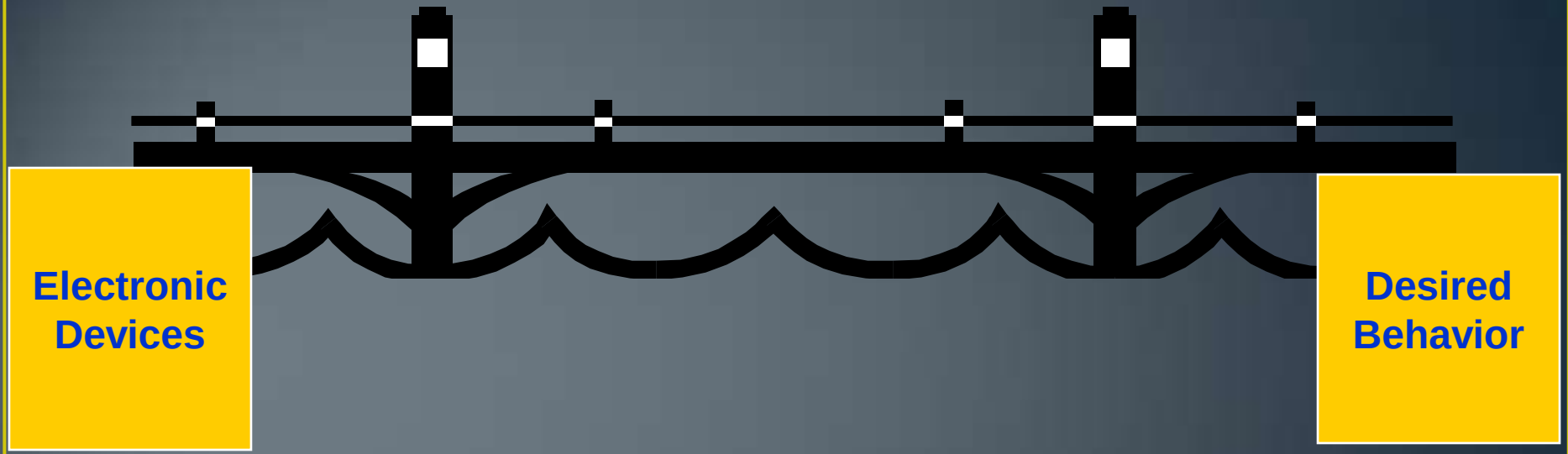
Computer Organization and Architecture

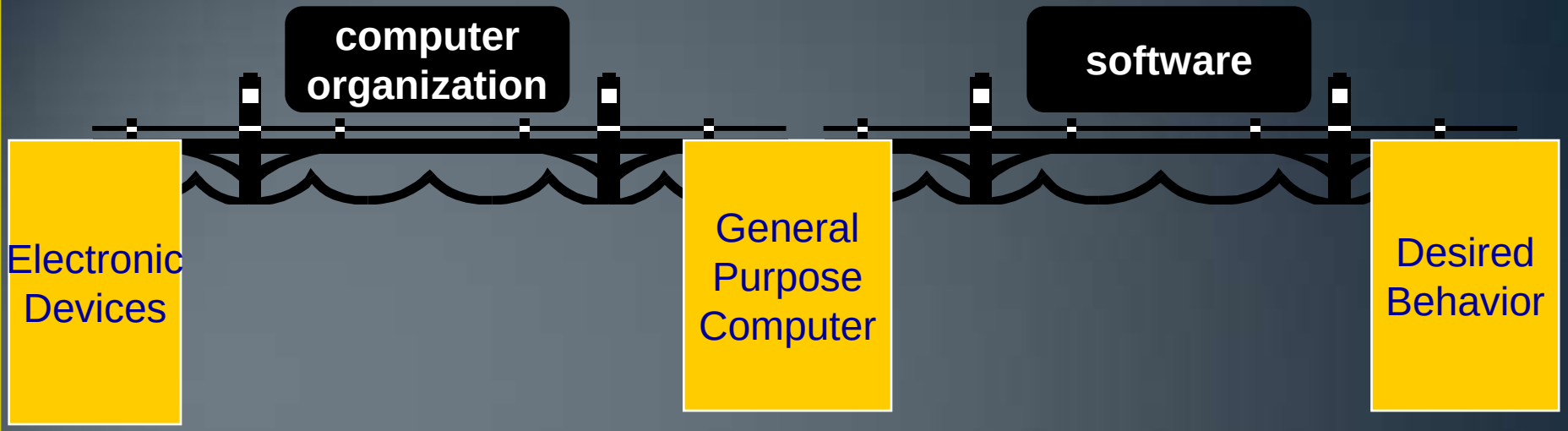
What is Computer Organization?



… a very wide semantic gap lies between the intended behavior and the workings of the underlying electronic devices that will actually do all the work.

The forerunners to modern computers attempted to assemble the raw devices (mechanical, electrical, or electronic) into a separate purpose-built machine for each desired behavior.

Role of General Purpose Computers



A general purpose computer is like an island that helps span the gap between the desired behavior (application) and the basic building blocks (electronic devices).

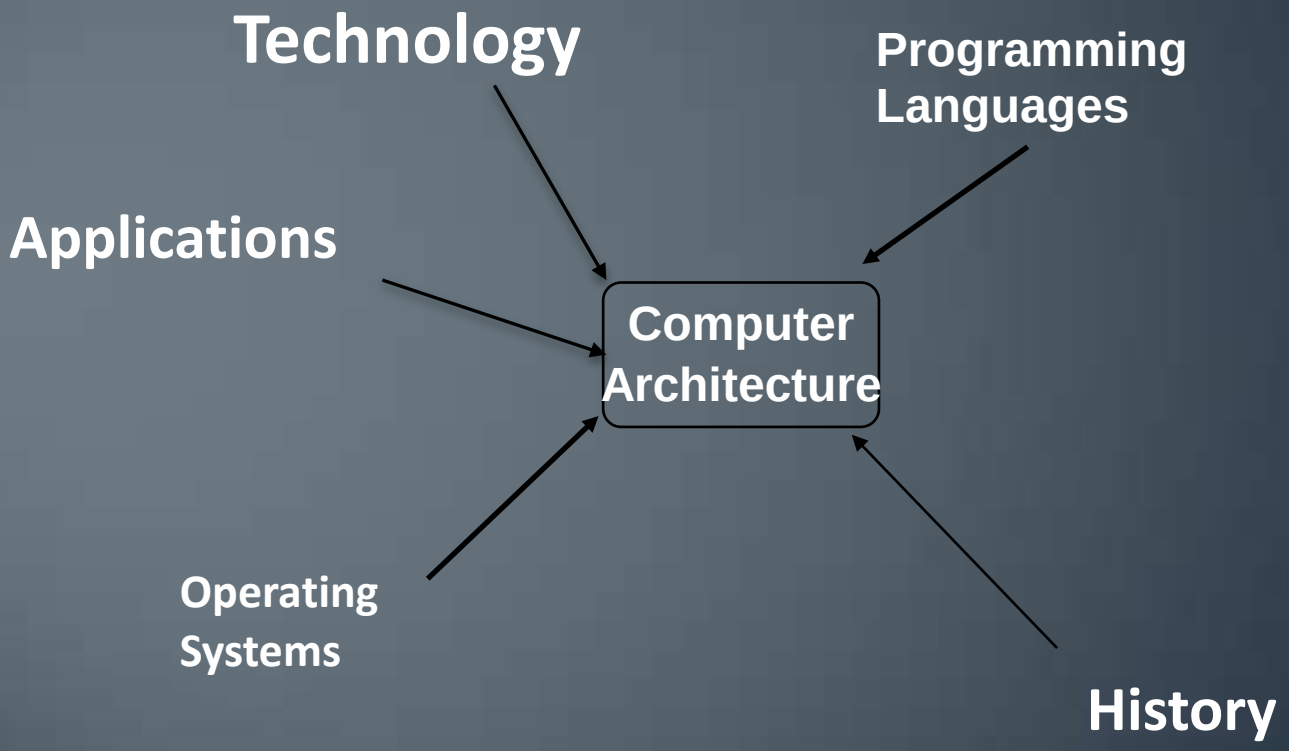
Computer Architecture - Definition

- Computer Architecture = ISA + MO
- Instruction Set Architecture
  - **What** the executable can “see” as underlying hardware
  - Logical View
- Machine Organization
  - **How** the hardware implements ISA ?
  - Physical View

