum tube
  - IBM 7094
    - 1962
    - Use data channels : independent I/O with its own processor and its own instruction set
    - CPU doesn’t execute detailed I/O instruction (before: transfer data from main to secondary tie up CPU)
      - I/O transfer by send signal to data channel instructing it to execute a sequence of instruction in memory
      - Data data channel perform, send signal back to CPU
    - Allow batch processing→require batch process monitor (permanently resides in main memory =>rudimentary version of OS)

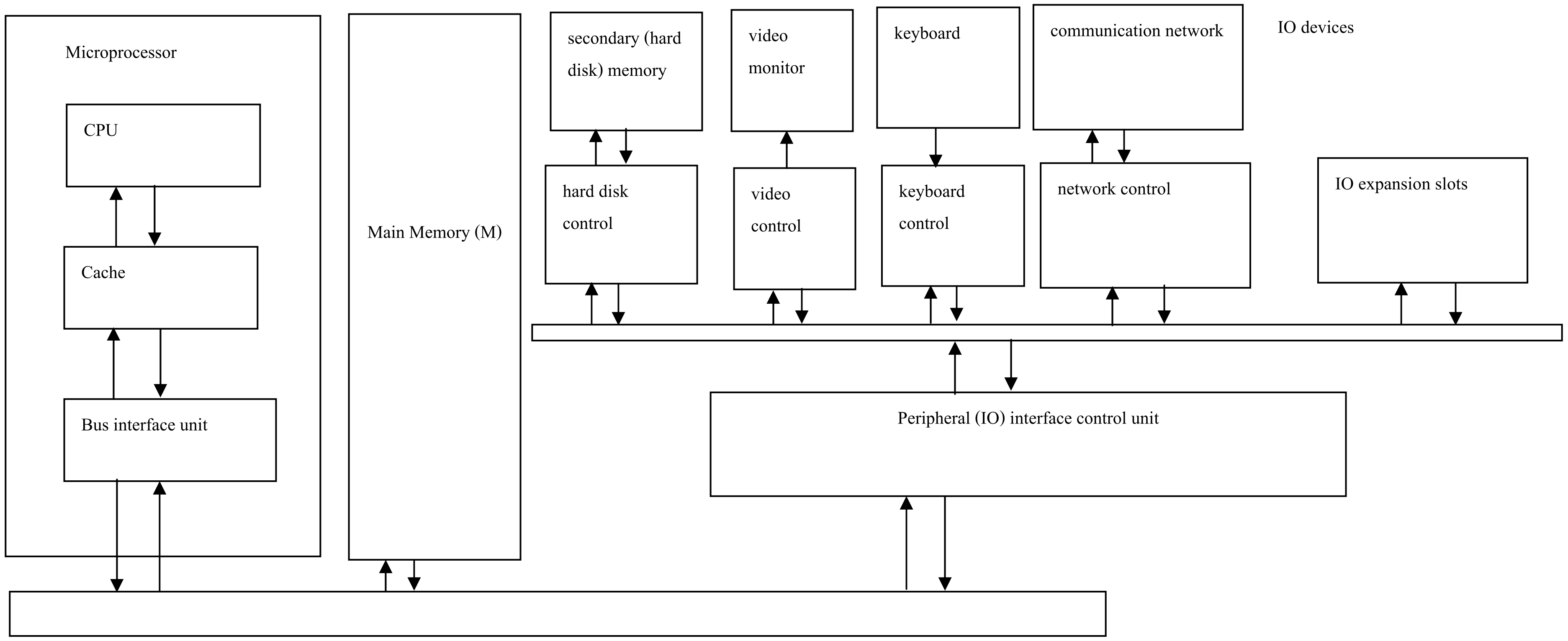
History

- Third generation : ICs (integrated circuits)
  - 1965
  - Problem→ a few dozens manufactures of computer around the world were each producing machine that were incompatible with those of other manufacturers →software's cost > hardware's cost
  - IBM developed System/360
    - Cover a wide range of computing performance
    - Software compatible
    - 1972
    - 32 bit (4 bytes) = 1 word size → become synonymous in the context of large computer

