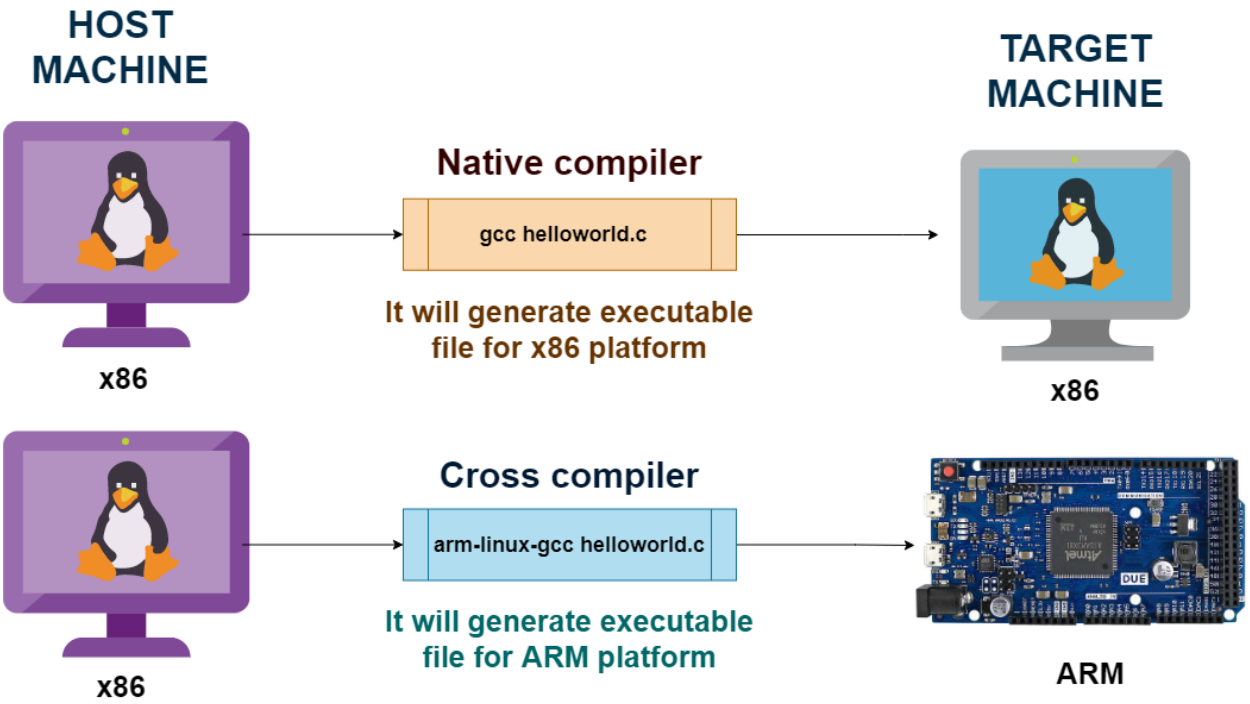
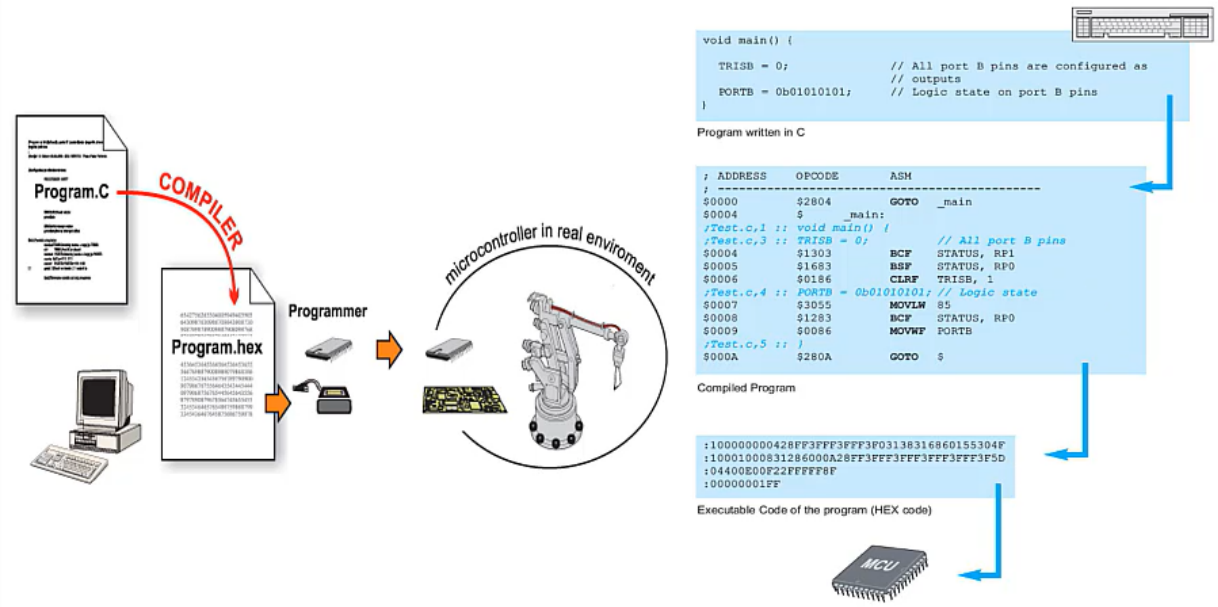
Introduction to Microcontroller Interfacing

Flashing(Burning) Code on MCU

- First we need to know the difference between **Cross** compiler and **Native** compiler
- **Native compiler :** Host will compile the code and run it at the same platform(Host platform)
- **Cross compiler :** Host will compile the code and run it for specfic target(Ex ARM platform) across burner(**Programmer Debugger**)(Ex: for PIC we use **PicKit3**) by taking the code from host and burn it on target

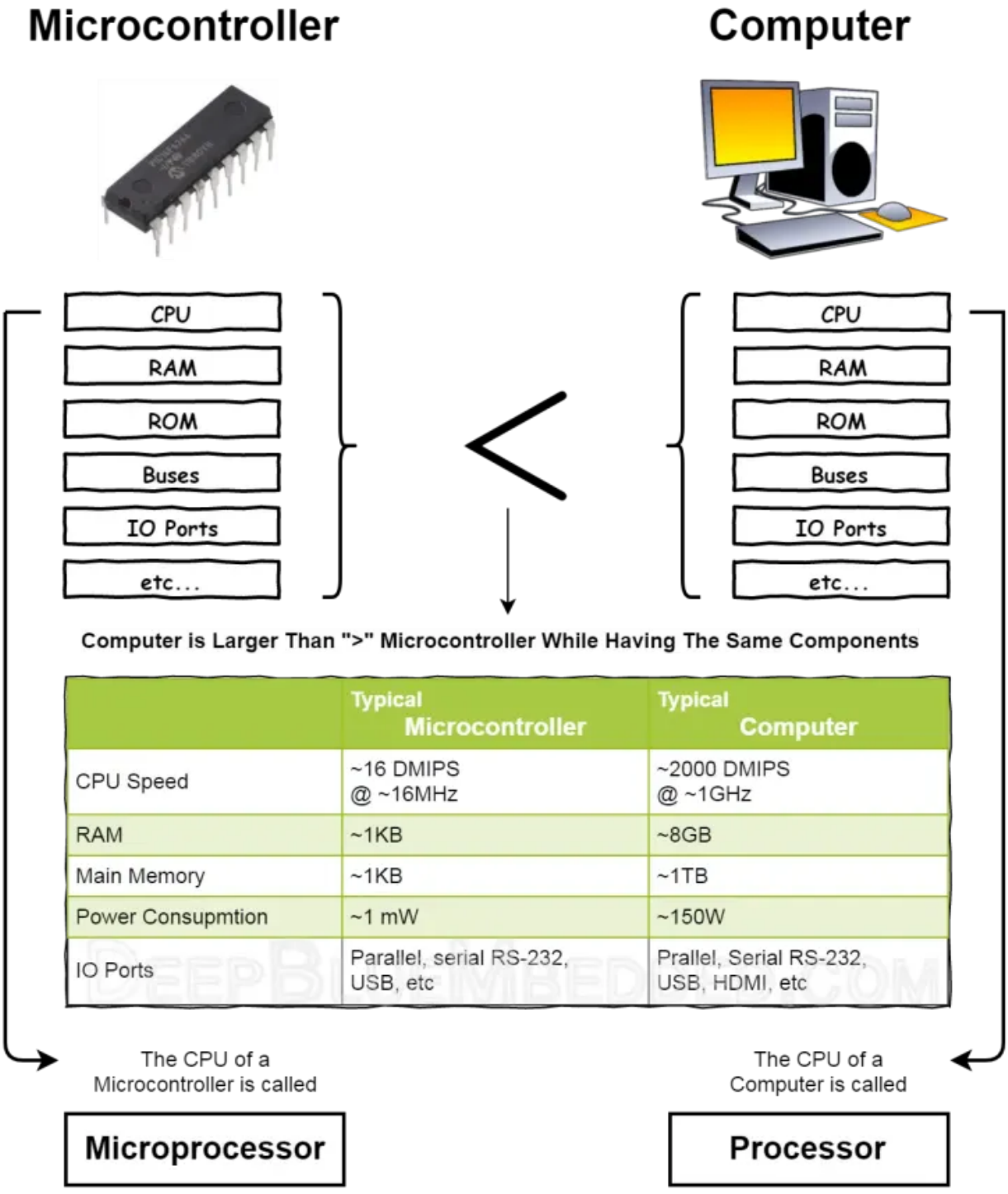


- Compiler take the Program file and compile it by using compiler **toolchain**
- Then Comipler give me the executable file(Ex: HEX, BIN, ...)
- MCU now can understand this code