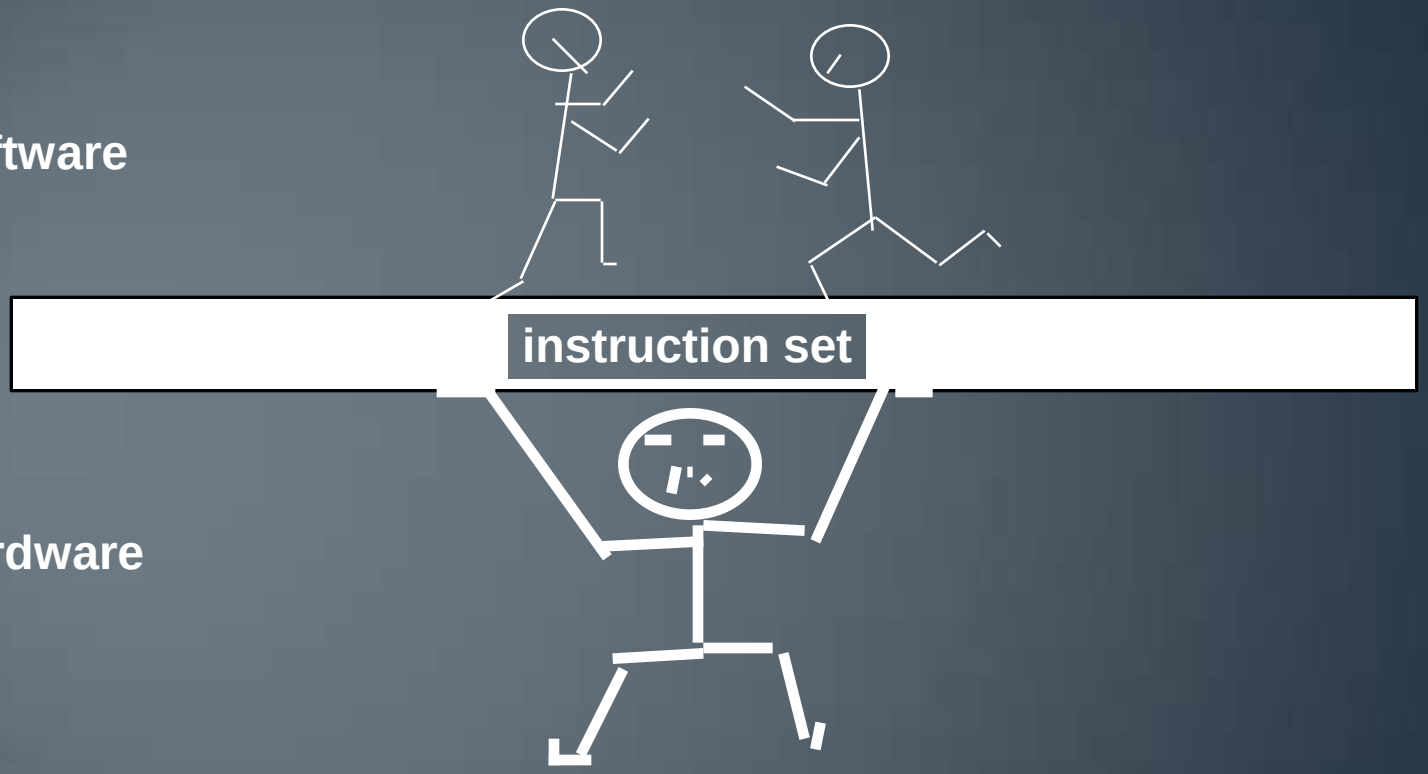
Impact of changing ISA

- Early 1990's Apple switched instruction set architecture of the Macintosh
  - From Motorola 68000-based machines
  - To PowerPC architecture
- Intel 80x86 Family: many implementations of same architecture
  - program written in 1978 for 8086 can be run on latest Pentium chip

Factors affecting ISA ???



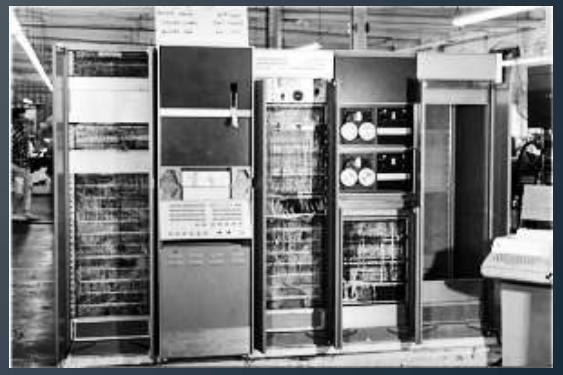
ISA: Critical Interface



Examples: 80x86 50,000,000 vs. MIPS 5500,000 ???

Designing Computers

- All computers more or less based on the same basic design, the Von Neumann Architecture!

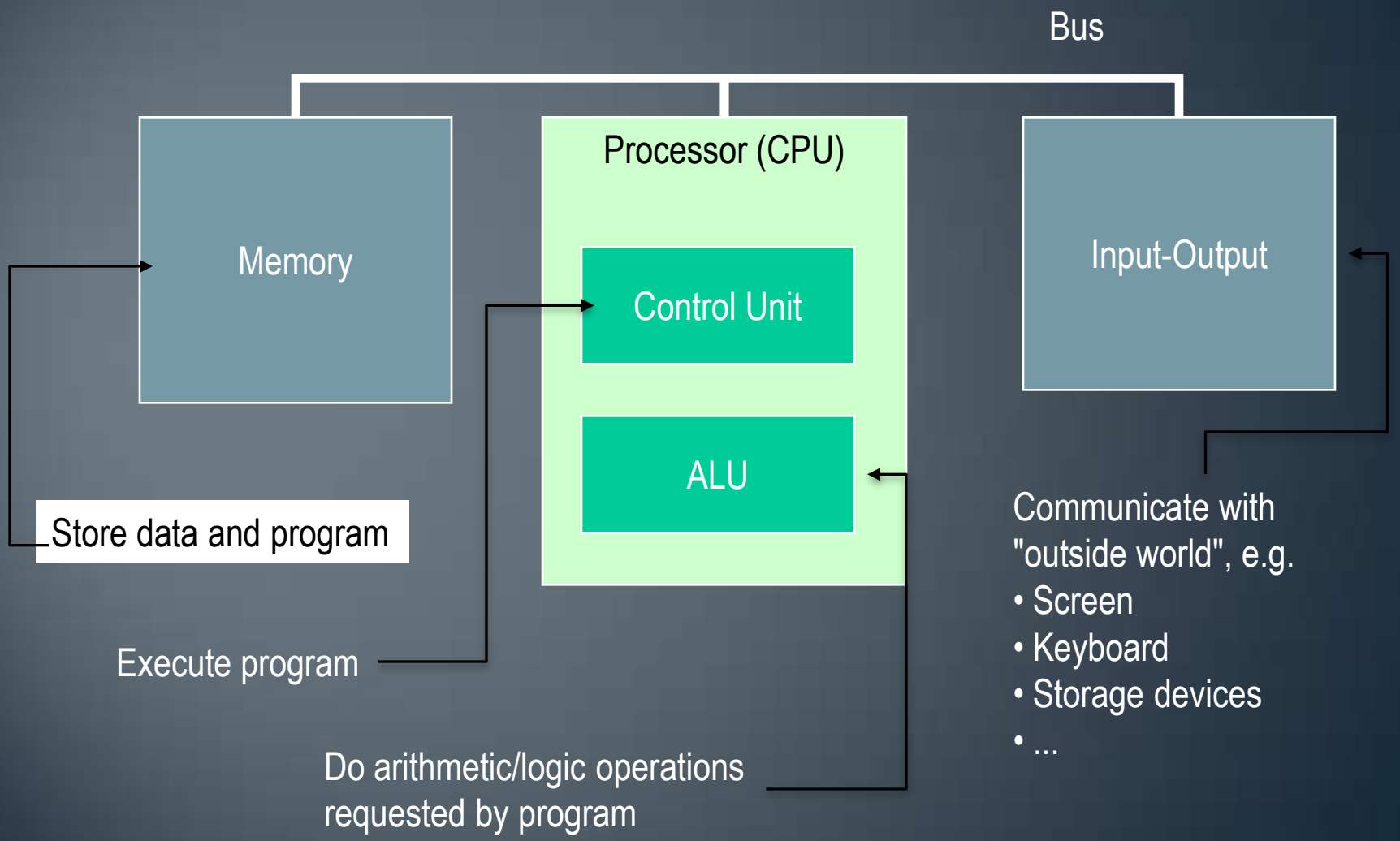


The Von Neumann Architecture

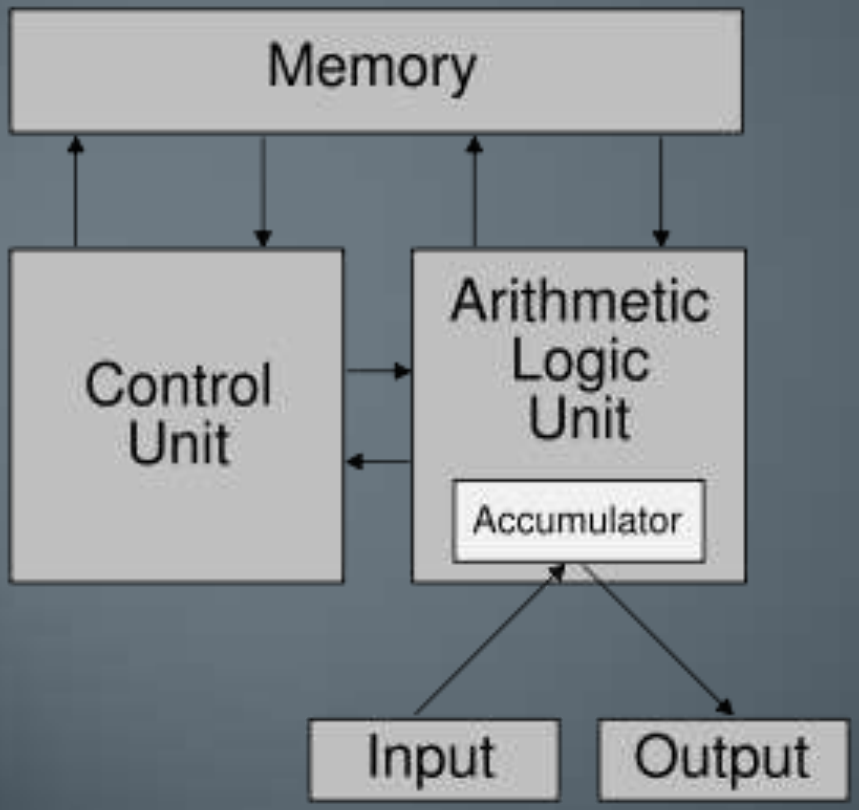
- Model for designing and building computers, based on the following three characteristics:

