Computer Architecture

Date / /

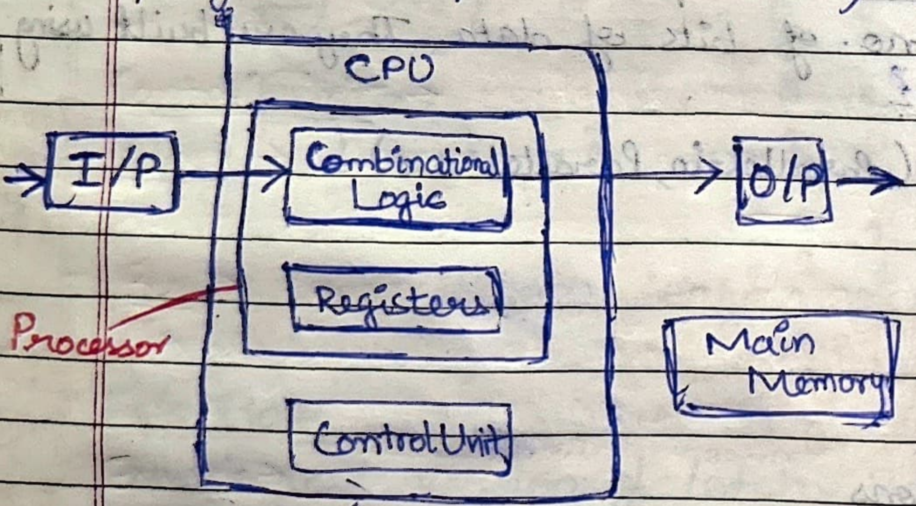
Introduction

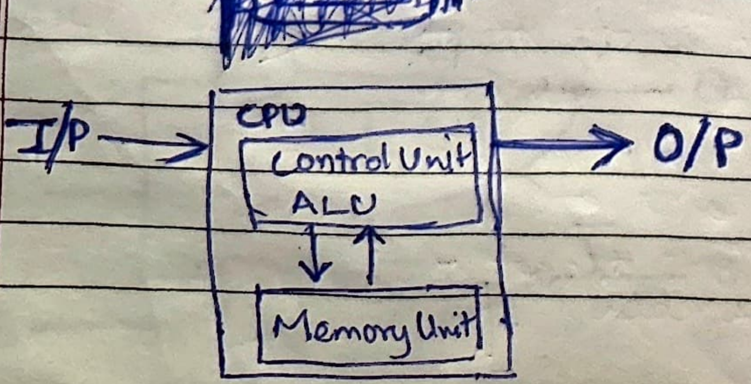
Claude Shannon was the one who took boolean algebra & used it in electronics to build logic gates.

2 types of Computer → Embedded : Very specific Ex - Washing machine
→ General Purpose

Concept of turing & implementations by Claude Shanon was brought together by Von-Neumam.

Modern computers are based on stored program concept introduced by John Von Neuman



