COMPUTER ORGANIZATION

UNIT-3

CPU Design:

Timing and control:-

The timing for all registers in the basic computer is controlled by a master clock generator.

The clock pulses are applied to all flip-flops and registers in the system, including the flip-flops and registers in the control unit.

The clock pulses do not change the state of a register unless the register is enabled by a control signal.

The control signals are generated in the control unit and provide control inputs for the multiplexers in the common bus, control inputs in processor registers, and micro-operations for the accumulator.

There are two major types of control organization:-

1. Hard wired control
2. Micro programmed control

The differences between hardwired and micro programmed control are

Hardwired control	Microprogrammed control
The control logic is implemented with gates, flip-flops, decoders, and other digital circuits.	The control information is stored in a control memory. The control memory is programmed to initiate the required sequence of micro-operations.
The advantage that it can be optimized to produce a fast mode of operation.	Compared with the hardwired control operation is slow.
Requires changes in the wiring among the various components if the design has to be modified or changed.	Required changes or modifications can be done by updating the micro program in control memory.

Instruction Cycle:

It is defined as the basic cycle carried out in a computer system in which the computer system fetched the instruction from memory, decodes the instruction and then executes the instruction. It is also known as Fetch-Execute-Cycle.

In the computer system, all the instructions are executed in the RAM of the computer system. The CPU is responsible for executing the instruction. The CPU system first fetches the data and

instruction from the main memory and store in the temporary memory, which is known as registers. This phase is known as the fetch cycle.

After fetching an instruction from memory, the next step taken by the CPU is decoding of fetched instruction. This phase is known as the decode phase.

The CPU contains instructions set which contain all the predefined list of instructions. In the last phase of the instruction cycle, data processing takes place. The instruction executes in this phase, and the result is stored in another register.

After the fetch-decode-execute cycle, the CPU reset itself for another instruction cycle. The CPU system is considered the basic operation cycle, which is carried out in RAM of the central processing unit and executes the instruction. It is repeatedly continuously in the CPU when the computer system boots and then shut down.

It is executed in a sequential manner means when one instruction cycle is completed, then only the next instruction cycle begins. But in the modern era central processing unit, the instructions can be executed in a parallel manner.

Stages of Instruction Cycle:

There are basically five stages of the instruction cycle which are described below:

1. Initiating Cycle

In this phase, the computer system boots up, and the operating system loads in the main memory (read-only memory) of the central processing unit. It immediately begins when the computer system starts.

2. Fetching of Instruction

The fetching of instruction is the first phase. The fetch instruction is common for each instruction executes in a central processing unit. In this phase, the central processing unit sends the PC to MAR and then sends the READ command into a control bus. After sending a read command on the data bus, the memory returns the instruction, which is stored at that particular address in the memory. Then, the CPU copies data from the data bus into MBR and then copies the data from MBR to registers. After all this, the pointer is incremented to the next memory location so that the next instruction can be fetched from memory.

3. Decoding of Instruction

The decoding of instruction is the second phase. In this phase, the CPU determines which instruction is fetched from the instruction and what action needs to be performed on the instruction. The opcode for the instruction is also fetched from memory and decodes the related operation which needs to be performed for the related instruction.

4. Read of an Effective Address

The reading of an effective address is the third phase. This phase deals with the decision of the operation. The operation can be of any type of memory type non-memory type operation. Memory instruction can be categorized into two categories direct memory instruction and indirect memory instruction.

5. Executing of Instruction

The executing of instruction is the last phase. In this stage, the instruction is finally executed. The instruction is executed, and the result of the instruction is stored in the register. After the execution of an instruction, the CPU prepares itself for the execution of the next instruction. For every instruction, the execution time is calculated, which is used to tell the processing speed of the processor.

Importance of Instruction Cycle:

- It is important for the processor system for the central processing unit as the instructions are the basic operations that are performed in the main memory of the central processing unit.
- It is a set of steps that help to understand the flow of instruction. By the instruction cycle, the end to end the flow of instructions can be visualized in the computer processor.
- It is common for all instruction set it needs to properly understand so that all the operations can be performed easily.
- By the instruction cycle, the processing time of the program can be easily calculated, which helps to determine the speed of the processor.
- As the speed of the processor tells how many instructions can be simultaneously executed in the central processing unit.

