## 4 - Mark

1. Various functional units of computer:

⇒ There are fine units

→ Input :- Converts external world data to a binary format.
ex: Keyboard.

→ Output :- Converts the binary format to world data format.
ex: Printer.

→ Arithmetic & logic :- • ALU & CU form processer.
• Operands are brought into processer to perform required operation and is stored in registers
- ALU executes most operations.

→ Memory :- • Stores data/programs ; there are 2 classes.
(i) Primary storage :- Fast memory, main memory ex: RAM
(ii) Secondary Storage :- External/Auxiliary memory. ex: CD

→ Control Unit :- • Co-ordinates operations of all other units.
• Physically distributed throughout the machine
• Carries signals for timing/sync of events in all units.

2. Various types of Memory System:-

| <u>Primary Memory</u>. | <u>Secondary Memory</u> |
|---|---|
| → Directly accessed by CPU | → Not directly accessed by CPU. |
| → It is main memory | → It is backup memory. |
| → Computer can't run without it | → Computer can run without it. |
| → It is faster | → It is slower. |
| <u>ex:</u> ROM, RAM | <u>ex:</u> CD, DVD. |

3. Explain role of PC, IR, MAR and MDR in processer

=> PC - Program Counter
  -> Holds the memory address of the next instruction
  to be fetched and executed.

IR - Instruction Register
  -> Holds the instruction that is currently being executed.

MAR - Holds the address of memory to be accessed.

MDR - Contains the data to be written into / read out
  of the addressed location

MAR -> Memory Address Register } Facilitate communication
MDR -> Memory Data Register } with memory.

4. What is a Bus? Explain single & multiple bus structure.

=> <u>Bus</u> => Communication system that transfers data between
  components in or between computers.

<u>Single Bus</u> => Only one transfer at a time,
  two units active at any time
  · Simplest way to connect all functional units.
  · Simplest way to connect all peripheral devices.

<u>Advantage</u> :- Low cost, flexible to add peripheral devices.

<u>Multiple Bus</u> => More concurrency
  -> 2 or more transfers at the same time.

<u>Advantage</u> :- Improves performance.

<u>Disadvantage</u> :- High cost.

5. Register Transfer Notation.

=> · Used to describe how data is passed between CPU registers
  during the execution of instructions.
- Written in human readable format.
- Locations are specified in instructions by symbolic names.
- Contents of location denoted by square bracket ones
  name of location.

<u>ex</u>: DATAIN, DATAOUT, INSTATUS, OUTSTATUS — IO registers.

6. Explain Assembly Language Notation:-

→ Notation to represent~~ation~~

→ machine instructions & program.

→ Mnemonics are used to define Assembly Language Instructions.

→ Different processers have different mnemonics.

ex:- • R1 ← [LOC] → Copies content from location LOC to register R1.

• R3 ← [R1] + [R2] → Adds content in R1 & R2 and moves to R3.

7. Different types of Instruction Format:-

⇒ (i) Three-address Instruction

Syntax :- Operation source 1, source 2, destination.

ex :- Add A, B, C.

Disadvantage :- Instruction is too long for one word.

(ii) Two-address Instruction

syntax:- Operation source, destination

• Single two-address Instruction isn't enough.

ex: MOVE B, C

ADD A, C.

(iii) One-address Instruction

syntax:- Operation source/destination.

• Either source / destination is mentioned.

ex: Load A.

8. Diff RISC & CISC

⇒

| RISC | CISC |
|---|---|
| → Reduced Instruction Set Computer | → complex Instruction set computer |
| → Uses more registers. | → less registers |
| → More addressing modes | → Less addressing modes |
| → Pipelining is difficult | → Pipelining is easy. |
| → Instructions take a varying amount of cycle time. | → Instructions take one cycle time. |

9. What are Assembler Directives?

⇒ Assembler directives are directions to the assembler to take some action / change a setting.

→ They do not represent instructions.

→ They aren't translated into machine code.

There are 4 assembler directives:-

- → .text
- → .data
- → .label
- → .number.

ex :- .label var1
.number 5

And, # is used to specify a comment.

10. Various Processer and CPU cores in ARM processes.

~~=> ARM7TDMI~~

=> ARM 7 TDMI Processer Core.

--> Commonly used for low cost/power applications.

--> Has 3 stage pipelining.

--> Supports Thumb & Standard instruction sets.

=> ARM9TDMI & ARM10TDMI Processer Core.

--> Seperate instruction & data ports for high performance.

--> 5 & 6 stage pipelining.

--> Decode Thumb instruction directly.

=> ARM 720T CPU core.

--> Consists of ARM7TDMI Processer core with 8k byte unified instruction & data cache.

--> clock rate can be up to 60MHz.

=> ARM920T & ARM1020E CPU core

--> Based on ARM9TDMI & ARM10TDMI processer core.

--> Have seperate instruction & data caches.

--> ARM920T --> 16k bytes

ARM1020E --> 32k bytes.