COA-Notes 1/11/23#

chapter #3: A Top-Level view of Computer functions and interconnections

→ At top level computer system consist of CPU, memory and I/O components, with one or more modules of each.

→ These components when inte-connected perform the basic functionality of a compute i.e to execute program.

3.1 Computer Components:

→ All computer designs are based on von-Neuman architecture.

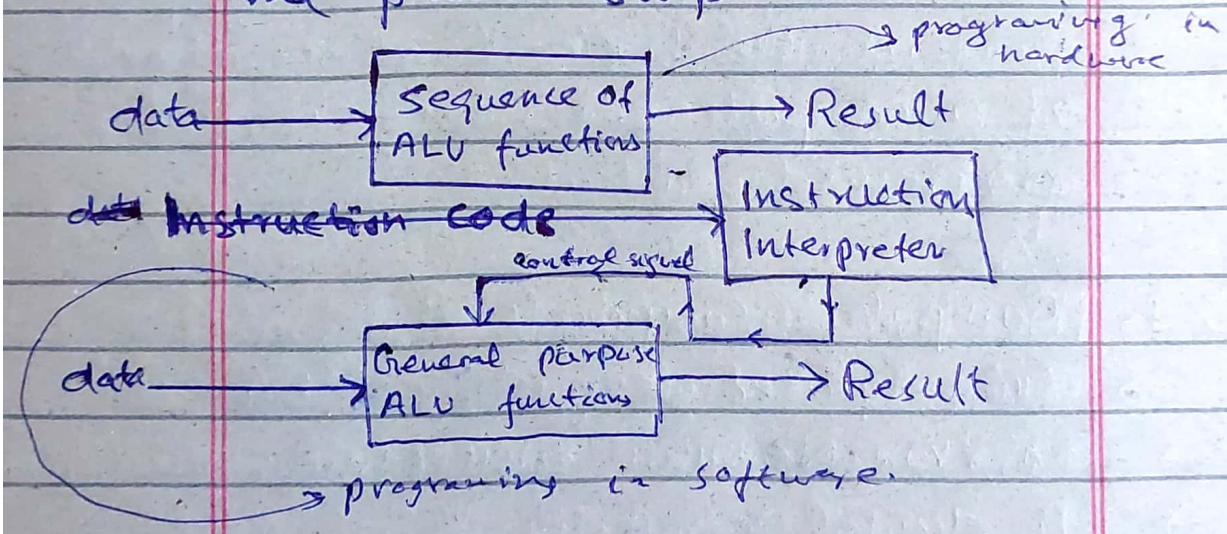
⇒ If we design a hardware for performing a special task and we can't modify it. The resulting program is in the form of hardware and is termed as 'hardwired program'.

→ Hardwired program are not scaleable because we will need rewiring.

(2)

→ If we design a general hardware and provide a control signal to it we can perform many functions and also this will be scaleable/flexable.

→ The former case accept data and produce output while the second case accept data and control signal and produce output.



