Chronological History of Pre and Early Computer Developments	
"Foreign" Milestones	**American Milestones**
500 B. C. Abacus	
1621 Slide Rule	
1623 Schickard's Adding Machine (Germany)	
1642 Pascal's Adding Machine (France)	
1673 Leibniz Calculator (Germany)	
1804 Jacquard Loom and Punch Cards	
1822 Babbage's Difference & Analytical Engine (England)	
1850 George Boole's Boolean Algebra (Irish)	
	1890 Census and Tabulating Machines
	1924 IBM
1937 Alan Turing and the Turing Machine	***1937 John Atanasoff and the ABC Computer (ISU)***
1938 Konrad Zuse's Z-series (Germany)	
	1939 Harvard MARK I
1943 COLOSSUS (England)	
	1945 ENIAC Mauchly and Eckert
1946 Manchester Mark I (England) (First Stored-Program Computer)	
	1947 Transistor Bell Labs
EDSAC 1949 (England)	.
	1951 UNIVAC Mauchly/Eckert Sperry Rand Corp Grace Hopper wrote first compiler for it

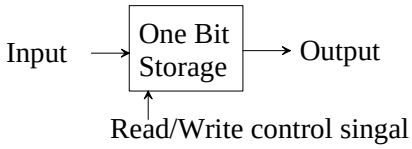
Computer Technology

Built from two types of components plus interconnection:

Gates:

A	B	OR
F	F	F
F	T	T
T	F	T
T	T	T

Memory Cells:



Circuit - collection of gates and/or memory cells to perform some function

Computer Generations

1st Generation - vacuum tubes with wires for interconnection (one gate per vacuum tube)

Examples: ENIAC - 1943-46 Army's Ballistics Lab Mauchly/Eckert (U. of Penn.) 1st general-purpose electronic digital computer
30 tons, 15,000 sq. ft., 18,000 vacuum tubes, 140 KW, 5000 adds/sec
Programmed with wires and switches

1945-52 - IAS John von Neumann (Princeton) used stored-program concept

2nd Generation - transistors - solid state device made from silicon, but single gate each
10,000 to 100,000 transistors soldered to circuit board
Advantages: improved speed, reliability, size, power consumption
Some system software: early OS and High-level programming languages

Computer Generations (continued)

3rd Generation - Small-scale integrated circuits (ICs)
Many gates on same wafer of silicon (chip) and connected to form circuits.
Advantages:
1) cheaper since the cost per chip the same, but fewer needed since more powerful
2) denser chips → shorter connections → faster
3) smaller computers → used in more places
4) reduced power and cooling consumption
5) interconnections on ICs more reliable that interconnections between ICs

4th Generation - Large-scale ICs - microprocessor
Enough gates per chip to implement whole CPU
Intel 4004 (1971) - 1st
Intel 8080 (1974) - lst general-purpose microprocessor

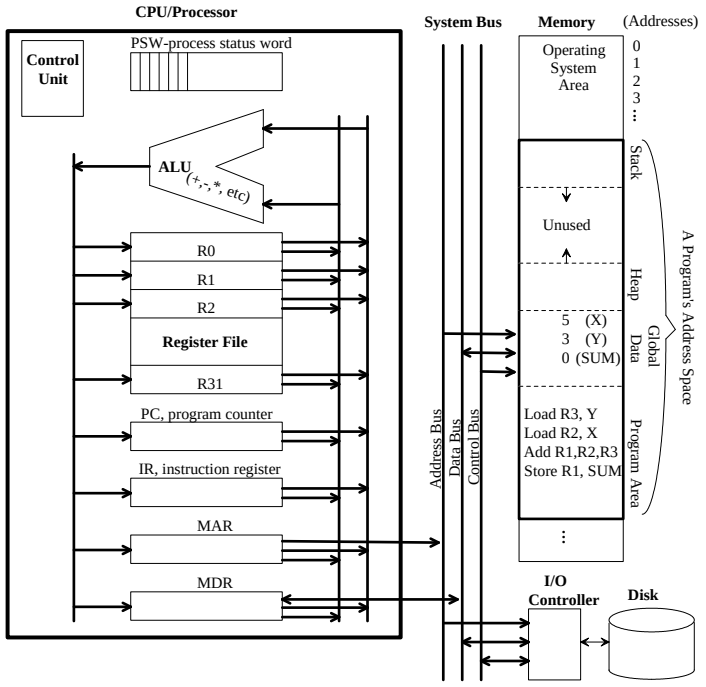
Moore's Law

Gordon Moore - Intel cofounder (1965)

Predicted that gate density would double every year into near future.