Memory Reference Instructions:

das Memory reference instructions are those commands or instructions which are in the custom to generate a reference to the memory and approval to a program to have an approach to the commanded information and that states as to from where the data is cache continually. These instructions are known as Memory Reference Instructions.

There are seven memory reference instructions which are as follows:

1. AND

The AND instruction implements the AND logic operation on the bit collection from the register and the memory word that is determined by the effective address. The result of this operation is moved back to the register.

2. ADD

The ADD instruction adds the content of the memory word that is denoted by the effective address to the value of the register.

3. LDA

The LDA instruction shares the memory word denoted by the effective address to the register.

4. STA

STA saves the content of the register into the memory word that is defined by the effective address. The output is next used to the common bus and the data input is linked to the bus. It needed only one micro-operation.

5. BUN

The Branch Unconditionally (BUN) instruction can send the instruction that is determined by the effective address. They understand that the address of the next instruction to be performed is held by the PC and it should be incremented by one to receive the address of the next instruction in the sequence. If the control needs to implement multiple instructions that are not next in the sequence, it can execute the BUN instruction.

6. BSA

BSA stands for Branch and Save return Address. These instructions can branch a part of the program (known as subroutine or procedure). When this instruction is performed, BSA will store the address of the next instruction from the PC into a memory location that is determined by the effective address.

7. ISZ

The Increment if Zero (ISZ) instruction increments the word determined by effective address. If the incremented cost is zero, thus PC is incremented by 1. A negative value is saved in the memory word through the programmer. It can influence the zero value after getting incremented repeatedly. Thus, the PC is incremented and the next instruction is skipped.

Input-output and interrupt:

Input or Output processes are a little complex than other memory or register reference processes. One most important thing behind it is the frequent interaction of h/w devices. To let the respective h/w do its task, two separate registers are used within the CPU. These are: INPR (Input Register) and OUTR (Output Register). Both of these registers are of 8-bit size, hence despite of the type of data they hold, we say that an INPR holds an input character that comes from some input device whereas an OUTR holds an output character that goes to some output device.

Two very specific flip-flops FGI and FGO are used to denote the forecast of some input or output operation about to happen. Set status of FGI tells that some input device wants to use INPR for its input data. Similarly, set status of FGO tells that some output data is ready to be taken from OUTR for output devices.

Programmed I/O: The systematic approach is that CPU, while doing it's own tasks, should check regularly the status of FGI and FGO flip-flops. If any of these flip-flops are set, the respective I/O process can start abruptly. This method is good enough but quite resource consuming. It is very clear that in case of a long silence in FGI and FGO, the CPU is wasting it's time by regularly checking their status.

Interrupt I/O: A good alternative to Programmed I/O is to let the device inform the control that some I/O transfer is needed. The CPU will halt it's ongoing task temporarily and branch to the requested I/O transfer. After the control is finished with I/O, it returns to the halted task. Here letting the device intervene CPU is called Interrupt. A third flip-flop IEN is used to denote the forecast of I/O transfer.

Interrupt Cycle:

Interrupt cycle is very similar to the instruction cycle. At the very start, the status of flip-flop R is checked. If it is 0 there is no interrupt and CPU can continue it's ongoing tasks. But when R=1, it denotes that the ongoing process should halt because an interrupt has occured.

When R=0, CPU continues its tasks checking the status of IEN in parallel. If it is 1, FGI and FGO are checked in a hierarchy. If any of these flip-flops are found set, R is immidiately set by 1.

When R=1, the content in PC (adress of next instruction in memory) is saved at M[0] and then PC is set by 1 enabling it to point the BUN operation. The instruction at M[1] is a BUN instruction that leads the control to approriate I/O ref. Instruction stored at some other location in the memory. Now separate Fetch, Decode and Execute phases are practised to entertain the I/O ref. instruction.

