Computer Architecture (CSE-2207)

Computer Evolution

Agnik Saha

Department of Computer Science and Engineering

R. P. Shaha University

September 04, 2023

Introduction

- We begin with a brief, introductory look at the components in a computer system
- We will then consider the evolution of computer hardware
- We end this chapter by considering the structure of the typical computer, known as a Von Neumann computer
- Its noteworthy that anything that can be done in software can also be done in hardware and vice versa
  - This is known as the principle of *equivalence of Hardware and Software*
    - general-purpose computers allow the instructions to be stored in memory and executed through a decoding process
    - we could take any program and “hard-wire” it to be executed directly without the decoding – this is faster, but not flexible

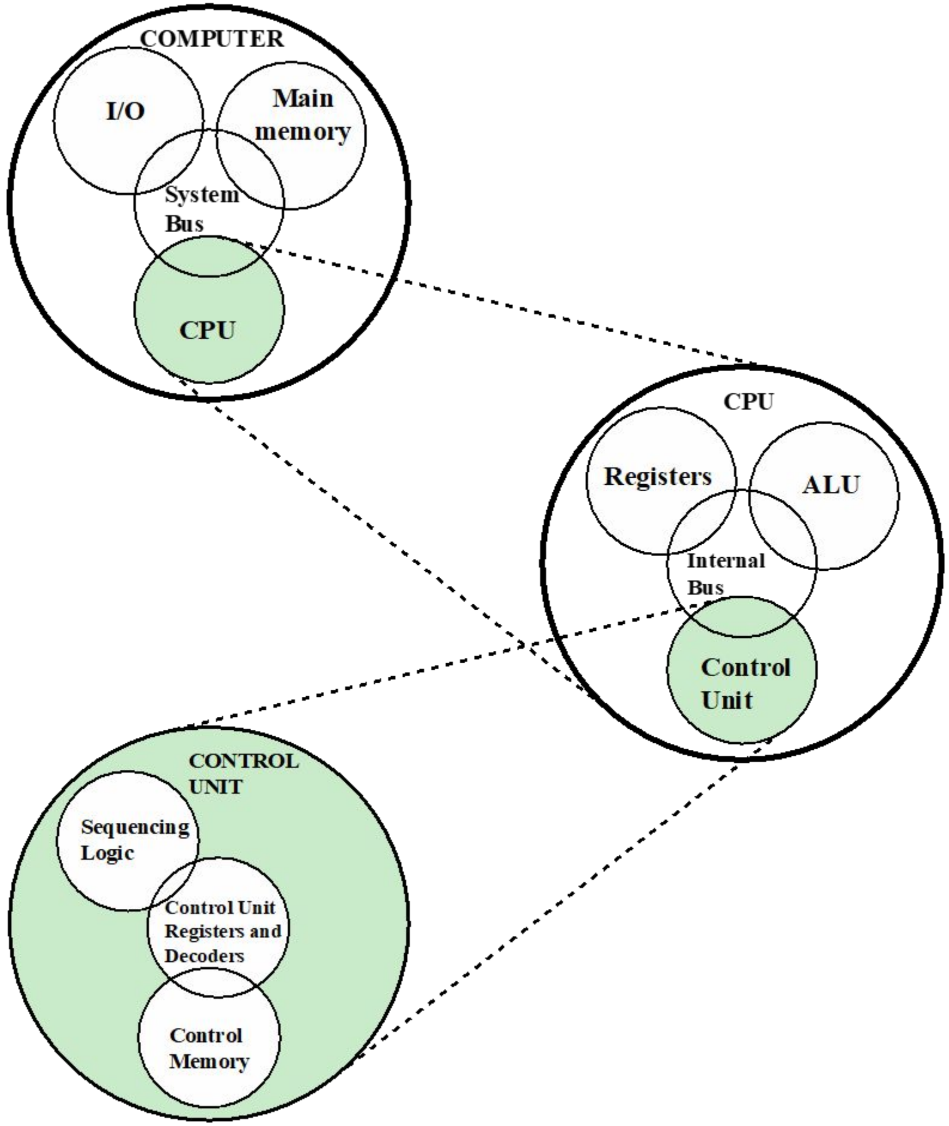


Figure 1.1 A Top-Down View of a Computer

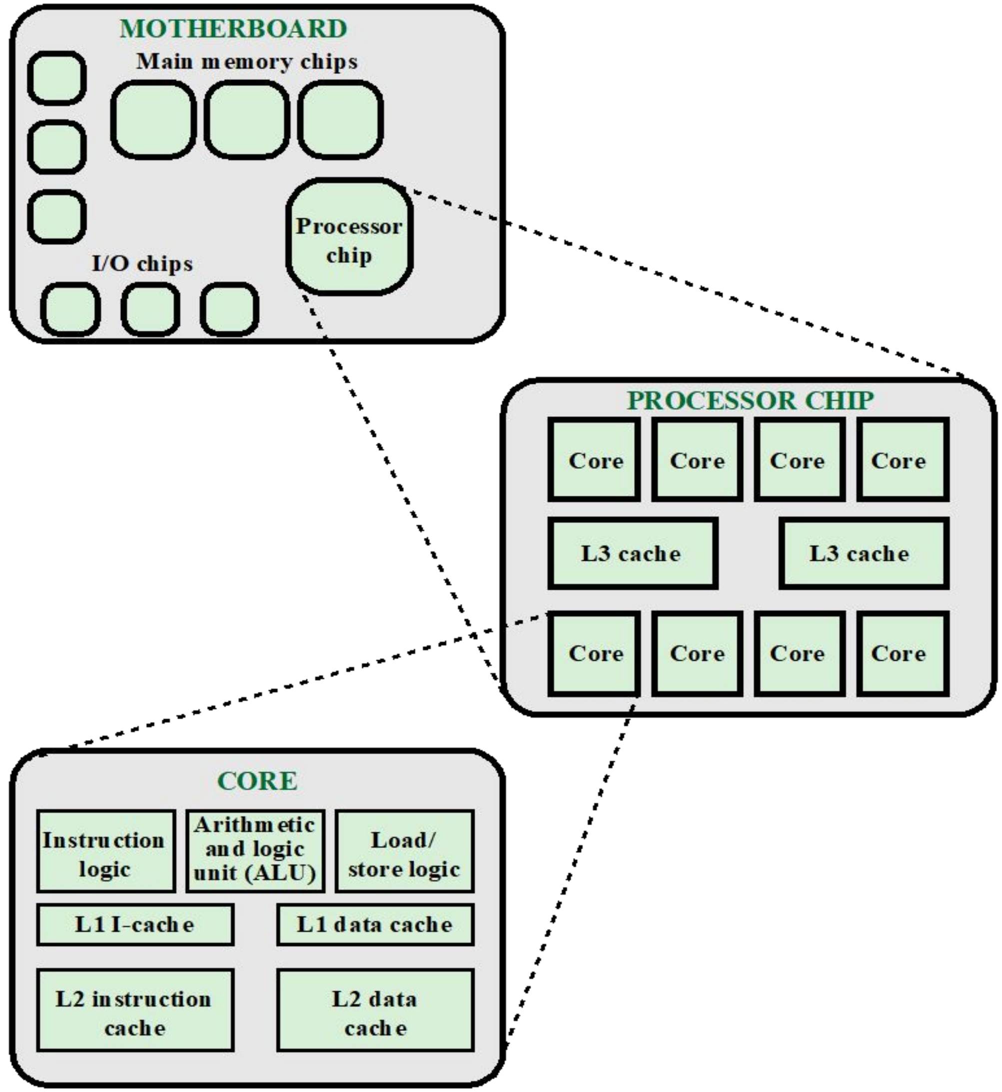
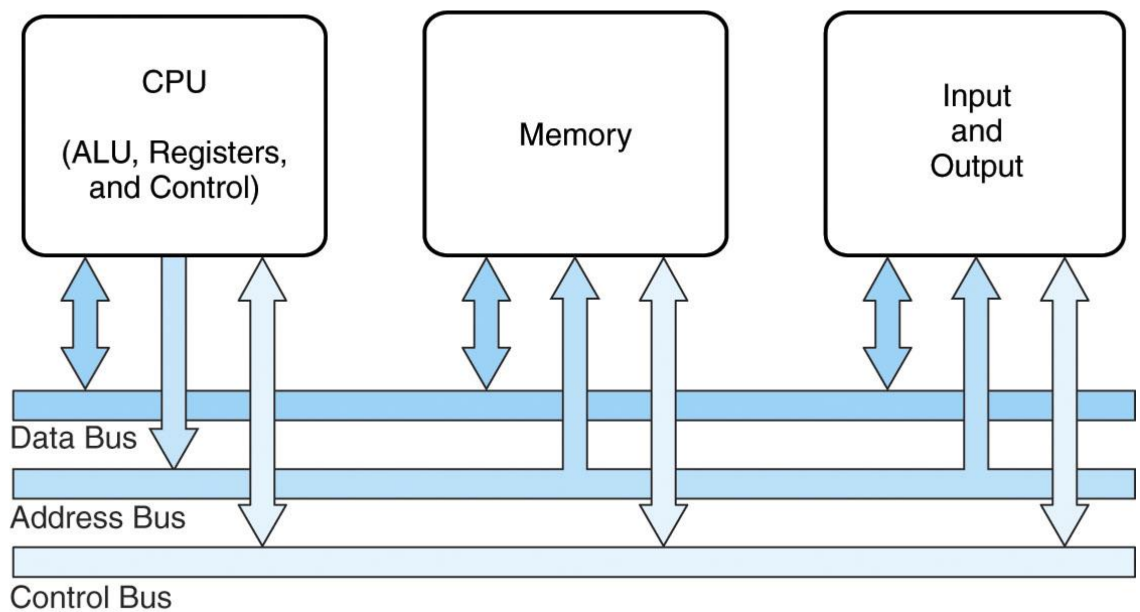