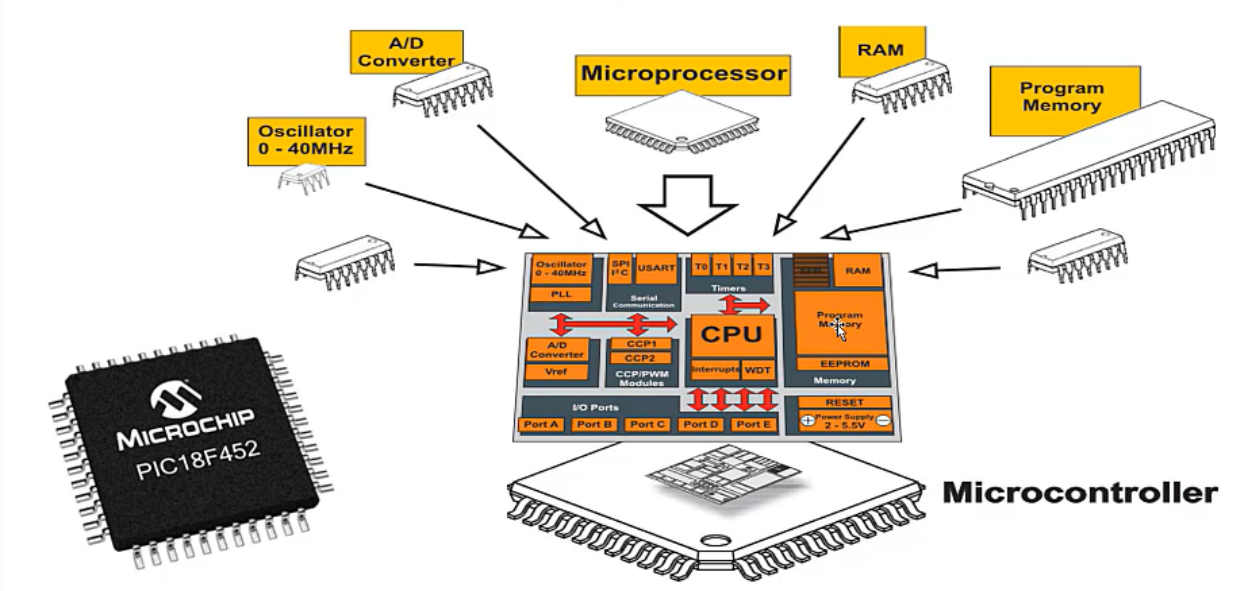


Microprocessor vs Microcontroller

- First we need to know the diiference betweeen Computer system and Microcontroller
- Microcontroller performs a **specific task** one at a time while computer performs **millions of instructions** at a time.


Microcontroller
Computer
CPU
RAM
ROM
Buses
IO Ports
etc...
CPU
RAM
ROM
Buses
IO Ports
etc...
Computer is Larger Than ">" Microcontroller While Having The Same Components
Typical Microcontroller
Typical Computer
CPU Speed
~16 DMIPS @ ~16MHz
~2000 DMIPS @ ~1GHz
RAM
~1KB
~8GB
Main Memory
~1KB
~1TB
Power Consupmtion
~1 mW
~150W
IO Ports
Parallel, serial RS-232, USB, etc
Prallel, Serial RS-232, USB, HDMI, etc
The CPU of a Microcontroller is called
Microprocessor
The CPU of a Computer is called
Processor
Low-End MCUs
High-End MCUs
Low-End Processors
High-End Processors
MIPS
20
100
>500
>5000
These Figures Are Not Exact By Any Means, I've Just Made Them Up For Demonstration Purposes Only!

- So **Microcontroller** is a CPU(**Microprocessor**) + Internal **Peripherlas**(RAM, ROM, ……)



- Any Embedded Systems Project must have some **Constraints** here is some of them

Characteristics of Embedded Systems Applications

- ✓ Single-functioned
  - Executes a single program, repeatedly
- ✓ Tightly-constrained
  - Low cost, low power, small, fast, etc.
- ✓ Reactive and real-time
  - Continually reacts to changes in the system's environment
  - Must compute certain results in real-time without delay
- ✓ Unit cost
  - The monetary cost of manufacturing each copy of the system, excluding NRE cost.
- ✓ NRE cost (Non-Recurring Engineering cost)
  - The one-time monetary cost of research, develop, design and test the system.
- ✓ Size
  - The physical space required by the system.
- ✓ Performance
  - The execution time or throughput of the system.
- ✓ Power
  - The amount of power consumed by the system.
- ✓ Flexibility
  - The ability to change the functionality of the system without incurring heavy NRE cost.

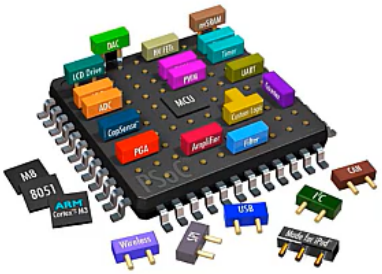
Characteristics of Embedded Systems Applications and Design Challenges

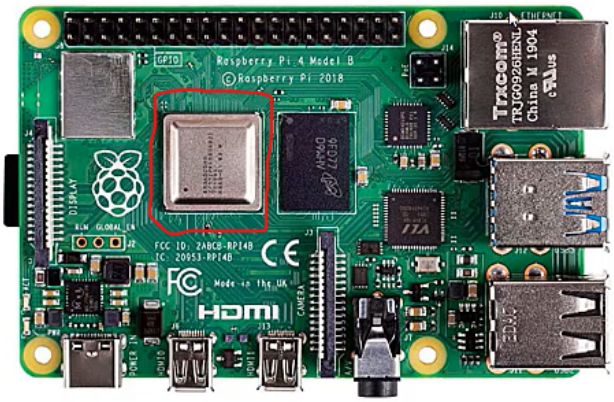
- ✓ Time-to-prototype
  - The time needed to build a working version of the system.
- ✓ Time-to-market
  - The time required to develop a system to the point that it can be released and sold to customers.
- ✓ Maintainability
  - The ability to modify the system after its initial release.
- ✓ Correctness, safety, Etc. .....

- Also we need to know **SoC(System On Chip)**

Characteristics of Embedded Systems Applications and Design Challenges

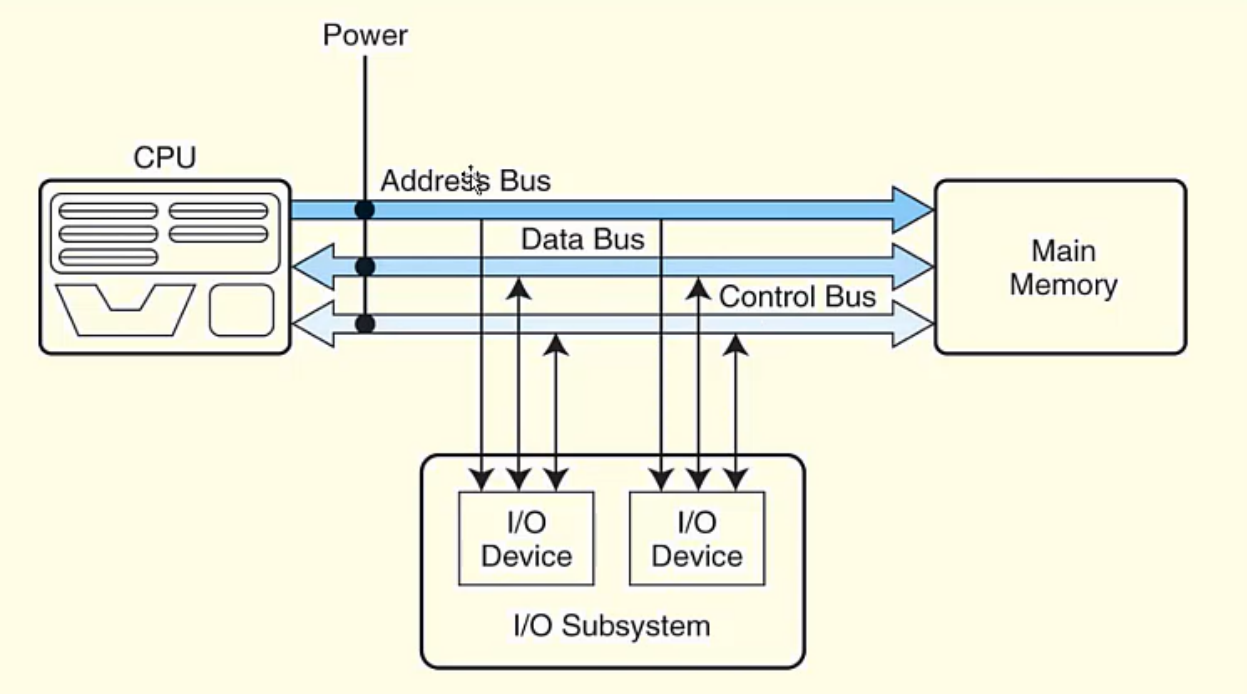
- ✓ An SoC "System On Chip", integrates almost all of these components into a single silicon chip.
- ✓ Along with a CPU, an SoC usually contains a GPU (a graphics processor), memory, USB controller, power management circuits, and wireless radios (Wi-Fi, 3G, 4G LTE, and so on).
- ✓ It's possible to build complete computers with just a single SoC.
- ✓ Applications
  - o Speech Signal Processing .
  - o Image and Video Signal Processing .





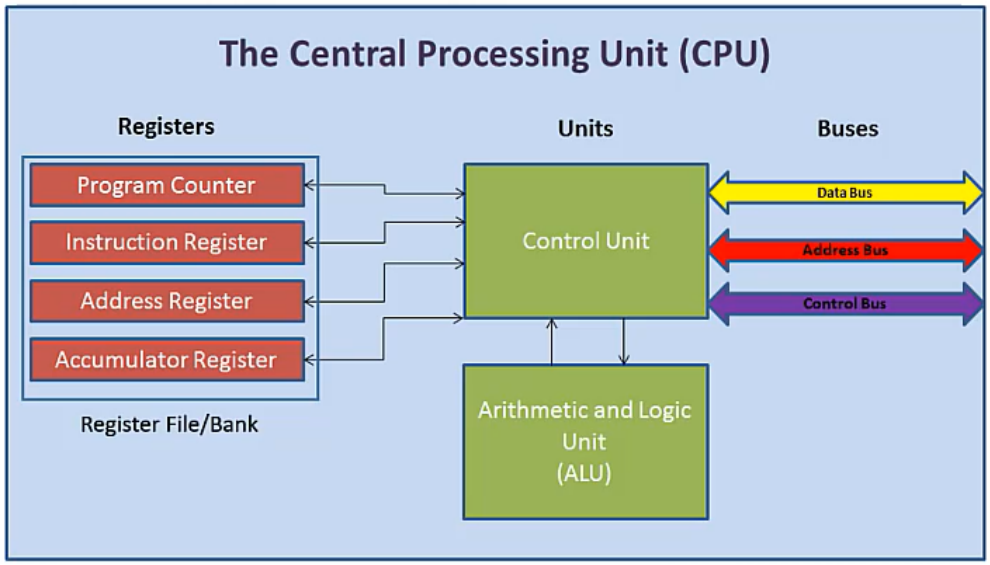
CPU and Main Memory

- What's CPU and Memory in deep
- How they communicate with each other



- CPU → execute code
- CPU = Control Unit + ALU + Register Bank + some Peripherlas