History

- VLSI ERA (personal computer era)
  - Very dense circuits
  - Contain thousand or million of transistor
  - Microprocessor (single-chip CPU) → intel’s 4004 →1971

A Typical personal computer system



History

- IBM choose intel 8088 built personal computer →publish how to build
  - 1981
  - 1990→manufacturer could fabricate the entire CPU of a System/460
  - Computer were classified into 3 main types (1980)
    - Mainframe
    - Minicomputer
    - Microcomputer
- PC has von Neumann organization →1970 →Altair computer → intel's 8088 (8-bit)
- IBM →open architecture →Intel →pentium…
- APPLE's Macintosh →1984→Motorola 68000 →Power PC…

History

- Other development
  - Supercomputer
    - 1976 Cray-1
      - Seymour Cray at Cray Research
        - » Faster, most expensive
  - Silicon Graphics bought Cray Research → no longer stand alone supercomputer company → 1996
  - Complex instruction set computer (CISC)
    - Aim to reduce N at the expense of CPI
    - Disadvantage: reduce performance (reduce program size but not faster execution) →reduce using complex instruction
  - Reduced instruction set computer (RISC)
    - SPARC
    - Reduce CPI at the expense of N
- Embedded computer: a computer inside another device used for running one predetermined application or collection of software

The Computer Revolution

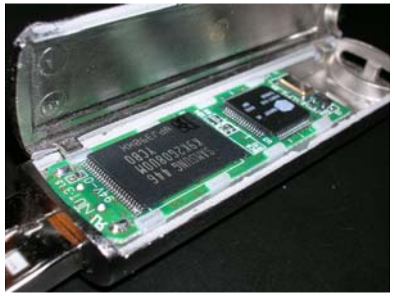
- Progress in computer technology
  - Underpinned by Moore’s Law (the transistor capacity doubles every 18-24 months)
- Makes novel applications feasible
  - Computers in automobiles
  - Cell phones
  - Human genome project
  - World Wide Web
  - Search Engines
- Computers are pervasive

Classes of Computers

- Desktop computers
  - General purpose, variety of software
  - Subject to cost/performance tradeoff
- Server computers
  - Network based
  - High capacity, performance, reliability
  - Range from small servers to building sized
- Embedded computers
  - Hidden as components of systems
  - Stringent power/performance/cost constraints

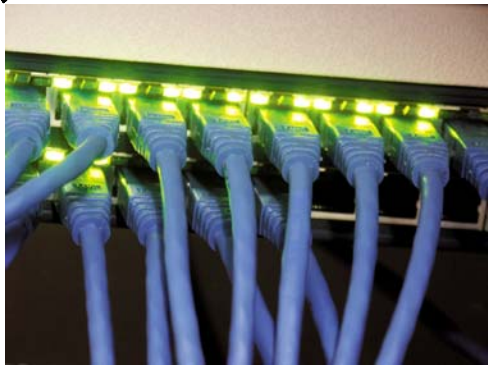
A Safe Place for Data

- Volatile main memory
  - Loses instructions and data when power off
- Non-volatile secondary memory
  - Magnetic disk
  - Flash memory
  - Optical disk (CDROM, DVD)