Complete Computer Description:

The complete computer description includes both, the instruction cycle and the interrupt cycle. Interrupt cycle because there may arise a case of I/O operation anytime during normal operation. Instruction cycle because in absence of interrupts, the CPU is always busy with stored program instructions. A flip flop R is used as a condition to determine the type of operation. When R=0, the instruction cycle continues. Similarly when R=1, the computer goes through an interrupt cycle. **Here it must be noted that an interrupt is signaled by IEN and R is just to make it sure that another interrupt does not get entertained while processing of an interrupt.**

A basic computer consists the following hardware components:

- A memory unit of 4096 X 16 bits
- Nine Registers- PC, AR, IR, DR, AC, TR, INPR, OUTR and SC
- Five Flip-Flops- I, R, IEN, FGI and FGO
- Two decoders- a 3 to 8 operation decoder and a 4 to 16 timing decoder
- A 16-bit common bus
- Control Logic Gates

Instruction Formats:

A computer performs a task based on the instruction provided. Instruction in computers comprises groups called fields. These fields contain different information as for computers everything is in 0 and 1 so each field has different significance based on which a CPU decides what to perform. The most common fields are:

- Operation field specifies the operation to be performed like addition.
- Address field which contains the location of the operand, i.e., register or memory location.
- Mode field which specifies how operand is to be founded.

Instruction is of variable length depending upon the number of addresses it contains. Generally, CPU organization is of three types based on the number of address fields:

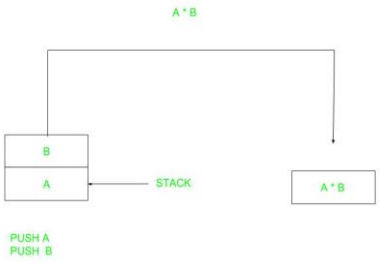
1. Single Accumulator organization
2. General register organization
3. Stack organization

In the first organization, the operation is done involving a special register called the accumulator. In second on multiple registers are used for the computation purpose. In the third organization the work on stack basis operation due to which it does not contain any address field. Only a single organization doesn't need to be applied, a blend of various organizations is mostly what we see generally.

Based on the number of addresses, instructions are classified as:

Note that we will use X = (A+B)*(C+D) expression to showcase the procedure.

1. Zero Address Instructions:



A stack-based computer does not use the address field in the instruction. To evaluate an expression first it is converted to reverse Polish Notation i.e. Postfix Notation.

Expression: X = (A+B)*(C+D)
Postfixed: X = AB+CD+*
TOP means top of stack
M[X] is any memory location

PUSH	A	TOP = A
PUSH	B	TOP = B
ADD		TOP = A+B
PUSH	C	TOP = C
PUSH	D	TOP = D
ADD		TOP = C+D
MUL		TOP = (C+D)*(A+B)
POP	X	M[X] = TOP

2. One Address Instructions:

This uses an implied ACCUMULATOR register for data manipulation. One operand is in the accumulator and the other is in the register or memory location. Implied means that the CPU already knows that one operand is in the accumulator so there is no need to specify it.

Expression: X = (A+B)*(C+D)
AC is accumulator
M[] is any memory location
M[T] is temporary location

LOAD	A	AC = M[A]
ADD	B	AC = AC + M[B]
STORE	T	M[T] = AC
LOAD	C	AC = M[C]
ADD	D	AC = AC + M[D]
MUL	T	AC = AC * M[T]
STORE	X	M[X] = AC

3. Two Address Instructions:

This is common in commercial computers. Here two addresses can be specified in the instruction. Unlike earlier in one address instruction, the result was stored in the accumulator, here the result can be stored at different locations rather than just accumulators, but require more number of bit to represent address.

Here destination address can also contain operand.