✓ CPU, is the "Central Processing Unit" of any microcontroller, the main purpose of it is to execute our code.
✓ In order to perform these functions, the CPU contains components

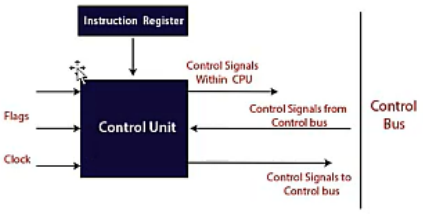


- After Flashing the program memory will have the execute file(0101101……)
- **Control Unit(CU)** : **Fetch** the execute file Instruction by Instruction then **Decode** it by **Instruction Set** and Send it to ALU

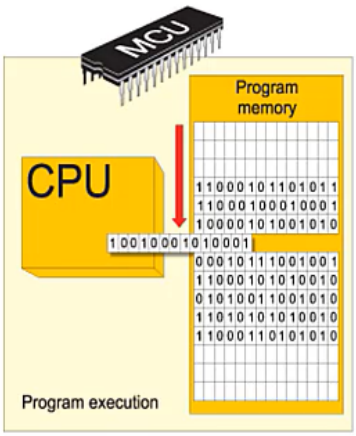
✓ The major components of the CPU:
  - Control Unit : CU
  - ALU : Arithmetic and Logical Unit
  - Register Bank/File : A group of special function and general purpose registers

✓ The Control Unit
  - Control unit consists of 2 major parts
    a) Fetch circuit
       - Fetch instruction from the program memory.
       - Each instruction is a pattern of zeros and Ones.

    b) Instruction Decoder "ID" circuit
       - It is used to control the operation of the processor.
       - Interprets or Decode instructions.
       - Signal the ALU to respond to the instructions.



Block Diagram of the Control Unit

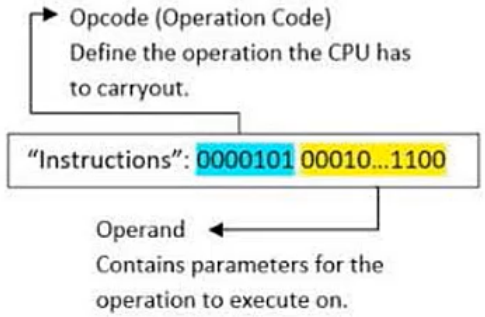


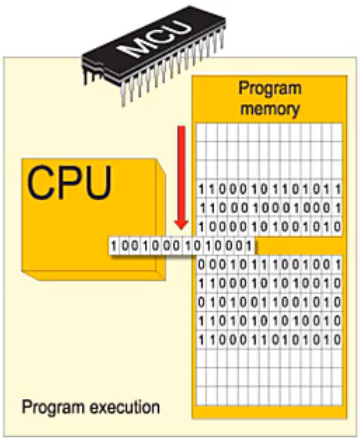
14

- ✓ Each CPU has its (Instruction Set) which is the group of instructions that could be executed by the CPU.
- ✓ Each Instruction has a unique "Opcode".
- ✓ It is the responsibility of the instruction decoder to decode the instruction.
- ✓ It is the responsibility of the Control Unit to signal the ALU to perform the needed instruction.

Instruction Set

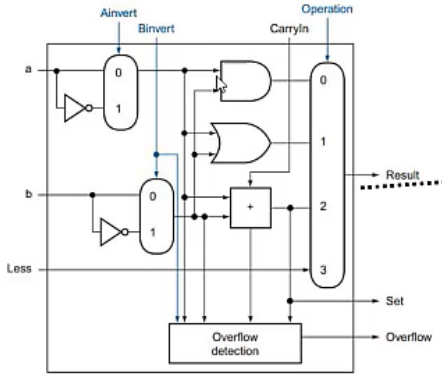
Instruction	Op Code
ADD	011
SUB	101
AND	110
OR	001
XOR	111
SHIFT	100

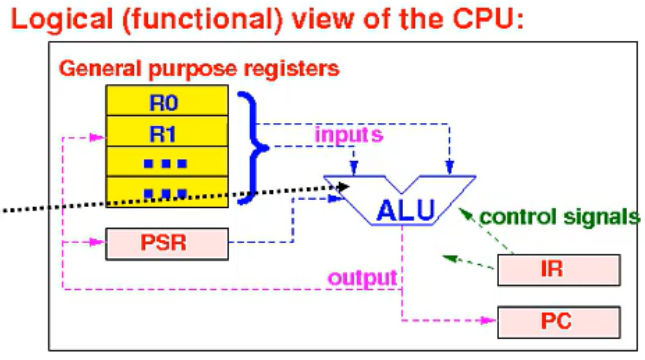




- **Arithmetic Logic Unit (ALU) :** take the decoded Instruction and Execute it and send it to the Program Memory

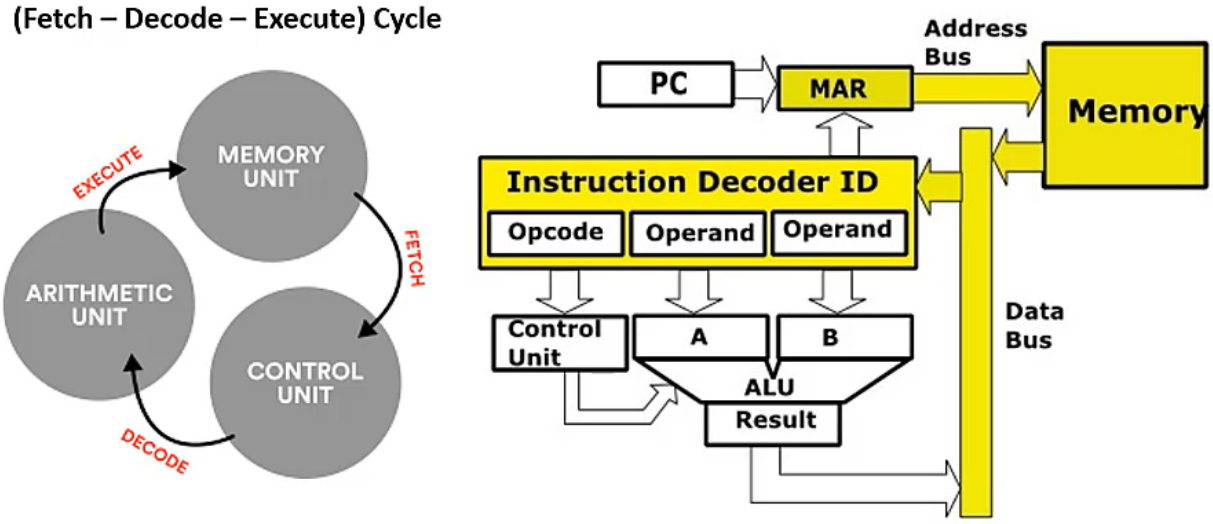
- ✓ The Arithmetic Logic Unit "ALU" performs all calculations, operating on binary data with arithmetic and logical functions.
  - ✓ Arithmetic operations include additions, division, exponentials, logarithms.
  - ✓ Logical functions (AND – OR) include comparisons (< > =) and transformations
- ✓ The signal from the control unit will activate special circuit in the ALU to execute the needed operation.





- Then Repeat this Cycle **Fetch** - **Decode** - **Execute**

✓ Once the instruction decoder decoded the instruction, the control unit signals the ALU to perform the needed operation on the operands.



Register Bank : Memory element internal the CPU to increase the speed of processing,

and all these register **dosn't have address** so if we want to access it we use assembly instructions.

GPRs : Hold variable of datatype **register** (Ex: **register unsigned char** var)

✓ What is the register ?
- A register is a high-speed memory storing units.
- A register is a memory device that can store 8, 16, or 32 bits values
- It is much faster to shift data to and from the registers rather than in and out of the cache or RAM (Random Access Memory) and so this speeds up the processing time.
- There are many registers available inside the CPU, some of which have a specific name and purpose.

✓ The internal register bank of register-based microprocessors consists of both general purpose and special purpose registers.

✓ There are two types of registers in the register bank:

a) General Purpose Register ➔ GPRs
   - Normally used to hold the data to operates on it.

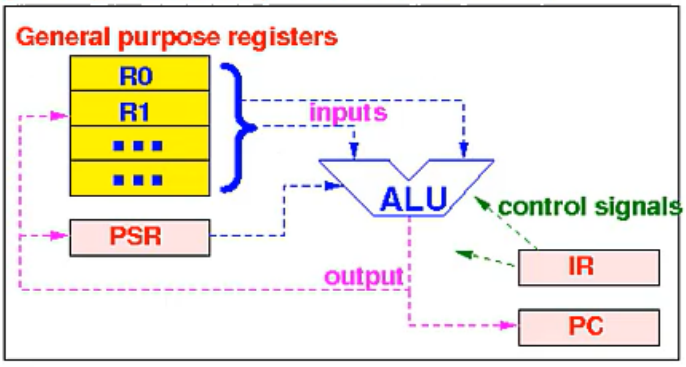
b) Special Purpose Register
   - Program Counter Register ➔ PC : Hold the memory address of the instruction to be executed.
     - Fetch circuit usually use this PC to know the instruction to be fetched each time.
   - Instruction Register ➔ IR : Hold the fetched instruction to be executed.
     - Instruction decoder decodes the fetched instruction stored in the IR.
   - Accumulator Register ➔ AC : Has the result of the operation done by the ALU.
   - Processor Status Word ➔ PSW : Hold the status of the last operation done by the ALU.
     - This register consists of a group of bits called "Flags" and each flag represent a specific status.
   - Stack Pointer ➔ SP : Hold the address of a special chunk of main memory used for temporary storage during program execution.
   - Link Register ➔ LR
     - This is used for holding the return address when calling a function or subroutine.

✓ General-purpose means that the register can be used in multiple ways.

✓ Example, suppose that you need to add to numbers.

- Move the fist number to the GPR : R0
- Move the second number to the GPR : R1
- Add the to registers R0 and R1
- The result will be in the AC register.
- The status of the operation will be in the PSR/PSW.