Figure 1.2 Simplified View of Major Elements of a Multicore Computer

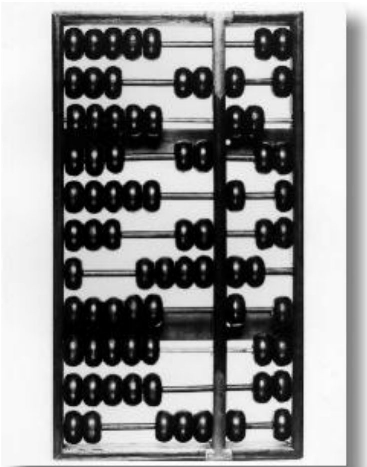
The Main Components

- CPU
  - does all processing and controls the other elements of the computer
    - it contains circuits to perform the execution of all arithmetic and logic operations (ALU), temporary storage (Registers) and the circuits to control the entire computer
- Memory
  - stores data and program instructions
    - includes cache, RAM memory, ROM memory
- Input and Output (I/O)
  - to communicate between the computer and the world
- The Bus
  - to move information from one component to another
  - divided into three sub-buses, one each for data, addresses and control signals



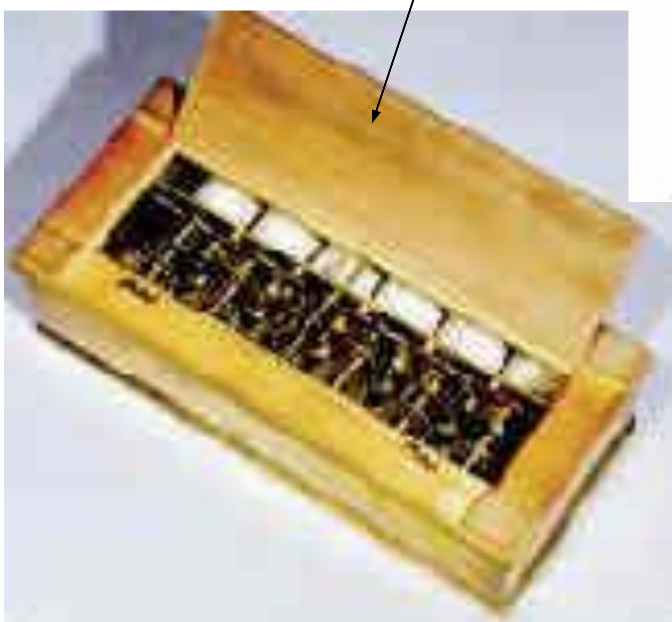
A History Lesson

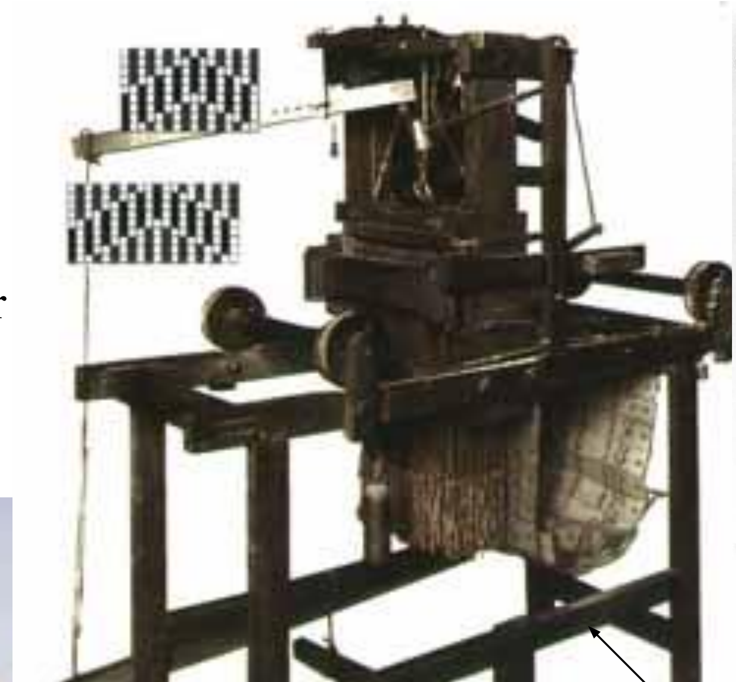
Early mechanical computational devices

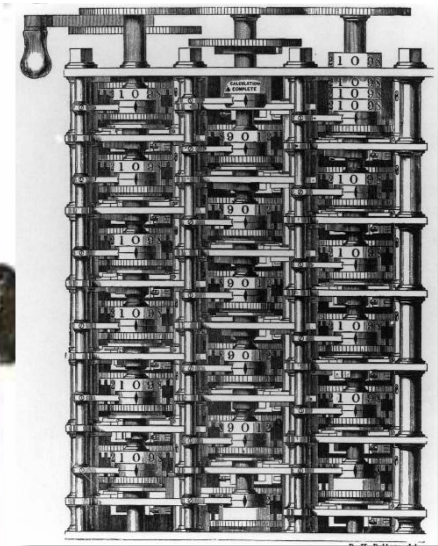


Abacus

Pascal's Calculator (1600s)







Early programmable devices:

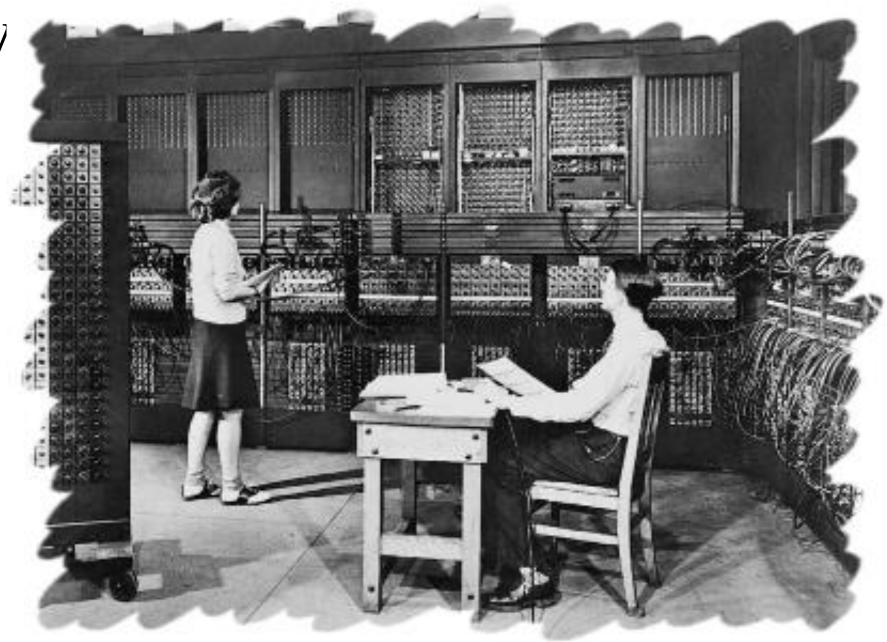
Jacquard's Loom (1800)