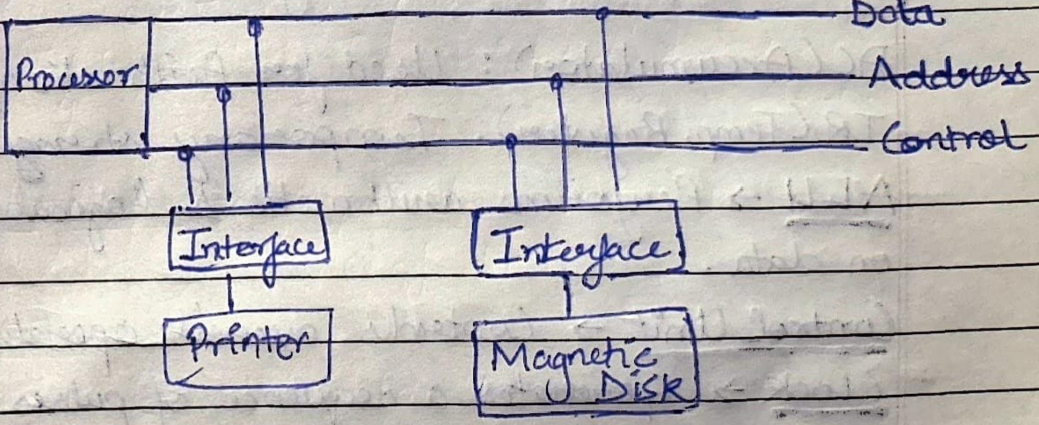


Buses

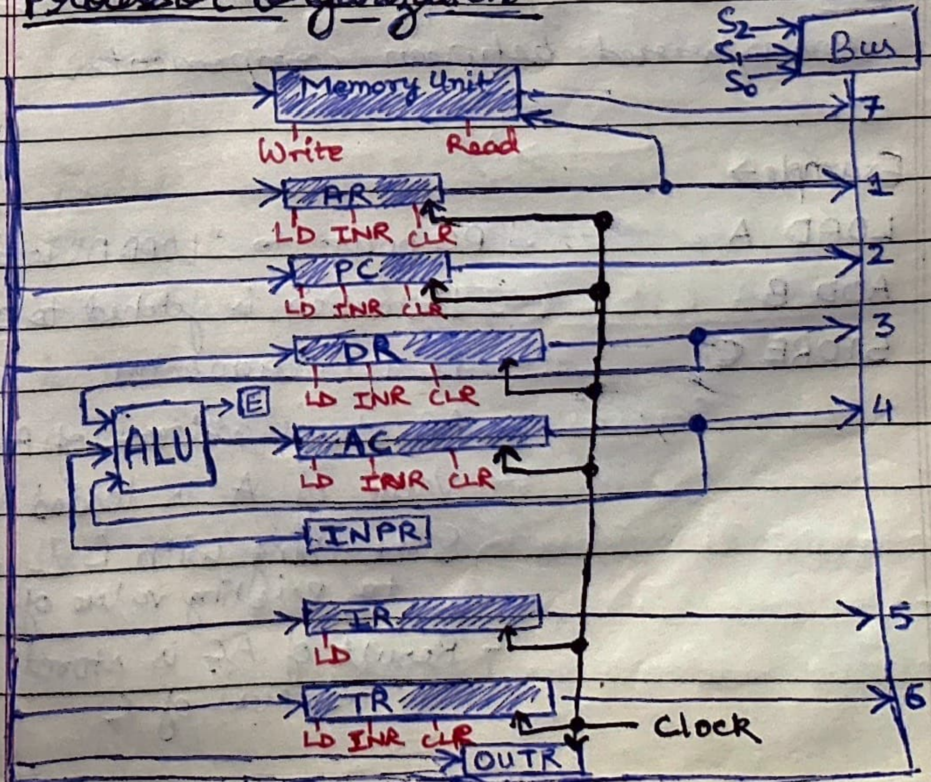
Address Bus → Used to identify correct I/O devices, so CPU put an address of specific I/O device & the devices will decode & match.

Control Bus → After selecting I/O, CPU sends functional code on control line

Data Bus → In final step, data will go to or from CPU & I/O device



Processor Organization



Memory Unit → Data & instructions are stored

Registers → Small, high-speed storage locations in CPU

PC (Program Counter): Holds address of next istruction

AR (Address Register): Temporarily stores address of memory location.

DR (Data Register): Holds data that is being transferrded b/w CPU & memory

IR (Instruction Register): Holds current instruction being executed.

AC (Accumulator): Used for Arithmetic operations

TR (Temp. Register): Temporary storagage.

ALU → Performs arithmetic & logical operations on data.

Control Unit → Controls overall operations

Clock → Generates a sequence of pulses to synchronize operation of processor.

Bus → Common path for data to be transferred between components.

Example→

```
LOAD A
ADD B
STORE C
```

- PC points to "LOAD A" instruction
- Instruction is fetched to IR
- PC is incremented.
- AR stores address of A
- Value of A is loaded to AC
- Same thing with B & added to existing value of AC
- Result of AC is stored at address of C

Addressing Modes

Define how operands (data) are located in memory.

Types →

→ Direct Addressing : Operand address is specified in instruction
Ex: LOAD 100 (load value at memory location 100)

→ Indirect Addressing : Address of operand is in register
Ex: LOAD (R1) (load value at memory loc. specified in R1)

→ Indexed Addressing: Address calculated by adding base address to index value.
Ex: LOAD 100(R2) (load value at memory location 100 + R2)

→ Register Addressing: Operand is in Register
Ex: ADD R1, R2 (Add values in R1 & R2)

→ Immediate Addressing: Operand is in instruction
Ex: ADD #10 (Add value 10)

Ex → Array access might use indexed addressing, pointer arithmetic can use indirect addressing

Instruction Set Architecture

It defines set of instructions the processor can execute. It acts as interface b\w H\w & S\w, specifying operational capability, instuction format, registers & data types.

Ex → x86 architecture

Features→

- Data types : Supports bytes, word (16 bit), 32 & 64 bit
- Registers : Includes general purpose (EAX, EBX), segment & special-purpose register
- Addressing Modes : Supports immediate, direct, indirect, register & indexed addressing

- Example instruction →

```
MOV EAX, 1;
ADD EAX, EBX;
JMP 0x00400000;
```

Control Unit

Fundamental component of a CPU whose primary role is to manage & coordinate operations of the CPU, directing flow of data & instructions b\w CPU & other parts of computer system.

Types → Hardwired (fixed logic gates for implementation, very fast)

→ Microprogrammed (used instructions stored in memory)

RISC & CISC (ISA Encodings)

RISC → Focuses on simplifying instructions executed by CPU

- Simple Instructions
- Uniform Instruction Length
- Few Addressing Modes

- Load/Store Arch. → Only Load & Store can access memory.
- High Performance

Ex → ~~Loa~~

```
LOAD R1, A;
LOAD R2, B;
ADD R3, R1, R2;
STORE R3, C;
```

CISC → Aims to reduce no. of instructions per program.

- Complex Instructions
- Variable Length Instructions
- Many Addressing Modes
- Many Instructions

Ex → ADD A, B, C;

—-x-—