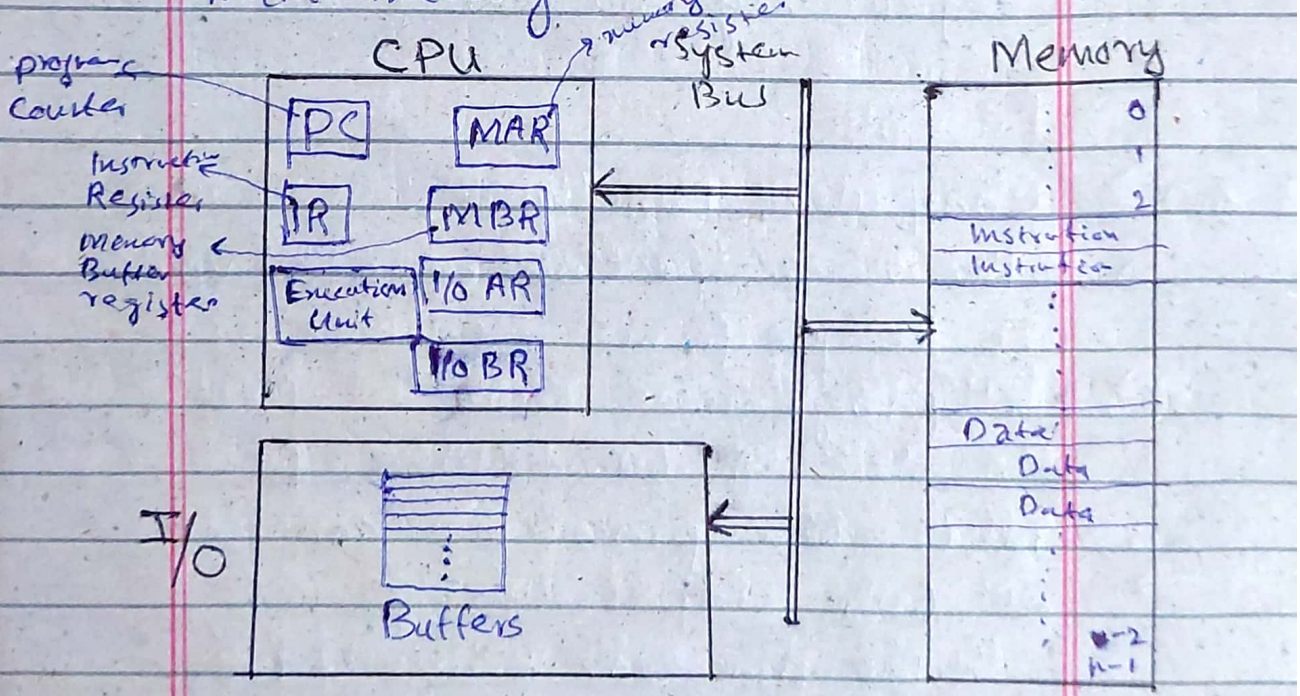
→ Program is actuall a sequence of steps. At each step some arithematic or logical operation is performed on some data. For each step a new set of control signal is needed.
Now for each possible set of control signal a unique code.

(opcode) is provided.

→ This new method of programing is called software which is a sequence of codes and/or instructions.

→ To take data and instructions and display the result we must need some devices called I/O components like keyboard, mouse, monitor, speaker etc

→ To store both data and instruction we will need a device called main memory.



- MAR (Memory address register): Specifies address in memory for next read or write.
- MBR (Memory buffero register): contains the data read from memory or to be written into memory.
- I/O AR (I/O address register): specifies a particular I/O device
- I/O BR (I/O buffer register): ~~Spec~~ is used to exchange data b/w I/O module and the CPU.
- A memory module consists of a set of locations defined by sequentially numbered addresses.

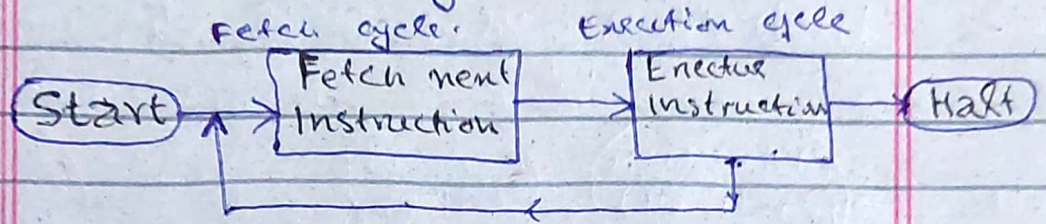
3.2 Computer Function:

→ The basic fuction of a computer is to perform execution of a program.

→ Instruction processing consists of two steps:

- The processor reads (fetch) instructions from a memory one at a time and execute each instruction.

→ processing required for a single instruction is called instruction cycle.



