Computer Architecture and Organization (MCSE503L)

School of Computer Science and Engineering (SCOPE)
Vellore Institute of Technology (VIT) ,Chennai

Books to Refer

- **Text Books (T)**
    1. William Stallings, Computer Organization and Architecture: Designing for Performance, Pearson, 2022, 11th Edition, Pearson
    2. Gerassimos Barlas, Multicore and GPU Programming: An Integrated Approach, 2022, 2nd edition, Morgan Kaufmann.
- **Reference Books (R)**
    1. J.L. Hennessy and D.A. Patterson. Computer Architecture: A Quantitative Approach. 5th Edition, 2012, Morgan Kauffmann Publishers.
    2. Shameem Akhter, Jason Roberts, Multi-core Programming: Increasing Performance Through Software Multi-threading, 2010, Intel Press, BPB Publications

Computer organization/Computer architecture

- Computer organization
  - Computer organization refers to the **operational units** and their **interconnections** that realize the architectural specifications.
  - physical aspects of computer systems.
  - E.g., circuit design, control signals, memory types.

- Computer architecture
  - Logical aspects of system as seen by the programmer.
  - E.g., instruction sets, instruction formats, data types, addressing modes.
  - Computer architecture refers to those attributes of a system visible to a programmer or those attributes that have a direct impact on the logical execution of a program.
  - A term that is often used interchangeably with computer architecture is **Instruction Set Architecture (ISA)**.

Architectural Design Issue or Organizational Issue?

- For example, it is an architectural design issue whether a computer will have a multiply instruction ?
- It is an organizational issue whether that instruction will be implemented by a **special multiply unit** or by a mechanism that makes **repeated use of the add unit** of the system.
- The organizational decision may be based on the **anticipated frequency of use of the multiply instruction**, the **relative speed** of the two approaches, and the **cost and physical size of a special multiply unit**.

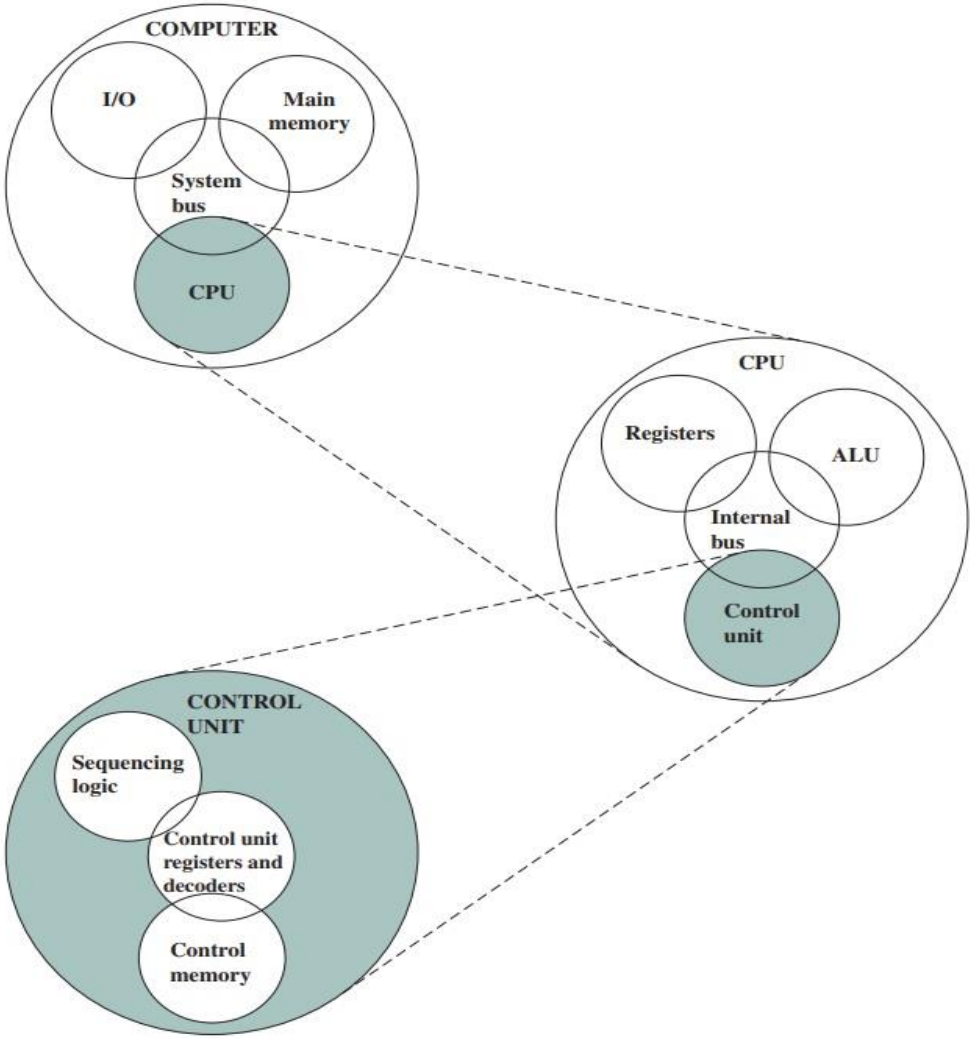
Computer Architecture Vs Computer Organization

- Historically, and still today, the distinction between architecture and organization has been an important one.
- Many computer manufacturers offer a **family of computer models**, all with the **same architecture but with differences in organization**.
- Consequently, the **different models** in the family have **different price and performance characteristics**.
- Furthermore, a **particular architecture may span many years** and encompass a number of different computer models, its **organization changing with changing technology**.

What is a computer?

- a computer is a sophisticated electronic calculating machine that:
  - Accepts input information,
  - Processes the information according to a list of internally stored instructions and
  - Produces the resulting output information.
- Functions performed by a computer are:
  - Accepting information to be processed as input.
  - Storing a list of instructions to process the information.
  - Processing the information according to the list of instructions.
  - Providing the results of the processing as output.
- What are the functional units of a computer?

Simple single-processor computer : Top Level Structure



Simple single-processor computer

- Figure provides a **hierarchical** view of the internal structure of a traditional **single-processor** computer.
- There are **four** main structural components.
    - **Central processing unit (CPU) :** Controls the operation of the computer and performs its data processing functions; often simply referred to as processor.
    - **Main memory :** Stores data
    - **I/O:** Moves data between the computer and its external environment.
    - **System interconnection:** Some mechanism that provides for communication among CPU, main memory, and I/O.
- A common example of system interconnection is by means of a **system bus**, consisting of a number of conducting wires to which all the other components attach.

CPU

- CPU major structural components are as follows,
  - **Control unit:** Controls the operation of the CPU and hence the computer.
  - **Arithmetic and logic unit (ALU):** Performs the computer's data processing functions.
  - **Registers:** Provides storage internal to the CPU.
  - **CPU interconnection:** Some mechanism that provides for communication among the control unit, ALU, and registers.

Multicore computer structure

- The Contemporary computers generally have **multiple processors.**
- When these processors all reside on a **single chip**, the term **multicore** computer is used, and
- Each processing unit (consisting of a control unit, ALU, registers, and perhaps cache) is called a **core**.