Babbage's Analytical Engine (1832)

Tabulating machine for 1890 census

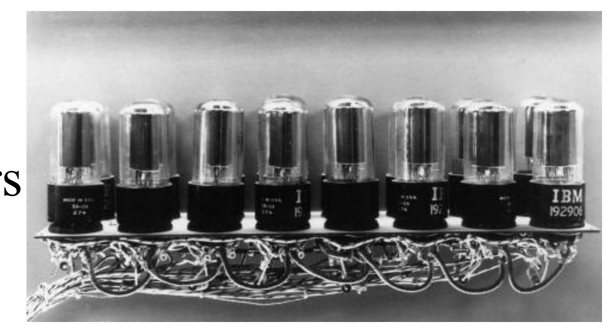
1st Generation Computers

- One of a kind laboratory machines
  - Used vacuum tubes for logic and storage (very little storage available)
  - Programmed in machine language
  - Often programmed by physical connection (hardwiring)
  - Slow, unreliable, expensive
    - Noteworthy computers
      - Z1
      - ABC
      - ENIAC



The ENIAC – often thought of as the first programmable electronic computer – 1946

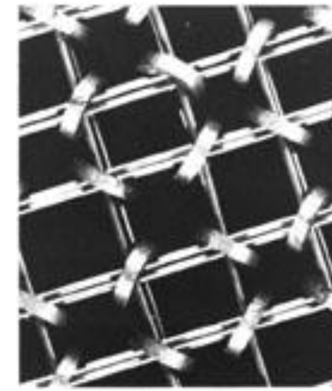
17468 vacuum tubes, 1800 square feet, 30 tons

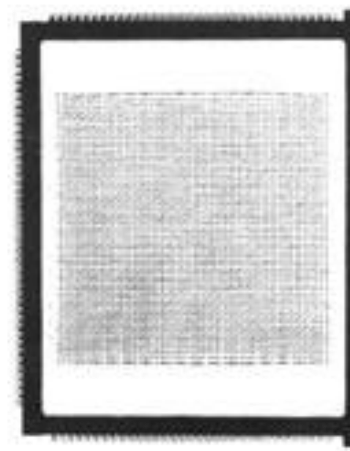


A vacuum-tube circuit storing 1 byte

2nd Generation Computers

- Transistors replaced vacuum tubes
- Magnetic core memory introduced
  - These changes in technology brought about cheaper and more reliable computers (vacuum tubes were very unreliable)
  - Because these units were smaller, they were closer together providing a speedup over vacuum tubes
  - Various programming languages introduced (assembly, high-level)
  - Rudimentary OS developed
    - The first supercomputer was introduced, CDC 6600 ($10 million)
    - Other noteworthy computers were the IBM 7094 and DEC PDP-1 mainframes





An array of magnetic core memory – very expensive – $1 million for 1 Mbyte!

3rd Generation Computers

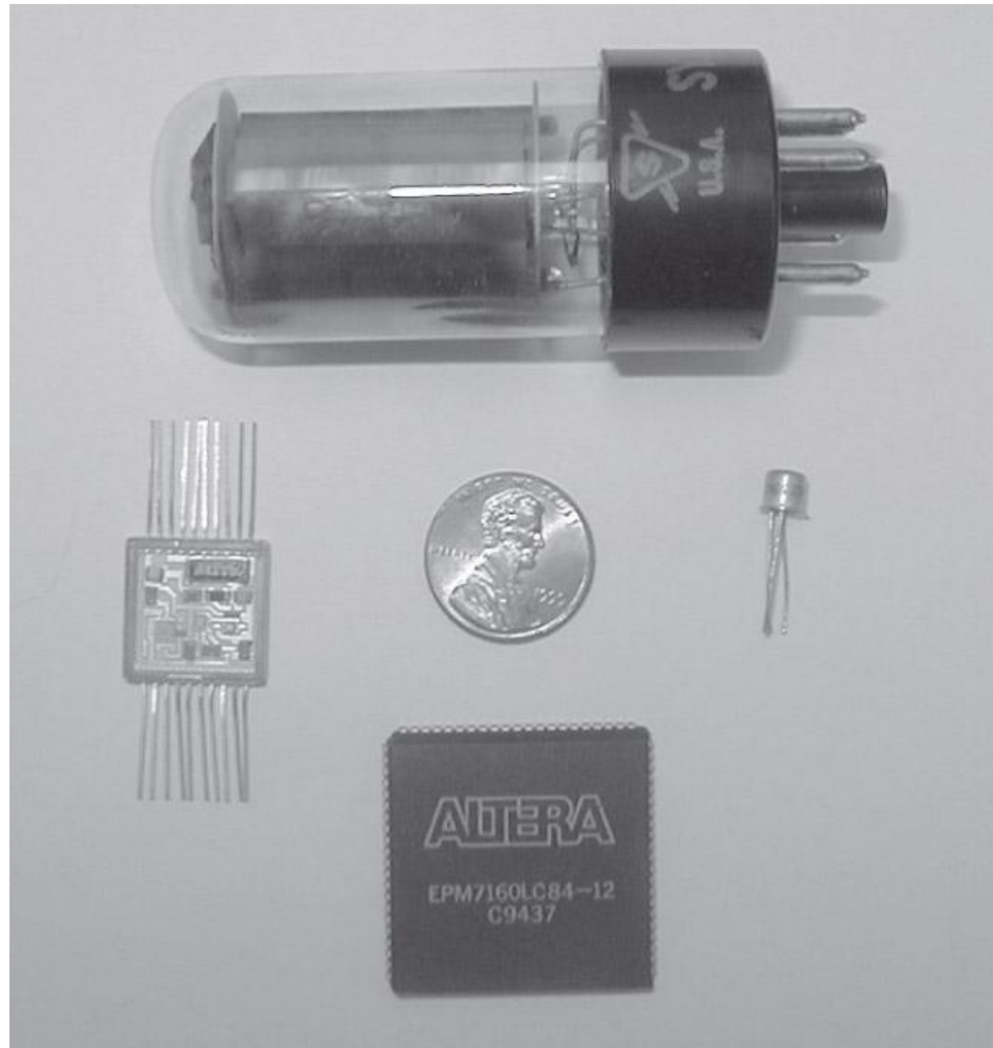
- Integrated circuit (IC) – or the ability to place circuits onto silicon chips
  - Replaced both transistors and magnetic core memory
  - Result was easily mass-produced components reducing the cost of computer manufacturing significantly
  - Also increased speed and memory capacity
  - Computer families introduced
  - Minicomputers introduced
  - More sophisticated programming languages and OS developed
    - Popular computers included PDP-8, PDP-11, IBM 360 and Cray produced their first supercomputer, Cray-1

Silicon chips now contained both logic (CPU) and memory

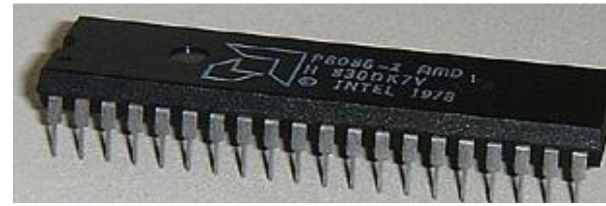
Large-scale computer usage led to time-sharing OS

Size Comparisons

- Here we see the size comparisons of
  - Vacuum tubes (1st generation technology)
  - Transistor (middle right, 2nd generation technology)
  - Integrated circuit (middle left, 3rd and 4th generation technology)
  - Chip (3rd and 4th generation technology)
  - And a penny for scale



4th Generation Computers



- Miniaturization took over
  - From SSI (10-100 components per chip) to
  - MSI (100-1000), LSI (1,000-10,000), VLSI (10,000+)
- Intel developed a CPU on a single chip – the microprocessor
  - This led to the development of microcomputers – PCs and later workstations and laptops
- Most of the 4th generation has revolved around not new technologies, but the ability to better use the available technology
  - with more components per chip, what are we going to use them for? More processing elements? More registers? More cache? Parallel processing? Pipelining? Etc.