Fetch cycle:

→ Program Counter (PC) hold address of next instruction to fetch.
→ Processor fetch instruction pointed by PC.
→ PC is incremented unless told otherwise
→ Instruction loaded in IR.

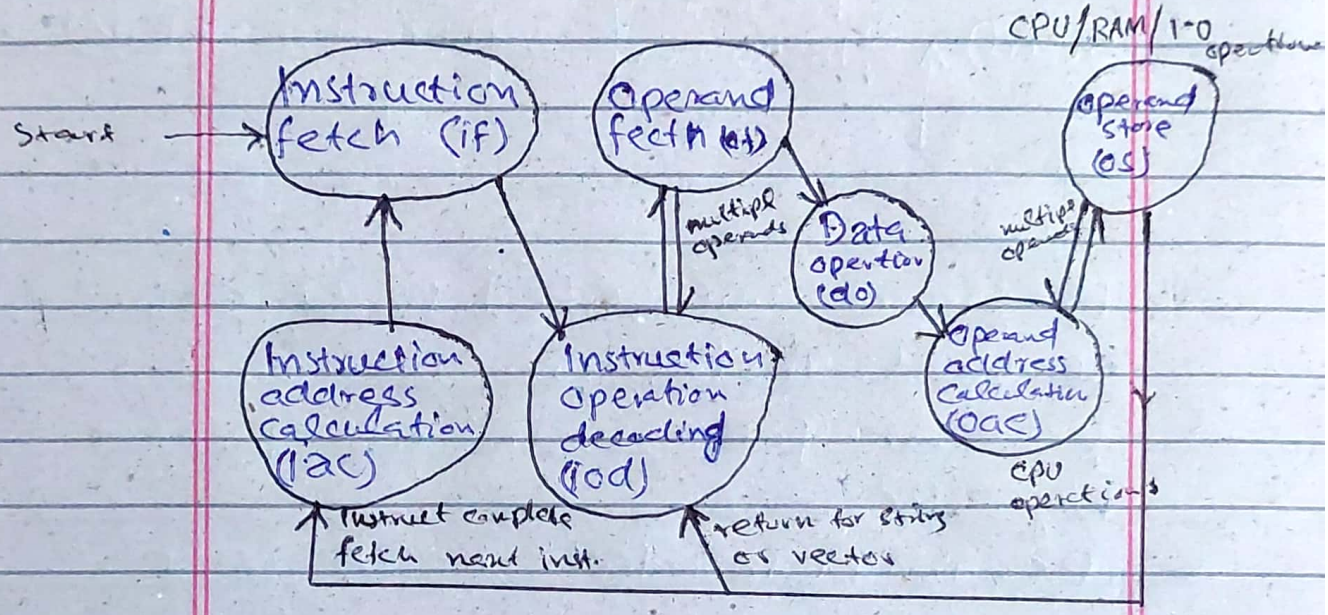
Execution cycle:

→ These instruction contain bits of actions. for processor to take. These action interpreted by processor are
- Processor – Memory: Exchange of data b/w processor and memory
- Processo – I/O: Exchange of data b/w processo and I/O.
- Data processing: processor perform some A/L operation on data.
- Control: altering of sequence

⑥

of operation, set PC.
• combination of above.

→ Instruction cycle state diagram:



Interrupts:

→ Mechanism by which other module (eg I/O) may interrupt normal sequence of processing.

→ Interrupts are provided primarily as a way to improve processing efficiency.

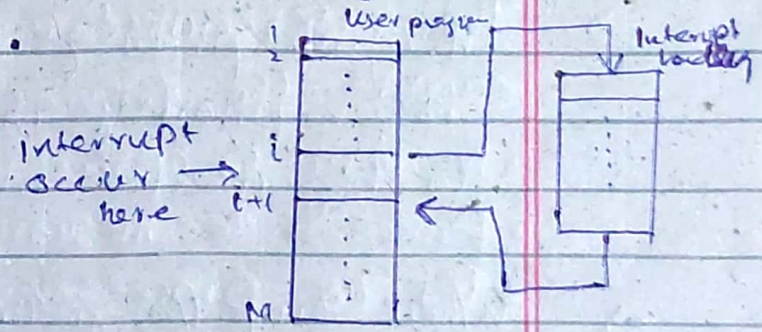
→ Some classes of interupts are
• Program : arithematic overflow, division by zero, etc
• Timer: generated by internal processor timer. use for

preintion.