Multicore computer structure

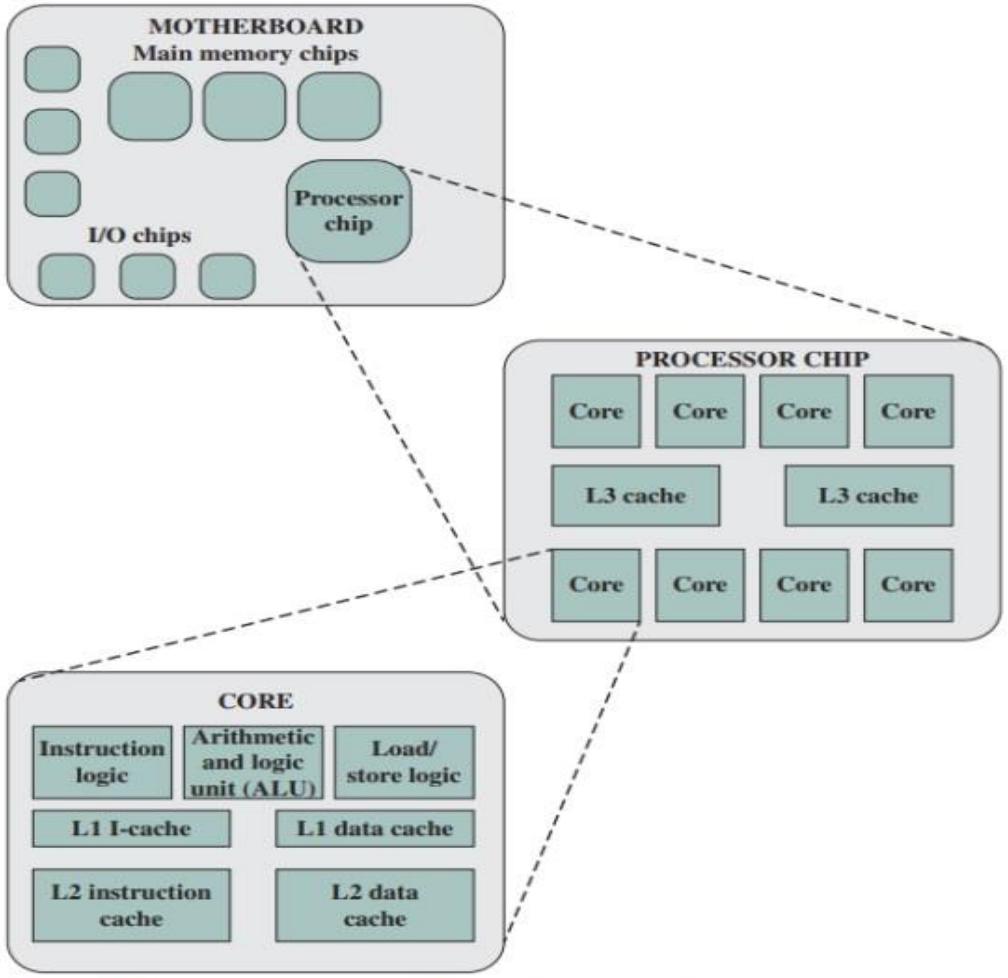
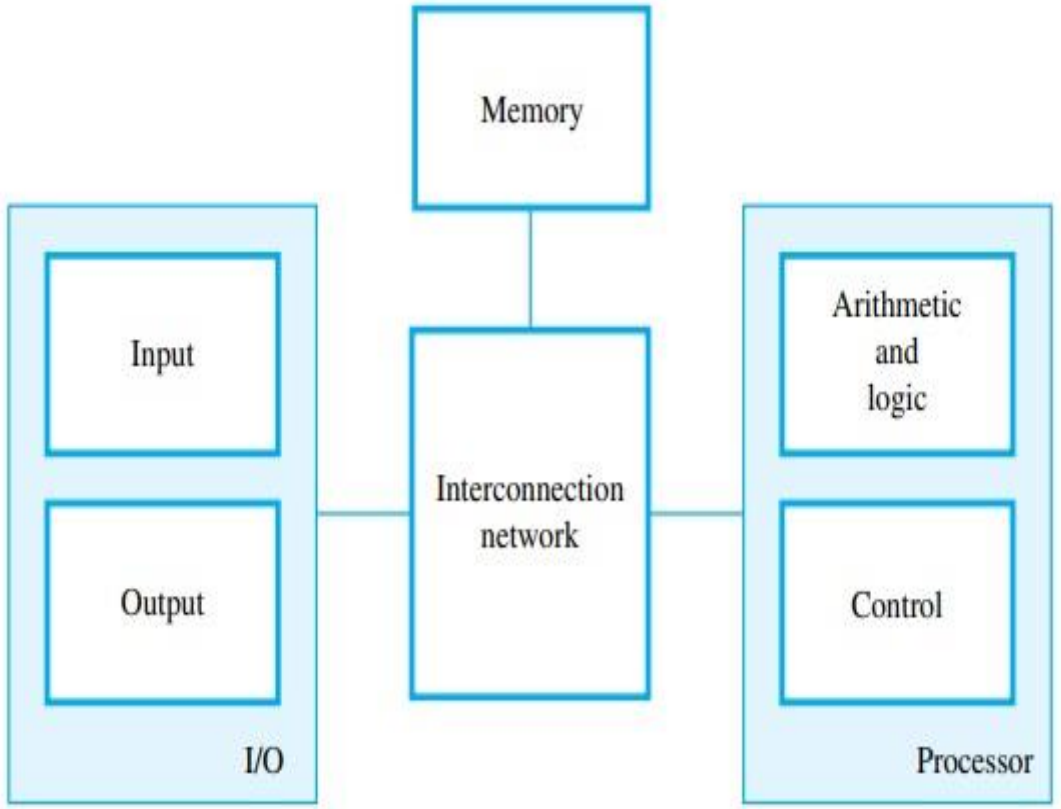


Figure 1.2 Simplified View of Major Elements of a Multicore Computer

Functional Components of a Computer



Instructions

- Instructions specify commands to:
  - Transfer information within a computer (e.g., from memory to ALU)
  - Transfer of information between the computer and I/O devices (e.g., from keyboard to computer, or computer to printer)
  - Perform arithmetic and logic operations (e.g., Add two numbers, Perform a logical AND).
- A sequence of instructions to perform a task is called a program, which is stored in the memory.
- Processor fetches instructions that make up a program from the memory and performs the operations stated in those instructions.
- What do the instructions operate upon?