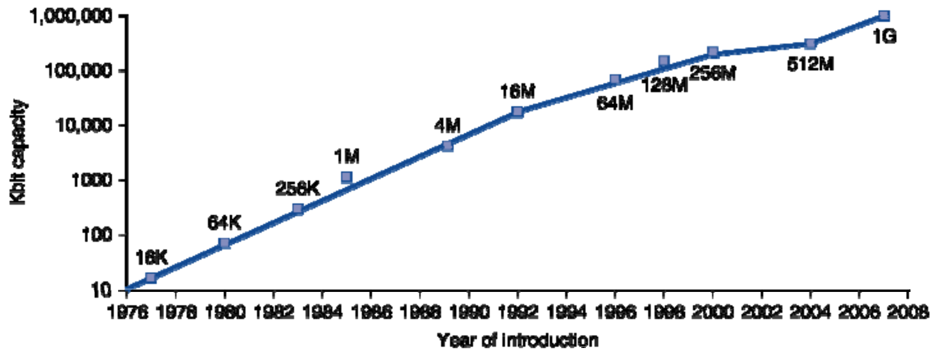
Networks

- Communication and resource sharing
- Local area network (LAN): Ethernet
  - Within a building
- Wide area network (WAN: the Internet
- Wireless network: WiFi, Bluetooth



Technology Trends

- Electronics technology continues to evolve
  - Increased capacity and performance
  - Reduced cost



DRAM capacity

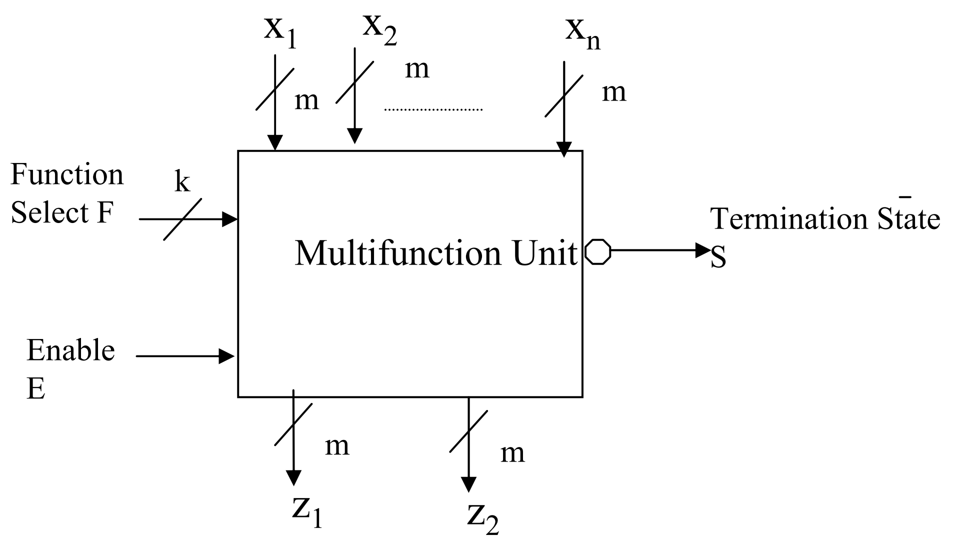
Year	Technology	Relative performance/cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit (IC)	900
1995	Very large scale IC (VLSI)	2,400,000
2005	Ultra large scale IC	6,200,000,000

Design Methodology

- Top-down design
  1. Specify the processor-level structure of system
  2. Specify the register-level structure of each component type identified in 1
  3. Specify the gate-level structure of each component type identified in 2
- Gate level
  - Combinational logic: a combinational function also referred to as a logic or Boolean function is a mapping from the set of 2^n input combination of n binary variables➔ output 0 or 1
  - AND, OR, NAND,NOR, XOR, NOT gates
  - SOP (AND-OR) using minterm, POS (OR-AND) using maxterm
  - Karnough map
  - Flip-Flop: JK, RS, D Flip-flop
  - Sequential circuit

Design Methodology

- Register Level



  - Termination State: control output , activate when value = 0, indicate when and how the unit complete its processing
  - Function Select F: specify one of several possible operator to perform
  - Enable: specify time or condition for a selected operation to be perform ➔ often connected to clock sources

Type	Component	Function
Combinational	Word gate	Logical (Boolean) operator
	Multiplexer	Data routing, general combinational function
	Decoders and encoders	Code checking and conversion
	Adders	Addition and subtraction
	Arithmetic-logic unit	Numerical and logical operations
	Programmable-Logic devices	General combinational function
Sequential	(Parallel) register	Information storage
	Shift register	Information storage, serial-parallel conversion
	Counters	Control/timing signal generation
	Programmable-Logic devices	General sequential function