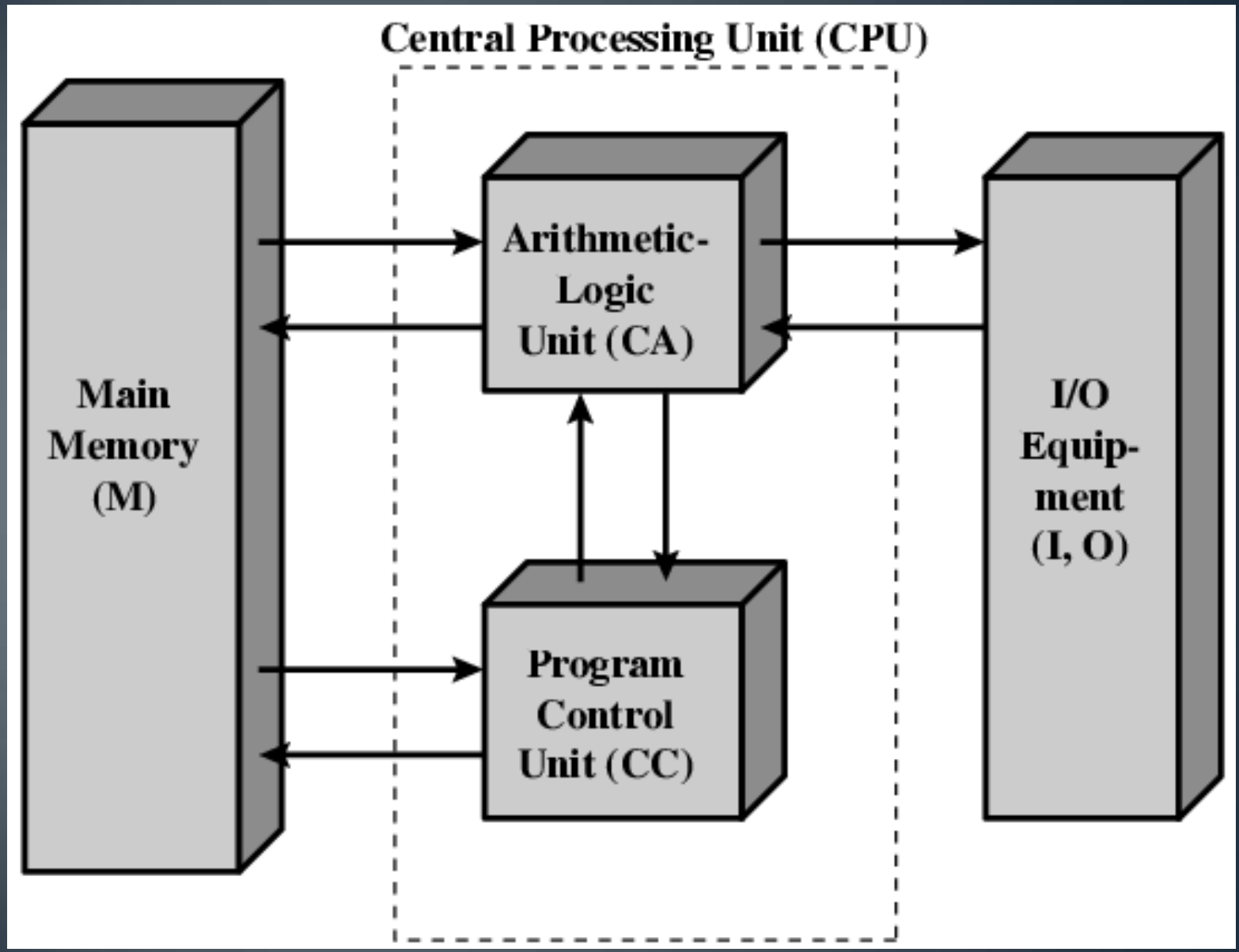
1) The computer consists of four main sub-systems:
   - Memory
   - ALU (Arithmetic/Logic Unit)
   - Control Unit
   - Input / Output System (I/O)
2) Program is stored in memory during execution.
3) Program instructions are executed sequentially.

The Von Neumann Architecture



Simplified Architecture




Central Processing Unit (CPU)
Arithmetic-
Logic
Unit (CA)
Main
Memory
(M)
I/O
Equip-
ment
(I, O)
Program
Control
Unit (CC)

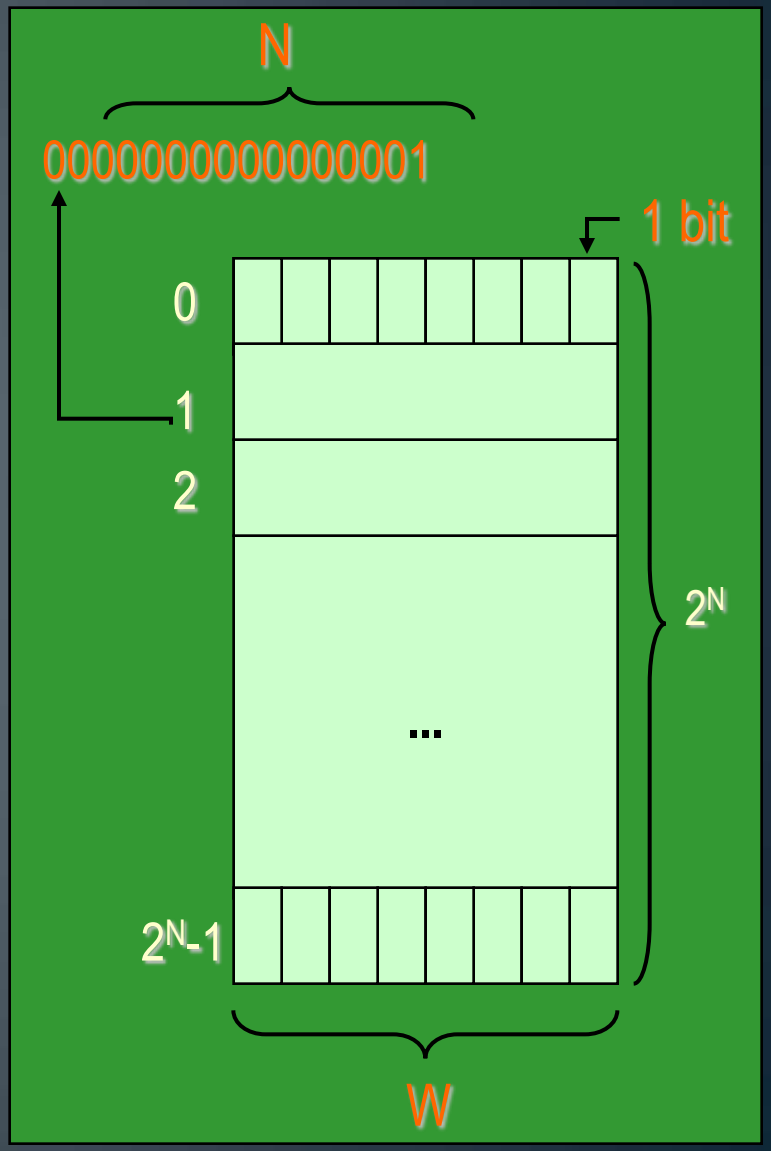
Memory Subsystem

- Memory, also called RAM (Random Access Memory),
  - Consists of many memory cells (storage units) of a fixed size. Each cell has an address associated with it: 0, 1, …
  - All accesses to memory are to a specified address. A cell is the minimum unit of access (fetch/store a complete cell).
  - The time it takes to fetch/store a cell is the same for all cells.
- When the computer is running, both
  - Program
  - Data (variables)

  are stored in the memory.

RAM

- Need to distinguish between
  - the address of a memory cell and the content of a memory cell
- Memory width (W):
  - How many bits is each memory cell, typically one byte (=8 bits)
- Address width (N):
  - How many bits used to represent each address, determines the maximum memory size = address space
  - If address width is N-bits, then address space is 2^N (0,1,...,2^N-1)



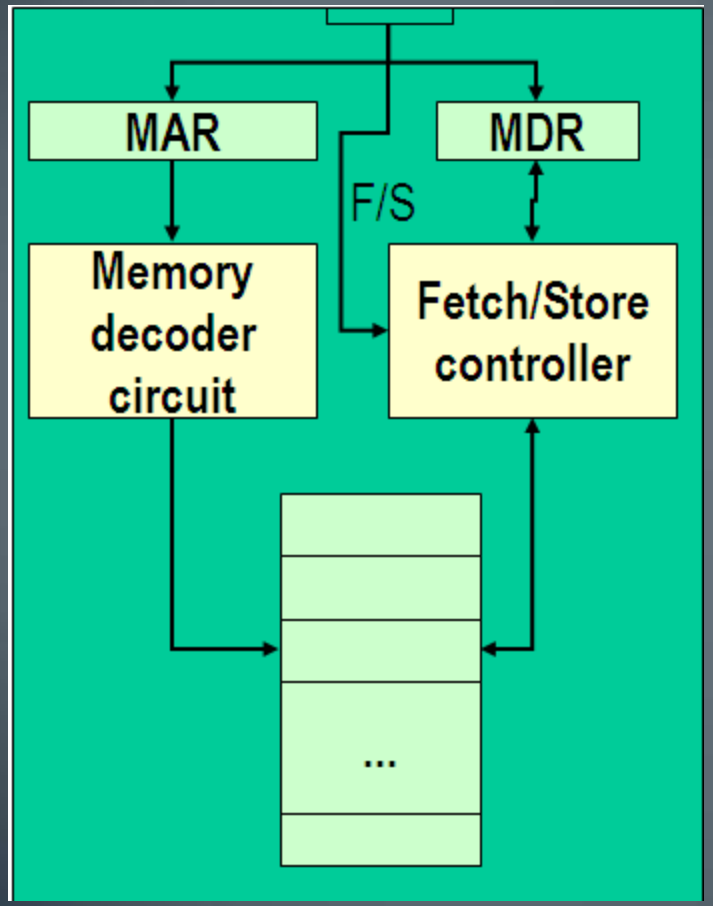
Memory Size / Speed

- Typical memory in a personal computer (PC):
  - 64MB - 256MB
- Memory sizes:
  - Kilobyte (KB) = 2^{10} = 1,024 bytes ~ 1 thousand
  - Megabyte(MB) = 2^{20} = 1,048,576 bytes ~ 1 million
  - Gigabyte (GB) = 2^{30} = 1,073,741,824 bytes ~ 1 billion
- Memory Access Time (read from/ write to memory)
  - 50-75 nanoseconds (1 nsec. = 0.000000001 sec.)
- RAM is
  - volatile (can only store when power is on)
  - relatively expensive

Operations on Memory

- Fetch (address):
  - Fetch a copy of the content of memory cell with the specified address.
  - Non-destructive, copies value in memory cell.
- Store (address, value):
  - Store the specified value into the memory cell specified by address.
  - Destructive, overwrites the previous value of the memory cell.
- The memory system is interfaced via:
  - Memory Address Register (MAR)
  - Memory Data Register (MDR)
  - Fetch/Store signal

Structure of the Memory Subsystem



- Fetch(address)
  - Load address into MAR.
  - Decode the address in MAR.
  - Copy the content of memory cell with specified address into MDR.
- Store(address, value)
  - Load the address into MAR.
  - Load the value into MDR.
  - Decode the address in MAR
  - Copy the content of MDR into memory cell with the specified address.

