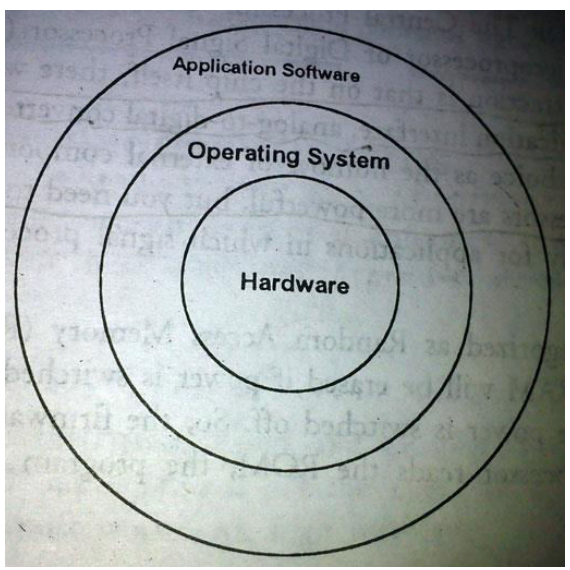
Chapter 2
Architecture of Embedded Systems

Overview of Embedded System Architecture

- Every embedded system consists of custom-built hardware built around a Central Processing Unit (CPU).
- The software residing on the memory chip is also called the ‘firmware’.

- The operating system runs above the hardware, the application software runs above the operating system.
- To have operating system in Embedded System is not compulsory. For example in case of small appliances such as remote control units, air-conditioners, toy etc there is no need of operating system
- It is possible to write small software specific to that application.



Pseudocode

```
Function Main_Function()
{
  Initialization();
  Do_Forever
  {
    Check_Status();
    Do_Calculations();
    Output_Response();
  }
}
```

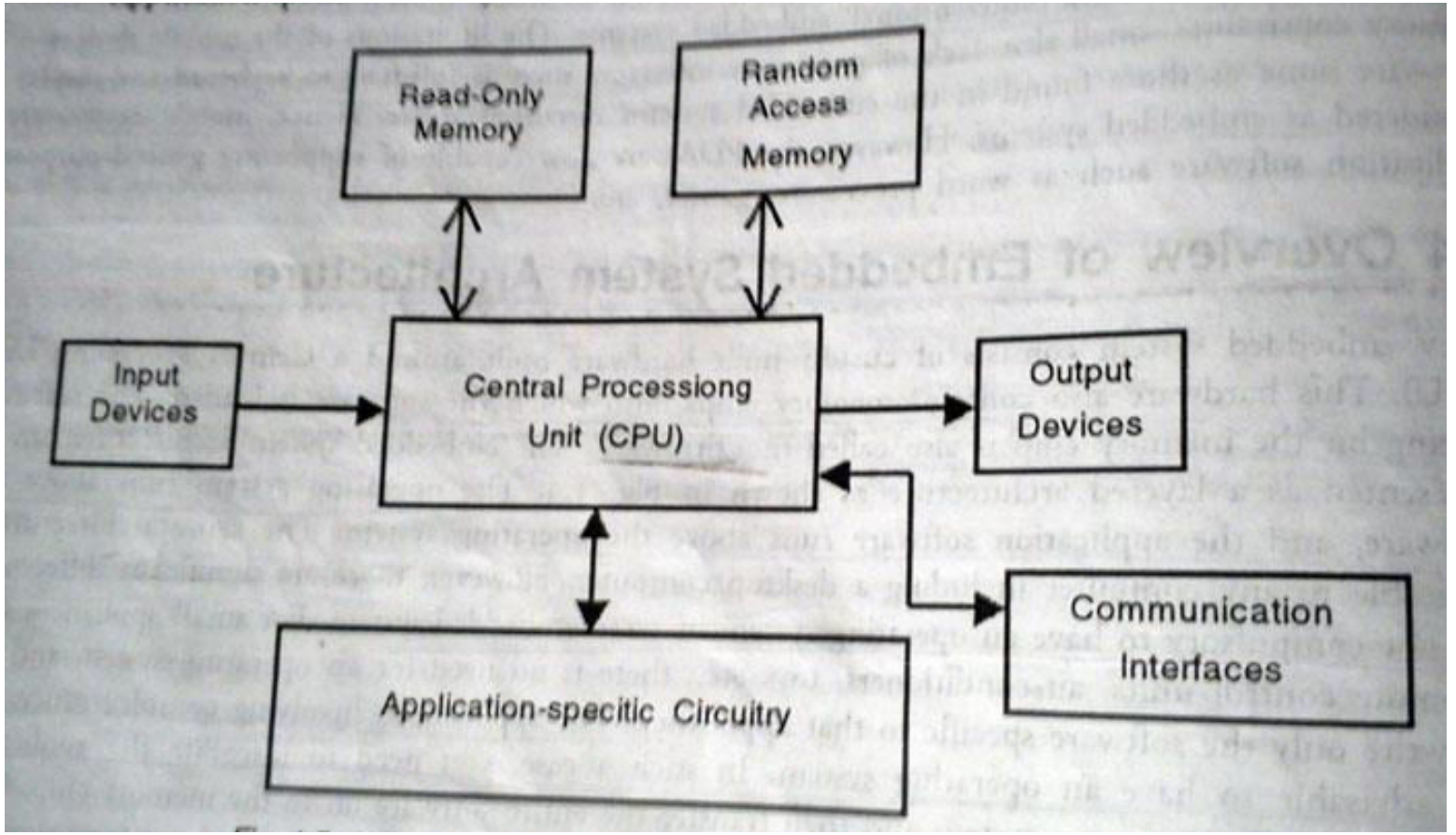
- For the complex processing, it is better to have operating system, in which deployment of application in the memory chips after operating system is integrated with the application.
- Once the software is transferred to the memory chip, the software will continue to run for a long time and don't need to reload new software.