- $X_1, X_2, \ldots, X_n$ be m-bit binary words
- $X_i = (X_{i1}, X_{i2}, \ldots, X_{im})$ for i=1,2,…,n

- $Z(X_1, X_2, \ldots, X_n) = [Z(X_{11}, X_{12}, \ldots, X_{1m}), Z(X_{21}, X_{22}, \ldots, X_{2m}), \ldots, Z(X_{n1}, X_{n2}, \ldots, X_{nm})]$

- Ex:. $Z = XY$ (and each bit)

  $(Z_1, Z_2, \ldots Z_m) = (X_1Y_1, X_2Y_2, \ldots X_mY_m)$
- Ex: $X_1 + X_2 + \ldots + X_n = (X_{11} + X_{21} + \ldots + X_{n1}, X_{12} + X_{22} + \ldots + X_{n2}, \ldots, X_{1m} + X_{2m} + \ldots + X_{nm})$
- Ex: $aX = (aX_1, a X_2, \ldots, aX_m)$
- Ex: $a+X = (a+X_1, a+X_2, \ldots, a+X_m)$
- Ex:

$$\overline{XY} = \left(\overline{X_1Y_1}, \overline{X_2Y_2}, \ldots, \overline{X_mY_m}\right)$$

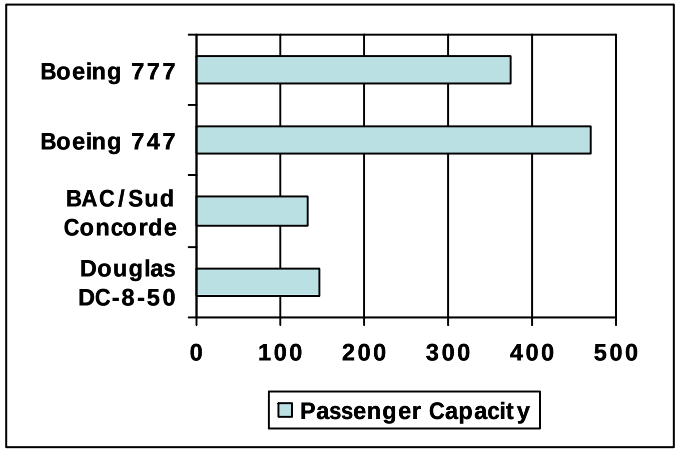
- MUX
  - Select 1 of input to transmit to the final output, intend to route data from 1 of several sources to a common destination
  - Select F: p bit, input X: 2^P-1 input each has m bit, output Z has m bit. If F = 000…01 then X_1 is selected
- Decoder
  - 1 out of 2^n or 1/ 2^n decoder, routing data from a common source to one of several destination
  - Combination circuit with n input data line and 2^n output data lines such that each of the 2^n possible input combination X_i activates (sets to 1) exactly 1 of the output lines Z_i
  - Routing data from common source to one of several destination
- Encoder
  - Inverse of a decoder
  - Use to generate the address or name of an active input line
  - 2^k input data line and k output data lines
  - Ex. k=3 ➔data input lines $x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_7$ =00000010➔ data output lines $z_2 z_1 z_0$ = 110
  - Problem: if more than 1 input active➔use priority encoder