This held true until early 1970s when the rate slowed to doubling every 18 months.

von Neumann Model: Stored-Program Concept



Instruction/Machine Cycle of stored-program computer - repeat all day

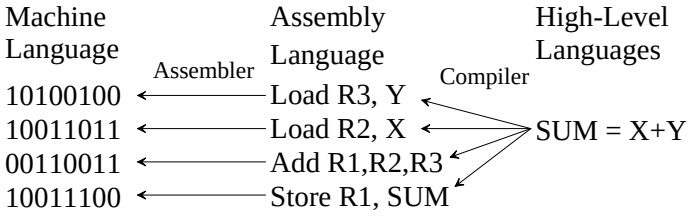
1. Fetch Instruction - read instruction pointed at by the program counter (PC) from memory into Instr. Reg. (IR)
2. Decode Instruction - figure out what kind of instruction was read
3. Fetch Operands - get operand values from the memory or registers
4. Execute Instruction - do some operation with the operands to get some result
5. Write Result - put the result into a register or in a memory location

(Note: Sometime during the above steps, the PC is updated to point to the next instruction.)

Today's stored-program computers have the following characteristics:

- –Three hardware systems:
  - •A central processing unit (CPU)
  - •A main memory system
  - •An I/O system
- –The capacity to carry out sequential instruction processing.
- –A single data path between the CPU and main memory.
  - •This single path is known as the *von Neumann bottleneck.*
  - •Register File - store a small amount of data in the CPU close to the computation circuits
  - •Level 1 (L1) and Level 2 (L2) caches on CPU store larger amounts of data and instruction on CPU

Programming Languages



Instruction Set Architecture (ISA)/Family of Computers
Several implementation of computer can run the same assembly/machine language with different cost/performance ratios.
Examples:
IBM 700/7000 Series (overhead)
Intel 486, Pentium I, II, III, IV (overhead)

Advantages of High-Level Programming Languages:

1) Portability - computer independent
2) Productivity - more and better software in less time
3) Application specific languages

Computer Networking

1962 - RAND (Research and Development a nonprofit institution) and Paul Baran begin research into packet switching.

1965 - ARPA - Advanced Research Projects Agency sponsors networking research

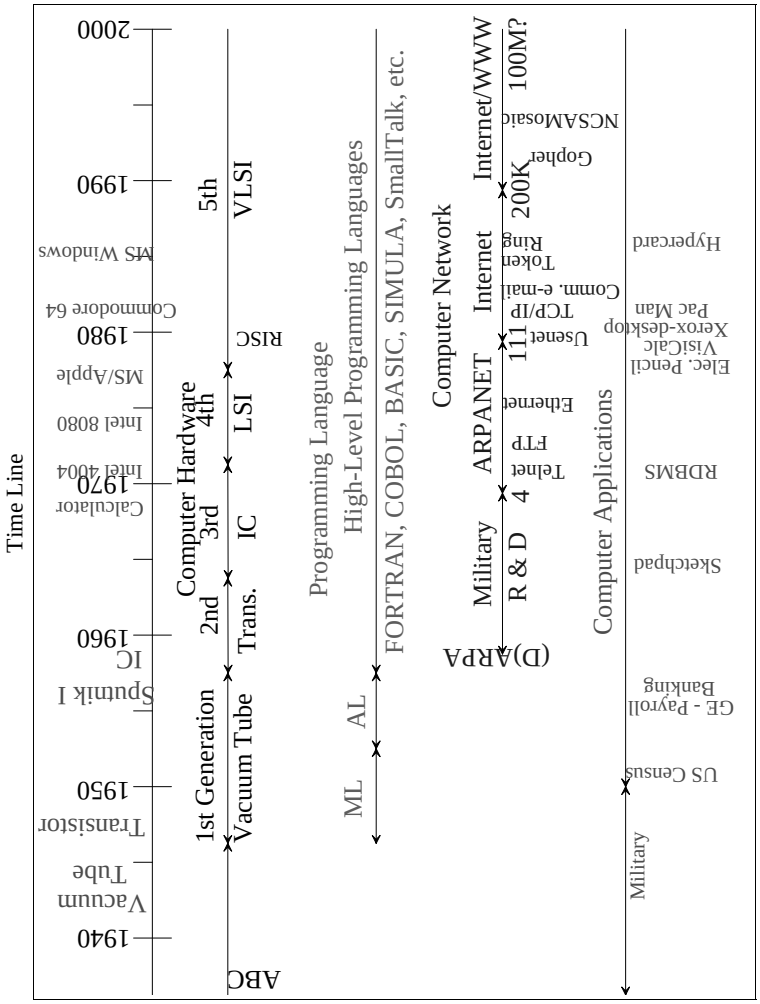
1969 - ARPA commissioned a Cambridge, Mass. company (Bolt, Beranek, and Newman) to build first packet switches Later, four computers connected