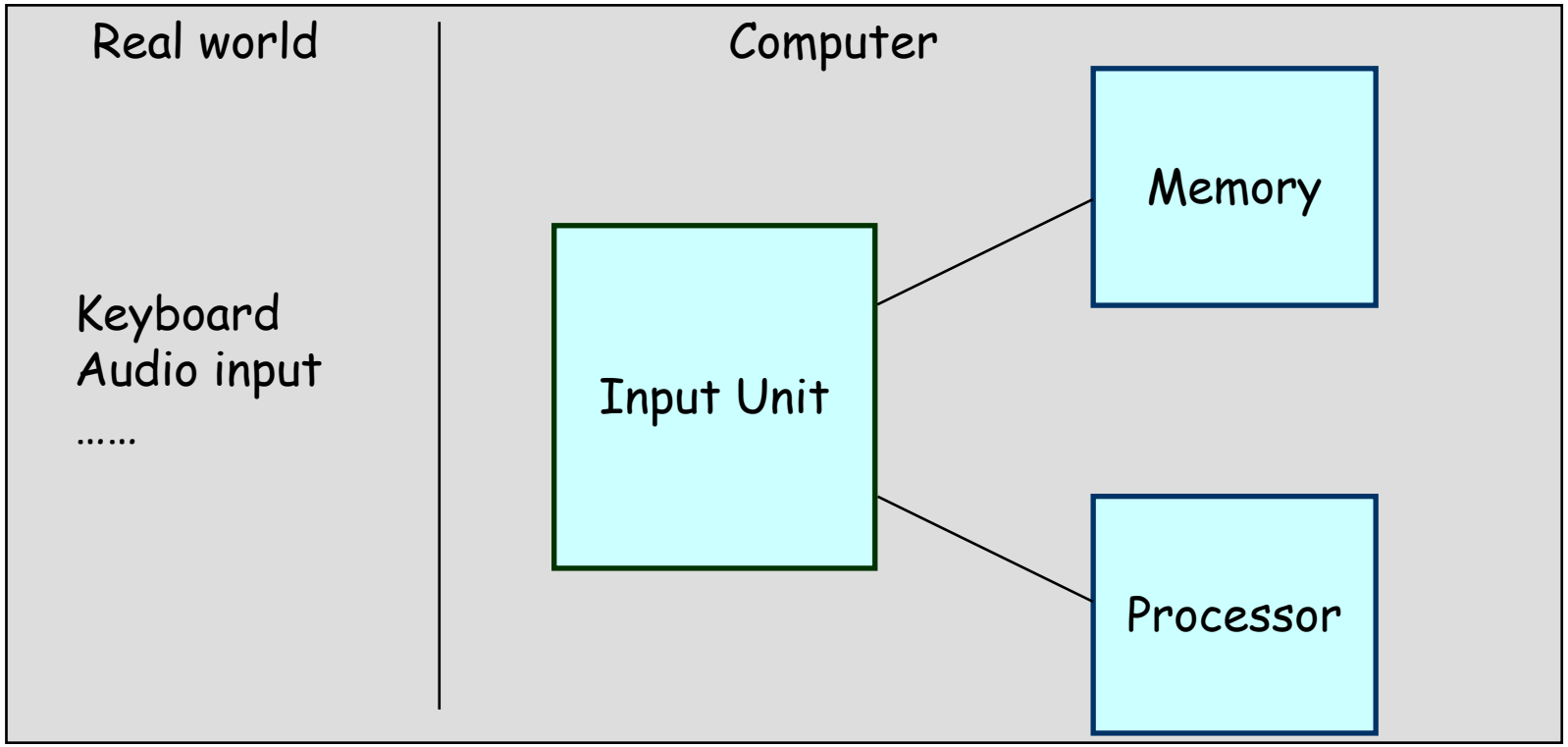
Data

- Data are the “operands” upon which instructions operate.
- Data could be:
  - Numbers,
  - Encoded characters.
- Data, in a broad sense means any digital information.
- Computers use data that is encoded as a string of binary digits called bits.

Input unit

Binary information must be presented to a computer in a specific format. This task is performed by the input unit:

- Interfaces with input devices.
- Accepts binary information from the input devices.
- Presents this binary information in a format expected by the computer.
- Transfers this information to the memory or processor.



Memory unit

- Memory unit stores instructions and data.
  - Recall, data is represented as a series of bits.
  - To store data, memory unit thus stores bits.
- Processor reads instructions and reads/writes data from/to the memory during the execution of a program.
  - In theory, instructions and data could be fetched one bit at a time.
  - In practice, a group of bits is fetched at a time.
  - Group of bits stored or retrieved at a time is termed as "word"
  - Number of bits in a word is termed as the "word length" of a computer.
- In order to read/write to and from memory, a processor should know where to look:
  - "Address" is associated with each word location.

Memory unit (contd..)

- Processor reads/writes to/from memory based on the memory address:
  - Access any word location in a short and fixed amount of time based on the address.
  - Random Access Memory (RAM) provides fixed access time independent of the location of the word.
  - Access time is known as "Memory Access Time".
- Memory and processor have to "communicate" with each other in order to read/write information.
  - In order to reduce "communication time", a small amount of RAM (known as Cache) is tightly coupled with the processor.
- Modern computers have three to four levels of RAM units with different speeds and sizes:
  - Fastest, smallest known as Cache
  - Slowest, largest known as Main memory.

Memory unit (contd..)

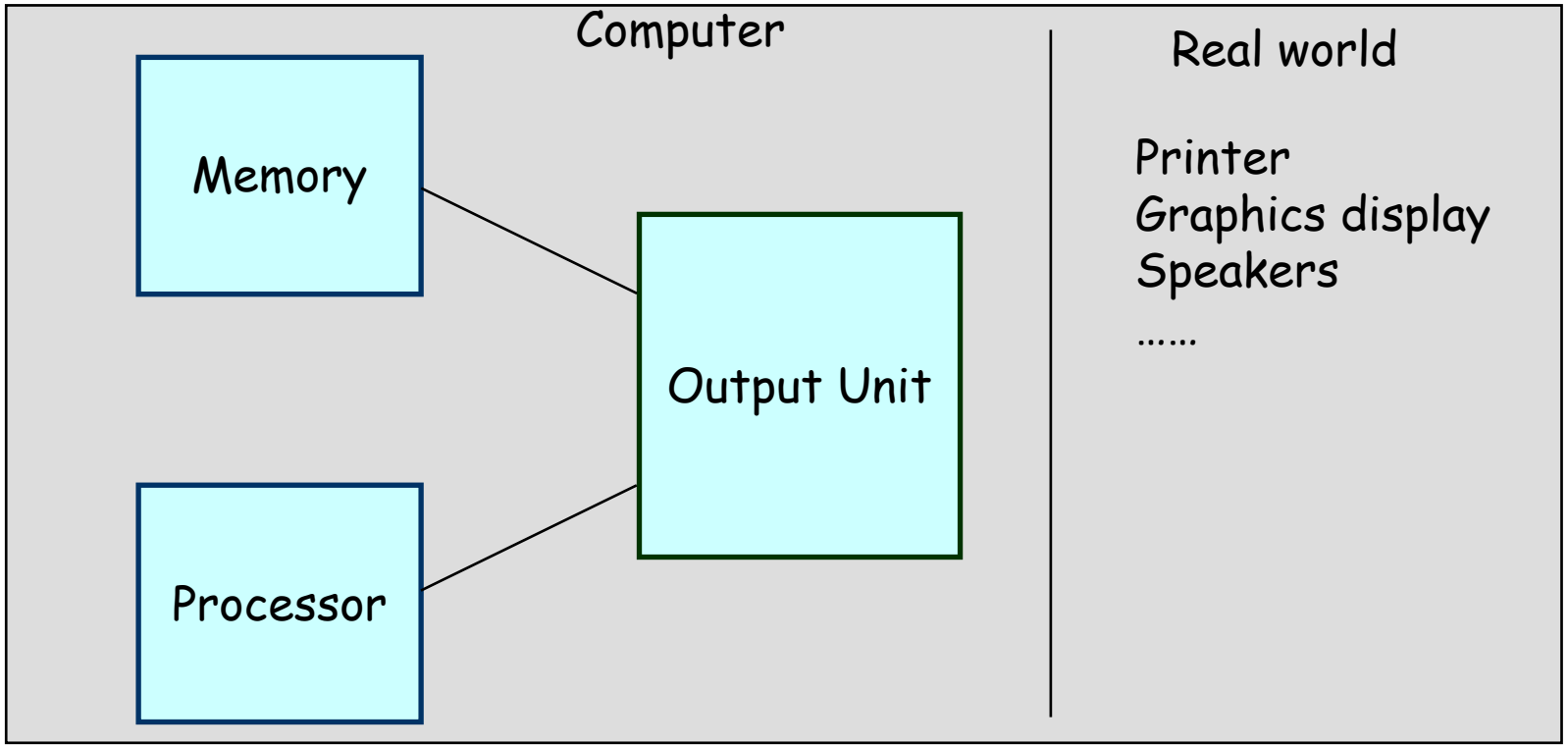
- Primary storage of the computer consists of RAM units.
  - Fastest, smallest unit is Cache.
  - Slowest, largest unit is Main Memory.
- Primary storage is insufficient to store large amounts of data and programs.
  - Primary storage can be added, but it is expensive.
- Store large amounts of data on secondary storage devices:
  - Magnetic disks and tapes,
  - Optical disks (CD-ROMS).
  - Access to the data stored in secondary storage in slower, but take advantage of the fact that some information may be accessed infrequently.
- Cost of a memory unit depends on its access time, lesser access time implies higher cost.

Arithmetic and logic unit (ALU)

- Operations are executed in the Arithmetic and Logic Unit (ALU).
  - Arithmetic operations such as addition, subtraction.
  - Logic operations such as comparison of numbers.
- In order to execute an instruction, operands need to be brought into the ALU from the memory.
  - Operands are stored in general purpose registers available in the ALU.
  - Access times of general purpose registers are faster than the cache.
- Results of the operations are stored back in the memory or retained in the processor for immediate use.

Output unit

- Computers represent information in a specific binary form. Output units:
  - Interface with output devices.
  - Accept processed results provided by the computer in specific binary form.
  - Convert the information in binary form to a form understood by an output device.