Logical (functional) view of the CPU:

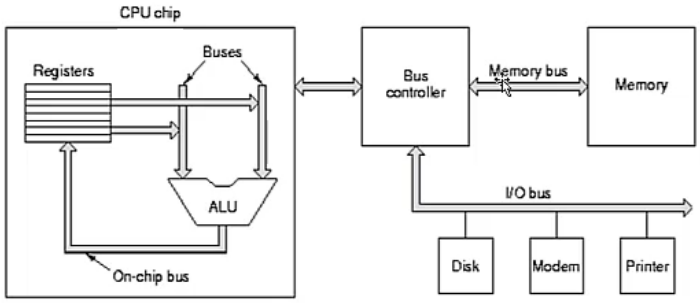


- know we need to know what's **BUS** to understand how these operation work

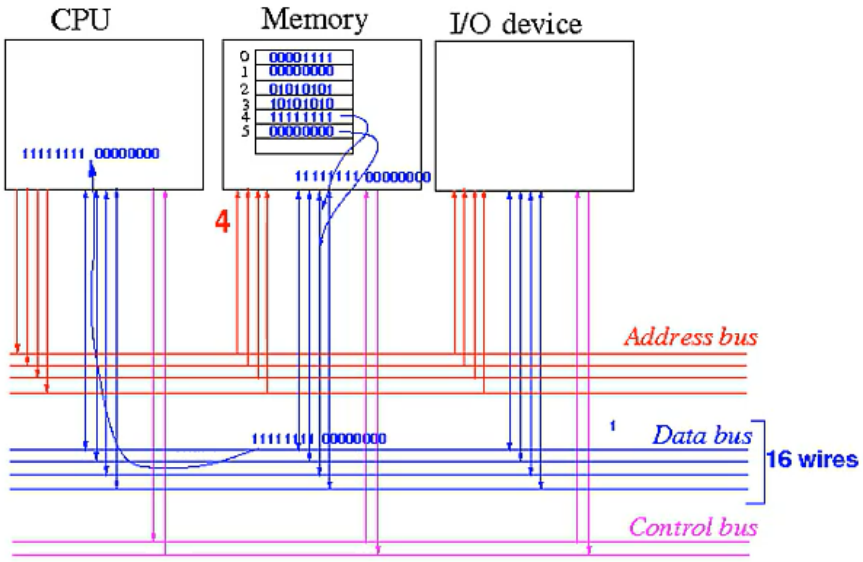
- ✓ A "Bus" is a common electrical pathway between multiple devices.
  - ✓ Can be internal to the CPU to transport data to and from the ALU.
  - ✓ Can be external to the CPU, to connect it to memory or to I/O devices.

- ✓ Early PCs had a single external bus or "System Bus".

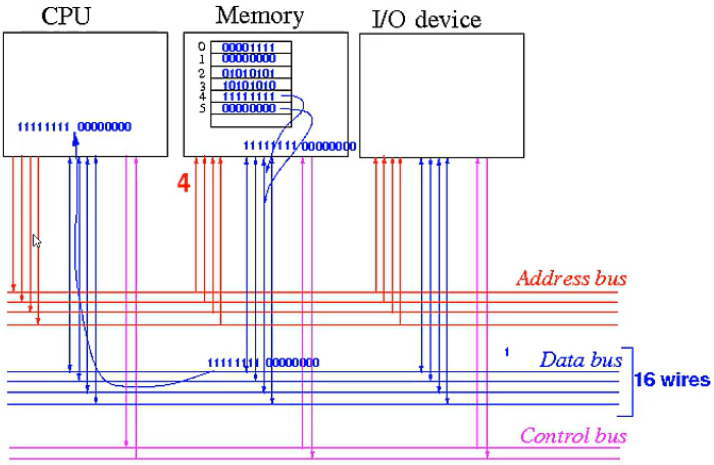
- ✓ Modern PCs have a special-purpose bus between the CPU and memory and (at least) one other bus for the I/O devices.



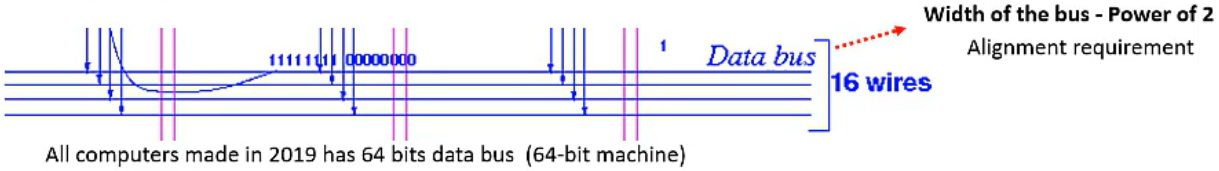
- ✓ There are three types of buses in any microcontroller
  - Address Bus
  - Data Bus
  - Control Bus



- ✓ The data bus is used to transfer the data bits (binary digits) between the CPU and memory.
- ✓ Example:
  - Data stored in memory address 4 (11111111 00000000) is transferred over the data bus to the CPU.
- ✓ Each bit of the data must be transferred on a different wire of the data bus
- ✓ When you send more than 1 electrical signal on a wire, it causes a short circuit.



- ✓ The more wires you use, the more costly the computer system will become, it's like the number of lanes in a high way system.



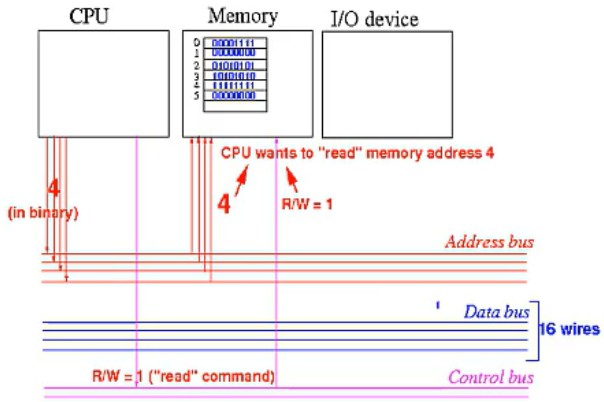
All computers made in 2019 has 64 bits data bus (64-bit machine)

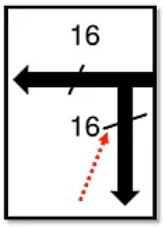
- ✓ A data bus that consists of 8 bits, can transfer 1 byte of data per reading / writing operation
- ✓ A data bus that consists of 16 bits, can transfer 2 bytes of data per reading / writing operation
- ✓ A data bus that consists of 32 bits, can transfer 4 bytes of data per reading / writing operation

- ✓ The width of the data bus determines the amount of data transferred per memory transfer operation.
- ✓ The wider the data bus, the more data you can transfer per time unit and this will result in a faster-running computer.

bit 31.....bit 0				
Address 0	byte 3	byte 2	byte 1	byte 0
Address 4	byte 7	byte 6	byte 5	byte 4
Address 8	byte 11	byte 10	byte 9	byte 8
Address 12	byte 15	byte 14	byte 13	byte 12

- ✓ The address bus is used to send the address (location) information to memory.
- ✓ Example:
  - When the CPU wants to read data stored in a memory location (address) 4, the CPU sends the value 4 on the address bus.
- ✓ Each wire of the address bus can transfer 1 bit of the address.
- ✓ Each memory byte is identified by a unique memory address.





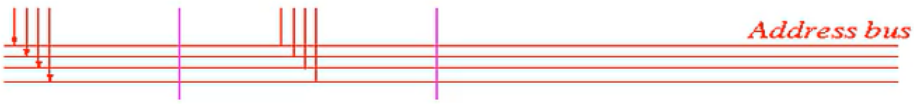
- ✓ An address bus that consists of 8 bits, can convey 2^8 = 256 different addresses.
- ✓ An address bus that consists of 16 bits, can convey 2^16 = 64K different addresses.
- ✓ An address bus that consists of 24 bits, can convey 224 = 16M different addresses.
- ✓ An address bus that consists of 32 bits, can convey 232 = 4G different addresses.

- ✓ The address bus does not have to be the same width as the data bus.

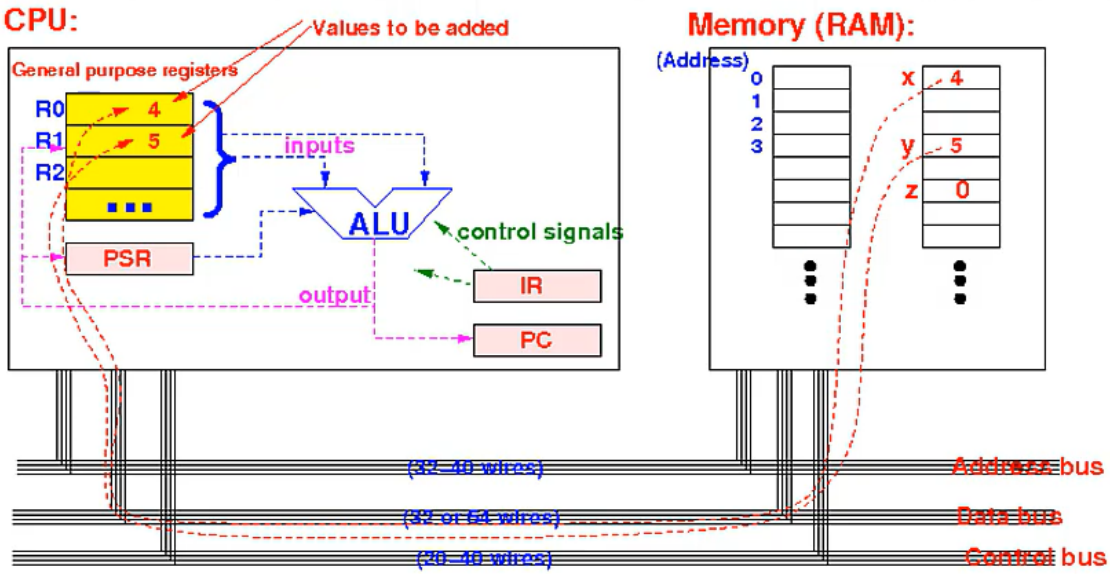
- ✓ The width of the address bus determines the size of the memory that the computer can use.

- ✓ The wider the address bus, the more memory a computer can use.

- ✓ More memory allows the computer to store more data and solve larger size problems.