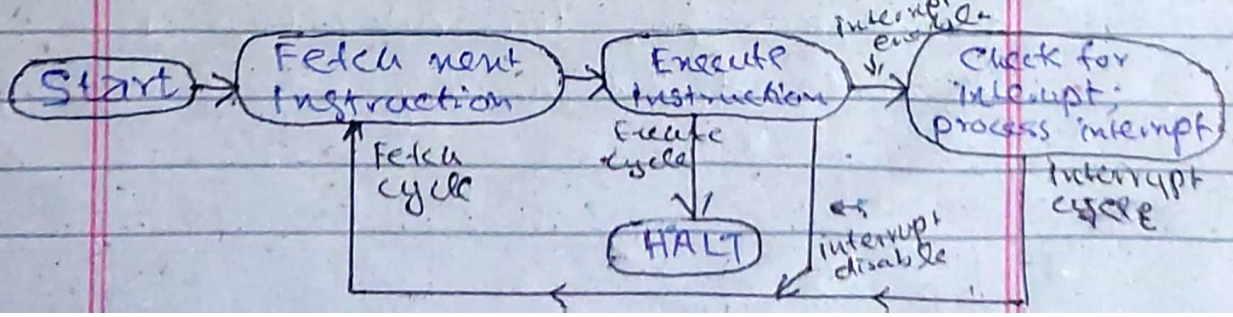
- I/O : generated by I/O controller,
- Hardware failure: power failure, memory parity error.

→ With interupts, the processor can be engaged in executing other instructions while an I/O operation is in progress.

→ When external program is ready to service — the I/O model sends an 'interrupt request signal' to the processor. The processor respond to signal by suspending current opperation and branching to I/O program.



→ To accomodate interrupt cycle is added to instruction cycle.



→ What if more than one interrupt occur at same time.

→ Two approaches can be take to deal with multiple interrupts:

• disable interrupts: simply processor can and will ignore the interrupt request signal. disable interupts while interupt handler is in process and enable it when it is ended.

→ The drawback of this approach is that it doe't take into account relative priority or time-critical needs.

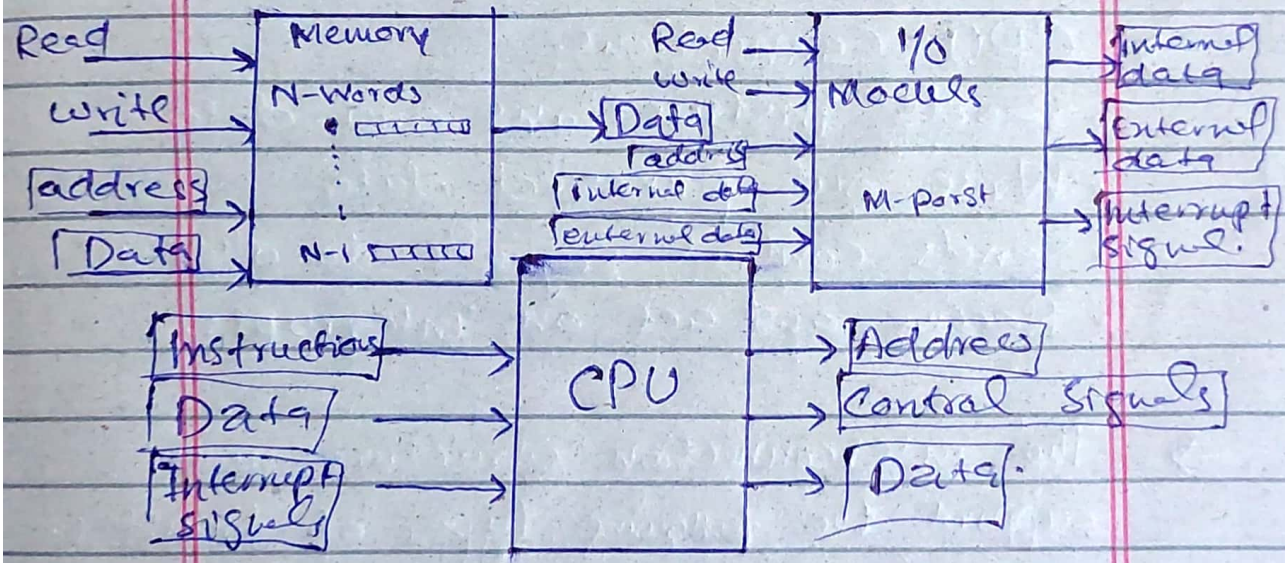
→ • Priortize interupts: A second approach is to define prioritise for interupts and to allow an interupt of heigher priority to cause a lower-priority interupt handler to be itself interrupted.