Control unit

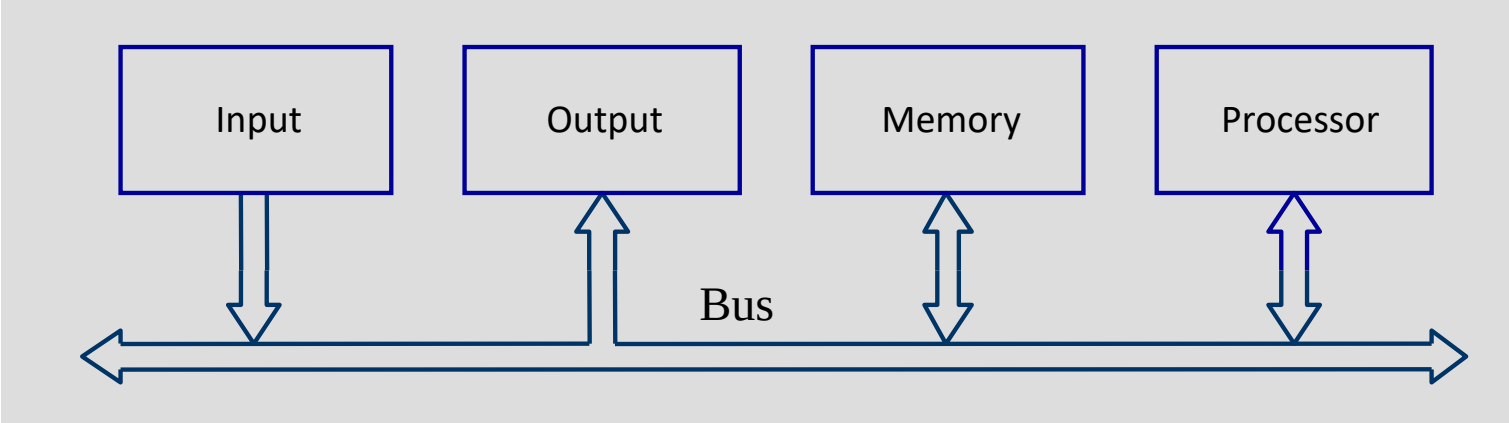
- Operations of Input unit, Memory, ALU and Output unit are coordinated by Control unit.
- Instructions control “what” operations take place (e.g. data transfer, processing).
- Control unit generates timing signals which determines “when” a particular operation takes place.

Control Unit Types

- Two distinct classes:
  - Non-programmable or *hard-wired*
  - Programmable or *microprogrammable*
- A hard-wired control units does not fetch or sequence instructions from a memory
- A programmable control unit has:
  - A program counter (PC) or other sequencing register with contents that points to the next instruction to be executed
  - An external ROM or RAM array for storing instructions and control information
  - Decision logic for determining the sequence of operations and logic to interpret the instructions

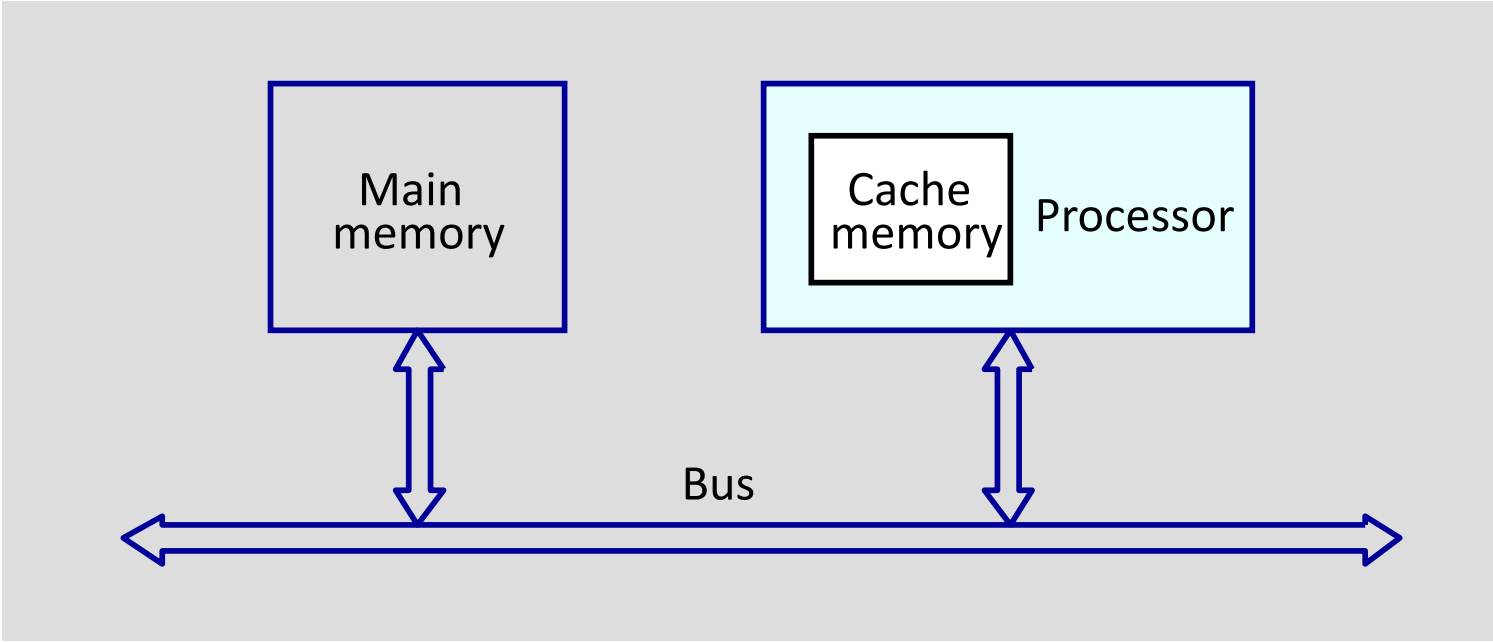
How are the functional units connected?

•For a computer to achieve its operation, the functional units need to communicate with each other.
•In order to communicate, they need to be connected.



•Functional units may be connected by a group of parallel wires.
•The group of parallel wires is called a ***bus***.
•Each wire in a bus can transfer one bit of information.
•The number of parallel wires in a bus is equal to the word length of a computer

Organization of cache and main memory



Registers

In addition to the ALU and the control circuitry, the processor contains number of registers used for several different purposes.

Instruction register (IR):

•It holds the instruction that is currently being executed.

•Its output is available to the control circuits which generates the Timing signals that control the various processing elements involved in executing the instruction.

Program counter (PC):

PC is another specialized register.

•It keeps track of the execution of a program. It contains the memory address of the next instructionto be fetched and executed.

•During the execution of an instruction, the contents of the PC updated to correspond to the address of the next instruction to executed.