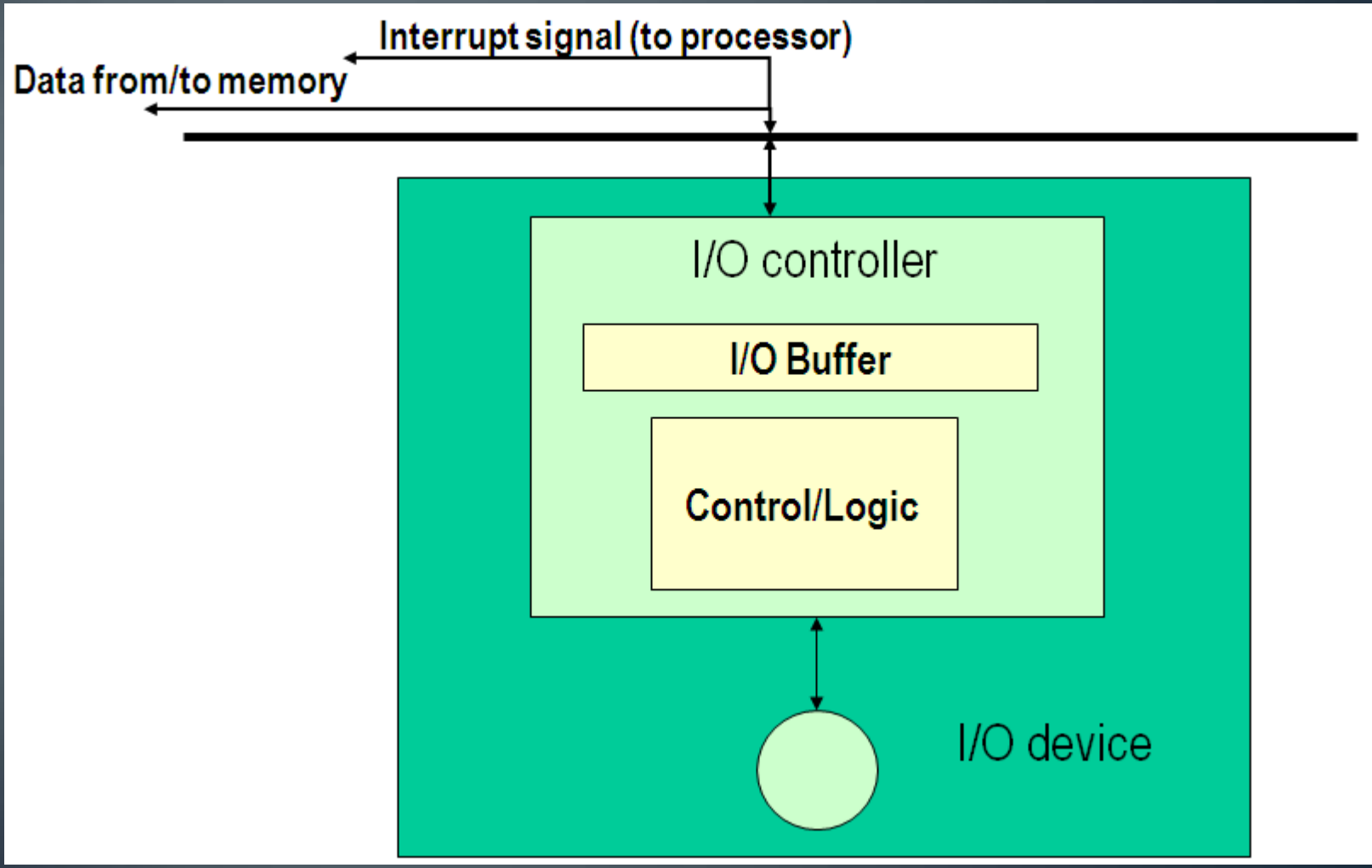
Input / Output Subsystem

- Handles devices that allow the computer system to:
  - Communicate and interact with the outside world
    - Screen, keyboard, printer, ...
  - Store information (mass-storage)
    - Hard-drives, floppies, CD, tapes, …
- Mass-Storage Device Access Methods:
  - Direct Access Storage Devices (DASDs)
    - Hard-drives, floppy-disks, CD-ROMs, ...
  - Sequential Access Storage Devices (SASDs)
    - Tapes (for example, used as backup devices)

I/O Controllers

- Speed of I/O devices is slow compared to RAM
  - RAM ~ 50 nsec.
  - Hard-Drive ~ 10msec. = (10,000,000 nsec)
- Solution:
  - I/O Controller, a special purpose processor:
    - Has a small memory buffer, and a control logic to control I/O device (e.g. move disk arm).
    - Sends an interrupt signal to CPU when done read/write.
  - Data transferred between RAM and memory buffer.
  - Processor free to do something else while I/O controller reads/writes data from/to device into I/O buffer.

Structure of the I/O Subsystem

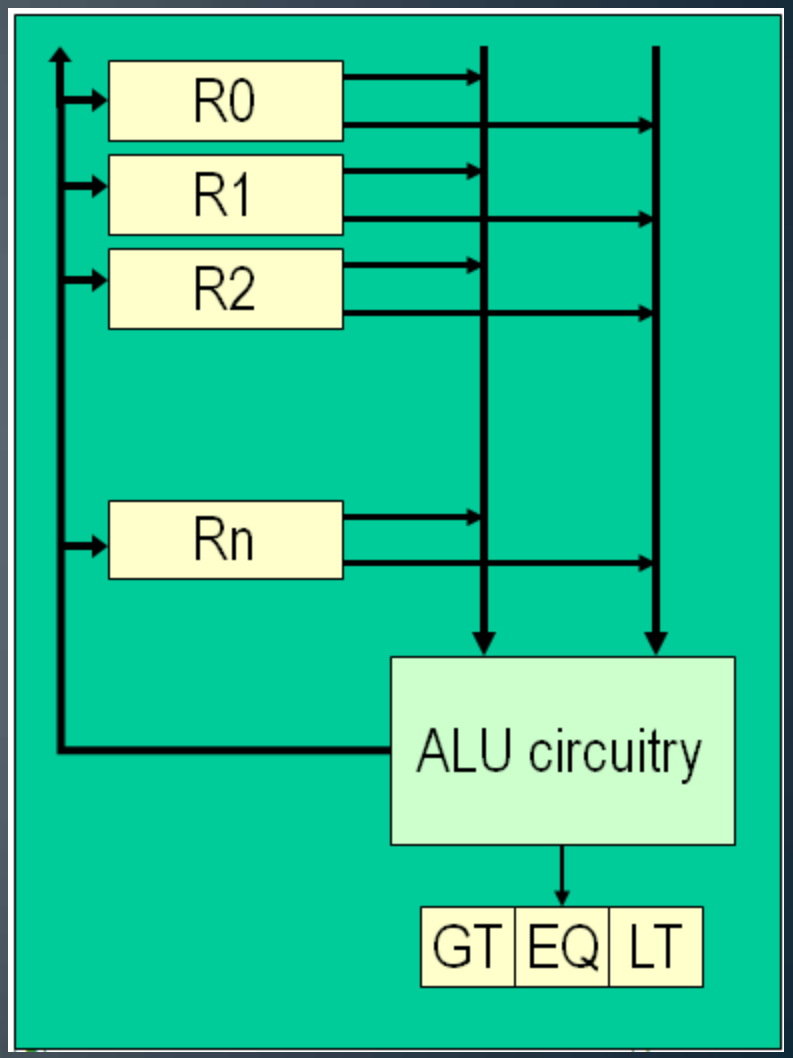


The ALU Subsystem

- The ALU (Arithmetic/Logic Unit) performs
  - mathematical operations (+, -, x, /, …)
  - logic operations (=, <, >, and, or, not, ...)
- In today's computers integrated into the CPU
- Consists of:
  - Circuits to do the arithmetic/logic operations.
  - Registers (fast storage units) to store intermediate computational results.
  - Bus that connects the two.

Structure of the ALU

- Registers:
  - Very fast local memory cells, that store operands of operations and intermediate results.
  - CCR (condition code register), a special purpose register that stores the result of <, = , > operations
- ALU circuitry:
  - Contains an array of circuits to do mathematical/logic operations.
- Bus:
  - Data path interconnecting the registers to the ALU circuitry.