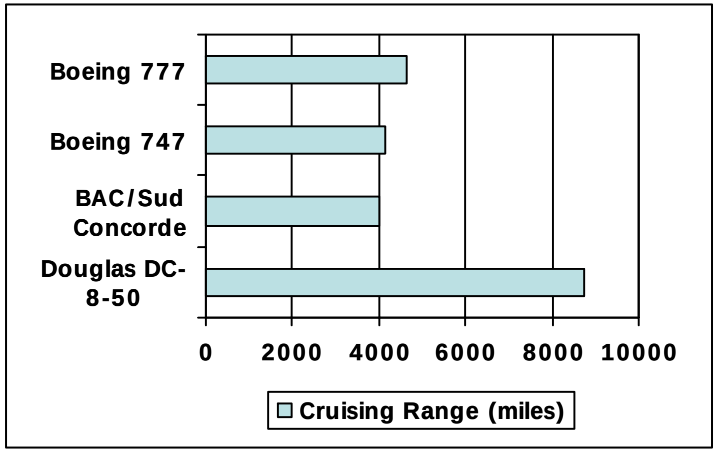
- Arithmetic elements
  - Fixed point (binary part are placed in any fixed position within number word format)
  - Ex: +0.5 = 0100000000…0
- Register
  - M-bit register is an ordered set of m flip-flops designed to store an m-bit word ($z_0, z_1, \ldots, z_{m-1}$) ➔each bit is in one flip-flop
  - Shift register
    - Right shift ($z_0, z_1, \ldots, z_{m-1}$) ➔ ($x, z_0, z_1, \ldots, z_{m-2}$)
    - Left shift ($z_0, z_1, \ldots, z_{m-1}$) ➔ ($z_1, z_2, \ldots, z_{m-1}, x$)
- Programmable Logic Device (PLD)
  - 2 tech are used to program PLDs
    - Mask programming → few special step in IC chip manufacturing process
    - Field programming →done by designer or end user "in the field"
  - Programmable array → connection to and from logic element contain transistor switches that can be programmed to be permanently on or off
- Programmable ,\logic array (PLA)
  - Intend to realize a set of combinational logic function in minimized SOP form

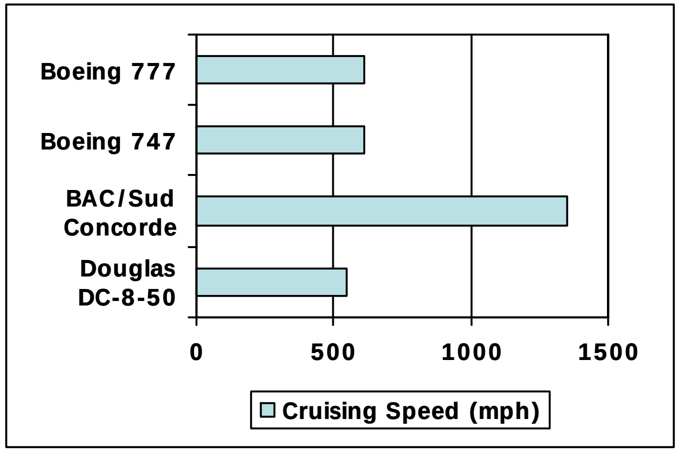
- Processor Level
  - CPU
    - I-unit fetch instruction from cache or M and decodes them to derive the control signals needed for their execution
    - E-unit has arithmetic-logic circuit that execute most instruction →data processing unit (data path)
    - 1 instruction cycle ➔ fetch, decode, execute
  - Memory
    - Main memory → word organized addressable
    - Secondary memory →connected indirectly (via M) to CPU and form part of the computer's I/O
    - Cache→ positioned between CPU and main memory → used random access or associative or content addressing
  - IO
  - Interconnection network
  - Processor level design: usually take a prototype design of known performance and modify it where necessary to accommodate new technology or meet new performance requirements, example,
    - The computer should be capable of executing s instruction of type b per second
    - The computer should be able to support c memory or I/O device of type d
    - The computer should be compatible with computer of type e
    - The total cost of the system should not exceed f