•PC *points to the next instruction that is to be fetched from the memory.*

Registers

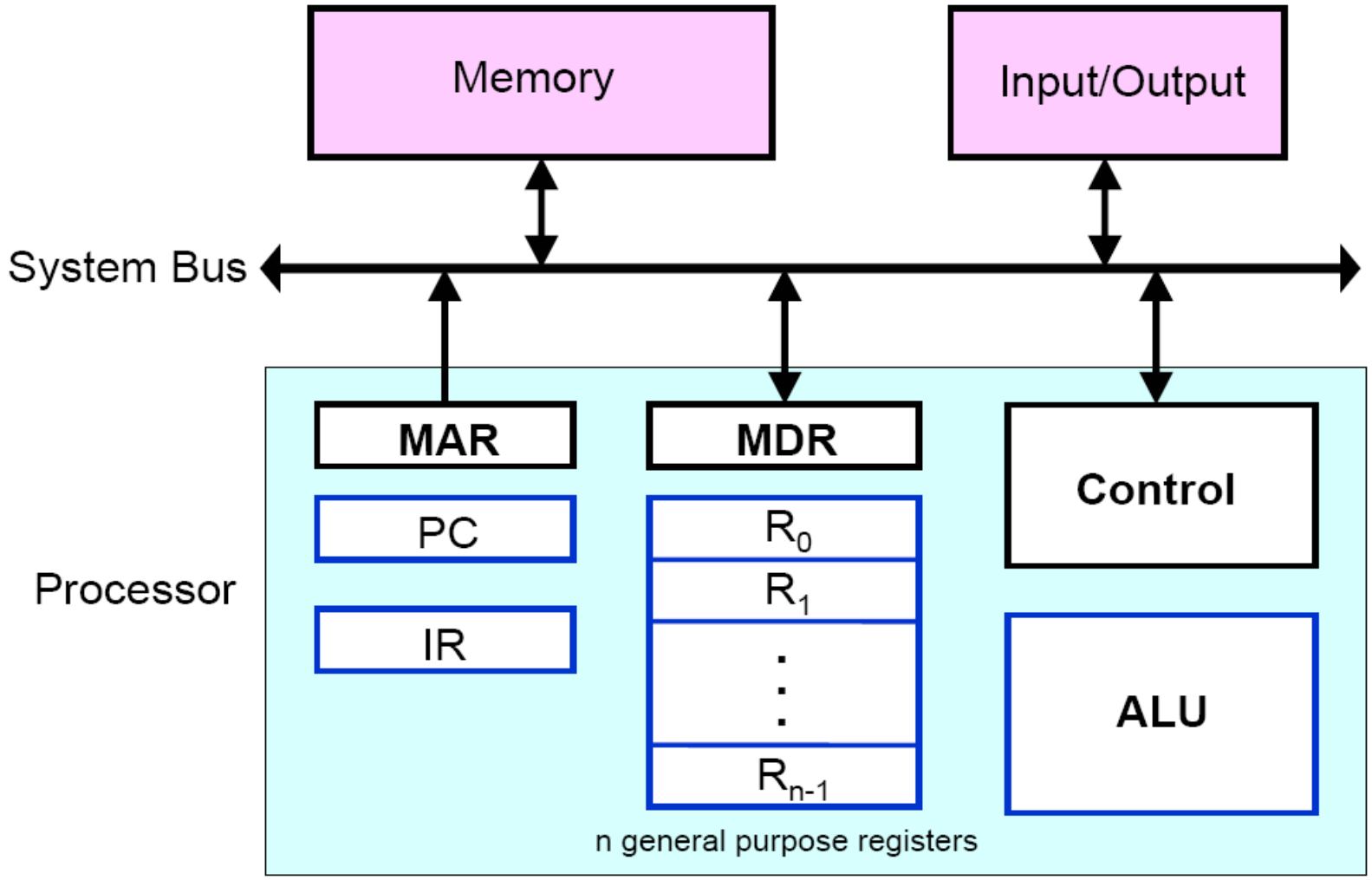
Two registers facilitate communication with the memory

Memory address register (MAR):
holds the address of the memory location to be accessed.

Memory data register (MDR):
MDR contains the data to be written into or read out of the addressed location.

General-purpose register (R0 – Rn-1)

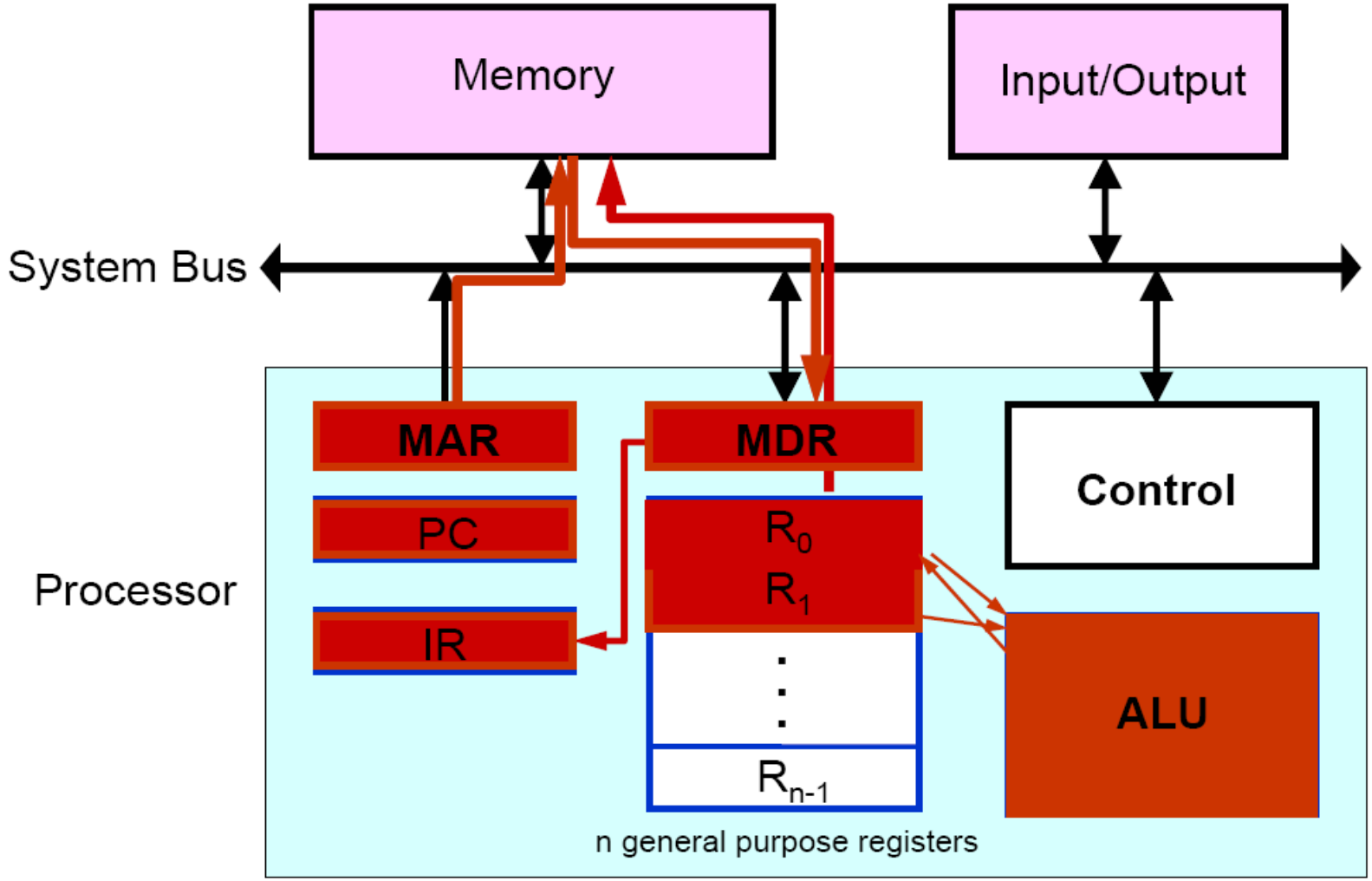
Computer Components: Top-Level View



Typical Operating Steps

- Programs reside in the memory through input devices
- PC is set to point to the first instruction
- The contents of PC are transferred to MAR
- A Read signal is sent to the memory
- The first instruction is read out and loaded into MDR
- The contents of MDR are transferred to IR
- Decode and execute the instruction
- Get operands for ALU
    - General-purpose register
    - Memory (address to MAR – Read – MDR to ALU)
- Perform operation in ALU
- Store the result back
    - To general-purpose register
    - To memory (address to MAR, result to MDR – Write)
- During the execution, PC is incremented to the next instruction

Basic Operational Concepts



Interrupt

- Normal execution of programs may be interrupted if some device requires urgent servicing
  - To deal with the situation immediately, the normal execution of the current program must be interrupted

- Procedure of interrupt operation
  - The device raises an interrupt signal
  - The processor provides the requested service by executing an appropriate interrupt-service routine
  - The state of the processor is first saved before servicing the interrupt
    - Normally, the contents of the PC, the general registers, and some control information are stored in memory
  - When the interrupt-service routine is completed, the state of the processor is restored so that the interrupted program may continue

stored program concept

In the stored program concept, both the instructions and the data (that the instructions operate on) are stored in the computer memory itself.