.
.
.

Soon networking bandwidth will be unlimited???

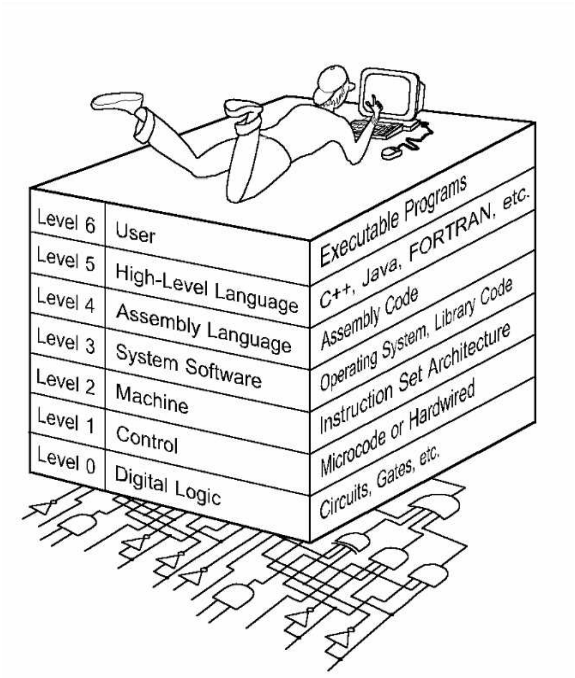
Computer Applications

Where are applications going???



The Computer Level Hierarchy

- Computers consist of many things besides chips.
- Before a computer can do anything worthwhile, it must also use software.
- Writing complex programs requires a “divide and conquer” approach, where each program module solves a smaller problem.
- Complex computer systems employ a similar technique through a series of virtual machine layers.



The machines at each level execute their own particular instructions, calling upon machines at lower levels to perform tasks as required.

- Level 6: The User Level
  - –Program execution and user interface level.
  - –The level with which we are most familiar.
- Level 5: High-Level Language Level
  - –The level with which we interact when we write programs in languages such as C, Pascal, Lisp, and Java.
- Level 4: Assembly Language Level
  - –Acts upon assembly language produced from Level 5, as well as instructions programmed directly at this level.
- Level 3: System Software Level
  - –Controls executing processes on the system.
  - –Protects system resources.
  - –Assembly language instructions often pass through Level 3 without modification.

- Level 2: Machine Level
  - –Also known as the Instruction Set Architecture (ISA) Level.
  - –Consists of instructions that are particular to the architecture of the machine.
  - –Programs written in machine language need no compilers, interpreters, or assemblers.
- Level 1: Control Level
  - –A *control unit* decodes and executes instructions and moves data through the system.
  - –Control units can be *microprogrammed* or *hardwired*.
  - –A microprogram is a program written in a low-level language that is implemented by the hardware.
  - –Hardwired control units consist of hardware that directly executes machine instructions.

- Level 0: Digital Logic Level
  - –This level is where we find digital circuits (the chips).
  - –Digital circuits consist of gates and wires.
  - –These components implement the mathematical logic of all other levels.