The Control Unit

- Program is stored in memory
  - as machine language instructions, in binary
- The task of the control unit is to execute programs by repeatedly:
  - Fetch from memory the next instruction to be executed.
  - Decode it, that is, determine what is to be done.
  - Execute it by issuing the appropriate signals to the ALU, memory, and I/O subsystems.
  - Continues until the HALT instruction

Machine Language Instructions

- A machine language instruction consists of:
  - Operation code, telling which operation to perform
  - Address field(s), telling the memory addresses of the values on which the operation works.
- Example: ADD X, Y (Add content of memory locations X and Y, and store back in memory location Y).
- Assume: opcode for ADD is 9, and addresses X=99, Y=100

Opcode (8 bits)	Address 1 (16 bits)	Address 2 (16 bits)
00001001	0000000001100011	0000000001100100

Instruction Set Design

- Two different approaches:
  - Reduced Instruction Set Computers (RISC)
    - Instruction set as small and simple as possible.
    - Minimizes amount of circuitry --> faster computers
  - Complex Instruction Set Computers (CISC)
    - More instructions, many very complex
    - Each instruction can do more work, but require more circuitry.

Typical Machine Instructions

- Notation:
  - We use X, Y, Z to denote RAM cells
  - Assume only one register R (for simplicity)
  - Use English-like descriptions (should be binary)
- Data Transfer Instructions
  - LOAD X — Load content of memory location X to R
  - STORE X — Load content of R to memory location X
  - MOVE X, Y — Copy content of memory location X to loc. Y (not absolutely necessary)

Machine Instructions (cont.)

- Arithmetic
  - ADD X, Y, Z CON(Z) = CON(X) + CON(Y)
  - ADD X, Y CON(Y) = CON(X) + CON(Y)
  - ADD X R = CON(X) + R
  - similar instructions for other operators, e.g. SUBTR,OR, ...
- Compare
  - COMPARE X, Y
    Compare the content of memory cell X to the content of memory cell Y and set the condition codes (CCR) accordingly.
  - E.g. If CON(X) = R then set EQ=1, GT=0, LT=0

Machine Instructions (cont.)

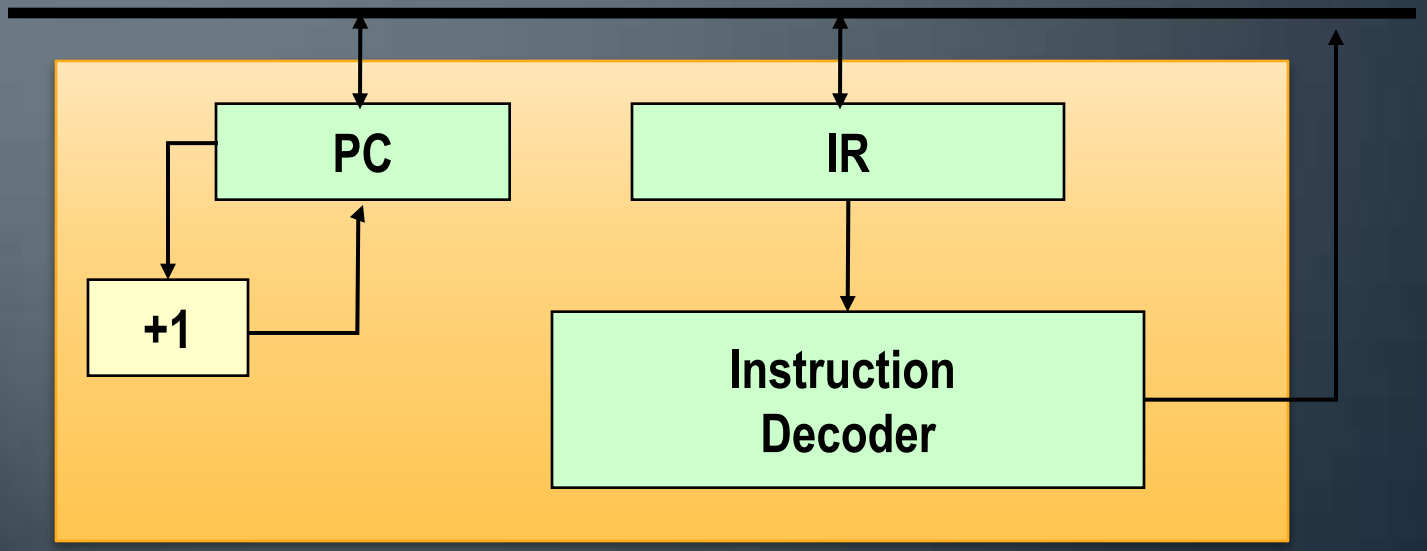
- Branch
  - JUMP X — Load next instruction from memory loc. X
  - JUMPGT X — Load next instruction from memory loc. X only if GT flag in CCR is set, otherwise load statement from next sequence loc. as usual.
    - JUMPEQ, JUMPLT, JUMPGE, JUMPLE,JUMPNEQ
- Control
  - HALT — Stop program execution.

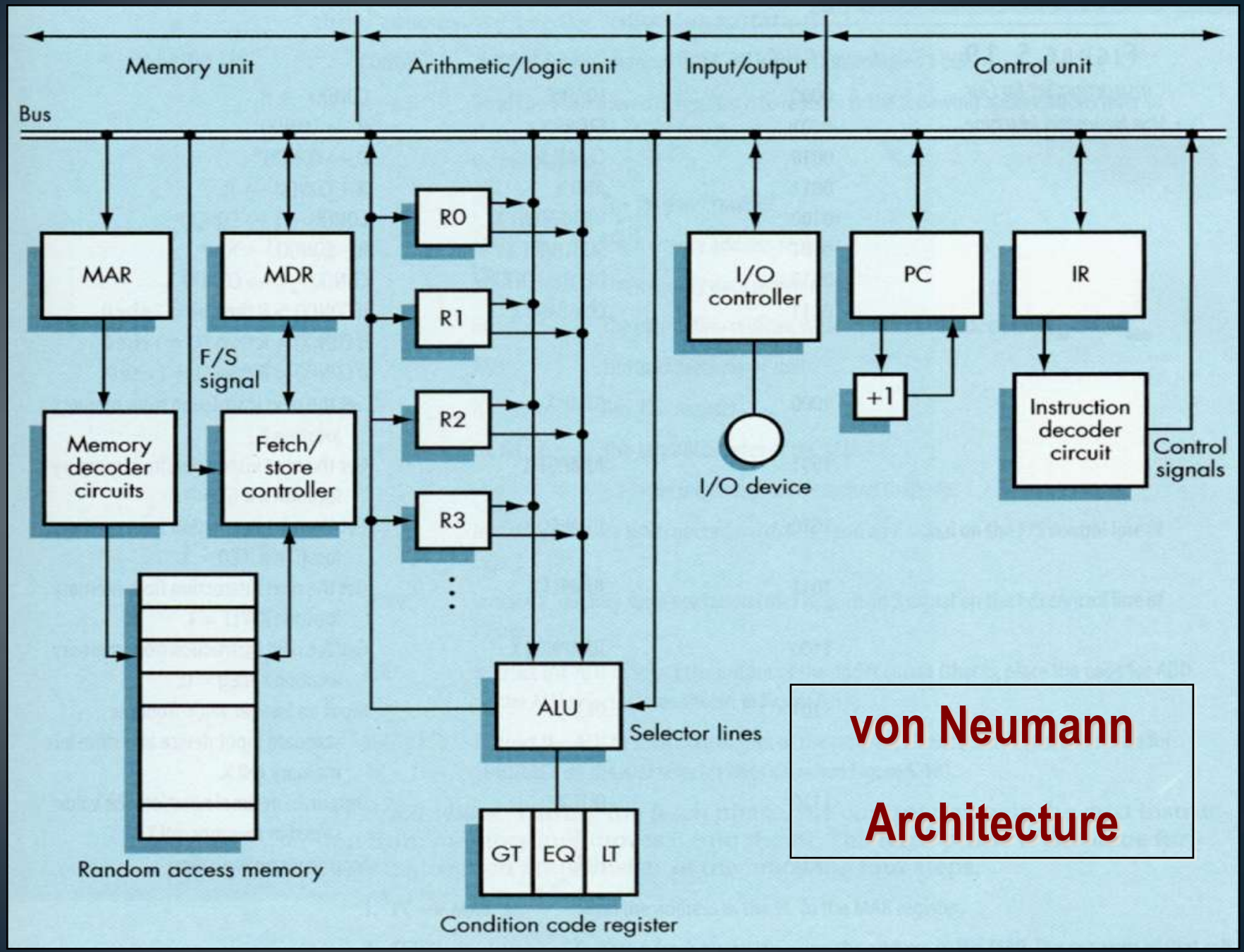
Example

- Pseudo-code: Set A to B + C
- Assuming variable:
  - A stored in memory cell 100, B stored in memory cell 150, C stored in memory cell 151
- Machine language (really in binary)
  - LOAD 150
  - ADD 151
  - STORE 100
  - or
  - (ADD 150, 151, 100)

Structure of the Control Unit

- PC (Program Counter):
  - stores the address of next instruction to fetch
- IR (Instruction Register):
  - stores the instruction fetched from memory
- Instruction Decoder:
  - Decodes instruction and activates necessary circuitry




Memory unit
Arithmetic/logic unit
Input/output
Control unit
Bus
MAR
MDR
R0
R1
R2
R3
I/O controller
PC
IR
F/S signal
Memory decoder circuits
Fetch/ store controller
+1
Instruction decoder circuit
Control signals
I/O device
ALU
Selector lines
GT
EQ
LT
Random access memory
Condition code register
von Neumann Architecture

How does this all work together?

- Program Execution:
  - PC is set to the address where the first program instruction is stored in memory.
  - Repeat until HALT instruction or fatal error

Fetch instruction

Decode instruction

Execute instruction

End of loop

Program Execution (cont.)

- Fetch phase
  - PC --> MAR (put address in PC into MAR)
  - Fetch signal (signal memory to fetch value into MDR)
  - MDR --> IR (move value to Instruction Register)
  - PC + 1 --> PC (Increase address in program counter)
- Decode Phase
  - IR -> Instruction decoder (decode instruction in IR)
  - Instruction decoder will then generate the signals to activate the circuitry to carry out the instruction

Program Execution (cont.)