CA

- Refer Moodle (1)

Various building blocks of the hardware of an embedded system.

- Central Processing Unit (CPU)
- Memory (Read-only Memory and Random Access Memory)
- Input Devices
- Output Devices
- Communication Interfaces
- Application-specific circuitry



- Simplified Hardware Architecture of an Embedded System

Central Processing Unit (CPU)

- The Central Processing Unit (Processor) can be any of the following:
  - Micro-controller,
  - Microprocessor
  - Digital Signal Processor (DSP)

- For small applications microcontroller is the best choice as the number of external components required very less.

- Microprocessor is very powerful, but it needs many external components

- DSP is used mainly for applications for signal processing is involved such as audio and video processing.

Memory

- Memory are categorized as
  - Random Access Memory (RAM)
  - Read Only Memory (ROM)
- The contents of the RAM will be erased if power is switched off to the chip, whereas ROM retains the contents even if the power is switched off. So, the firmware is stored in the ROM.
- When power is switched on, the processor read the ROM, the program is transferred to RAM and the program is executed.

Input Devices

- Unlike the desktops, the input devices to an embedded system have very limited capability. There will be no keyword or a mouse and hence interacting with the embedded system is no easy task.
- Many will have small keypad.
- They take inputs from sensors or transducers and produce electrical signals that are in turn fed to others systems.

Output devices

- Mostly output of embedded system is also limited capability.
- Some embedded systems will have a few Light Emitting Diodes (LEDs).
- A small Liquid Crystal Display (LCD)