The PC Market

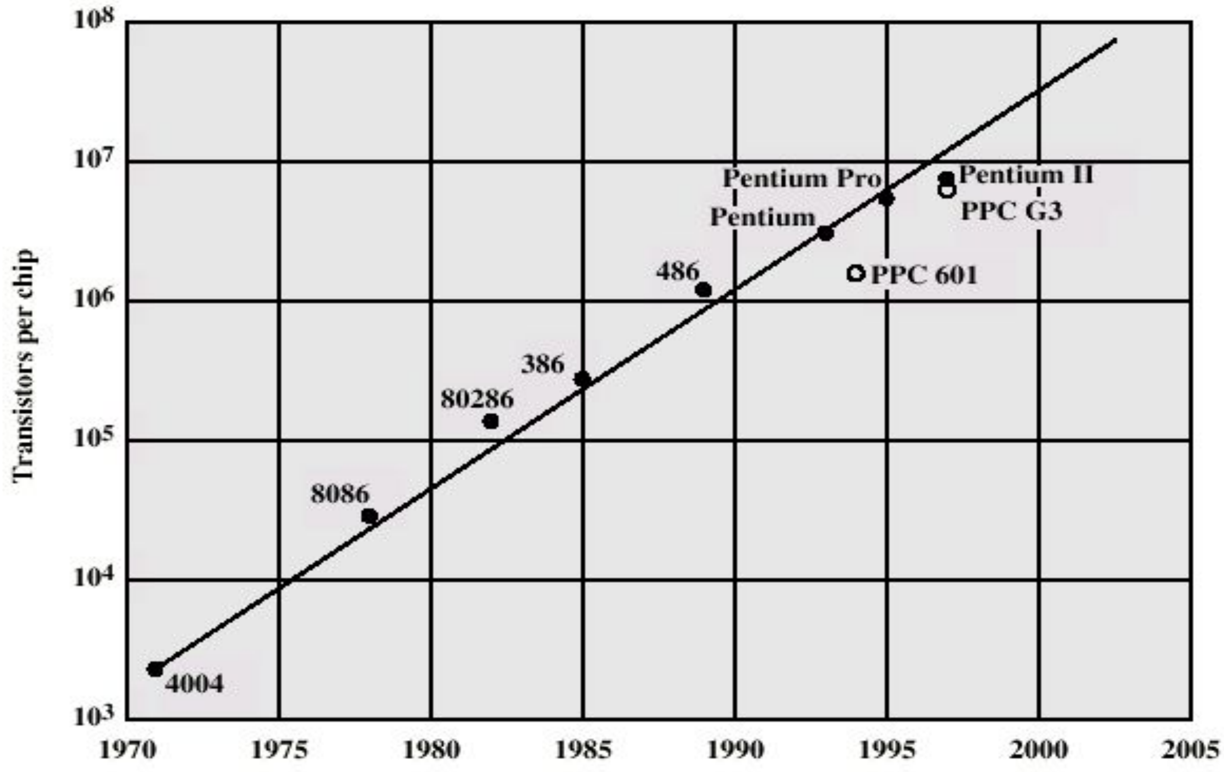
- The impact on miniaturization was not predicted
  - Who would have thought that a personal computer would be of any interest?
  - Early PCs were hobbyist toys and included Radio Shack, Commodore, Apple, Texas Instruments, and Altair
  - In 1981, IBM introduced their first PC
    - they decided to publish their architecture which led to clones or compatible computers
    - Microsoft wrote business software for the IBM platform thus making the machine more appealing
  - These two situations allowed IBM to capture a large part of the PC marketplace
  - Over the years since 1981, PC development has been one of the biggest concerns of the computer industry
    - More memory, faster processors, better I/O devices and interfaces, more sophisticated OS and software

Other Computer Developments

- During the 4th generation, we have seen improvements to other platforms as well
  - Mainframe and minicomputers much faster with substantially larger main memories
  - Workstations introduced to provide multitasking for scientific applications
  - Supercomputers reaching 10s or 100s of trillions of instructions per second speed
  - Massive parallel processing machines
  - Servers for networking
  - Architectural innovations have included
    - Floating point and multimedia hardware, parallel processing, pipelining, superscalar pipelines, speculative hardware, cache, RISC

Moore's Law

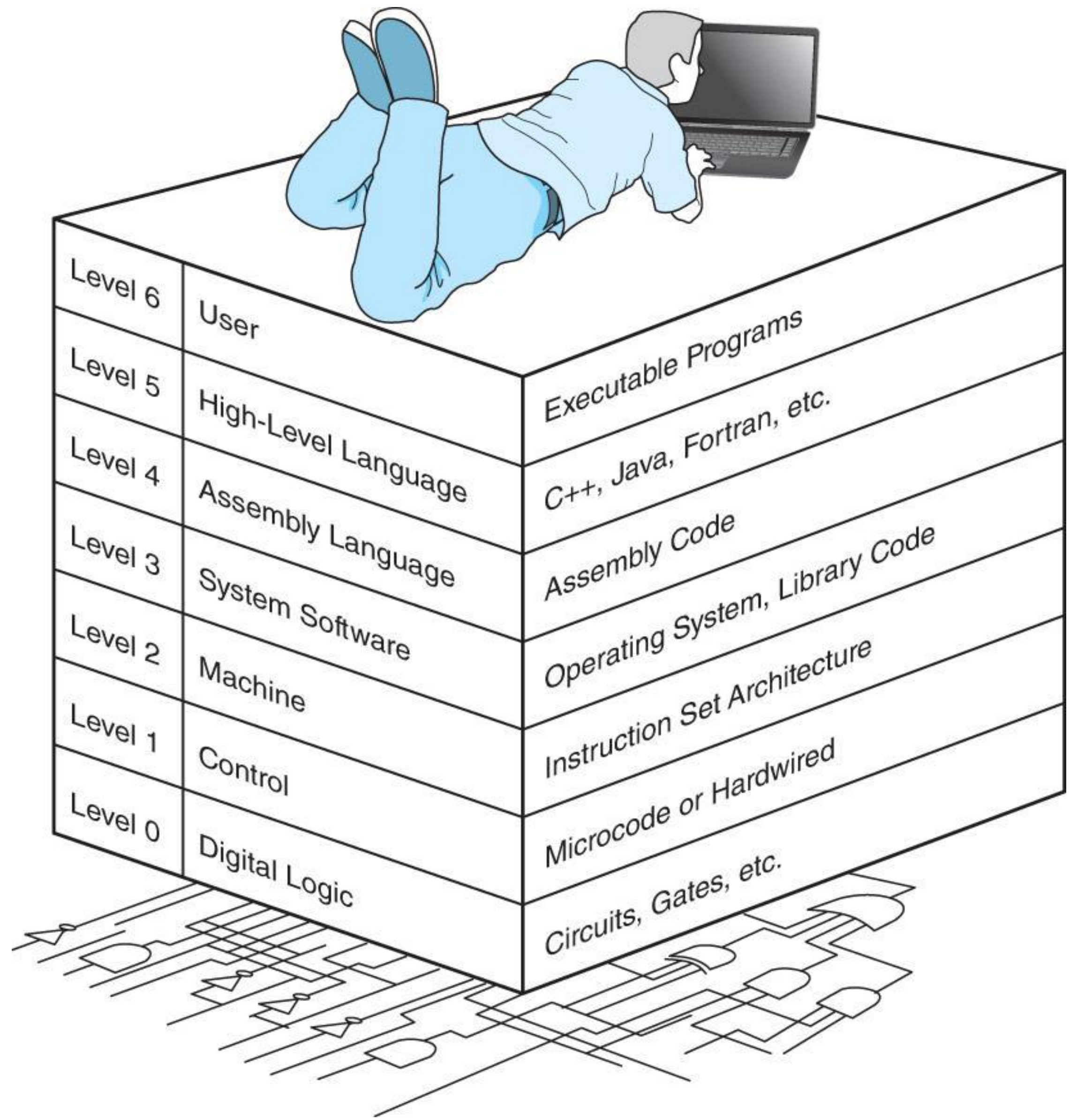
- Gordon Moore (Intel founder) noted that transistor density was increasing by a factor of 2 every 2 years
  - This observation or prediction has held out pretty well since he made it in 1965 (transistor count doubles roughly every 2 years)