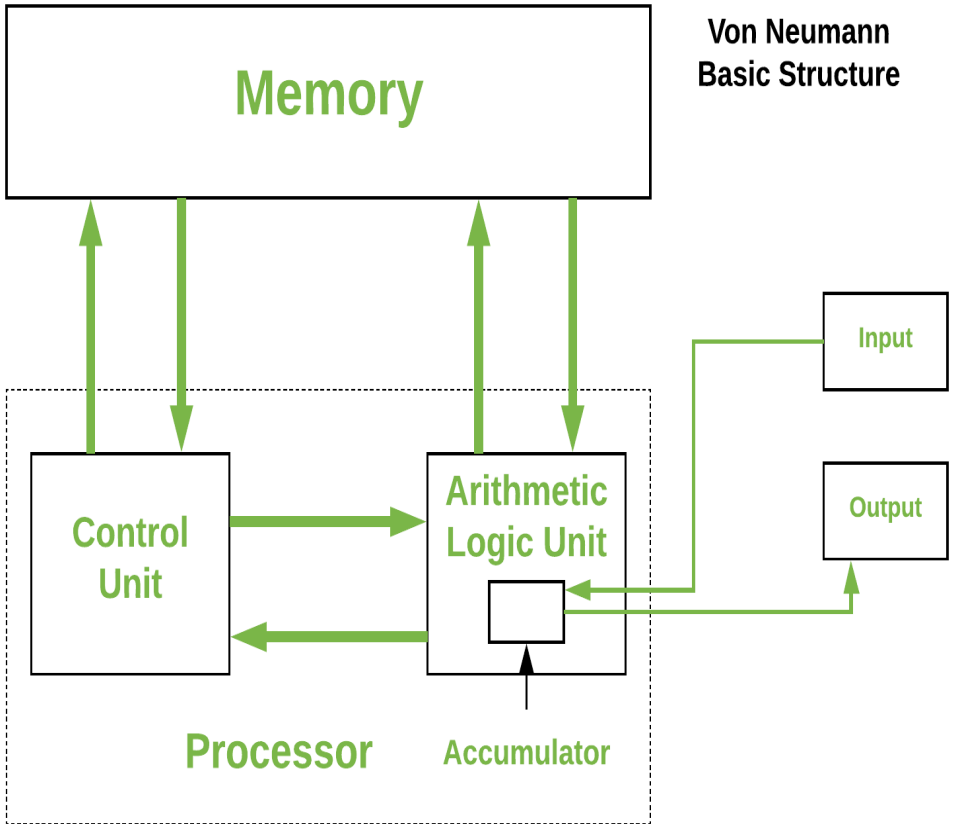
Before the introduction of this idea, instructions and data were considered two totally different entities and were thus stored separately.

Computers that store both instructions and data on the same memory are said to be based on the **Von Neumann architecture.**

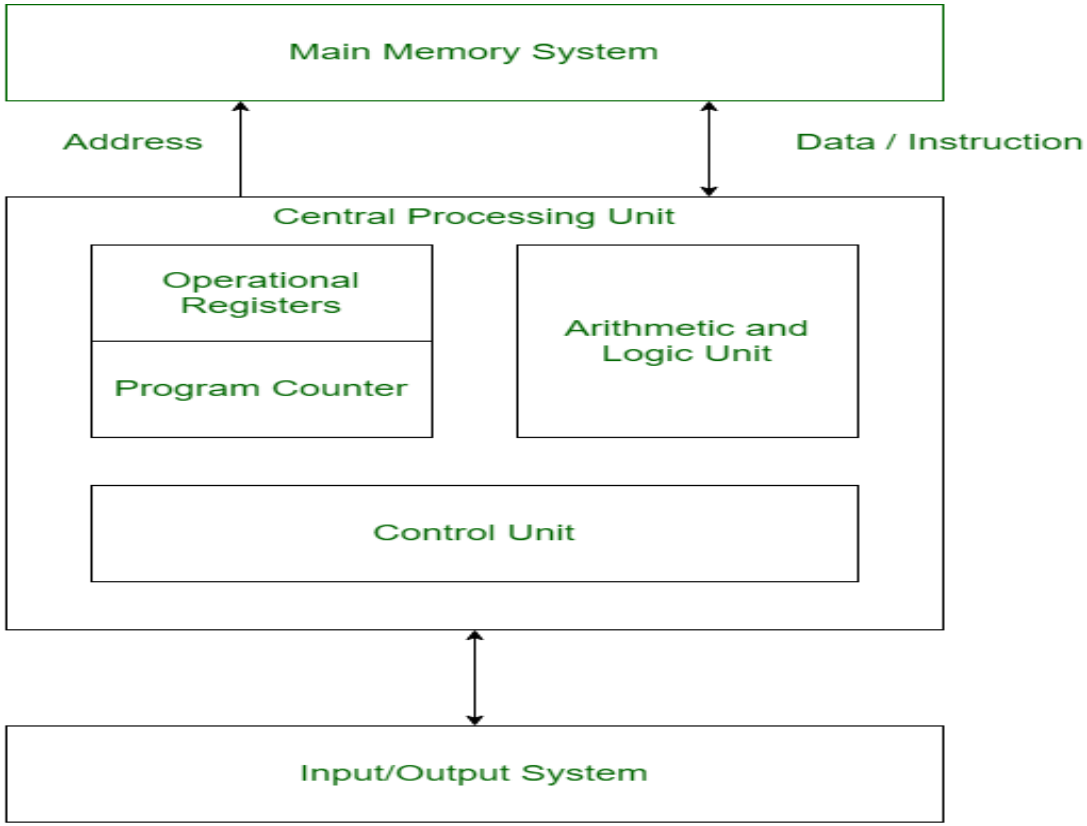
Modern desktop computers are still based on the same stored program concept.

Harvard machine instructions and data are present in Separate memory.

Von-Neumann Architecture



Von-Neumann Architecture



Von Neumann Architecture

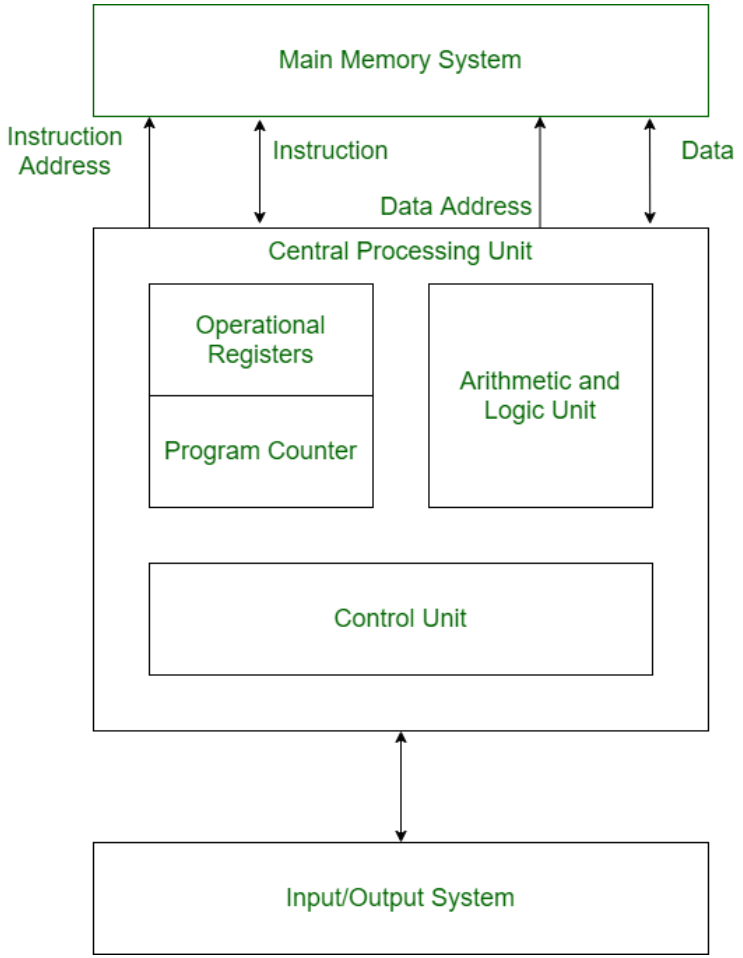
Von Neumann bottleneck

- Whatever we do to enhance performance, we cannot get away from the fact that **instructions can only be done one at a time** and **can only be carried out sequentially.**
- Both of these factors hold back the competence of the CPU. This is commonly referred to as the **‘Von Neumann bottleneck’.**
- We can provide a Von Neumann processor with more cache, more RAM, or faster components but if original gains are to be made in CPU performance then an influential inspection needs to take place of CPU configuration.
- This architecture is very important and is used in our PCs and even in Super Computers.