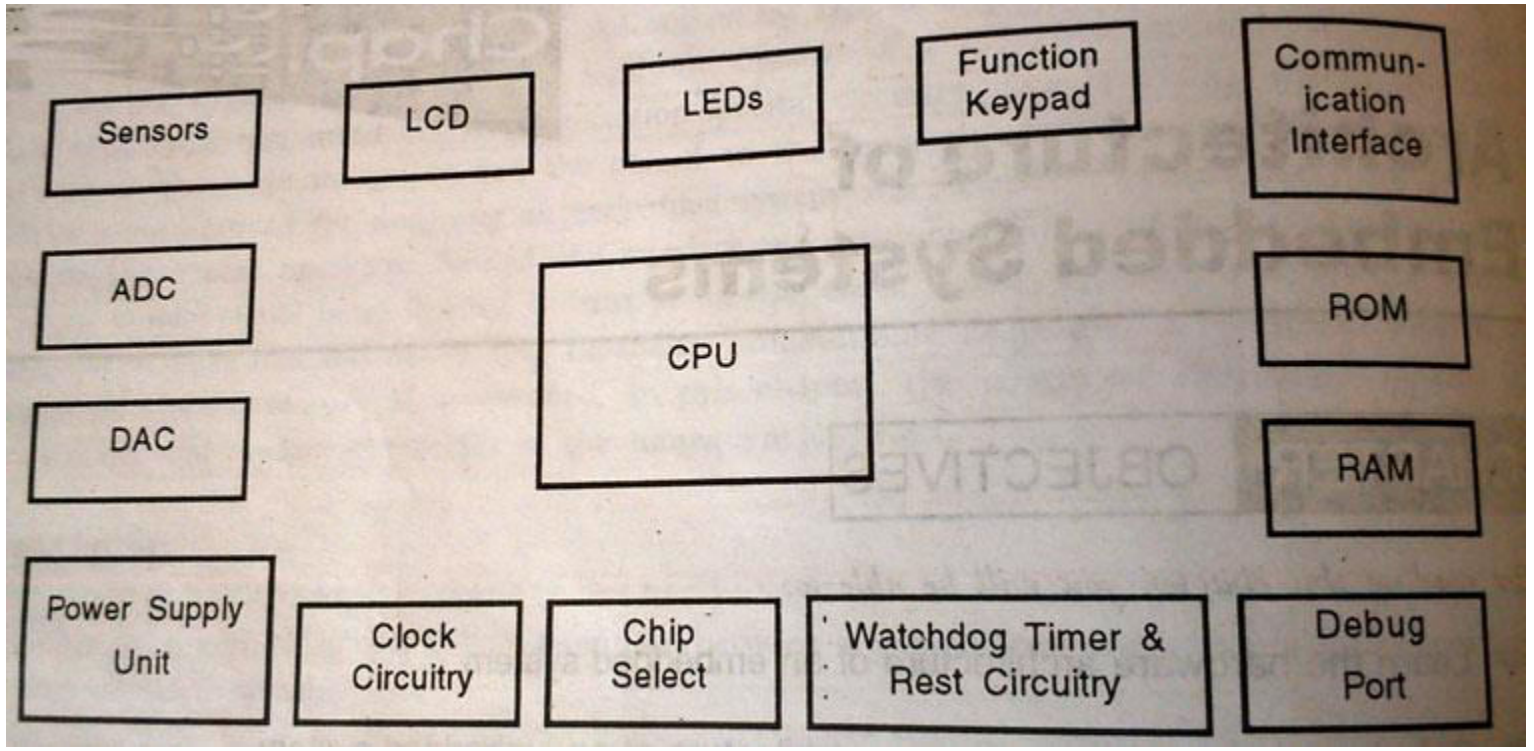
Communication Interfaces

- The embedded systems may need to interact with other embedded systems.
- Very few communication interfaces such as RS232, RS422, RS485, Universal Serial Bus (USB), IEEE 1394, Ethernet etc

Application-specific Circuitry

- Sensors, transducers, special processing and control circuitry may be required for an embedded system, depending on its application. This circuitry interacts with the processor to carry out the necessary work.
- The entire hardware has to be given power supply either through the 220 volts main supply or through a battery.
- The hardware has to be designed in such a way that the power consumption is minimized.
  - Sensors are devices which mainly sense any thing and give output in any desired form; where as transducers are devices that transform one form of energy in to another form.
  - transducer is a device which converts one form of signal to another. For example, an LED is a transducer which converts electrical energy to light, or an antenna converts EM wave energy to electrical signal.