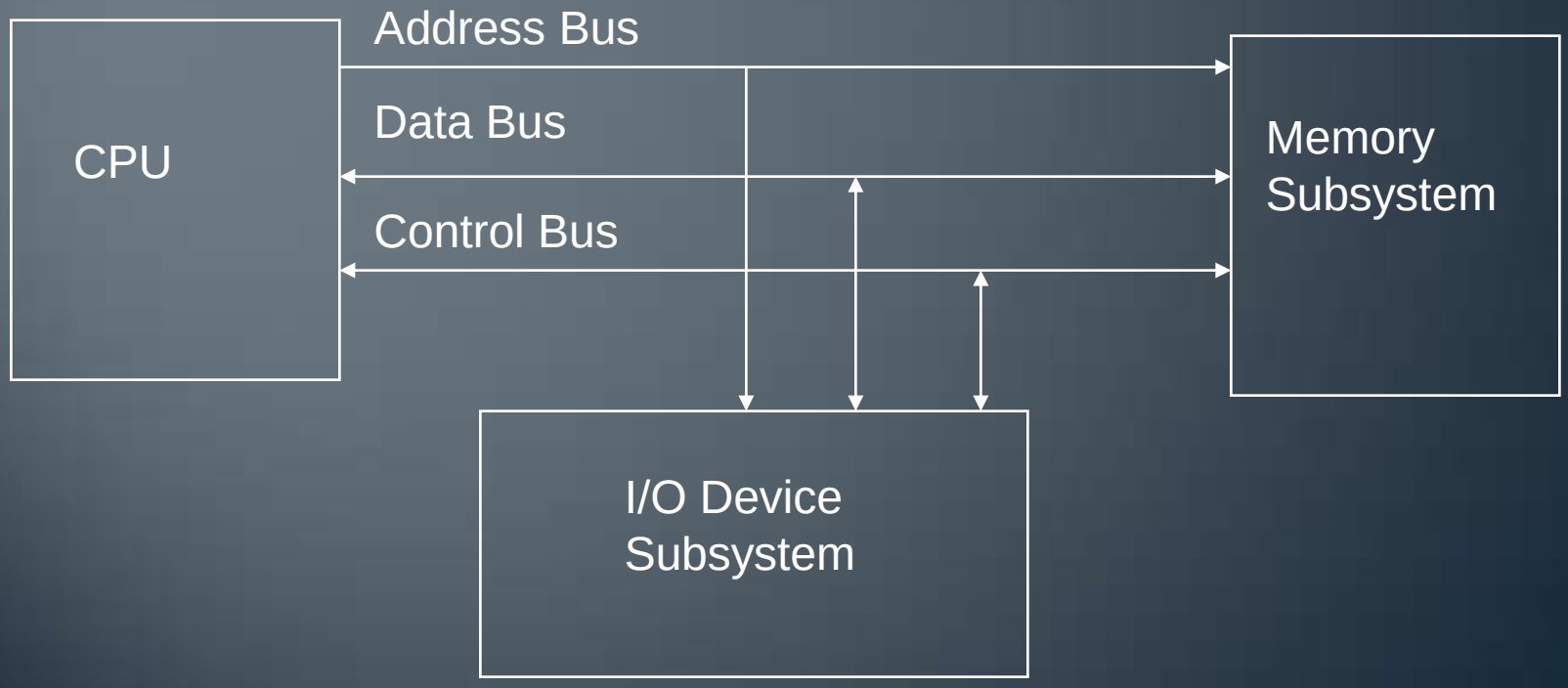
- Execute Phase
  - Differs from one instruction to the next.
- Example:
  - LOAD X (load value in addr. X into register)
    - IR_address -> MAR
    - Fetch signal
    - MDR --> R
  - ADD X
    - left as an exercise

Instruction Set for Our Von Neumann Machine

Opcode	Operation	Meaning
0000	LOAD X	CON(X) --> R
0001	STORE X	R --> CON(X)
0010	CLEAR X	0 --> CON(X)
0011	ADD X	R + CON(X) --> R
0100	INCREMENT X	CON(X) + 1 --> CON(X)
0101	SUBTRACT X	R - CON(X) --> R
0101	DECREMENT X	CON(X) - 1 --> CON(X)
0111	COMPARE X	If CON(X) > R then GT = 1 else 0 If CON(X) = R then EQ = 1 else 0 If CON(X) < R then LT = 1 else 0
1000	JUMP X	Get next instruction from memory location X
1001	JUMPGT X	Get next instruction from memory loc. X if GT=1
...	JUMPxx X	xx = LT / EQ / NEQ
1101	IN X	Input an integer value and store in X
1110	OUT X	Output, in decimal notation, content of mem. loc. X
1111	HALT	Stop program execution

Fundamental Components of Computer

- The CPU (ALU, Control Unit, Registers)
- The Memory Subsystem (Stored Data)
- The I/O subsystem (I/O devices)



Each of these Components are connected through Buses.

- BUS - Physically a set of wires. The components of the Computer are connected to these buses.
- Address Bus
- Data Bus
- Control Bus

Address Bus

- Used to specify the address of the memory location to access.
- Each I/O devices has a unique address.
  (monitor, mouse, cd-rom)
- CPU reads data or instructions from other locations by specifying the address of its location.
- CPU always outputs to the address bus and never reads from it.

Data Bus

- Actual data is transferred via the data bus.
- When the cpu sends an address to memory, the memory will send data via the data bus in return to the cpu.

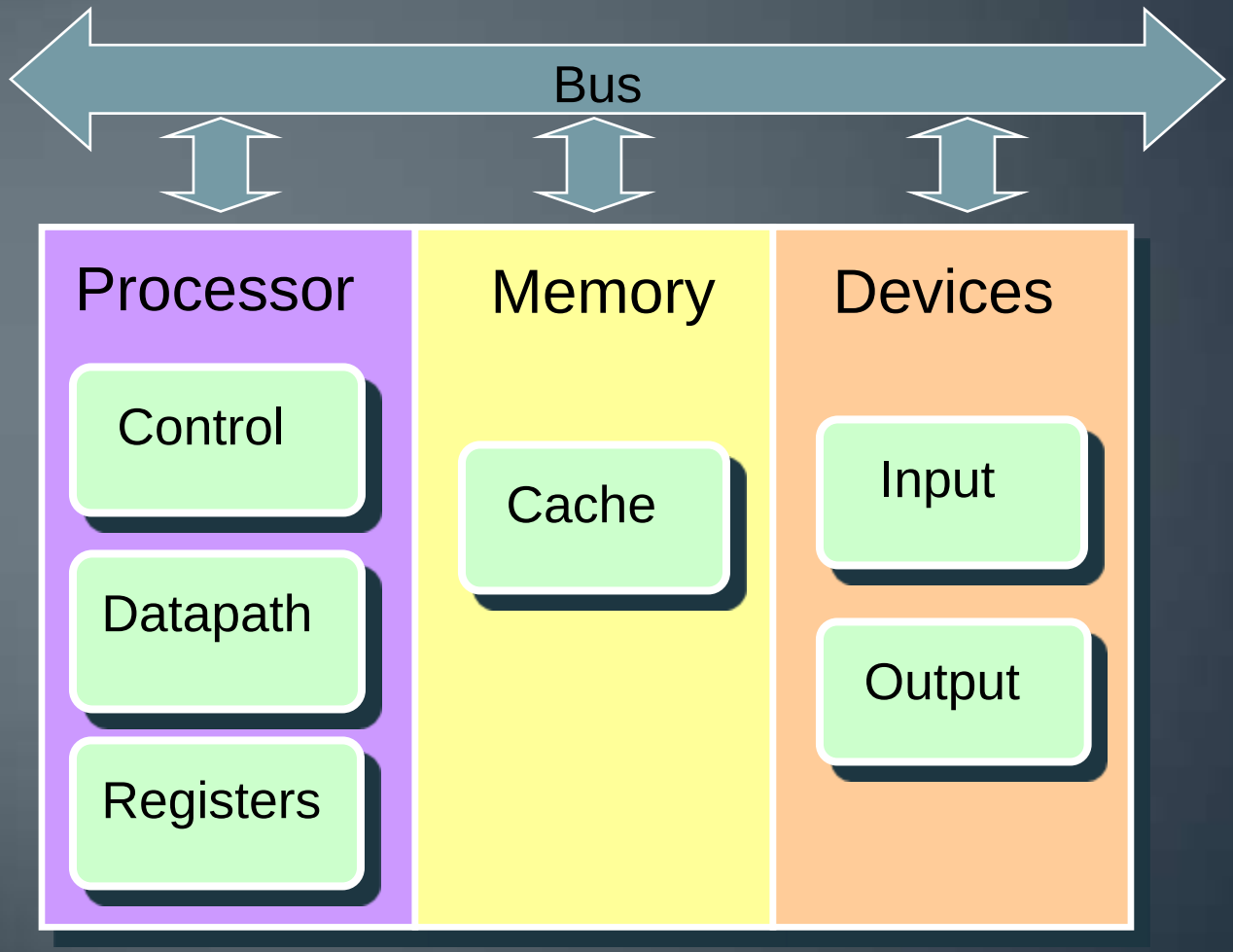
Control Bus

- Collection of individual control signals.
- Whether the cpu will read or write data.
- CPU is accessing memory or an I/O device
- Memory or I/O is ready to transfer data

I/O Bus or Local Bus

- In today's computers the the I/O controller will have an extra bus called the I/O bus.
- The I/O bus will be used to access all other I/O devices connected to the system.
- Example: PCI bus