→ When an I/O module issue read or write command to memory, relieving the processor of responsibility for the exchange. This operation is called direct Memory access (DMA).

3.3 Interconnection Structure:

→ The collection of paths for connecting the modules (basic types like processor, I/O, memory) is called interconnection structure.

→ Types of exchanges that are needed by indicating its inputs and outputs for each module



→ Memory connections:
- Receive and sends data
- Receive addresses (of locations)
- Receive control signals (Read, write).

→ I/O connections are similar to memory from compute point of view:

- Receive data from computer.
- Receive data from phariphoral.
- Send data to phariphoral.
- Send data to computer.
- Receive control signal from computer
- Send control signal to phariperal
- Receive addresses from computer (port no. to identify phariperals).
- Send interrupt signals (control).

→ CPU connections:
- reads instructions and data
- Write out processed data
- Sends control signals to other unit.
- Receive and act on interrupts.

→ The interconnection structure must support the following types of transfer.
- Memory to processor.
- Processor to memory.
- I/O to processor.
- processor to I/O.
- I/O to/from memory.

3.4 Bus Interconnection:

→ A bus is a communication pathway connecting two or more devices.

→ A bus consists of multiple communication pathways or lines. Each line can transmit signals representing binary 1 or 0.

→ Bus structure may be single or multiple.

→ There are many buses in a computer. The bus which connects major components (I/O, Memory, Processor) is called a system bus.

→ A typical bus consists of 50 to hundreds of seperate lines, asigned with particular meaning or function.

→ In any bus, lines are classified into three functional groups: data, address and control lines. There may be also power distribution lines.

→ Data lines provide a path for moving data among system modules often called data bus collectively.

→ Data bus may consists of 32, 64, 128

(12) → width of data bus determine overall system performance.

or even more lines, This is refered to width of data bus.

→ Address lines are used to designate the source or distination of data on the data bus.
→ Width of ~~data~~ address bus determine maximum possible memory capacity of a system.
→ Address bus is also used to address I/O ports.

→ The control lines are used to control the access to and the use of data and address lines.
→ Control signals transmit both commands and timing information.
→ Typical control signals are: memory read/write, I/O read/write, Transfer ACK, Bus request/grant, Interrupt request/ACK, Clock, reset.