Hardware Architecture



- Building Blocks of the Hardware in an Embedded System

Central Processing Unit

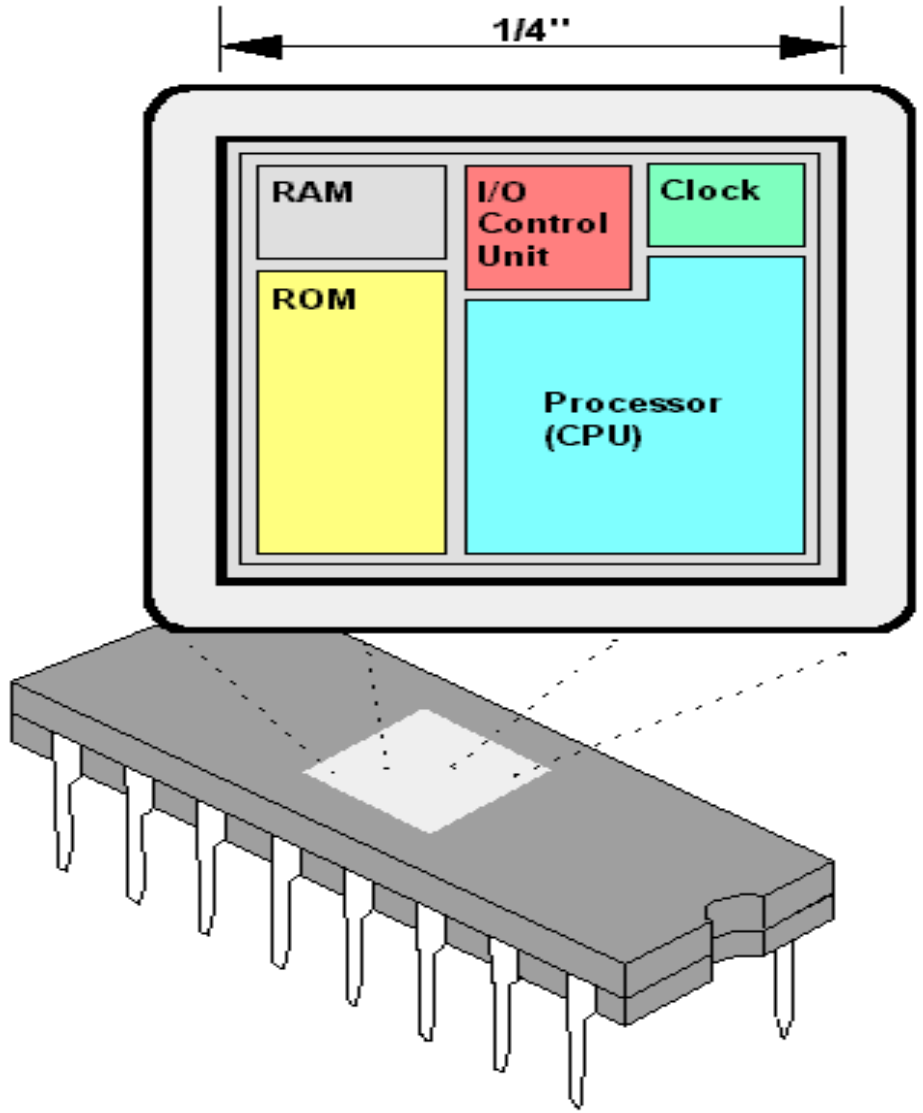
- The central Processing Unit (CPU) used in an embedded system can be one of the following two categories:
  - General Purpose Processor (GPP)
  - Digital Signal Processor (DSP)

- In case of General Purpose Processor (GPP) are further classified as micro-controllers and microprocessors.
  - Micro Controller has a memory and other peripheral on the chip itself; hence it is the best choice for small embedded Systems.
  - Microprocessor is more powerful but requires a large number of external components.

Microcontroller

- A single chip that contains the processor (the CPU), non-volatile memory for the program (ROM or flash), volatile memory for input and output (RAM), a clock and an I/O control unit.
- Also called a "computer on a chip," billions of microcontroller units (MCUs) are embedded each year in a myriad of products from toys to appliances to automobiles. For example, a single vehicle can use 70 or more microcontrollers.