Harvard Architecture

- In a normal computer that follows von Neumann architecture, instructions, and data both are stored in the same memory.
- So same buses are used to fetch instructions and data.
- This means the CPU cannot do both things together (read the instruction and read/write data).
- **Harvard Architecture** is the computer architecture that contains separate storage and separate buses (signal path) for instruction and data.
- It was basically developed to overcome the bottleneck of Von Neumann's Architecture.
- The main advantage of having separate buses for instruction and data is that the CPU can access instructions and read/write data at the same time.

Structure of Harvard Architecture



Harvard Architecture - Features

- **Separate memory spaces:** In Harvard architecture, there are separate memory spaces for instructions and data. This separation ensures that the processor can access both the instruction and data memories simultaneously, allowing for faster and more efficient data retrieval.
- **Fixed instruction length:** In Harvard architecture, instructions are typically of fixed length, which simplifies the instruction fetch process and allows for faster instruction processing.
- **Parallel instruction and data access:** Since Harvard architecture separates the memory spaces for instructions and data, the processor can access both memory spaces simultaneously, allowing for parallel instruction and data processing.
- **More efficient memory usage:** Harvard architecture allows for more efficient use of memory as the data and instruction memories can be optimized independently, which can lead to better performance.
- **Suitable for embedded systems:** Harvard architecture is commonly used in embedded systems because it provides fast and efficient access to both instructions and data, which is critical in real-time applications.
- **Limited flexibility:** The separate memory spaces in Harvard architecture limit the flexibility of the processor to perform certain tasks, such as modifying instructions at runtime. This is because modifying instructions requires access to the instruction memory, which is separate from the data memory.

Advantage of Harvard Architecture

- Harvard architecture has two separate buses for instruction and data. Hence, the CPU can access instructions and read/write data at the same time.
- This is the major advantage of Harvard architecture.

Fast and efficient data access: Since Harvard architecture has separate memory spaces for instructions and data, it allows for parallel and simultaneous access to both memory spaces, which leads to faster and more efficient data access.

Better performance: The use of fixed instruction length, parallel processing, and optimized memory usage in Harvard architecture can lead to improved performance and faster execution of instructions.

Suitable for real-time applications: Harvard architecture is commonly used in embedded systems and other real-time applications where speed and efficiency are critical.

Security: The separation of instruction and data memory spaces can also provide a degree of security against certain types of attacks, such as buffer overflow attacks.

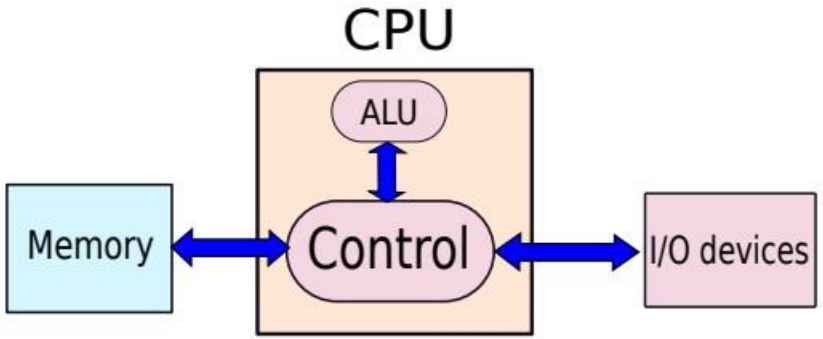
Disadvantages of Harvard Architecture

- **Complexity:** The use of separate memory spaces for instructions and data in Harvard architecture adds to the complexity of the processor design and can increase the cost of manufacturing.
- **Limited flexibility:** Harvard architecture has limited flexibility in terms of modifying instructions at runtime because instructions and data are stored in separate memory spaces. This can make certain types of programming more difficult or impossible to implement.
- **Higher memory requirements:** Harvard architecture requires more memory than Von Neumann architecture, which can lead to higher costs and power consumption.
- **Code size limitations:** Fixed instruction length in Harvard architecture can limit the size of code that can be executed, making it unsuitable for some applications with larger code bases.

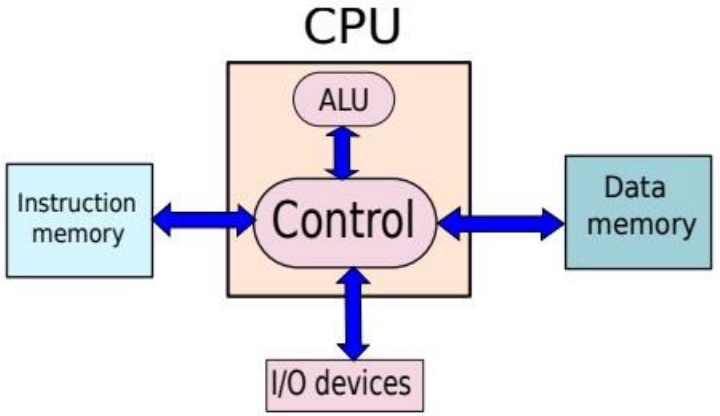
Difference between Von Neumann and Harvard Architecture

VON NEUMANN ARCHITECTURE	HARVARD ARCHITECTURE
It is ancient computer architecture based on **stored program** computer concept.	It is modern computer architecture based on **Harvard Mark I relay** based model.
Same physical memory address is used for instructions and data.	**Separate physical memory address** is used for instructions and data.
There is **common bus for data and instruction** transfer.	**Separate buses** are used for transferring data and instruction.
Two clock cycles are required **to execute single instruction.**	An instruction is executed in a **single cycle.**
It is **cheaper in cost.**	It is **costly** than Von Neumann Architecture.
CPU can not access instructions and read/write at the same time.	CPU can access instructions and read/write at the same time.
It is used in **personal computers and small computers.**	It is used in **micro controllers and signal processing.**

Von Neumann vs Harvard Architecture



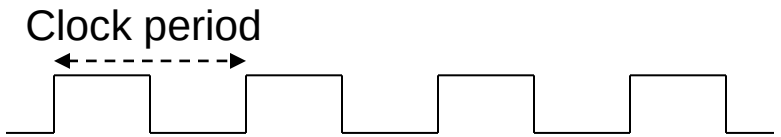
Von Neumann Architecture



Harvard Architecture

CPU Speed

- Processor circuits are controlled by a timing signal called a Clock.
- The clock goes through a regular/repetitive action. In a binary system, a cycle consists of a 1 and a 0 (a high followed by a low). – Clock Cycles



- The clock is usually a quartz oscillator that is **external to the microprocessor**.

CPU Speed (Cont.)

- The speed is measured in **Hertz**, which are cycles per second.
  - KiloHertz, kHz, is thousands (10^3) of cycles per second
  - MegaHertz, MHz, is millions (10^6) of cycles per second
  - GigaHertz, GHz, is billions (10^9) of cycles per second

CPU Speed (Cont.)

- The clock speed is also known as the clock's **frequency** (the number of cycles per second).
- A related quantity is called the **period** which is the time required for one cycle (a.k.a. as a clock tick).
- A clock's frequency and period are reciprocals.
  - **f = 1/T** or **T = 1/f**, where f is frequency and T is period
  - E.g. a frequency of 60 Hertz (cycles per second) corresponds to a period of 1/60 = 0.0167 seconds per cycle

CPU Speed (Cont.)

- A frequency of **1 kHz** [a thousand cycles per second] corresponds to a period (tick) of **1 millisecond (ms)** [a thousandth (10^{-3}) of a second per cycle].
- A frequency of **1 MHz** [a million cycles per second] corresponds to a period (tick) of **1 microsecond (μs)** [a millionth (10^{-6}) of a second per cycle].
- A frequency of **1 GHz** [a billion cycles per second] corresponds to a period (tick) of 1 **nanosecond (ns)** [a billionth (10^{-9}) of a second per cycle].