→ If a module want to send data:
- obtain use of bus
- transfer data via bus

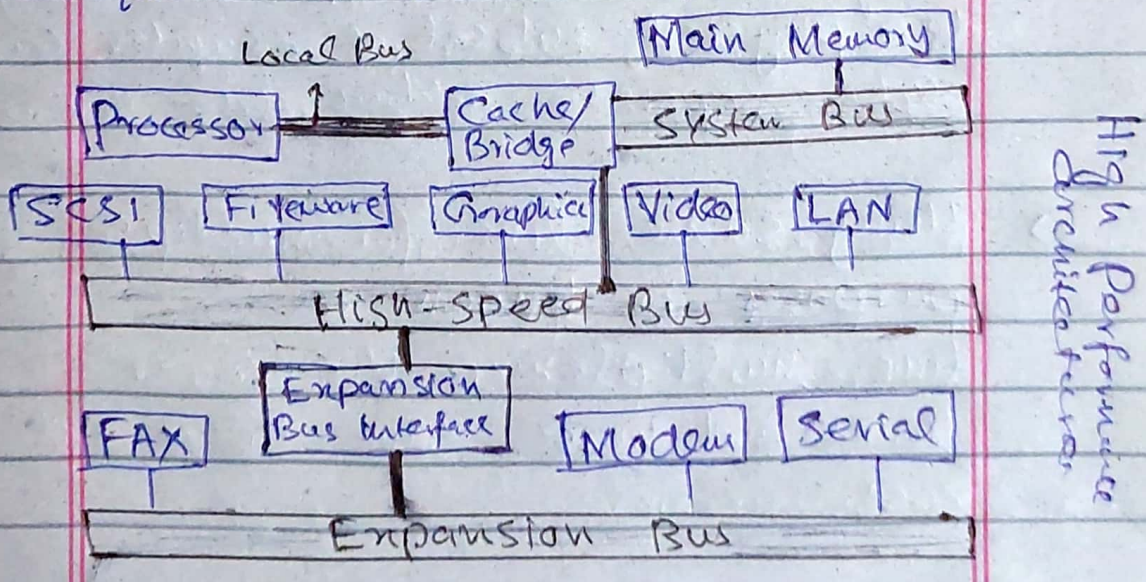
→ If a module request data:

- obtain use of bus.
- transfer request to other module.
- wait for module to send data..

→ An on-chip bus connects processor and cache memory while an on-board bus may connects the processor to memory and other models.

→ Most computer uses multiple buses because a single bus can become a bottleneck and cause propagation delay.

→ We can also use expansion bus. it interface buffer data transfer between the system bus and I/O controller on expansion bus.



(14)

→ The advantage of mezzanine/high performance architecture is that high speed bus brings high-demand devices into closer integration with the processor and at the same time is independent of the processor.

→ Some Elements of a bus design are:
- Type: dedicated or multiplexed
- Bus Width: address/data
- Method of arbitration: centralised/distributed
- Timing: Synchronous/Asynchronous
- Data transfer type: read, write, read-modify-write, read-after-write, ~~modify~~ block.

→ A dedicated bus line is permanently assigned either to one function or to a physical subset of computer component. like separate address, data line.

→ Using same line for multiple purposes is known as time multiplexing. Use fewer lines and hence save space and cost.

→ In a centralised scheme, a single hardware device, refered to as a bus controller or arbiter, is responsible for allocating time on the bus.

→ In a distributed scheme, There is a no central controller, rather each module contains access logic and the modules act together to share the bus.

→ Timing means the way in which events are cordinated on the bus.

→ With synchronous timing, the occurance of events on the bus is determined by a clock.

→ A single 1-0 transmission is reffered to as a clock cycle or bus cycle and defines a time slot.

→ With asyncuronous timing, the occurence of one event on a bus follows and depends on the occurence of a previous event.

(16) $W_{D \cdot B} \propto$ Performance, $W_{A \cdot B} \propto$ Locations in memory.

→ The width of data bus has an impact on system performance while the width of address bus has an impact on system capacity

→ All busses support both write (master to slave) and read (slave to master) transfer

→ A read-modify-write operation is simply a read followed ~~by~~ imidiatly by a write to same address.

→ Read-after-write is an indivisible operation consisting of a write followed imidiatly by a read from the same address. The read operation may be performed for checking purposes.