MAR MBR → Data to be read/written

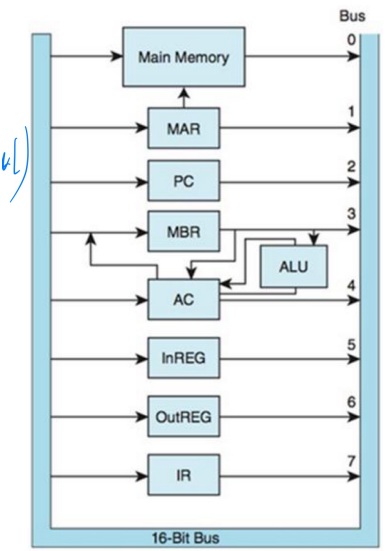
PC IR → next instruction to be executed

PC → Address

IR → Data

AC → holds Data to be processed by CPU

ALU operations Add, sub etc done in AC



Need to understand this

as parallel operations in store I don't understand

Fetch decode execute

1) fetch instruction from memory
2) decode instruction using opcode
3) execute performing the operation

CPU time = seconds/program

To improve CPU time

* Reduce Number of instructions
* Reduce Number of cycles per instruction
* Reduce Number time per cycle

* Bus: Set of wires that simultaneously convey a single bit along each line

Types of Buses: point to point / Multipoint
Connects 2 specific point — Connects multiple devices

Buses Consist of:
- Data lines: send Data between Devices
- Control lines: Determine Direction of Data flow (permission)
- Address lines: Determine Location of source and destination

---

* CPU Basics: Fetch Decode execute program instructions

2 principal parts:
- Datapath: ALU and registers interconnected by Data bus
- Control unit: Responsible for creating sequenced operations

* Registers are made using D flip flops

* Control unit:
  * Monitors the execution of All instructions
  * Extracts instructions from memory
  * it uses a program Counter register to find the next instruction to be executed

* Clock:
  * At least 1 clock is used to synchronize the components activities
  * Clock frequency exp: 2.5 GHz
    means the 2.5 G cycles are done in 1 second

MARIE

The MARIE architecture has the following characteristics:

- ❑ Binary, two's complement data representation.
- ❑ **4K words** of **word-addressable** main memory. 4K x 16
- ❑ **16-bit** data words.
- ❑ **16-bit** instructions, **4 for the opcode** and **12 for the address**.
- ❑ A **16-bit** arithmetic logic unit (**ALU**).
- ❑ **Seven registers** for control and data movement.

MAR MBR
PC IR
↓ ↓
Addrss Data

7 Registers:

1) MAR : Memory Address Register

holds the Address of the Data being Referenced

2) PC : Program Counter

holds the Address of the next Instruction to be executed

3) MBR : Memory Buffer Register

holds the Data read from memory or to be written

4) AC : ACCumulator

holds the data that needs to be processed by CPU [Mathematical / Logical operations]

5) IR : Instruction Register

holds the next instruction to be excuted

6) InReG : Input Register

holds Data from input Device

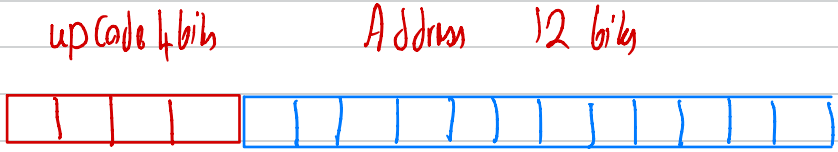
7) outReG : output

holds Data from output Device

MARIE Instruction Set Architecture:

* specifies the instructions that a computer can perform and the format

format



4 bits $0 \rightarrow 2^4 - 1$ Max 16 Instructions

Instruction Number			
Binary	Hex	Instruction	Meaning
0001	1	Load X	Load contents of address X into AC.
0010	2	Store X	Store the contents of AC at address X.
0011	3	Add X	Add the contents of address X to AC.
0100	4	Subt X	Subtract the contents of address X from AC.
0101	5	Input	Input a value from the keyboard into AC.
0110	6	Output	Output the value in AC to the display.
0111	7	Halt	Terminate program.
1000	8	Skipcond	Skip next instruction on condition.(While, If)
1001	9	Jump X	Load the value of X into PC.

→ Data

* M[x] notation is used to indicate Data stored at address x

Load x : Load Contents of address x into AC

MAR ← X
MBR ← M[MAR] , AC ← MBR

Store x : Store Contents of AC into Address x

MAR ← X
MBR ← AC , M[MAR] ← MBR

Add x :

MAR ← X
MBR ← M[MAR]
AC ← MBR + AC