Question

- If the processor speed is 3.20 GHz, what is the corresponding period?
- Approach 1:
  - 3.20 GHz → 3.20 x 10^9 Hz
  - T = 1/f → 1/3.20 x 10^9 Hz = 0.000000003125 s = 0.0000003125 ms = 0.0003125 µs = 0.3125 ns
- Approach 2:
  - 1/3.20 GHz = 0.3125 ns (just know reciprocal of GHz is ns)

What Is an Instruction Set Architecture?

1. The Instruction Set Architecture (ISA) is the part of the processor that is visible to the programmer or compiler writer.
2. The ISA acts as an interface between the hardware and the software, specifying both what the processor is capable of doing as well as how it gets done.
3. The ISA provides the only way through which a user is able to interact with the hardware.
4. The ISA defines the supported data types, the registers, how the hardware manages main memory.

ISAs

Ideally the only part of the machine visible to the programmer/compiler

- ***Available instructions (Opcodes)***
- ***Formats***
- ***Registers, number and type***
- ***Addressing modes, access mechanisms***
- ***Exception conditions etc.***

3 most common types of ISAs

- **Stack -** The operands are implicitly on top of the stack.
- **Accumulator -** One operand is implicitly the accumulator.
- **General Purpose Register (GPR) -** All operands are explicitly mentioned, they are either registers or memory locations.

in all 3 architectures:

Lets look at the assembly code of

C = A + B;

Stack	Accumulator	GPR
PUSH A	LOAD A	LOAD R1,A
PUSH B	ADD B	ADD R1,B
ADD	STORE C	STORE R1,C
POP C		

Stack

- **Advantages:** Simple Model of expression evaluation (reverse polish). Short instructions.
  Disadvantages: A stack can't be randomly accessed This makes it hard to generate efficient code. The stack itself is accessed every operation and becomes a bottleneck.

Accumulator

- **Advantages:** Short instructions.
  Disadvantages: The accumulator is only temporary storage so memory traffic is the highest for this approach.

GPR

- **Advantages:** Makes code generation easy. Data can be stored for long periods in registers.
  Disadvantages: All operands must be named leading to longer instructions.

- Earlier CPUs were of the first 2 types but in the last 15 years all CPUs made are GPR processors. The 2 major reasons are that **registers are faster than memory**, the more data that can be kept internally in the CPU the faster the program wil run. The other reason is that **registers are easier for a compiler to use.**

Instruction Formats

- On the basis of number of address instruction are classified.
- **The layout of bits of instruction** is called an instruction format.

- Three-Address Instructions
  - ADD R1, R2, R3 R3 ← [R1] +[R2]
  - Operation Destination,Source1,Source2

  This has three address field to specify a register or a memory location.

Instruction Formats

- Two-Address Instructions
  - ADD R1, R2 R1 ← [R1] + [R2]
  - Operation Destination,Source

  Here two address can be specified in the instruction.
  here result cab be stored at different location.

Instruction Formats

- One-Address Instructions
  - ADD B AC ← AC + M[B]

This use a implied ACCUMULATOR register for data manipulation.

One operand is in accumulator and other is in register or memory location.

Implied means that the CPU already know that one operand is in accumulator so there is no need to specify it.

Instruction Formats

- Zero-Address Instructions
  - ADD — TOS ← TOS + (TOS – 1)

    TOS- Top of stack

  A stack based computer do not use address field in instruction.

  To evaluate a expression first it is converted to revere Polish Notation i.e. Post fix Notation.

Instruction Formats

Example: Evaluate x = (A+B) * (C+D)

- Three-Address

```
1. ADD   R1, A, B      ; R1 ← M[A] + M[B]
2. ADD   R2, C, D      ; R2 ← M[C] + M[D]
3. MUL   X, R1, R2     ; M[X] ← R1 * R2
```

Instruction Formats

Example: Evaluate x = (A+B) * (C+D)

- Two-Address

```
1. MOV   R1, A      ; R1 ← M[A]
2. ADD   R1, B      ; R1 ← R1 + M[B]
3. MOV   R2, C      ; R2 ← M[C]
4. ADD   R2, D      ; R2 ← R2 + M[D]
5. MUL   R1, R2     ; R1 ← R1 * R2
6. MOV   X, R1      ; M[X] ← R1
```

Instruction Formats

Example: Evaluate x = (A+B) * (C+D)

- One-Address

1.	LOAD	A	; AC ← M[A]
2.	ADD	B	; AC ← AC + M[B]
3.	STORE	T	; M[T] ← AC
4.	LOAD	C	; AC ← M[C]
5.	ADD	D	; AC ← AC + M[D]
6.	MUL	T	; AC ← AC * M[T]
7.	STORE	X	; M[X] ← AC

Zero Address

Example: Evaluate x = (A+B) * (C+D)

PUSH	A	TOP = A
PUSH	B	TOP = B
ADD		TOP = A+B
PUSH	C	TOP = C
PUSH	D	TOP = D
ADD		TOP = C+D
MUL		TOP = (C+D)*(A+B)
POP	X	M[X] = TOP

Write a program for zero address instructions to evaluate the following arithmetic instruction.
((a-b)) / ((c)+(d*e))

Write a program for zero address instructions to evaluate the following arithmetic instruction.
((a-b)) / ((c)+(d*e))

- Postfix Notation: ab-cde *+/

X= (a-b) / (c+(d*e))	
Micro-Instructions	Top of Stack Value - Explanation
PUSH a	a
PUSH b	b
SUB	a-b
PUSH c	c
PUSH d	d
PUSH e	e
MUL	d*e
ADD	c+d*e
DIV	(a-b) / (c+(d*e))
POP X	Move top value of stack in variable X

RISC

1. RISC stands for Reduced Instruction Set Computer.
2. RISC processors have simple instructions taking about one clock cycle. The average clock cycle per instruction (CPI) is 1.5
3. It has a hard-wired unit of programming.
4. The instruction set is reduced i.e. it has only a few instructions in the instruction set. Many of these instructions are very primitive.
5. Multiple register sets are present
6. RISC processors are highly pipelined
7. Execution time is very less
8. Code expansion can be a problem
9. The decoding of instructions is simple.
10. The most common RISC microprocessors are Alpha, ARC, ARM, AVR, MIPS, PA-RISC, PIC, Power Architecture, and SPARC.
11. RISC architecture is used in high-end applications such as video processing, telecommunications, and image processing.

CISC

1. CISC stands for Complex Instruction Set Computer.
2. CSIC processor has complex instructions that take up multiple clocks for execution. The average clock cycle per instruction (CPI) is in the range of 2 and 15.
3. It has a microprogramming unit.
4. The instruction set has a variety of different instructions that can be used for complex operations.
5. Only has a single register set
6. They are normally not pipelined or less pipelined
7. Execution time is very high
8. Code expansion is not a problem
9. Decoding of instructions is complex
10. Examples of CISC processors are the System/360, VAX, PDP-11, Motorola 68000 family, AMD, and Intel x86 CPUs.
11. CISC architecture is used in low-end applications such as security systems, home automation, etc.

Byte ordering, or endianness

- As an example, suppose we have the hexadecimal number 12345678.
- The big endian and small endian arrangements of the bytes are shown below.

Address ⟶	00	01	10	11
Big Endian	12	34	56	78
Little Endian	78	56	34	12

Quiz

1. System Bus is Collection of _____ lines.
    (a) Address Bus
    (b) Control Bus
    (c) Data Bus
    (d) All
2. In READ control signal data transfer is between _____ and _____.
    (a) Memory , CPU
    (b) Memory , I/O
    (c) CPU , Memory
    (d) I/O , Memory

Quiz

3. If there are 8 address lines then the number of memory locations in the memory are____

    (a) 8

    (b) 256

    (c) 128

    (d) 64

4. If there are 12 address lines then the memory capacity is_____

    (a) 4 Kilo Bits

    (b) 14 bits

    (c) 14 Kilo Bytes

    (d) 4 Kilo Bytes

Quiz

5. A 4GB memory with 8 bit data lines has _____address lines

    (a) 32

    (b) 64

    (c) 22

    (d) 12

6. The registers,ALU and the interconnection between them are collectively called as _____ .

    (a) Process route

    (b) Information trail

    (c) information path

    (d) data path