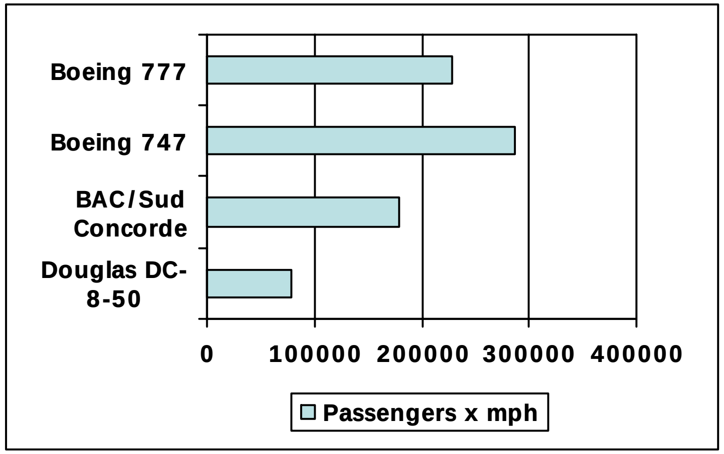
Defining Performance

- Which airplane has the best performance?









Performance Evaluation

- Throughput : total amount of work done in a given time
- Response time (execute time): time between start and completion of a task
- Want to reduce response time and increse throughput

  Performance = 1/execution time

  – X is n times faster than Y ➔ $\text{performance}_x/\text{performance}_y = n$
  – Clock period = length of each clock cycle
  – CPU time (CPU execution time= time CPU spend computing for a specified task) ➔ user CPU time + system CPU time
    - User CPU time = CPU time spent in a program itself
    - System CPU time = CPU time spent in OS performing tasks on behalf of the program
    - Clock cycle = time for 1 clock period usually of the processor clock which run at a constant rate
    - f MHz ➔ can perform 1 basic operation in (1/f) μs

CPU execution time for a program = CPU clock cycle for a program × clock cycle time (s/cycle)
CPU execution time for a program = CPU clock cycle for a program / clock rate (Hz or cycle/s)

Suppose program Q on a given CPU takes T seconds, N executed instruction (including repeated)
average number of instructions executed per second (IPS):

$$T = \frac{N}{IPS} \text{ s}$$

Average number of cycles per instruction (CPI): (f is in MHz)

$$CPI = \frac{\left(f \times 10^{-6}\right)}{IPS}$$

$$T = \frac{N \times CPI}{f \times 10^{-6}} \text{ s}$$

1. Software: reduce N → reduce T
2. Architecture: reduce CPI → reduce T
3. Hardware: increase f → reduce T

CPU clock cycle = Instruction for a program × CPI

CPU time = Instruction count × CPI × clock cycle time

$$\text{CPU time} = \frac{\text{Instruction count} \times CPI}{\text{clock rate}}$$

$$CPI = \frac{\text{clock rate}}{IPS}$$

$$T = \frac{N \times CPI}{f \times 10^{-6}} \text{ s}$$

Instruction count → can be measured without knowing all the details of implementation→depend on the architecture but not on the exact implementation

CPI depends on a wide variety of design details of the machine including both memory system and the processor structure as well as on the mix of instruction type executed in an application

$$\text{CPU clock cycle} = \sum_{i=1}^{n} \left(CPI_i \times C_i \right)$$

n : number of instruction class, CPI_i : CPI for class i, C_i : count of number of instruction of class i

	Comp A	Comp B
P1	1	10
P2	1000	100
	1001	110

PerformanceB/performanceA = (1001)/(110) = 9.1 ➔ B is faster than A → correct?

Or we could use

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

$Time_i$→execution time for the I program of a total of n in workload (a set of program run on a computer that is either the actual collection of application run by a user or is constructed from real program to approximate such a mix. A typical workload specified both the program as well as the relative frequencies)

If run P1 and P2 → not equal numbert of time, e.g., 20% of the task in the workload run P1, 80% of the rest run P2→ use

$$WAM = \frac{1}{n}\sum_{i=1}^{n} w_i Time_i \quad \text{with} \quad \sum_{i=1}^{n} w_i = 1$$

SPEC CPU Benchmark

- Programs used to measure performance
  - Supposedly typical of actual workload
- Standard Performance Evaluation Corp (SPEC)
  - Develops benchmarks for CPU, I/O, Web, …
- SPEC CPU2006
  - Elapsed time to execute a selection of programs
    - Negligible I/O, so focuses on CPU performance
  - Normalize relative to reference machine
  - Summarize as geometric mean of performance ratios
    - CINT2006 (integer) and CFP2006 (floating-point)