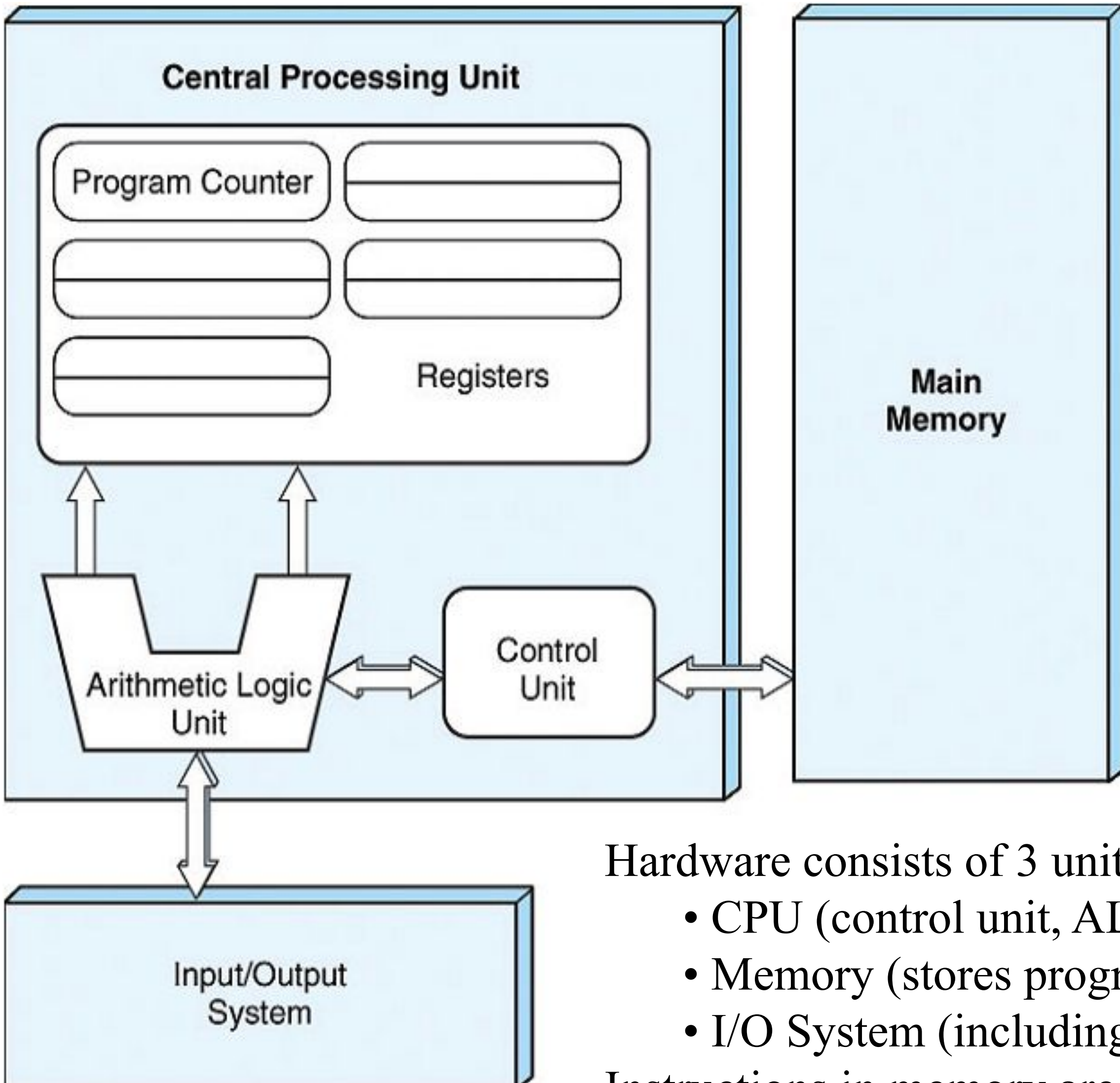
The growth has meant an increase in transistor count (and therefore memory capacity and CPU capability) of about 2^{20} since 1965, or computers 1 million times more capable!

How much longer can Moore's Law continue?

View of Computing Through Abstraction



The Von Neumann Architecture



Named after John von Neumann, Princeton, he designed a computer architecture whereby data and instructions would be retrieved from memory, operated on by an ALU, and moved back to memory (or I/O)

This architecture is the basis for most modern computers (only parallel processors and a few other unique architectures use a different model)

Hardware consists of 3 units

- CPU (control unit, ALU, registers)
- Memory (stores programs and data)
- I/O System (including secondary storage)

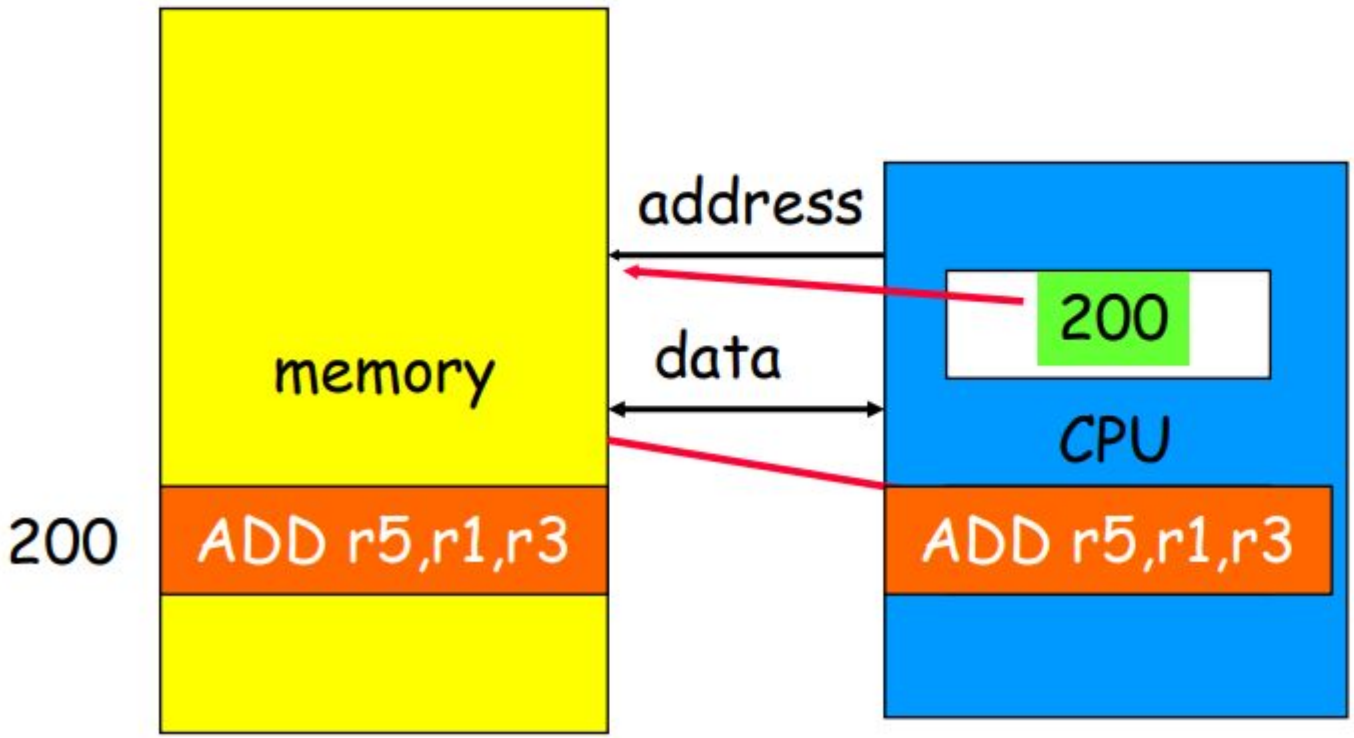
Instructions in memory are executed sequentially unless a program instruction explicitly changes the order