From Computer Desktop Encyclopedia

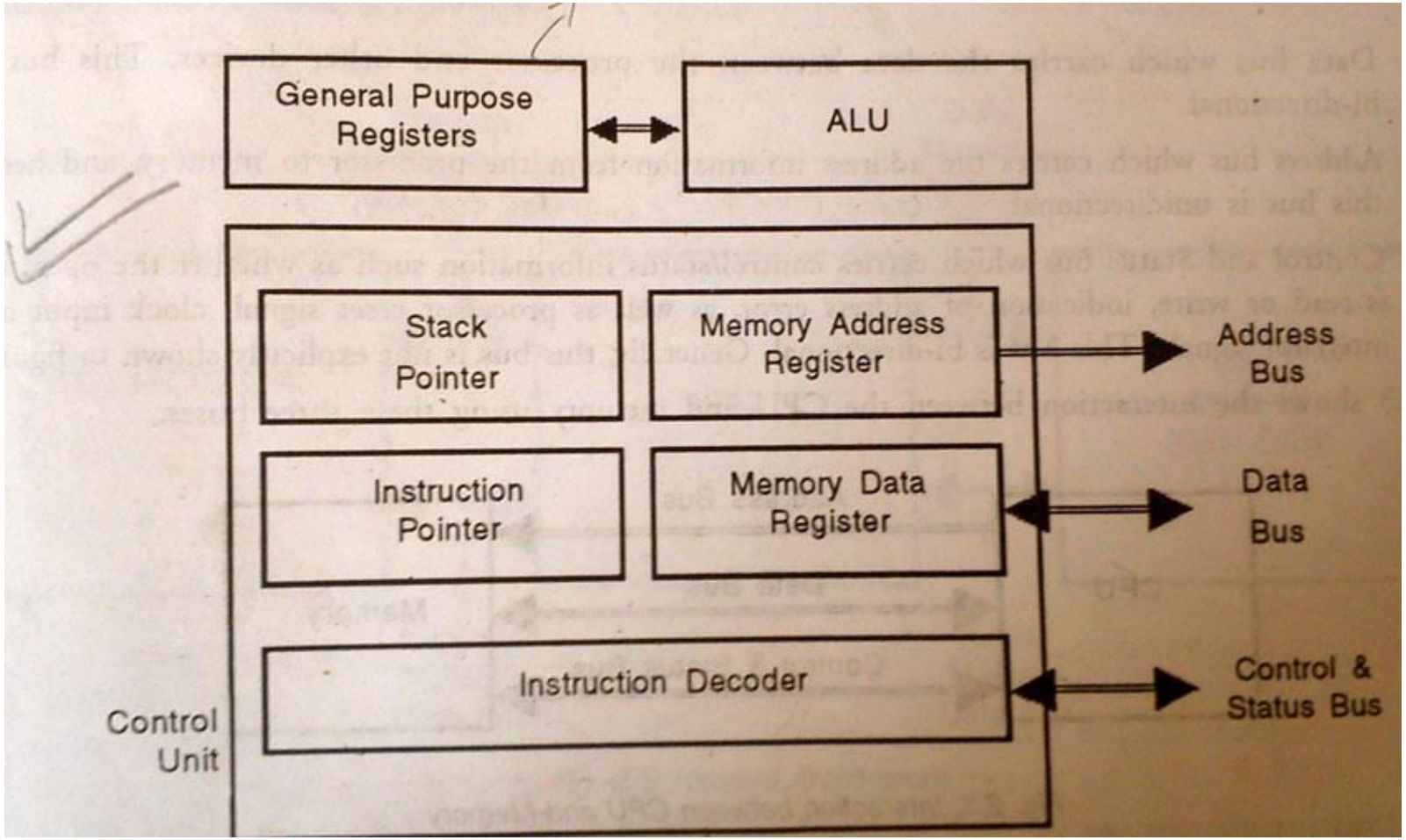


An entire computer on a single chip.

Microprocessor	Microcontroller
A silicon chip representing a central processing unit(CPU), which is capable of performing arithmetic as well as logical operations according to a pre-defined set of instructions	A microcontroller is a highly integrated chip that contains a CPU, RAM, special and general purpose register arrays, on chip ROM/ FLASH memory for program storage, timer and interrupt control units and dedicated I/O ports.
It is a dependent unit. It requires the combination of other chips like timers, program and data memory chips; interrupt controllers, etc. for functioning.	It is a self-contained unit and it doesn't require external interrupt controller, timer, UART, etc for its functioning.
Most of the time general purpose in design and operation.	Mostly application-oriented or domain-specific.
Limited power saving options compared to micro-controllers	Includes lot of power saving features

Links

- https://www.engineersgarage.com/tutorials/difference-between-microprocessor-and-microcontroller/



- Internal Architecture of a Processor

Processor Architectures

- The CPU consists of
  - Arithmetic Logic Unit (ALU) which performs arithmetic and Logic operations (such as add, multiply, subtract etc)

  - General Purpose Registers. These Registers constitute the processor's internal memory. The number of registers varies from processor to processor. Registers contain the current data and operands that are being manipulated by the processor.
  - Control unit that fetches the instructions from memory, decodes them and executes them.

Register

- Instruction Decoder

Its purpose is to translate an **instruction** code into the address in the micro memory where the micro code for the **instruction** starts.

Register

- Stack Pointer

  It stores the address of the last program request in a stack.

Register

- Instruction pointer

It holds the address of the next **instruction** to be executed. Control flow **instructions** such as jumps, conditional branches, subroutine calls and returns manipulate the **instruction pointer**.[PC(Program Counter), IAR(Instruction Address Register)]

Register

- Memory Address Register(MAR)

It is the CPU register that either stores the memory address from which data will be fetched to the CPU, or the address to which data will be sent and stored.