✓ Example: We need to sum 2 variables variable (x) and variable (y) and store the result in variable (z)



✓ Example: We need to sum 2 variables variable (x) and variable (y) and store the result in variable (z)

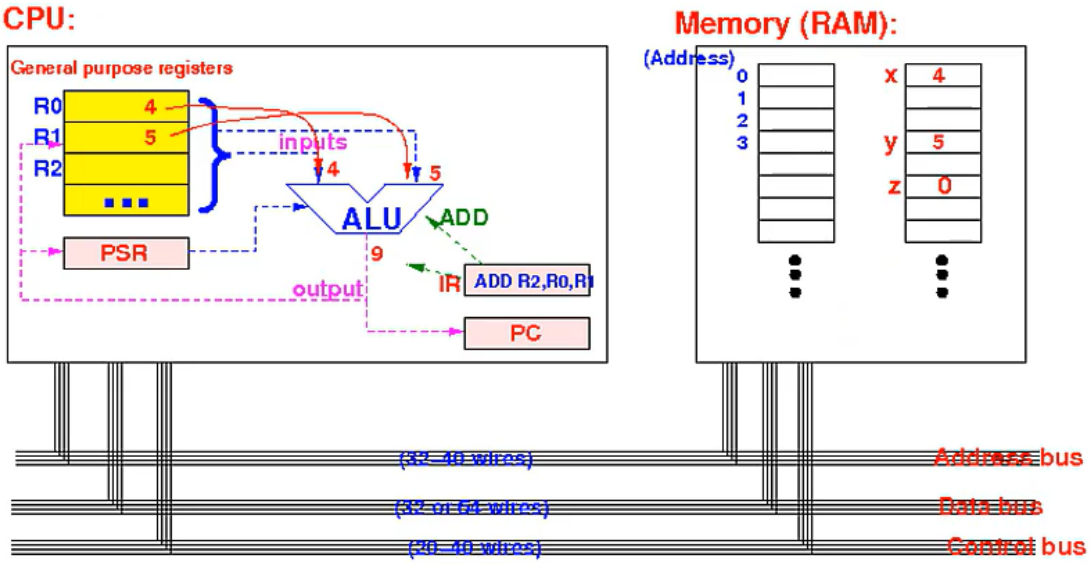
✓ Please refer the instruction set of the PIC microcontroller.

http://technology.niagarac.on.ca/staff/mboldin/18F_Instruction_Set/

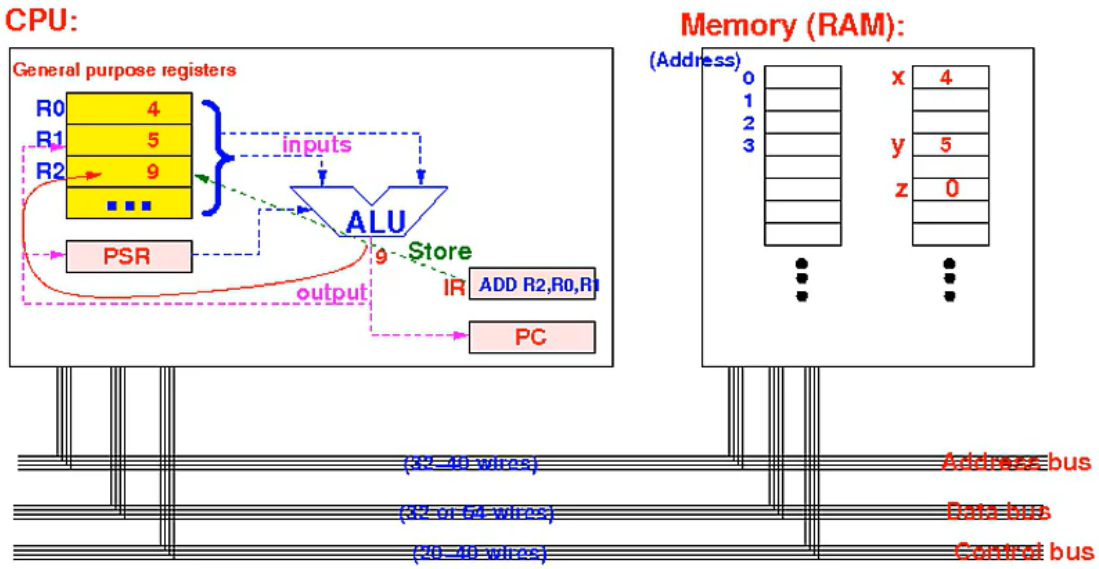
```c
int var1 = 0;
int var2 = 4;
int var3 = 5;

void main() {
    Var1 = Var2 + Var3;
}
```

```
2  _main:
3
4  ;MyProject.c,6 ::         void main() {
5  ;MyProject.c,7 ::         Var1 = Var2 + Var3;
6      MOVF         _var3+0, 0
7      ADDWF        _var2+0, 0
8      MOVWF        _var1+0
9      MOVF         _var3+1, 0
0      ADDWFC       _var2+1, 0
1      MOVWF        _var1+1
2  ;MyProject.c,8 ::         }
3  L_end_main:
4      GOTO         $+0
5  ; end of  main
```

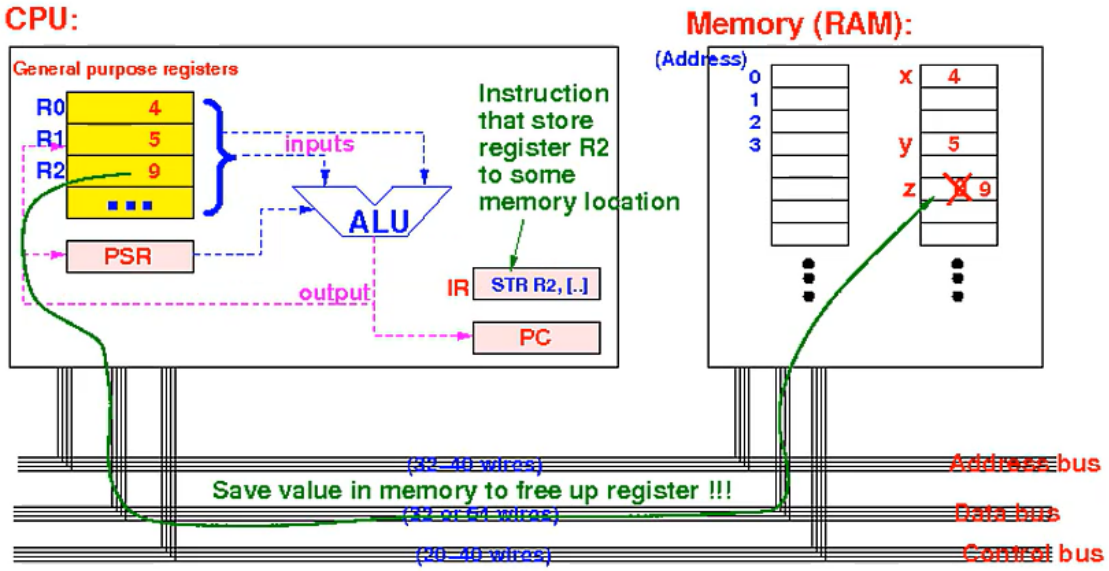
✓ Example: We need to sum 2 variables variable (x) and variable (y) and store the result in variable (z)



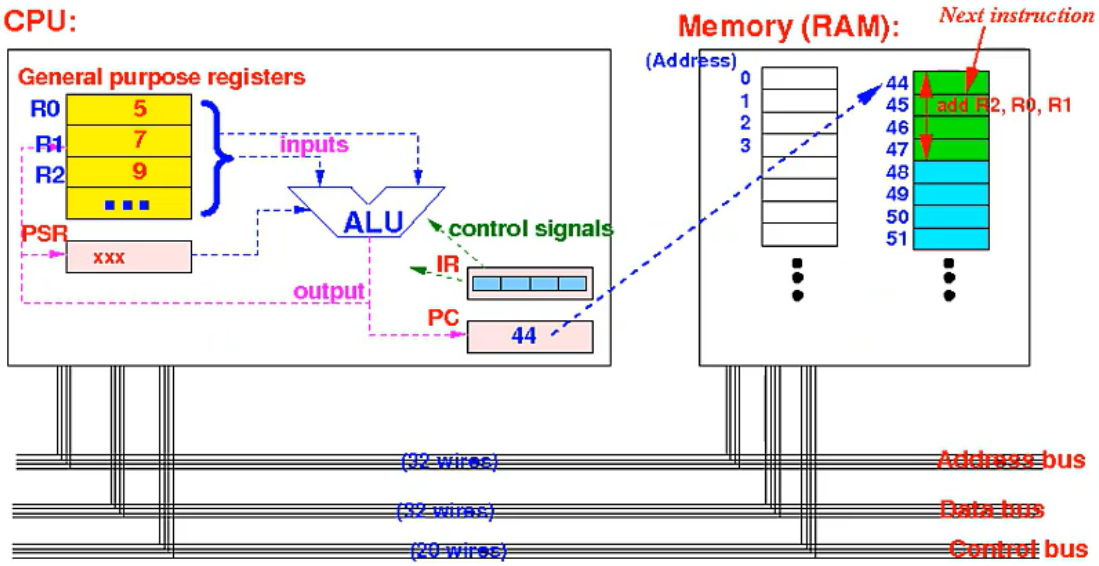
✓ Example: We need to sum 2 variables variable (x) and variable (y) and store the result in variable (z)



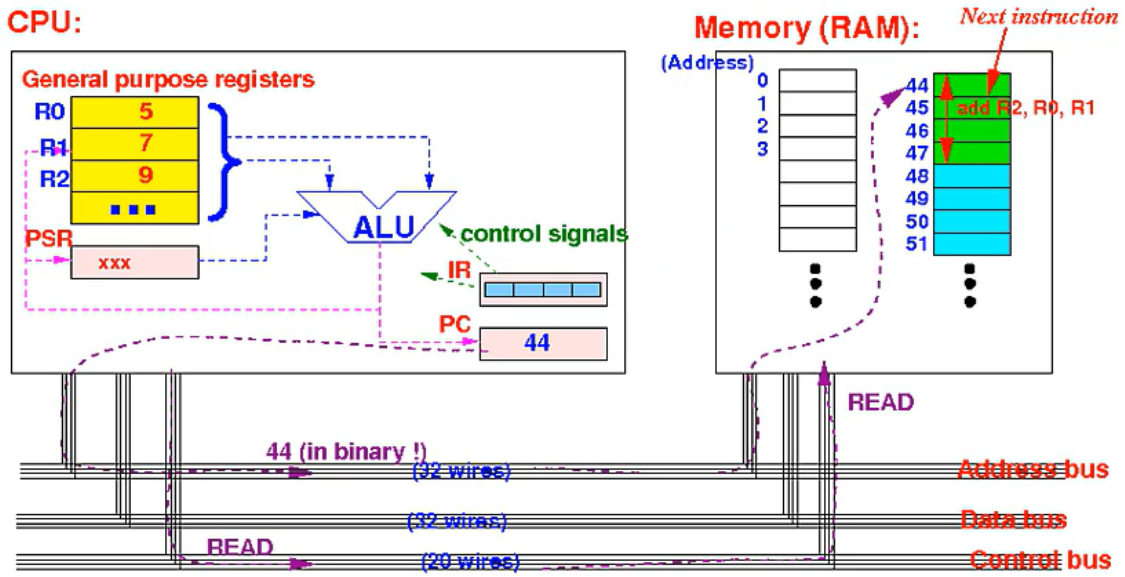
✓ Example: We need to sum 2 variables variable (x) and variable (y) and store the result in variable (z)