More on Von Neumann Architectures

- There is a single pathway used to move both data and instructions between memory, I/O and CPU
  - the pathway is implemented as a bus
  - the single pathway creates a bottleneck
    - known as the *von Neumann bottleneck*
  - A variation of this architecture is the *Harvard architecture* which separates data and instructions into two pathways
  - Another variation, used in most computers, is the system bus version in which there are different buses between CPU and memory and memory and I/O

- The von Neumann architecture operates on the *fetch-execute cycle*
  - Fetch an instruction from memory as indicated by the Program Counter register
  - Decode the instruction in the control unit
  - Data operands needed for the instruction are fetched from memory
  - Execute the instruction in the ALU storing the result in a register
  - Move the result back to memory if needed

Von Neumann Architectures



Von Neumann vs. Harvard

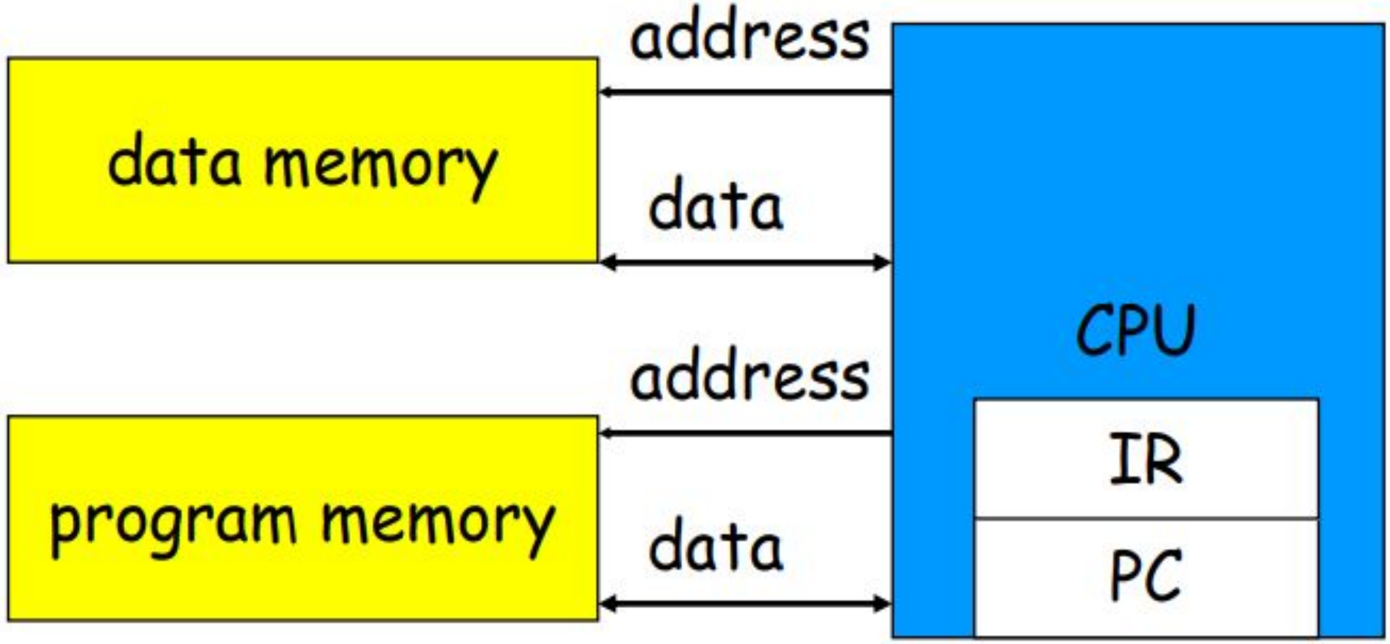
Von Neumann

- Same memory holds data, instructions.
- A single set of address/data buses between CPU and memory

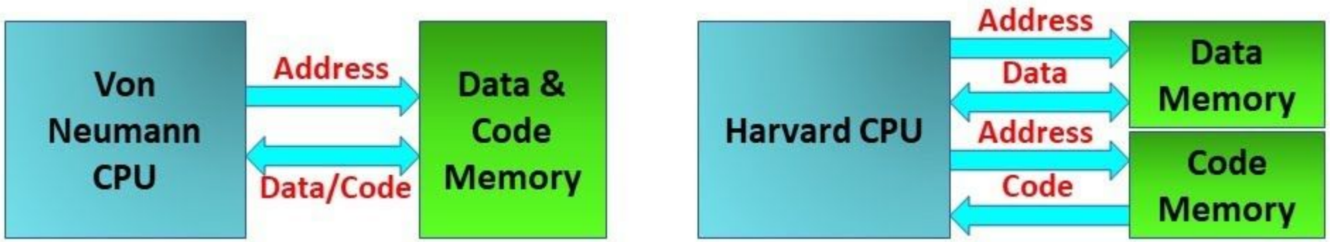
Harvard

- Separate memories for data and instructions.
- Two sets of address/data buses between CPU and memory

Harvard Architecture



Von Neumann vs. Harvard



Parameters	Von Neumann	Harvard
Memory	❖ Data and Program {Code} stored in same memory	❖ Data and Program {Code} stored in different memory
Memory Type	❖ It has only RAM for Data & Code	❖ It has RAM for Data and ROM for Code
Buses	❖ Common bus for Address & Data/Code	❖ Separate Bus Address & Data/Code
Program Execution	❖ Code is executed serially and takes more cycles	❖ Code is executed in parallel with data so it takes less cycles.
Data/Code Transfer	❖ Data or Code in one cycle	❖ Data and Code in One cycle
Control Signals	❖ Less	❖ More
Space	❖ It needs less Space	❖ It needs more space
Cost	❖ Less	❖ Costly

Microprocessors

	von Neumann	Harvard
RISC	ARM7	ARM9
CISC	Pentium	SHARC (DSP)