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

with $$\text{Execution time ratio} = \frac{\text{reference time}}{\text{measured time}}$$

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

- Pitfall:
  - Expecting the improvement of one aspect of a computer to increase performance by an amount proportional to the size of the improvement
  - Using MIPS (million instruction per second) as a performance metric (faster has higher MIPS rating)

$$\text{MIPS} = \frac{\text{Instruction Count}}{\text{Execution time} \times 10^6}$$

    - Problem:
      - MIPS specifies the instruction execution rate but does not take into account the capabilities of the instruction→ cannot compare computers with different instruction sets using MIPS since instruction counts will certainly differ
      - MIPS varies between program on the same computer, then a machine cannot have a single MIPS rating for all program
      - MIPS can vary inversely with performance
- Fallacy: Hardware-independent metrics predict performance
  - Use code size as a measure of speed

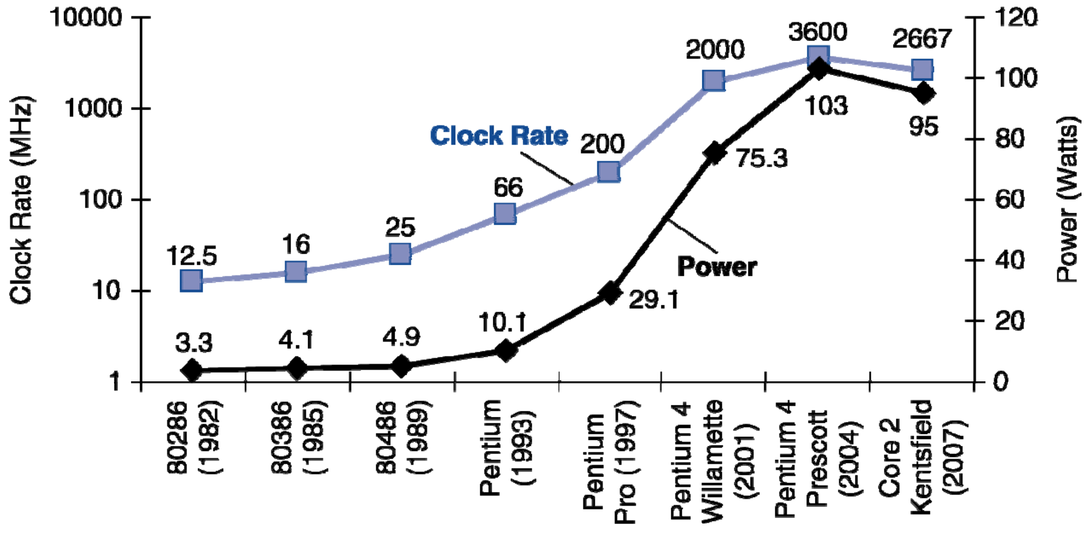
Performance Summary

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

Power Trends