Register

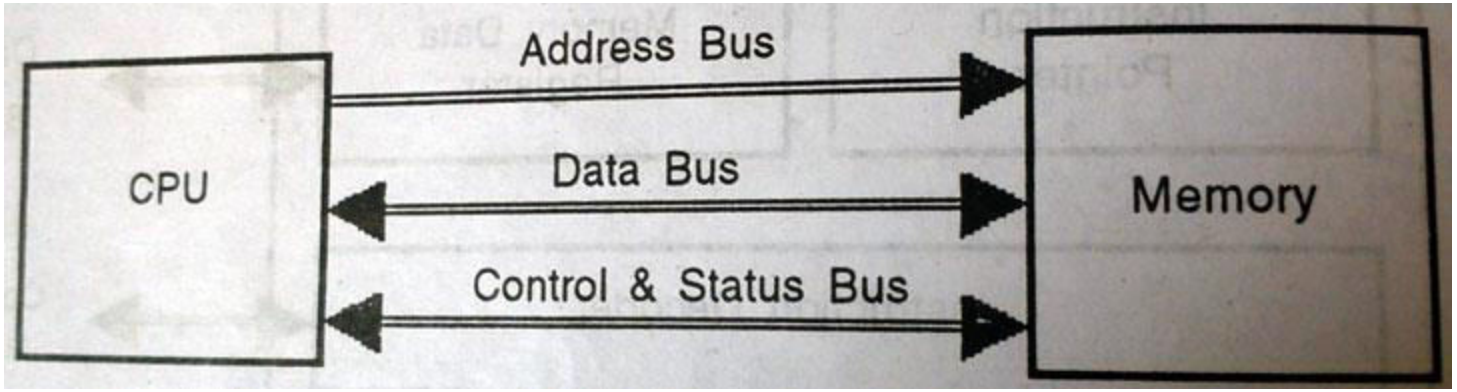
- Memory Buffer Register(MBR,MDR)

  It is the register in a computer's processor, or central processing unit, CPU, that stores the data being transferred to and from the immediate access storage.

Processor Architectures

- To manipulate data, processor's job is
  - to read data and instructions from memory,
  - read and write data to memory,
  - write data to output devices, and
  - read data from input devices.
- To do these functions, the processor communicates with other devices using three buses, a bus being a group of signals. These buses are:
  - Data bus
  - Address bus
  - Control and Status bus

- Data bus which carries the data between the processor and other devices. This bus is bi-directional.
- Address bus which carries the address information from the processor to memory, and hence this bus is unidirectional.