IAS

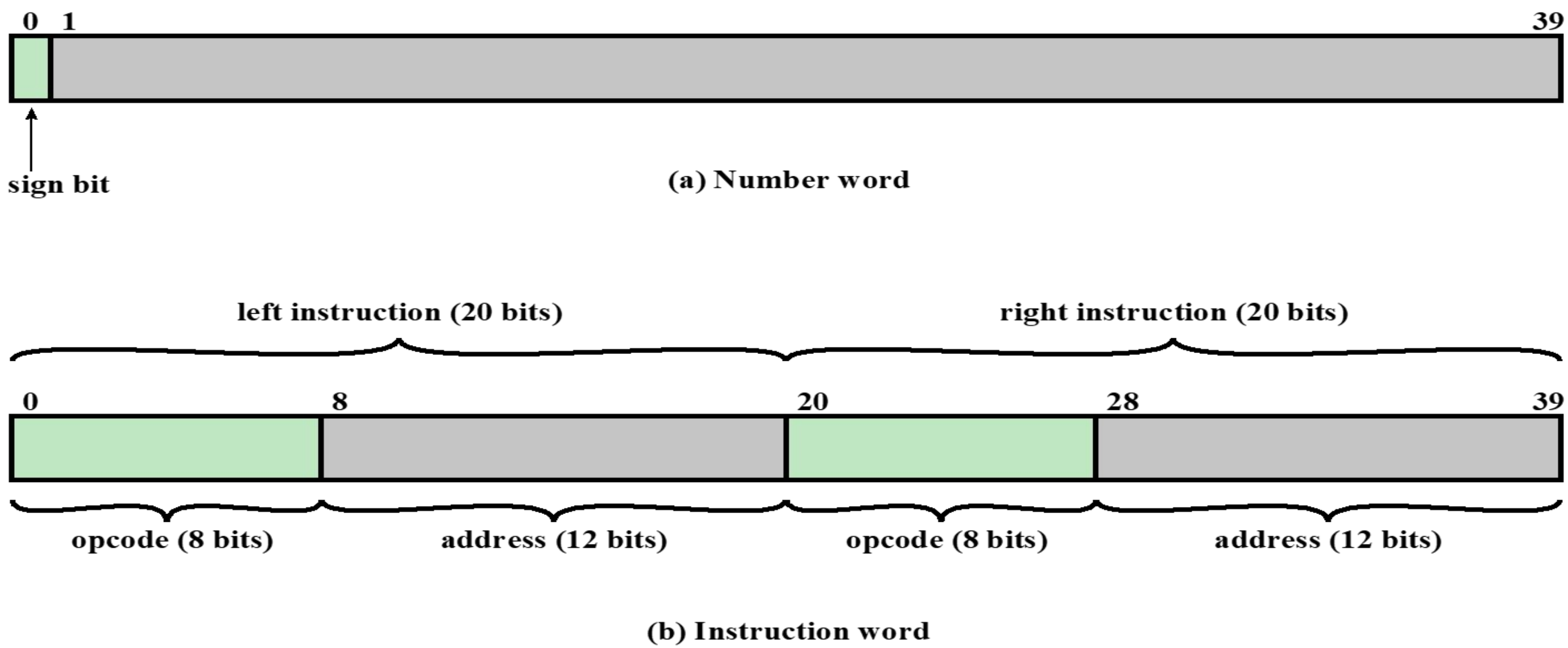


Figure 1.7 IAS Memory Formats

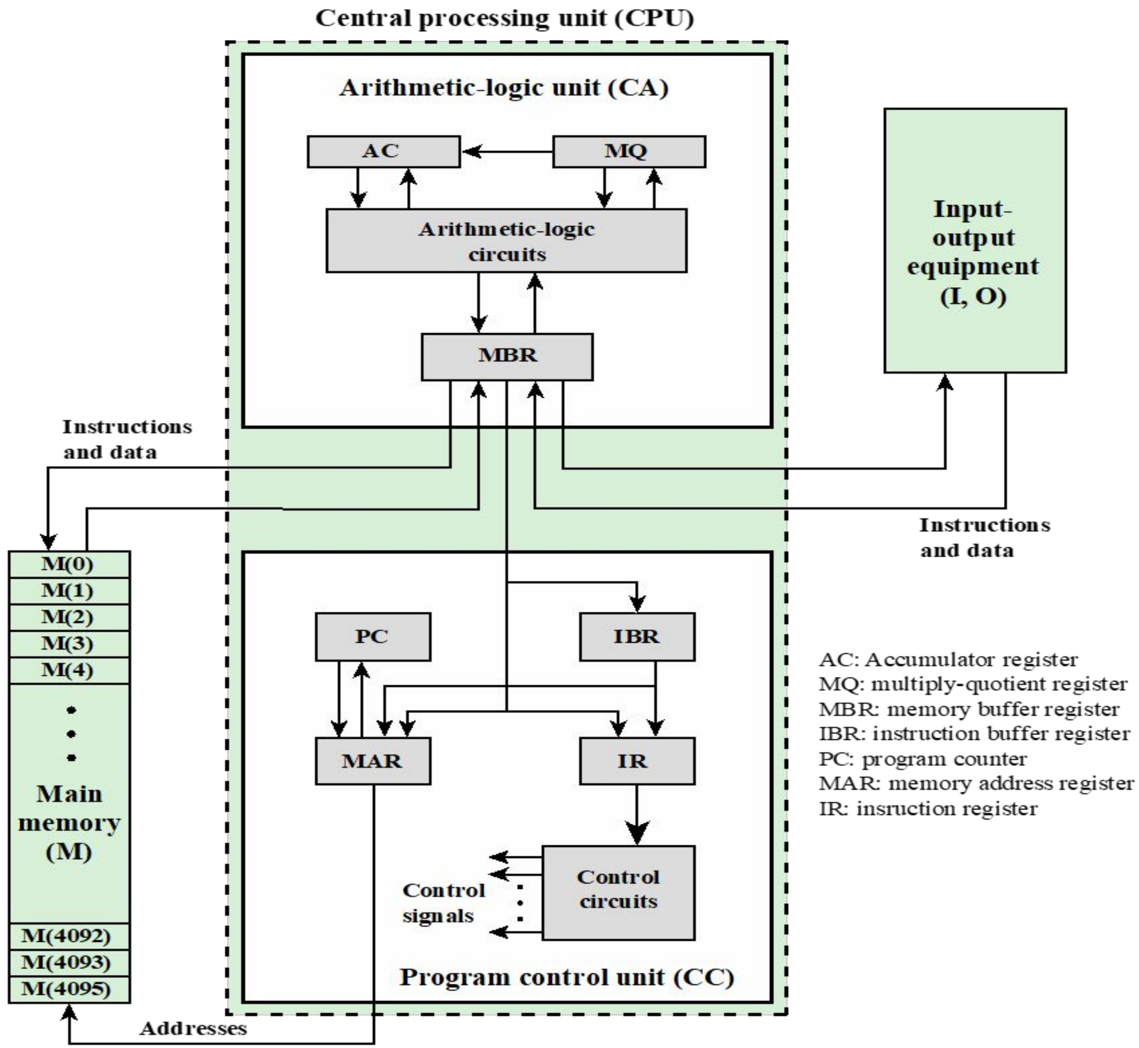


Figure 1.6 IAS Structure

Registers

Memory buffer register (MBR)

•Contains a word to be stored in memory or sent to the I/O unit

•Or is used to receive a word from memory or from the I/O unit

Memory address register (MAR)

•Specifies the address in memory of the word to be written from or read into the MBR

Instruction register (IR)

•Contains the 8-bit opcode instruction being executed

Instruction buffer register (IBR)

•Employed to temporarily hold the right-hand instruction from a word in memory

Program counter (PC)

•Contains the address of the next instruction pair to be fetched from memory

Accumulator (AC) and multiplier quotient (MQ)

•Employed to temporarily hold operands and results of ALU operations

Instruction Type	Opcode	Symbolic Representation	Description
Data transfer	00001010	LOAD MQ	Transfer contents of register MQ to the accumulator AC
	00001001	LOAD MQ,M(X)	Transfer contents of memory location X to MQ
	00100001	STOR M(X)	Transfer contents of accumulator to memory location X
	00000001	LOAD M(X)	Transfer M(X) to the accumulator
	00000010	LOAD –M(X)	Transfer –M(X) to the accumulator
	00000011	LOAD |M(X)|	Transfer absolute value of M(X) to the accumulator
	00000100	LOAD –|M(X)|	Transfer –|M(X)| to the accumulator
Unconditional branch	00001101	JUMP M(X,0:19)	Take next instruction from left half of M(X)
	00001110	JUMP M(X,20:39)	Take next instruction from right half of M(X)
Conditional branch	00001111	JUMP+ M(X,0:19)	If number in the accumulator is nonnegative, take next instruction from left half of M(X)
		JUMP+ M(X,20:39)	*If number in the accumulator is nonnegative, take next instruction from right half of M(X)*
Arithmetic	00000101	ADD M(X)	Add M(X) to AC; put the result in AC
	00000111	ADD |M(X)|	Add |M(X)| to AC; put the result in AC
	00000110	SUB M(X)	Subtract M(X) from AC; put the result in AC
	00001000	SUB |M(X)|	Subtract |M(X)| from AC; put the remainder in AC
	00001011	MUL M(X)	Multiply M(X) by MQ; put most significant bits of result in AC, put least significant bits in MQ
	00001100	DIV M(X)	Divide AC by M(X); put the quotient in MQ and the remainder in AC
	00010100	LSH	Multiply accumulator by 2; i.e., shift left one bit position
	00010101	RSH	Divide accumulator by 2; i.e., shift right one position
Address modify	00010010	STOR M(X,8:19)	Replace left address field at M(X) by 12 rightmost bits of AC
	00010011	STOR M(X,28:39)	Replace right address field at M(X) by 12 rightmost bits of AC

Designing an ISA

- Important questions that need to be answered :
  - How many instructions should we have ?
  - What should they do ?
  - How complicated should they be ?