Expression: X = (A+B)*(C+D)
R1, R2 are registers
M[] is any memory location

MOV	R1, A	R1 = M[A]
ADD	R1, B	R1 = R1 + M[B]
MOV	R2, C	R2 = C
ADD	R2, D	R2 = R2 + D
MUL	R1, R2	R1 = R1 * R2
MOV	X, R1	M[X] = R1

4.Three Address Instructions:

This has three address field to specify a register or a memory location. Program created are much short in size but number of bits per instruction increase. These instructions make creation of program much easier but it does not mean that program will run much faster because now instruction only contain more information but each micro-operation (changing content of register, loading address in address bus etc.) will be performed in one cycle only.

Expression: X = (A+B)*(C+D)
R1, R2 are registers
M[] is any memory location

ADD	R1, A, B	R1 = M[A] + M[B]
ADD	R2, C, D	R2 = M[C] + M[D]
MUL	X, R1, R2	M[X] = R1 * R2

Instruction and Addressing:

The operation field of an instruction specifies the operation to be performed. This operation will be executed on some data which is stored in computer registers or the main memory. The way any operand is selected during the program execution is dependent on the addressing mode of the instruction. The purpose of using addressing modes is as follows:

1. To give the programming versatility to the user.
2. To reduce the number of bits in addressing field of instruction.

Types of Addressing Modes:

Below we have discussed different types of addressing modes one by one:

1. Immediate Mode:

In this mode, the operand is specified in the instruction itself. An immediate mode instruction has an operand field rather than the address field.

For example: ADD 7, which says Add 7 to contents of accumulator. 7 is the operand here.

2. Mode:

In this mode the operand is stored in the register and this register is present in CPU. The instruction has the address of the Register where the operand is stored.

Advantages

- Shorter instructions and faster instruction fetch.
- Faster memory access to the operand(s)

Disadvantages

- Very limited address space
- Using multiple registers helps performance but it complicates the instructions.

3. Register Indirect Mode:

In this mode, the instruction specifies the register whose contents give us the address of operand which is in memory. Thus, the register contains the address of operand rather than the operand itself.

4. Auto Increment/Decrement Mode:

In this the register is incremented or decremented after or before its value is used.

5. Direct Addressing Mode:

In this mode, effective address of operand is present in instruction itself.

- Single memory reference to access data.
- No additional calculations to find the effective address of the operand.

For Example: ADD R1, 4000 - In this the 4000 is effective address of operand.

NOTE: Effective Address is the location where operand is present.

6. Indirect Addressing Mode:

In this, the address field of instruction gives the address where the effective address is stored in memory. This slows down the execution, as this includes multiple memory lookups to find the operand.

7. Displacement Addressing Mode:

In this the contents of the indexed register is added to the Address part of the instruction, to obtain the effective address of operand.

EA = A + (R), In this the address field holds two values, A(which is the base value) and R(that holds the displacement), or vice versa.

8. Relative Addressing Mode:

It is a version of Displacement addressing mode.

In this the contents of PC (Program Counter) are added to address part of instruction to obtain the effective address.

EA = A + (PC), where EA is effective address and PC is program counter.

The operand is A cells away from the current cell (the one pointed to by PC)

9. Base Register Addressing Mode:

It is again a version of Displacement addressing mode. This can be defined as EA = A + (R), where A is displacement and R holds pointer to base address.

10. Stack Addressing Mode:

In this mode, operand is at the top of the stack. For example: ADD, this instruction will *POP* top two items from the stack, add them, and will then *PUSH* the result to the top of the stack.

Instruction Cycle:

An instruction cycle, also known as **fetch-decode-execute cycle** is the basic operational process of a computer. This process is repeated continuously by CPU from boot up to shut down of computer.

Following are the steps that occur during an instruction cycle:

1. Fetch the Instruction

The instruction is fetched from memory address that is stored in PC(Program Counter) and stored in the instruction register IR. At the end of the fetch operation, PC is incremented by 1 and it then points to the next instruction to be executed.

2. Decode the Instruction

The instruction in the IR is executed by the decoder.

3. Read the Effective Address

If the instruction has an indirect address, the effective address is read from the memory. Otherwise, operands are directly read in case of immediate operand instruction.

4. Execute the Instruction

The Control Unit passes the information in the form of control signals to the functional unit of CPU. The result generated is stored in main memory or sent to an output device.