- In CMOS IC technology

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

$\times 30$

$5\text{V} \rightarrow 1\text{V}$

$\times 1000$

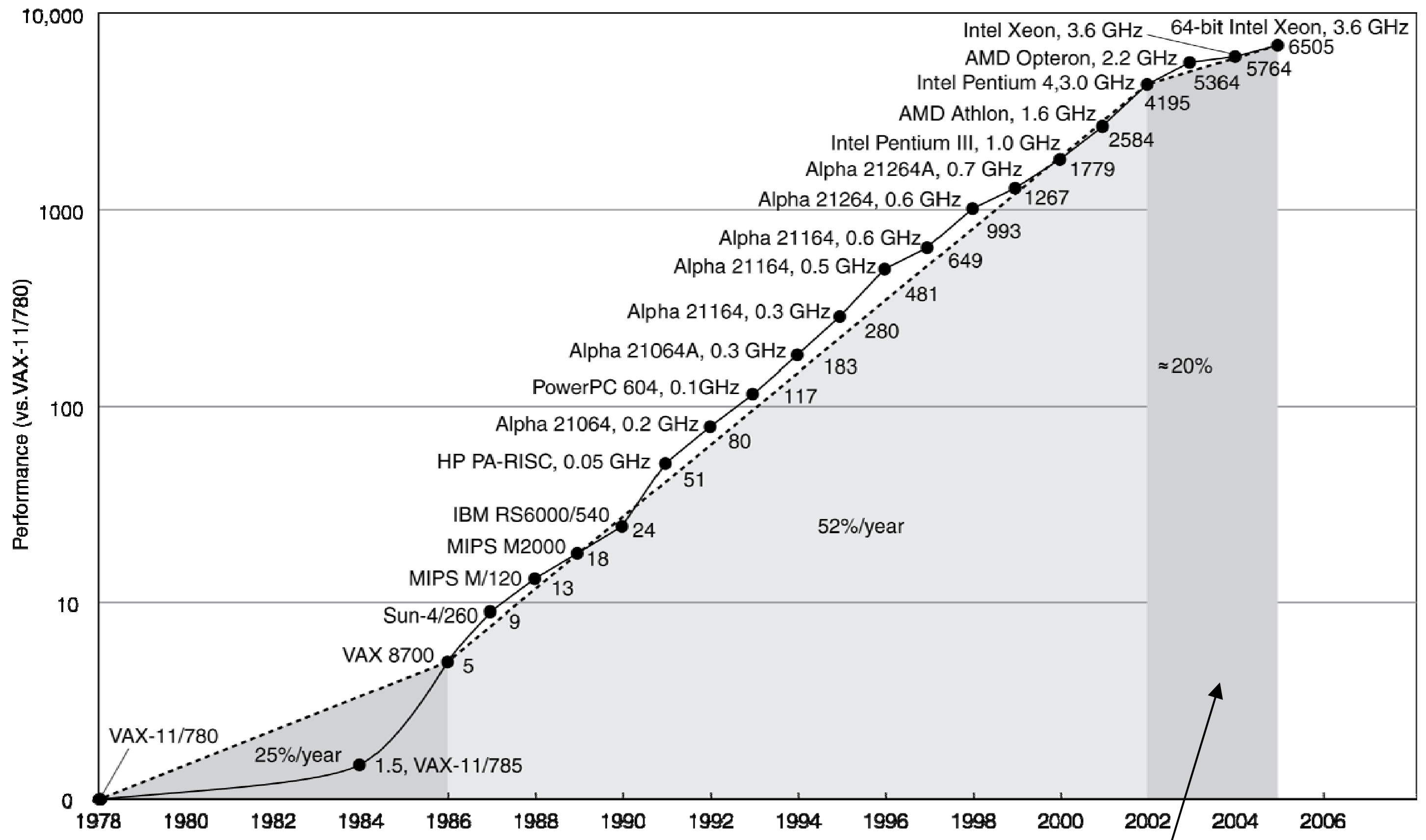
Reducing Power

- 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?

Uniprocessor Performance



Constrained by power, instruction-level parallelism, memory latency

Multiprocessors

- Multicore microprocessors
  - More than one processor per chip
- 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

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

FIGURE 1.17 Number of cores per chip, clock rate, and power for 2008 multicore microprocessors. Copyright © 2009 Elsevier, Inc. All rights reserved.

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%	920,35	233
30%	70,500	222
20%	47,126	206
10%	23,066	180
0%	0	141
Overall sum	1,283,590	2,605
∑ssj_ops/ ∑power		493

Concluding Remarks

- Cost/performance is improving
  - Due to underlying technology development
- Hierarchical layers of abstraction
  - In both hardware and software
- Instruction set architecture
  - The hardware/software interface
- Execution time: the best performance measure
- Power is a limiting factor
  - Use parallelism to improve performance