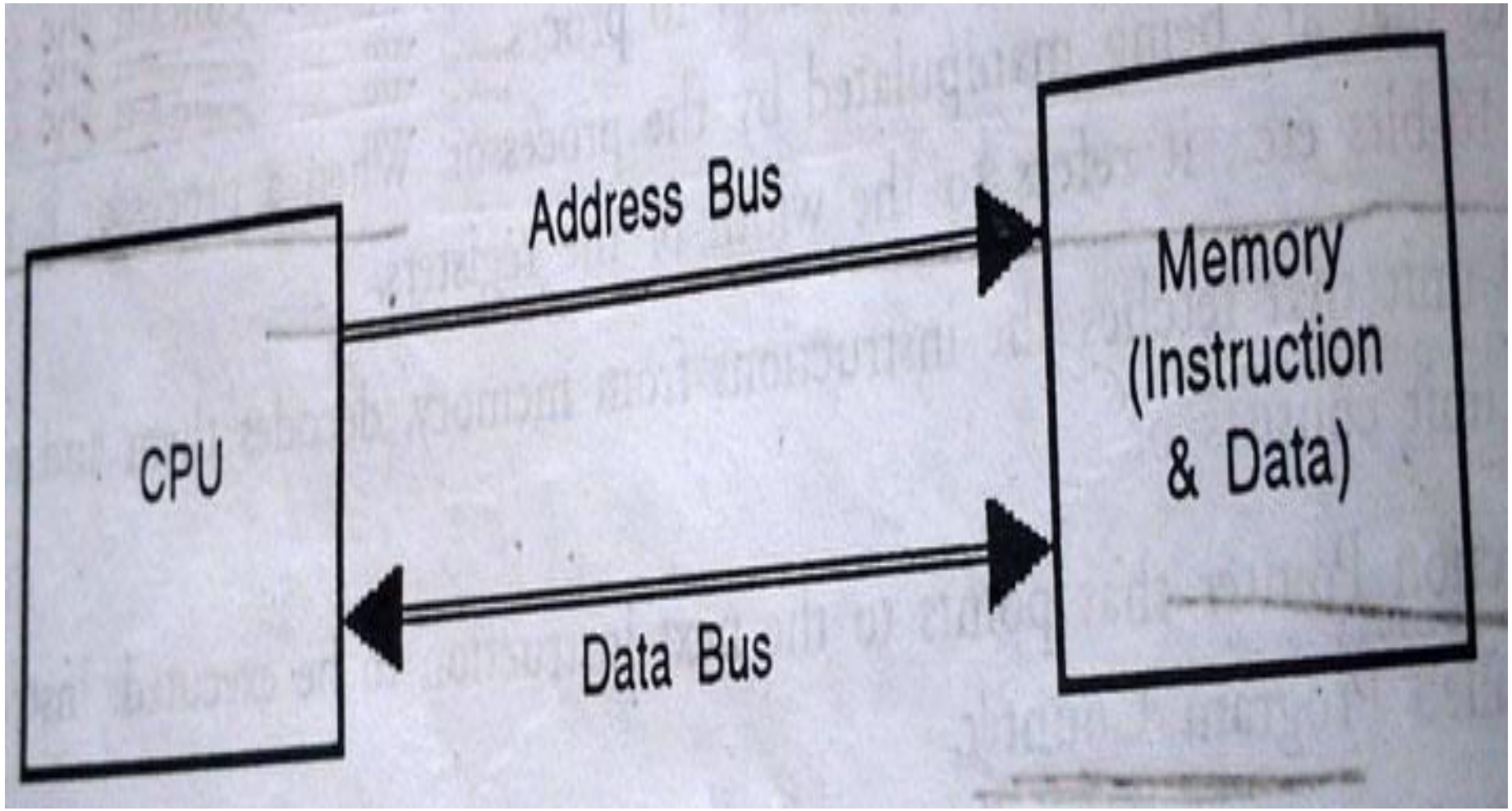


- Control and Status bus which carries control/status information such as whether the operation is read or write, indication of address error, as well as processor reset signal, clock input and interrupt signals.
- This bus is bi-directional.

- Based on the number of memory and data buses used, there are three types of architectures for the processors. These are
  - Von Neumann Architecture
  - Harvard Architecture
  - Super Harvard Architecture