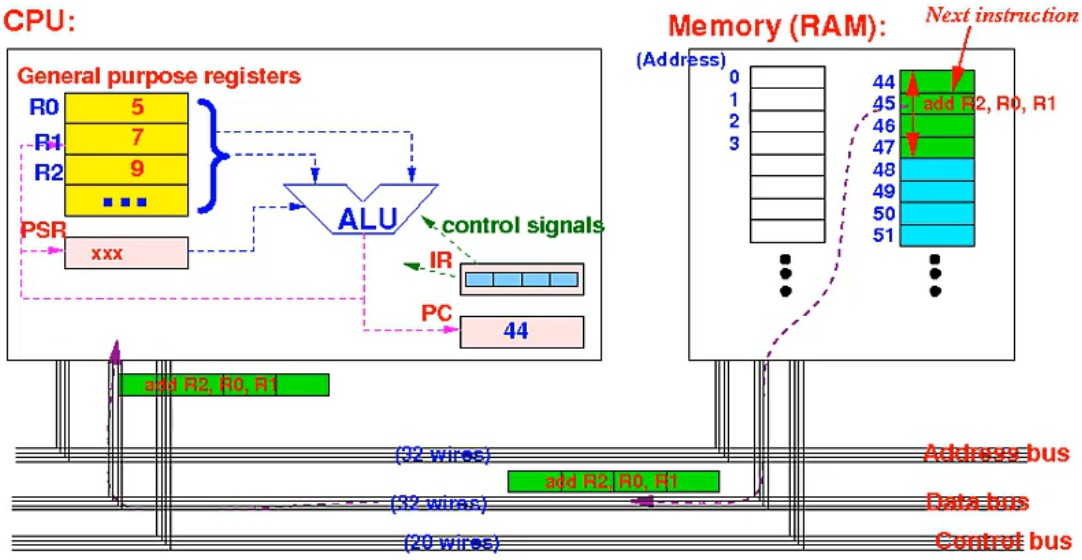
✓ Example: We need to sum 2 variables variable (x) and variable (y) and store the result in variable (z)



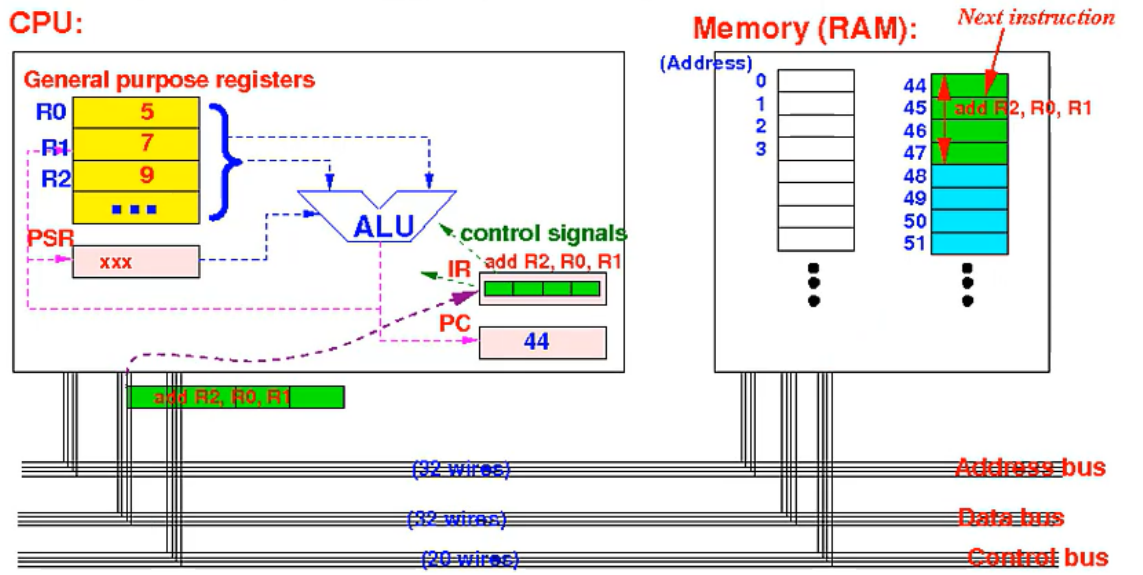
✓ Example: We need to sum 2 variables variable (x) and variable (y) and store the result in variable (z)



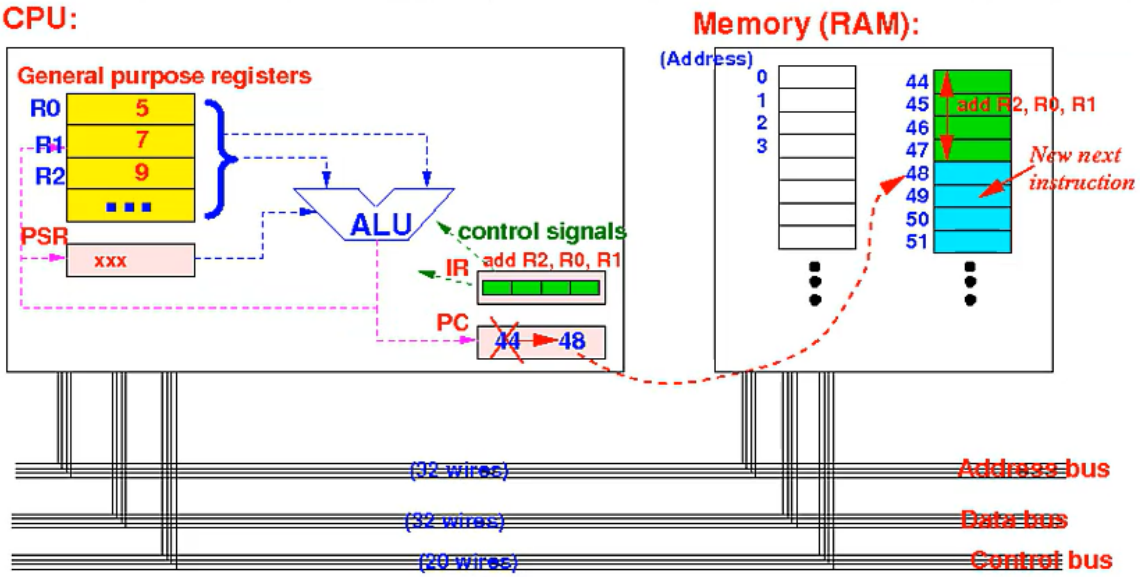
✓ Example: We need to sum 2 variables variable (x) and variable (y) and store the result in variable (z)



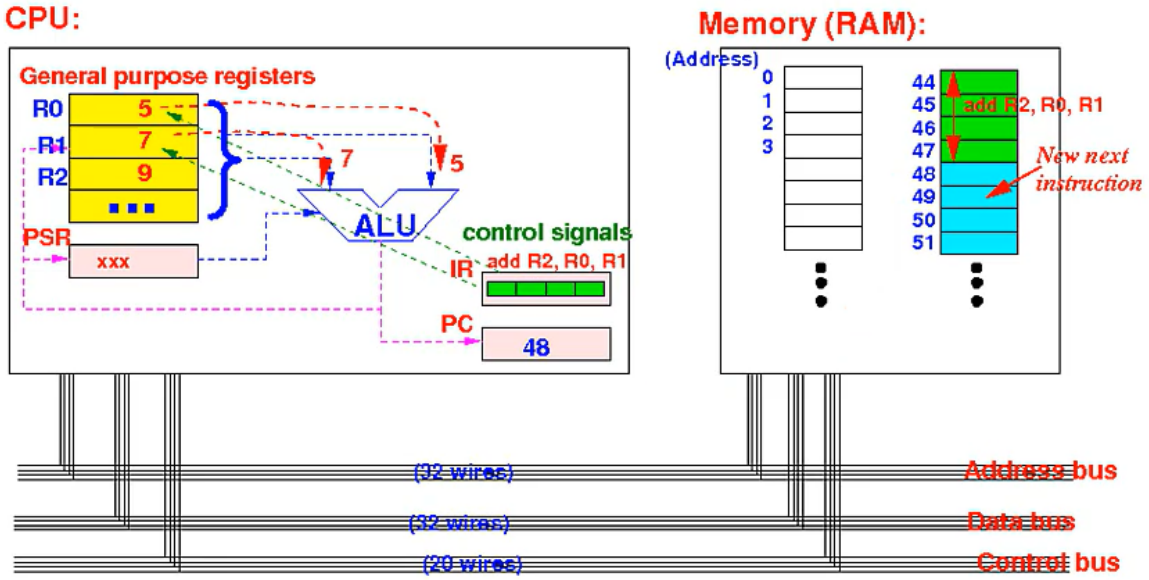
✓ Example: We need to sum 2 variables variable (x) and variable (y) and store the result in variable (z)



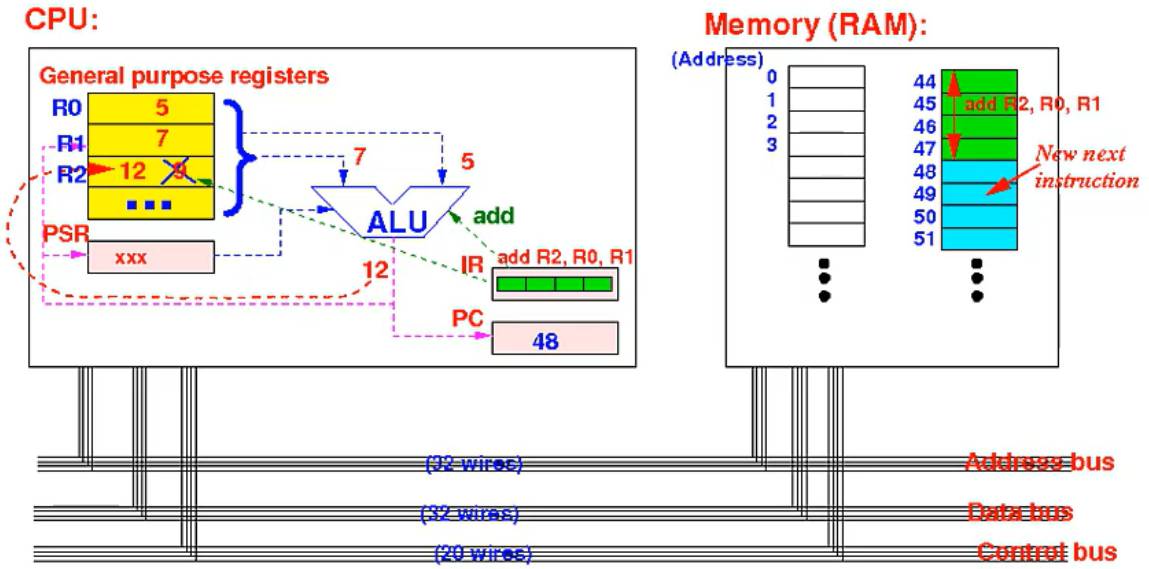
✓ Example: We need to sum 2 variables variable (x) and variable (y) and store the result in variable (z)