Bus Organisation

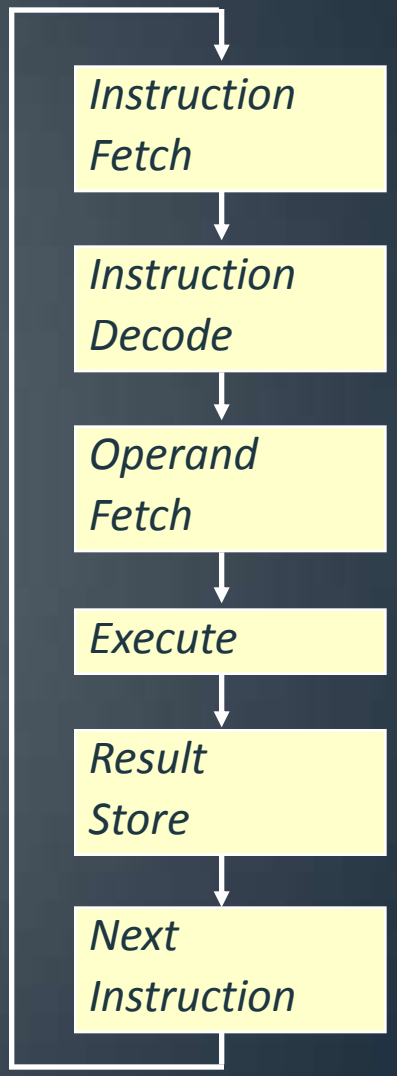


Fundamental Concepts

- Processor (CPU): the active part of the computer, which does all the work (data manipulation and decision-making).
- Datapath: portion of the processor which contains hardware necessary to perform all operations required by the computer (the brawn).
- Control: portion of the processor (also in hardware) which tells the datapath what needs to be done (the brain).

Fundamental Concepts (2)

- Instruction execution cycle:
  fetch, decode, execute.
  - Fetch: fetch next instruction (using PC) from memory into IR.
  - Decode: decode the instruction.
  - Execute: execute instruction.

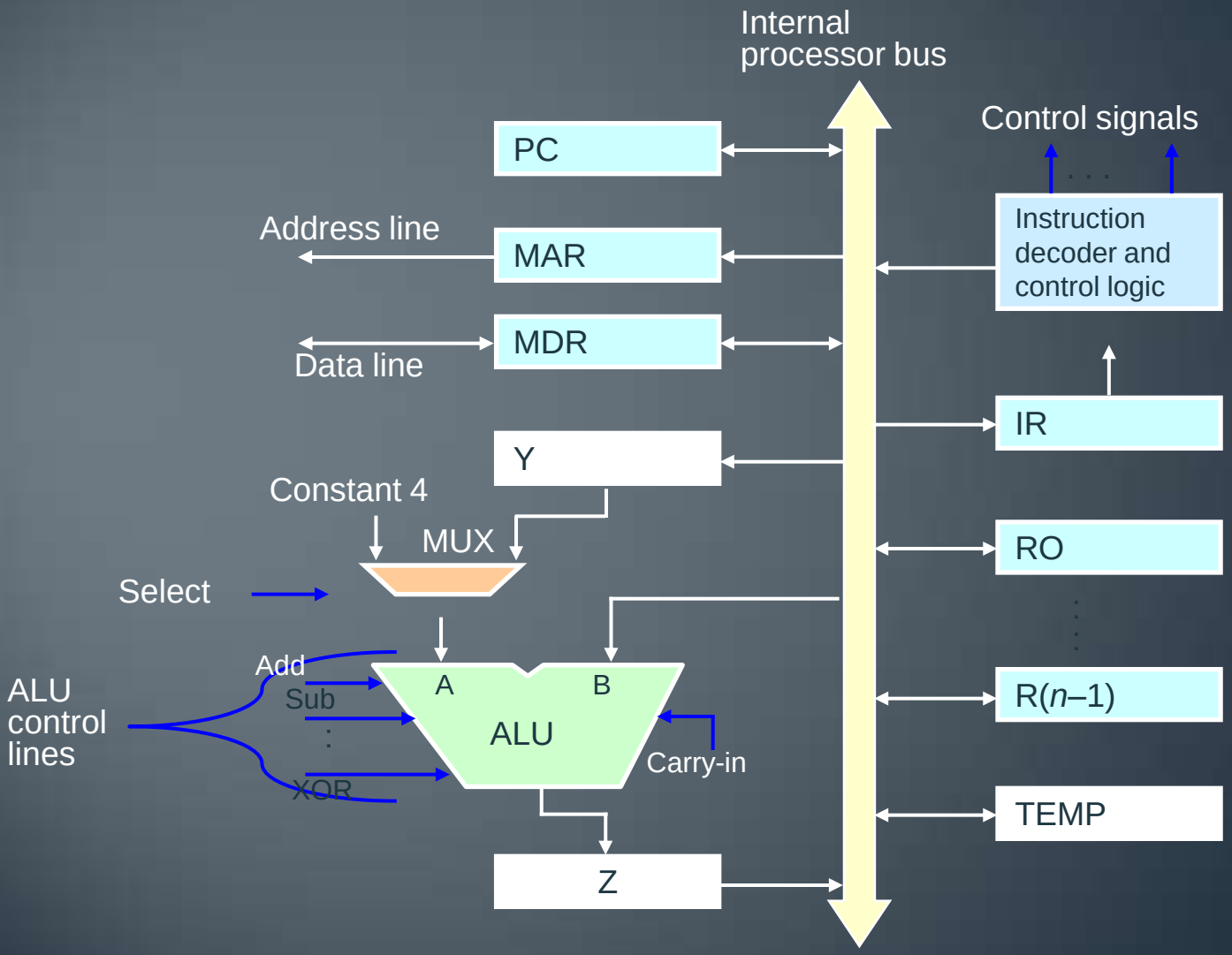


Fundamental Concepts (3)

- Fetch: Fetch next instruction into IR (Instruction Register).
  - Assume each word is 4 bytes and each instruction is stored in a word, and that the memory is byte addressable.
  - PC (Program Counter) contains address of next instruction.

$$IR \leftarrow [[PC]]$$
$$PC \leftarrow [PC] + 4$$

Single Bus Organization

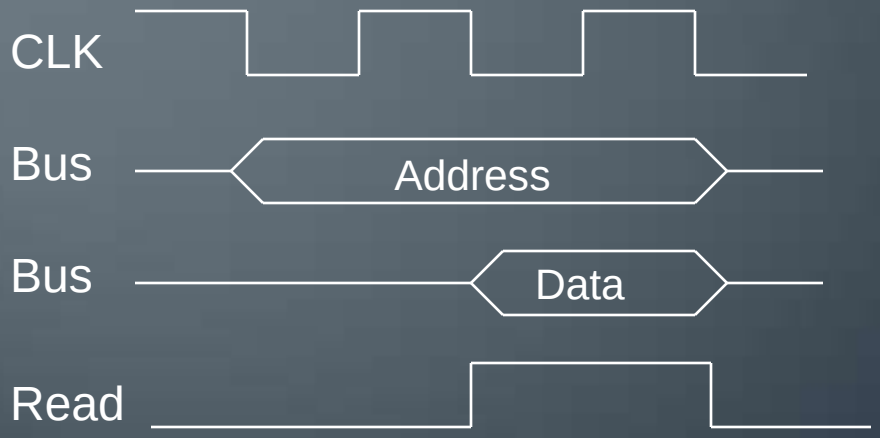


Instruction Cycles

- Procedure the CPU goes through to process an instruction.
- 1. Fetch - get instruction
- 2. Decode - interperate the instruction
- 3. Execute - run the instruction.

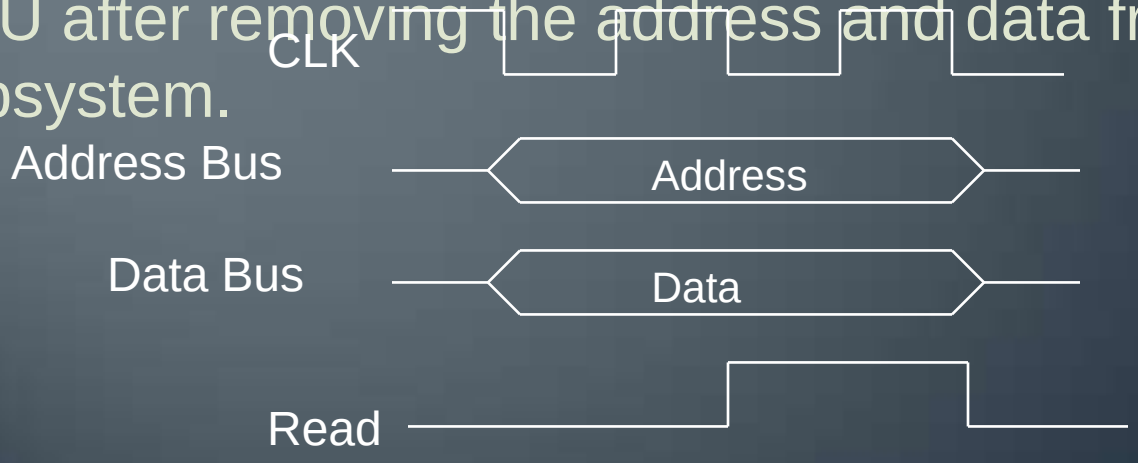
Timing Diagram: Memory Read

- Address is placed at beginning of clock
- after one clock cycle the CPU asserts the read.
- Causes the memory to place its data onto the data bus.
- CLK : System Clock used to synchronize



Timing Diagram : Memory Write

- CPU places the Address and data on the first clock cycle.
- At the start of the second clock the CPU will assert the write control signal.
- This will then start memory to store data.
- After some time the write is then deasserted by the CPU after removing the address and data from the subsystem.



I/O read and Write Cycles

- The I/O read and Write cycles are similar to the memory read and write.
- Memory mapped I/O : Same sequences as input output to read and write.
- The processor treats an I/O port as a memory location.
- This results in the same treatment as a memory access.

