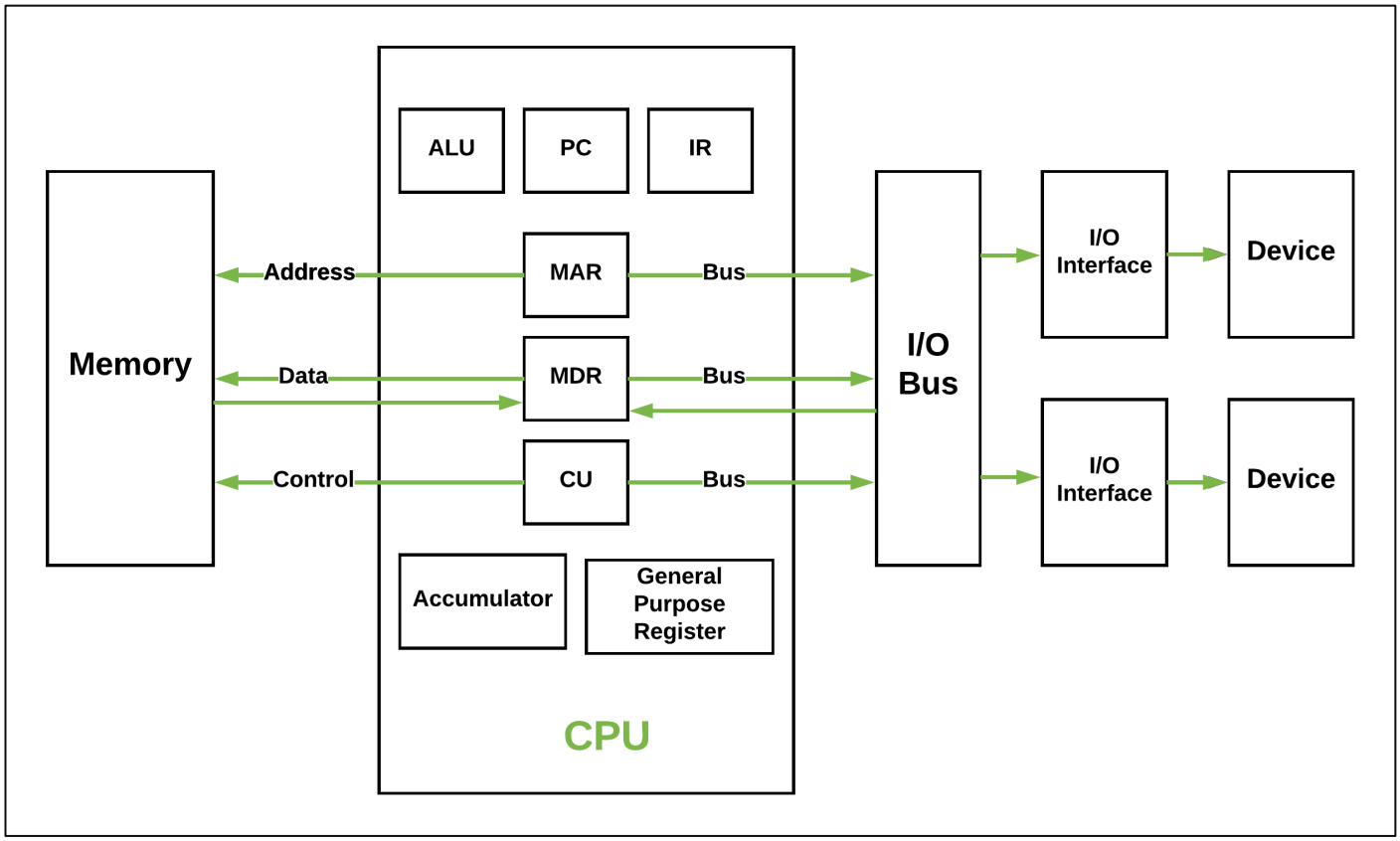
Computer Architecture - Lecture 1

- Vonn Neoman Model:



- Data bus and control bus are **bidirectional**.
- I/O interface is managed through a **controller**, it is slower than the main memory.
- Processor is faster than memory.
- Main memory is divided into **bytes**, each byte has an **address** and **control**.
- Processor can **read** and **write** on memory.
- To access a bit, the address is passed on **the address bus**.
- To **read** a byte:
  - CPU sends its address through the address bus.
  - CPU sends a read signal through the control bus.
  - CPU waits the receives the byte through the data bus.
- To **write** a byte:
  - CPU sends its address to the address bus.
  - CPU sends the data to the data bus.
  - CPU sends a write control signal.
  - Waits until memory function completed.
- Interrupt service routine is a software routine that hardware invokes in response to an **interrupt**, it moves data into a memory **buffer** and then return to what it was doing.
  - By looping (same as in factories).
  - By causing interrupts (same as our computers).
- Main memory is used to store **programs** and **data**.

- A program is a sequence of **instructions** (command to form simple operation).
- The instruction is **binary coded** (machine code).
    - Ex: C = A + B
        - Load A (instruction)
        - Load B
        - Add A and B
        - Store result in C
- Organization of a program:
    - Code segment (contains functions).
    - Data segment (contains variables).
    - Stack segment (contains local variables).
- 100 → jump 300 (skip the data from 100 and jump to 300) (data and code together).
- Compiler changes **assembly** code to **machine** code
- **Main Memory**:
    - RAM [volatile] → Random access memory (is used when generating data)
        - Sequential (ex. magnetic tape)
        - Direct (ex. hard disk)
    - ROM [permanent] → contains the basic input/output system [BIOS] program.
        - Self-test.
        - I/O routine setup.
        - Bootstrap loading. (loading: copying for secondary storage to RAM)
            - OS is loaded in the RAM. [BIOS → OS → Program]
- **CPU:**
    - Main function is to execute the program instructions in sequence.
- **Processor:**
    - **Fetch** (احضار) the next instruction.
    - **Decode** the instruction.
    - **Execute** the instruction.
- **CPU components:**
    - **ALU** → arithmetic and logic unit that performs arithmetic and logical operations.
    - **Control unit [CU]** → change instruction to micro instruction sequence (control signal generation).
    - **Register file.**
- Ex: Add C,A,B
    - Signal A
    - Signal B
    - A + B [Using ALU]
    - Temporary storage in a register.
    - Copy to memory.

- Registers → used for temporary storage.
- **Basic Registers:**
    - **Memory address register [MAR]** (unidirectional) (address bus)
        - Hold the address of the current instruction and data to be fetched (accessed).
    - **Memory data register [MDR]** (bidirectional) (data bus)
        - Hold the data coming from/to memory.
    - **Program counter [PC]**
        - Instruction pointer [IP] → stores the next instruction address.
        - Stores the next instruction address.
        - Incremented at the end of the instruction fetch.
    - **Instruction register [IR]**
        - Hold the currently executed instructions.
    - **General purpose registers**
        - Can be accessed by the instructions.
- **Instruction Cycle:**
    - The cycle in which fetch, decode and execute of an instruction take place.
- **Fetch stage:**
    - [MAR] ← [PC]
    - Send a read signal.
    - Wait for memory function to complete.
    - [MDR] ← [Memory]
    - [IR] ← [MDR]
    - Increment the PC.
- To branch to another instruction, we have to change the PC through:
    - Conditional jump instruction (checking using flags).
    - Unconditional jump instruction.
    - Call instruction (store next address then jump)(بيروح وبيرجع).
    - Return instruction (get the stored address then jump).
    - Interrupt.
- Jump always happen in the **decoder** phase.