✓ Example: We need to sum 2 variables variable (x) and variable (y) and store the result in variable (z)



✓ Example: We need to sum 2 variables variable (x) and variable (y) and store the result in variable (z)



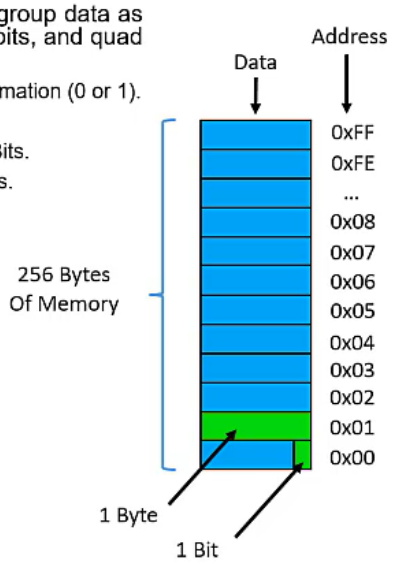
Memory Organization

✓ Introduction Memory Organization

- ✓ We need memory to store information that is used for binary code and group data as bytes, which contain 8 bits of information, double bytes, which have 16 bits, and quad bytes, which have 32 bits.
  - The "Bit" is the building block of memory and stores 1 piece of Boolean information (0 or 1).
  - A byte is 8 Bits and can store a binary value with a length of 8 Bits.
  - A double bytes are 16 Bits and can store a binary value with a length of 16 Bits.
  - A quad bytes are 32 Bits and can store a binary value with a length of 32 Bits.

- ✓ Memory Scale Examples:
  - Data Centers : Peta Bytes
  - Personal Computers : Mega Bytes – Tera Bytes
  - Embedded Systems : Kilo Bytes – Mega Bytes

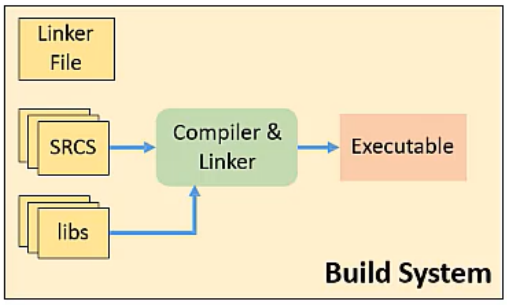
At a basic level, MCU contains volatile and non-volatile hardware components.

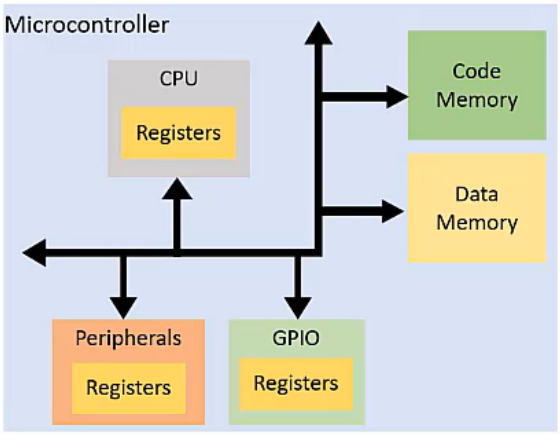


✓ Introduction Memory Organization

- Executable Program consists of program code and program data compiled for a particular architecture and platform.

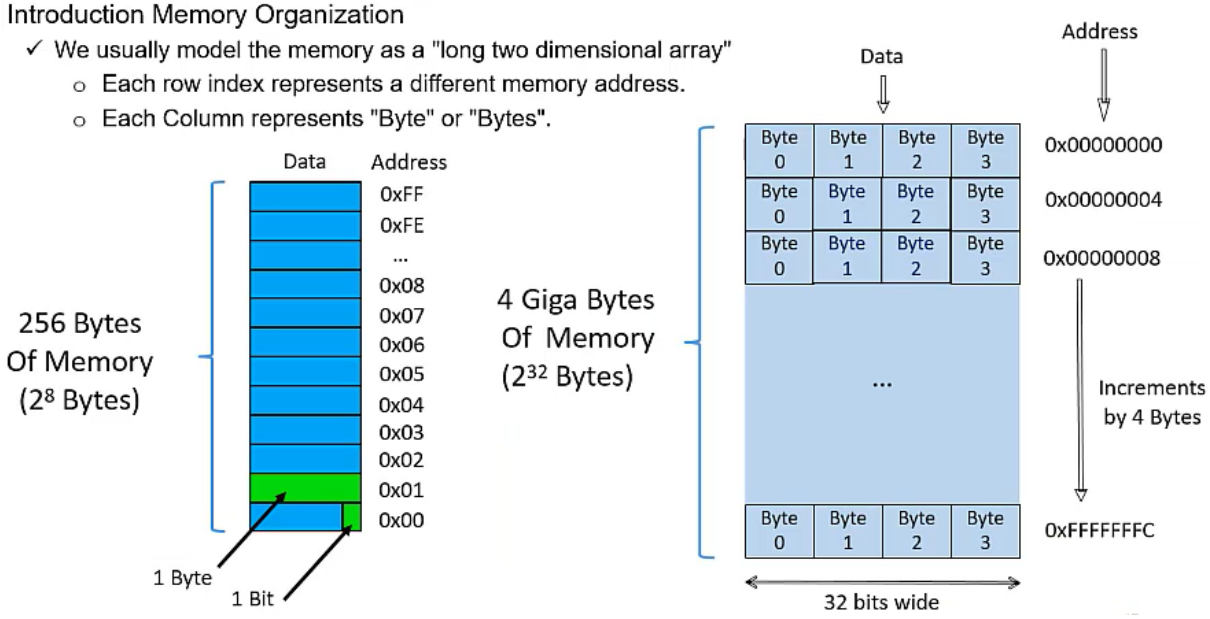
- Three types of storage needed for a program
  1) Code Memory
  2) Data Memory
  3) Runtime State of Program





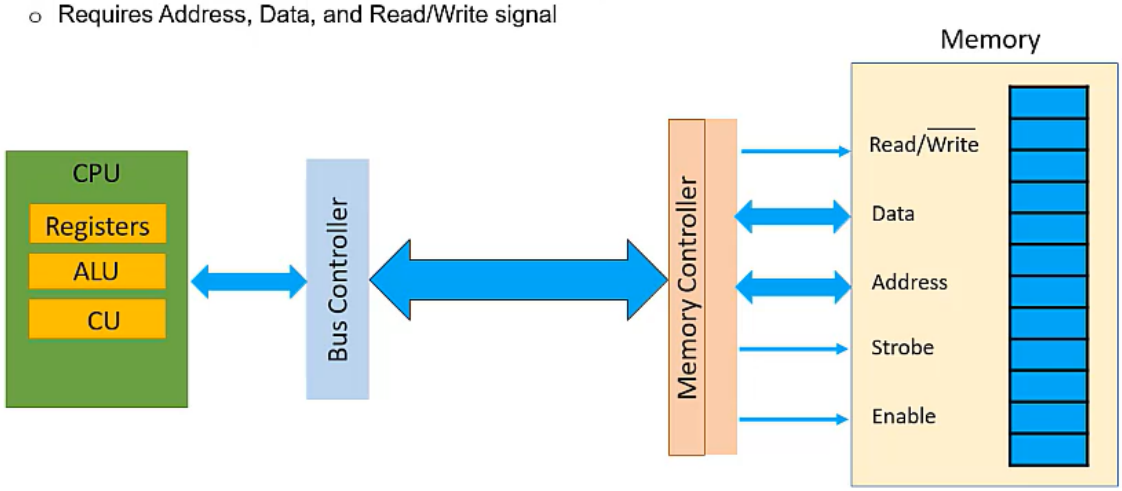
✓ Introduction Memory Organization

- ✓ We usually model the memory as a "long two dimensional array"
  - Each row index represents a different memory address.
  - Each Column represents "Byte" or "Bytes".



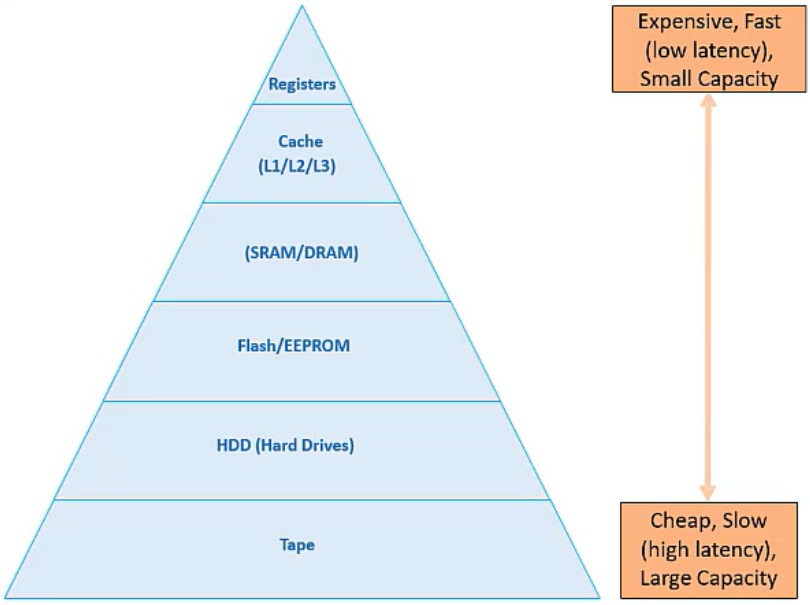
✓ Introduction Memory Organization

- ✓ CPU Reads or Writes data to Memory through Bus and Memory Controllers
  - Requires Address, Data, and Read/Write signal