Technology Trends

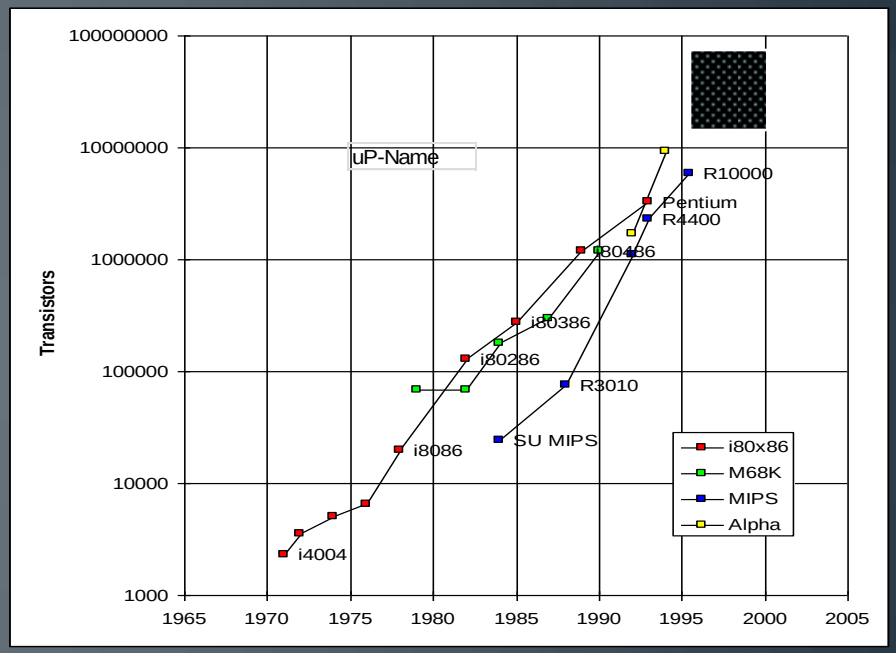
- **Processor**
  - logic capacity: about 30% per year
  - clock rate: about 20% per year
- **Memory**
  - DRAM capacity: about 60% per year (4x every 3 years)
  - Memory speed: about 10% per year
  - Cost per bit: improves about 25% per year
- **Disk**
  - capacity: about 60% per year
  - Total use of data: 100% per 9 months!
- **Network Bandwidth**
  - Bandwidth increasing more than 100% per year!

Technology Trends

DRAM chip capacity

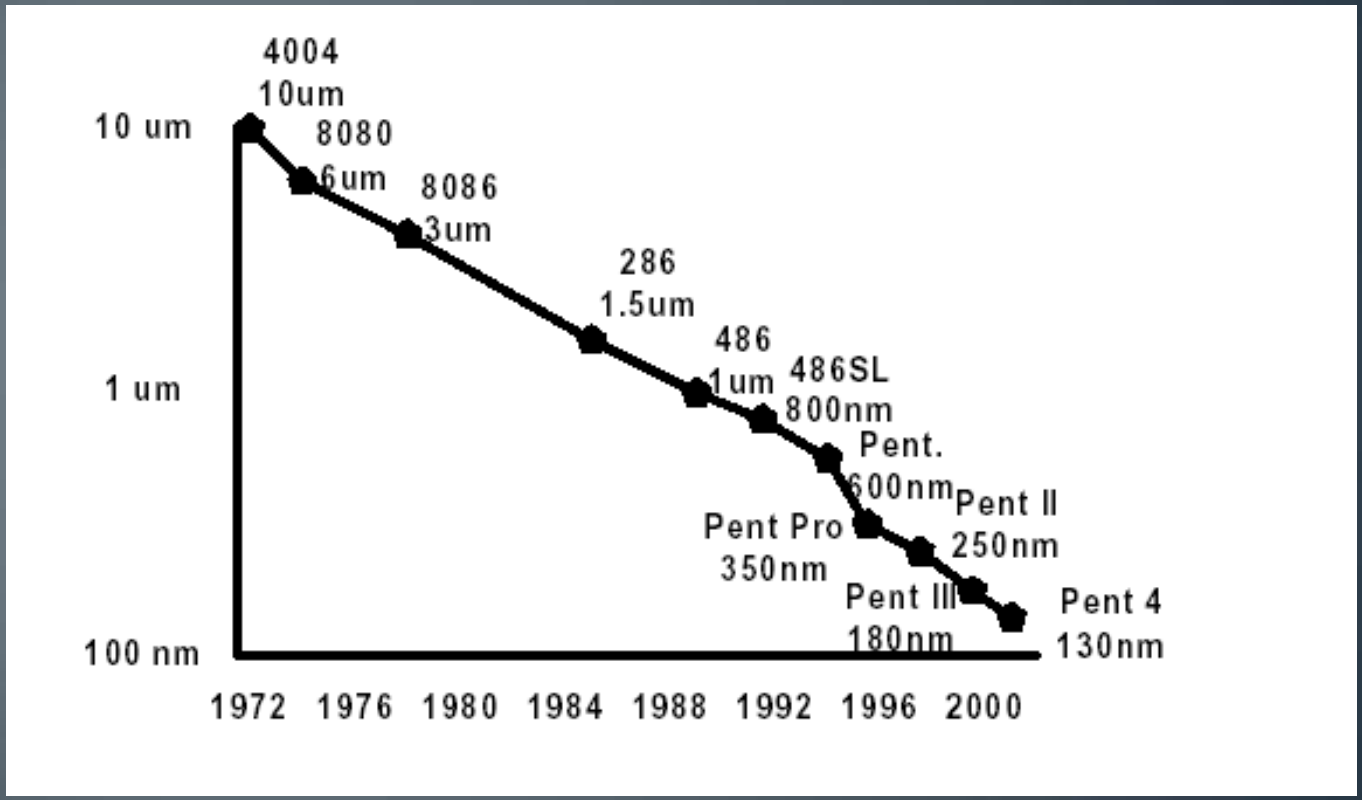
DRAM	
Year	Size
1980	64 Kb
1983	256 Kb
1986	1 Mb
1989	4 Mb
1992	16 Mb
1996	64 Mb
1999	256 Mb
2002	1 Gb

Microprocessor Logic Density



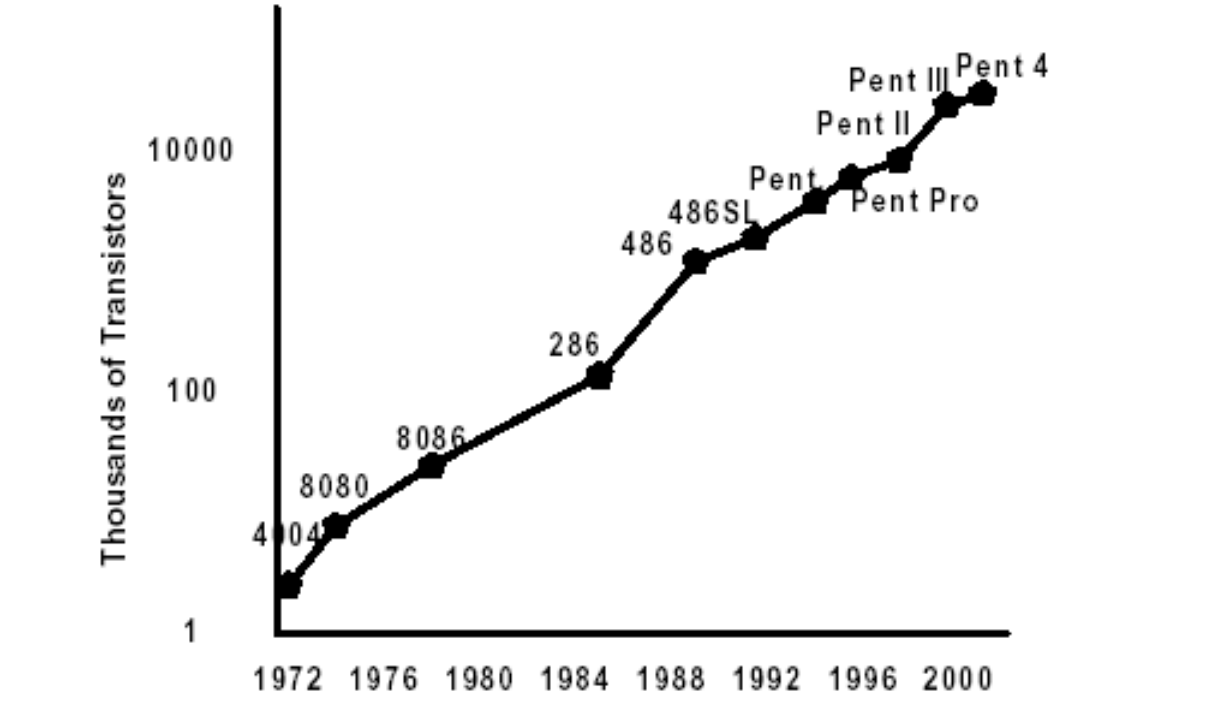
- In ~1985 the single-chip processor (32-bit) and the single-board computer emerged
- In the 2002+ timeframe, these may well look like mainframes compared single-chip computer (maybe 2 chips)

Technology Trends



Smaller feature sizes – higher speed, density

Technology Trends



Number of transistors doubles every 18 months

(amended to 24 months)

References

- Computer Organization and Architecture, Designing for performance by *William Stallings,* Prentice Hall of India.
- Modern Computer Architecture, by *Morris Mano,* Prentice Hall of India.
- Computer Architecture and Organization by *John P. Hayes,* McGraw Hill Publishing Company.
- Computer Organization by *V. Carl Hamacher, Zvonko G. Vranesic, Safwat G. Zaky,* McGraw Hill Publishing Company.