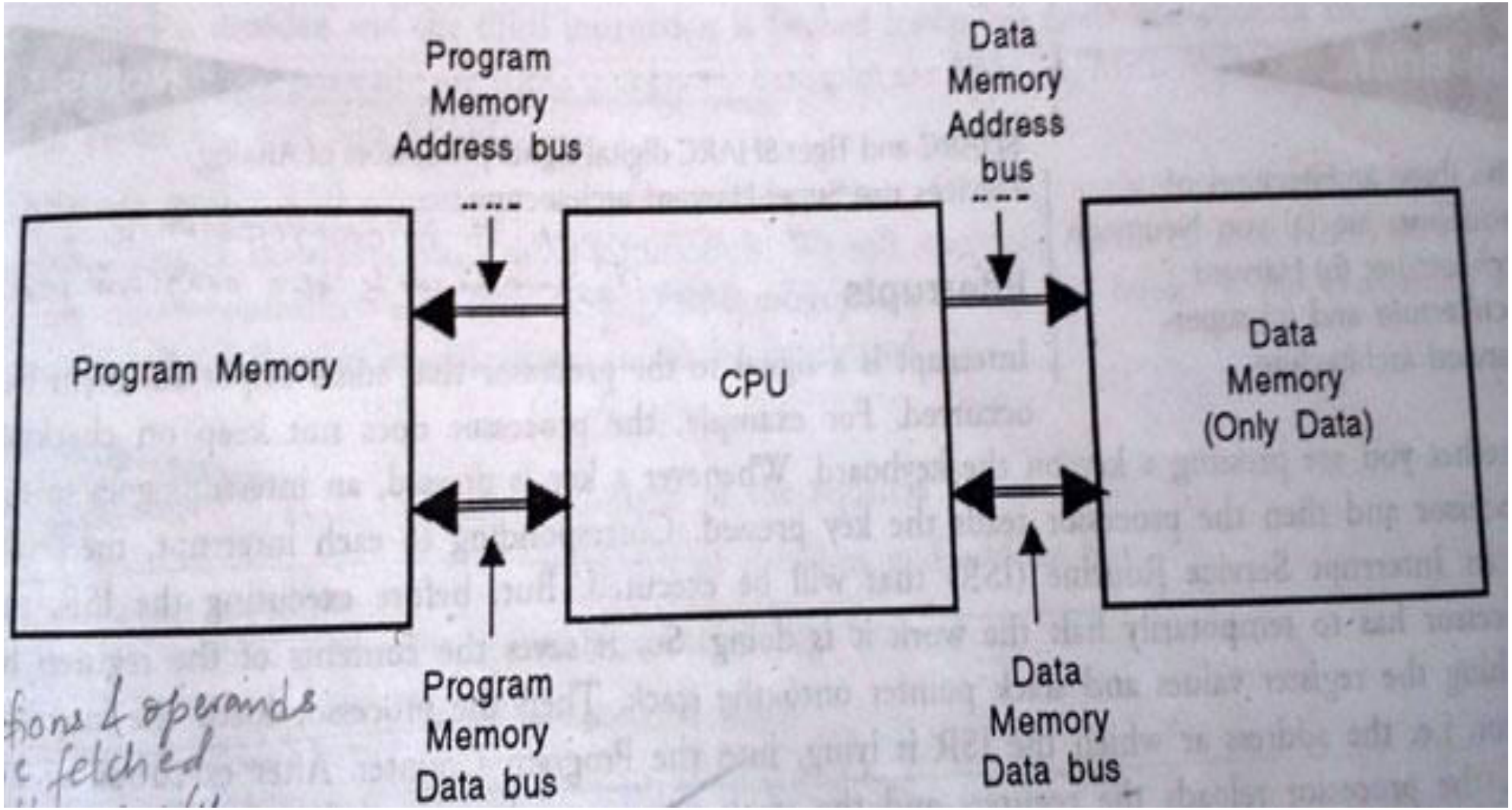
Von Neumann Architecture

- Microprocessors/controllers based on the Von-Neumann architecture shares a single common bus for fetching both instructions and data
- Von-Neumann architecture based processors/controllers first fetch an instruction and then fetch the data to support the instruction from code memory.
- The two separate fetches slows down the controller's operation.


Address Bus
CPU
Memory
(Instruction
& Data)
Data Bus

Harvard Architecture

- Harvard architecture will have separate data bus and instruction bus.
- This allows the data transfer and program fetching to occur simultaneously on both buses.
- In this model, the data memory can be read and written while the program memory is being accessed.
- These separated data memory and code memory buses allow one instruction to execute while the next instruction is fetched (“pre-fetching”).
- The pre-fetch theoretically allows much faster execution than Von-Neumann architecture


Program
Memory
Address bus
Data
Memory
Address
bus
Program Memory
CPU
Data
Memory
(Only Data)
Program
Memory
Data bus
Data
Memory
Data bus

- In this architecture, there are two separate memory blocks- one is program memory and the other is data memory. Program memory stores only instructions and data memory stores only data. Two pairs of data buses are used between the CPU and the memory blocks.

Von-Neumann Architecture	Harvard Architecture
Single shared bus for instruction and data fetching	Separate buses for instruction and data fetching
Low performance compared to Harvard architecture	Easier to pipeline, so high performance can be achieved
Cheaper	Comparatively high cost
Allows self modifying codes	No Memory alignment problems
Since data memory and program memory are stored physically in the same chip, chances for accidental corruption of program memory.	Since data memory and program memory are stored physically in different locations, no chances for accidental corruption of program memory

Note: Self-modifying code *is code that alters its own instructions while it is executing - usually to reduce the instruction path length and improve performance or simply to reduce otherwise repetitively similar code, thus simplifying maintenance.*

Atmel's AVRs have a single level pipeline design.
This means the next machine instruction is fetched as the current one is executing.

Super Harvard Architecture

- A slight but significant modification of the Harvard architecture.
- In Harvard architecture, the data memory is accessed more frequently than the program memory.
- So SHARC is design to store some secondary data in the program memory to balance the load on both memory blocks. This architecture is used in Digital Signal Processor.