✓ Introduction Memory Organization

✓ Memory Characteristics
- Capacity
- Volatility
- Access
- Power Consumption
- Latency
- Durability
- Transaction Size



✓ Introduction Memory Organization
- Capacity: the amount of storage a memory can hold
  - Embedded Systems do NOT need a lot of memory, they need optimized performance

↑ Capacity ≠ ↑ Performance

- Increasing capacity increases the complexity of design and size
  - Physical size and connection circuitry (potentially)

You want to LIMIT size, power & cost of system

↑ Capacity = ↑ Size = ↑ Power = ↑ Cost

- Volatility: The ability for memory to hold data without power
  - Volatile Memory – Loses data when power removed
  - Non-Volatile – Retains data when power is removed

- Endurance: Non-volatile semiconductor memories have a limited number of write-erase cycles before failure

- ✓ Types of Memory : The real difference between Volatile and Nonvolatile memories is **the speed/volatility**
  - Volatile memory : **Very fast, but it cannot hold data without power**
    - SRAM → Static Random Access Memory
      - Faster than the DRAM : **approximately 4 times faster.**
      - Since it needs more transistors per bit of data, it is also **more expensive** compared to DRAMs.
      - Earlier the SRAM was called just RAMs but later after the introduction of DRAMs, the term "static" got introduced into its name in order to differentiate it from the DRAM technology.
    - DRAM → Dynamic Random Access Memory
      - The reason behind its name comes from the fact that the **data stored in this RAM needs to be refreshed every few milliseconds or else it will end up being erased.**
      - Even if the power is being applied continuously the data still needs to be refreshed.
      - The reason behind this dynamic behavior is because of the capacitor present in its design.
      - This action is taken care of by a special device named **DRAM controllers.**
    - Embedded microcontrollers these days usually have a **mix of both SRAMs and DRAMs.**
    - SRAMs usually a small percentage of the total RAM (less than 10%), and it's the first primary storage that the processor sees.
    - Once the SRAM is full, the DRAM comes into the picture to store data.
    - If the hardware designer needs they can always more SRAM to their design to get more speed.

Why do we need Memory Stack in AUTOSAR ?

✓ Types of Memory : The real difference between Volatile and Nonvolatile memories is **the speed/volatility**
  - Volatile memory : **Very fast, but it cannot hold data without power**

	SRAM	DRAM
Construction Principle	It uses a cross-coupled flip flop configuration of transistors	It uses a capacitor transistor circuit to hold data
Cost	Relatively more expensive, it needs more transistors per bit of data it can store	Relatively less expensive, as fewer transistors per bit of storage are needed
Speed	4X more than DRAM	4X less speed
Volatility	As long as power is ON, it can store data since it uses no capacitors	Data needs to be continuously refreshed (usually in the order of 4 times a second) since the capacitors leak power.
Power consumption	Less	More
Addition components needed	None	DRAM controllers are needed to make it work like an SRAM. This controller offloads the data refreshing duties of a microprocessor and hence a DRAM coupled with a DRAM controller behaves more like an SRAM from the processor's perspective.
Application areas	Applications/scenarios that need very fast memory	Budget applications that do not need the speed of SRAMs

4

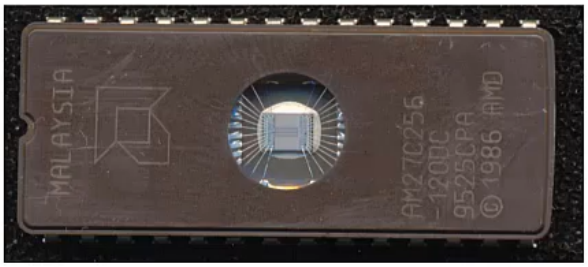
Why do we need Memory Stack in AUTOSAR ?

- ✓ Types of Memory : The real difference between Volatile and Nonvolatile memories is **the speed/volatility**
  - o Nonvolatile memory : **Does not care if the power is present or not, they are relatively slow**
    - o Masked ROM → Masked Read Only Memory
      - Data is written onto the device as it gets manufactured and it is impossible to change them.
      - This is done by designing the chip in such a manner so that it already contains the necessary data.
      - They usually serve the function of storing the bootloaders in microcontrollers and to store microcode on microprocessors.
      - This type of storage is generally used in mass-produced, long term devices where each penny counts.
      - The IC area per bit for masked ROMs is generally lower compared to other types.
      - Also since the area occupied is smaller, the overall device/chip's size is also reduced.

    - o PROM → Programmable Read Only Memory
      - These are programmable chips, the main characteristic being it can **only be programmed one time**.
      - It cannot be erased or reprogrammed.
      - They are also known as One Time Programmable devices or OTPs for short.
      - They are used to store the firmware and constants in the source code.
      - These are used on devices that are still mass-produced but have a relatively shorter life as compared to the devices that use Masked ROMs.
      - The shorter lifetime is generally due to the fact that the technology for this device is rapidly developing and they will be replaced by newer models on a yearly basis and hence not a good idea to actually design custom ROMs as the cost/benefit trade-off doesn't work in the manufacturer's favor.
      - Applications : home appliances like TV, washing machine and microwave ovens.

Why do we need Memory Stack in AUTOSAR ?

- ✓ Types of Memory : The real difference between Volatile and Nonvolatile memories is **the speed/volatility**
  - o Nonvolatile memory : **Does not care if the power is present or not, they are relatively slow**
    - o EPROM → Erasable Programmable Read Only Memory
      - These chips usually **have a small glass window on top and if you expose them to direct sunlight that will erase the chip's data.**
      - They can then be programmed again with fresh data.
      - Similar to PROM they are also used to store firmware and constants in the source code.
      - But this kind of chip is usually used in the development phase of the software as erasing and reprogramming the chips is a vital part of the development cycle.
      - Once the development is complete, the hardware designers usually change this chip to a PROM and hence the application areas and examples of PROMs apply also to the EPROMs



6

Why do we need Memory Stack in AUTOSAR ?

- ✓ Types of Memory : The real difference between Volatile and Nonvolatile memories is **the speed/volatility**
  - Nonvolatile memory : **Does not care if the power is present or not, they are relatively slow**
    - EEPROM → Electrically Erasable Programmable Read Only Memory
      - These chips can be **erased and reprogrammed using electricity** as opposed to exposing them to UV rays as EPROMs.
      - They can be used for 3 different functionalities
        1. Storing firmware during the development phase of the product
        2. Storing runtime constants after production.
        3. Storing updatable firmware after production.
      - Masked ROMs, PROMs, and EPROMs are true ROMs as they can only be read but EEPROMs are considered a hybrid between read-only and read-write memory as individual bits can be reprogrammed by the microcontroller and yet the data is non-volatile, unlike RAMs.
    - This quality is particularly useful in situations where even after the product reaches the customer's hand, the manufacturer may need to update the firmware and hence is a desirable quality.
    - Most of the microcontrollers includes EEPROM.
      - Mainly in Kbytes in 8-bit MCUs : 1K, 2k, max 4K
      - Mainly in Kbytes in 16-bit MCUs : 8K, max 16K
      - Mainly in Kbytes in 32-bit MCUs : max 32K
    - Mostly applications to store non volatile data during the runtime.

Why do we need Memory Stack in AUTOSAR ?

- ✓ Types of Memory : The real difference between Volatile and Nonvolatile memories is **the speed/volatility**
  - Nonvolatile memory : **Does not care if the power is present or not, they are relatively slow**
    - FLASH
      - This memory is the most popular one that you might come across on a day to day basis as an embedded developer.
      - It has all the qualities of EEPROMs except one.
        - ❑ **EEPROMs are reprogrammable one bit at a time, while flash storage is a reprogrammable one block at a time.**
      - The block size (X number of bytes) usually can be found on the given chip's specifications.
      - The main advantage of flash memory is that it is cost-effective (per byte) compared to EEPROMs and used for storing large files.
      - Usually, microcontrollers, these days use them for storing firmware of large sizes and other constant data and large lookup tables as needed by the application.
    - NVRAM → Non Volatile Random Access Memory

      - This is a special type of RAM that can store data permanently.
      - It's basically **an SRAM with a power supply** (usually a coin cell like CR2302).
      - This type of storage on the PC world has a special name called "RAM disks".
      - Imagine the speed of a RAM and non-volatility of data like ROM combined together, that is exactly what this type of storage aims at accomplishing.
      - The downside is that SRAM per byte is very expensive compared to any of the types mentioned.
      - This type of RAM is usually used in applications where startup time is extremely important, and we cannot afford to lose even microseconds of time trying to load a program from slower storages like flash or EEPROM.

Why do we need Memory Stack in AUTOSAR ?

- ✓ Types of Memory : The real difference between Volatile and Nonvolatile memories is **the speed/volatility**
  - o Nonvolatile memory : **Does not care if the power is present or not, they are relatively slow**
    - o NVRAM → Non Volatile Random Access Memory
      - They are usually accompanied by a backup memory (that stores the original code and compile-time data) of one of the types about so that when the time comes to replace the battery, all the runtime data can be backed up to another(slower) non volatile memory (like EEPROM or flash memory).
      - Once the battery is replaced, it can load the code and data from this memory and start working again from where it left off.
      - Examples include anything that can be done using EEPROMs and flash memories.

Interview Questions : Types of memory ☺

Why do we need Memory Stack in AUTOSAR ?

- ✓ Flash memory technologies
  - a) NAND Flash memory ➔ Used in FLASH
    - o Offer a more compact physical architecture.
    - o NAND memories are available in greater storage densities.
    - o Lower costs per bit than NOR-flash.
    - o NAND memories also have up to ten times the endurance of NOR-flash.
    - o NAND is more fit as storage media for large files including video and audio.
    - o NAND-flash has faster erase and write times compared to NOR Flash.
    - o Examples: USB thumb drives, SD cards and MMC cards.
  - b) NOR Flash memory ➔ Used in EEPROM
    - o NOR-flash memories offer complete address and data buses to randomly access any of its memory location (addressable to every byte).
    - o NOR memories endurance is 10,000 to 100,000 erase cycles.
    - o NOR-flash memories are slower in erase operation and write-operation compared to NAND-flash.
    - o NOR-flash can read data slightly faster than NAND.
- ✓ According to that (NAND Flash memory ➔ Used in FLASH) used more than the EEPROM in the automotive projects.
  - ✓ Mostly used path to store data : Application → RTE → NvM → MemIF → FEE → FLS