Two different paradigms : RISC and CISC

RISC
(Reduced Instruction Set Computer)

CISC
(Complex Instruction Set Computer)

RISC vs CISC

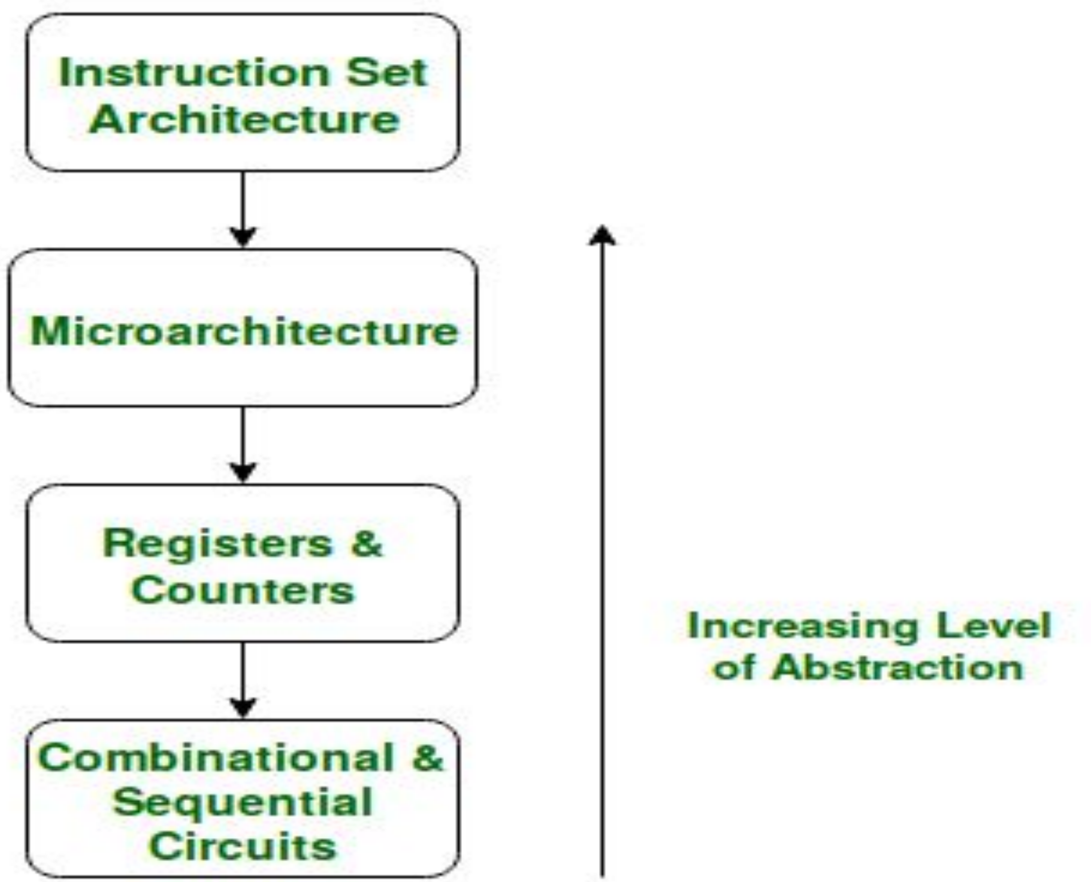
A reduced instruction set computer (RISC) implements simple instructions that have a simple and regular structure. The number of instructions is typically a small number (64 to 128). Examples: ARM, IBM PowerPC, HP PA-RISC

A complex instruction set computer (CISC) implements complex instructions that are highly irregular, take multiple operands, and implement complex functionalities. Secondly, the number of instructions is large (typically 500+). Examples: Intel x86, VAX

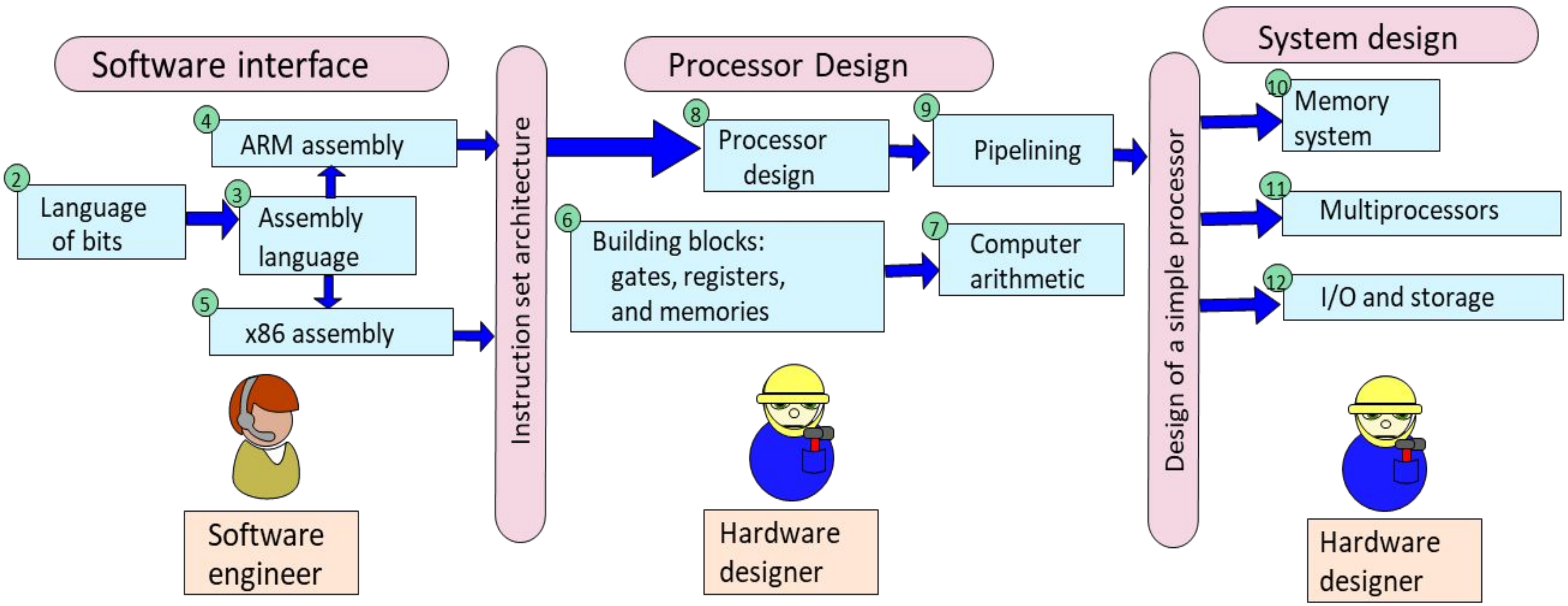
RISC vs CISC

Reduced Instruction Set Computer (RISC)
• Compact, uniform instructions -> facilitate pipelining
• More lines of code -> large memory footprint
• Allow effective compiler optimization Complex Instruction Set Computer (CISC)
• Many addressing modes and long instructions
• High code density
• Often require manual optimization of assembly code for embedded systems

Instruction Set Architecture



Roadmap



Software interface
Processor Design
System design
Instruction set architecture
Design of a simple processor
2
Language of bits
3
Assembly language
4
ARM assembly
5
x86 assembly
6
Building blocks: gates, registers, and memories
7
Computer arithmetic
8
Processor design
9
Pipelining
10
Memory system
11
Multiprocessors
12
I/O and storage
Software engineer
Hardware designer
Hardware designer

Summary

* Computers are dumb yet ultra-fast machines.
* Instructions are basic rudimentary commands used to communicate with the processor. A computer can execute billions of instructions per second.
* The compiler transforms a user program written in a high level language such as C to a program consisting of basic machine instructions.
* The instruction set architecture(ISA) refers to the semantics of all the instructions supported by a processor.
* The instruction set needs to be complete. It is desirable if it is also concise, generic, and simple.