Module-2, Unit-3
Instruction Execution

Question 1: Consider an instruction cycle consisting of fetch, operators fetch (immediate/direct/indirect), execute and interrupt cycles. Explain the purpose of these four cycles.

Solution 1:

The life of an instruction passes through four phases—(i) Fetch, (ii) Decode and operators fetch, (iii) execute and (iv) interrupt. The purposes of these phases are as follows

1. *Fetch*
   We know that in the stored program concept, all instructions are also present in the memory along with data. So the first phase is the "fetch", which begins with retrieving the address stored in the Program Counter (PC). The address stored in the PC refers to the memory location holding the instruction to be executed next.
   Following that, the address present in the PC is given to the address bus and the memory is set to read mode. The contents of the corresponding memory location (i.e., the instruction) are transferred to a special register called the Instruction Register (IR) via the data bus. IR holds the instruction to be executed. The PC is incremented to point to the next address from which the next instruction is to be fetched

   So basically the fetch phase consists of four steps:
   a) MAR <= PC (Address of next instruction from Program counter is placed into the MAR)
   b) MBR<=(MEMORY) (the contents of Data bus is copied into the MBR)
   c) PC<=PC+1 (PC gets incremented by instruction length)
   d) IR<=MBR (Data i.e., instruction is transferred from MBR to IR and MBR then gets freed for future data fetches)

2. *Decode and Operands Fetch*
   Next the instruction is decoded. Now the instruction is partitioned into two parts—(i) operation to be performed i.e., determined by the OP Code and (ii) fetching the operands on which the operation is to be performed. As the OP code is present in the instruction itself, it can be directly used. However, the operands specified in the instruction are needed to be fetched. Now, based on the mode of addressing used in the instruction the steps (and path) of fetching the operands differ. Some of the most important types of addressing used and the corresponding steps to fetch the operands are as follows:

   a. Immediate: In this case the operands are specified in the instruction itself. So, there is no requirement to explicitly fetch the operands.

b. Direct: In this case the memory location of the operand is specified in the instruction. This is a direct fetch phase and operands are brought to the location (e.g., user registers, accumulator, MBR etc.) as per the instruction.

It involves the following steps

- Step 1: MAR <= address of operand in IR
- Step 2: MBR <= memory cell whose address is given in MAR

c. Indirect: In this case, the memory location of the operand is specified in another memory location whose address is present in the IR. So this is called indirect fetch phase.

It involves the following basic steps

- Step 1: MAR <=IR address (IR address has the address of a memory location that has the address of the operand)
- Step 2: MBR <= memory cell whose address is given in MAR
- Step 3: IR address <= MBR (IR is now in same state as if direct addressing had been used, and the next steps are that of direct addressing)

3. Execute

This phase, as the name suggests simply executes the instructions based on the OP Code on the operands that have been fetched.

The results of the execution of instructions can be classified as

(a) Data transfer

Examples are data transfer from a memory location to a register, or vice versa, Read and write data from hardware devices etc.

(b) Arithmetic and logic operations

Examples are addition, subtraction, multiplication, negation of each bit, comparison etc.

(c) Control flow operations

Examples are Branch to another location in the program, conditionally branch to another location if a certain condition holds etc.

4. Interrupt

Broadly speaking the life cycle of an instruction passes through ``fetch-decode and operands fetch-execute" phases. In other words, a program which is in terms of instructions passes through these three phase, from the first instruction to the last one. However, interrupt also a part of the cycle because sometimes an event external to the currently executing program needs to get the CPU that causes a

change in the normal flow of instruction execution. Interrupt is usually generated by hardware devices external to the CPU like Keyboard, mouse, screen etc. These devices also need CPU service but we cannot predict exactly when such a necessity arises. So if the CPU keeps waiting for such a need this would lead to wastage of CPU time. An example is when a printer completed one printing task and requests for another.

To address this situation we have the interrupt phase. In other words, the CPU keeps executing the instructions of a pogrom in sequence one by one, however, after completing execution of one instruction it checks if there is any interrupt. If an interrupt has arrived, the CPU first serves the interrupt and then goes for the next instruction. The task to be accomplished in an interrupt is defined in the corresponding Interrupt Service Routine (ISR).

An interrupt phase in generally have these steps

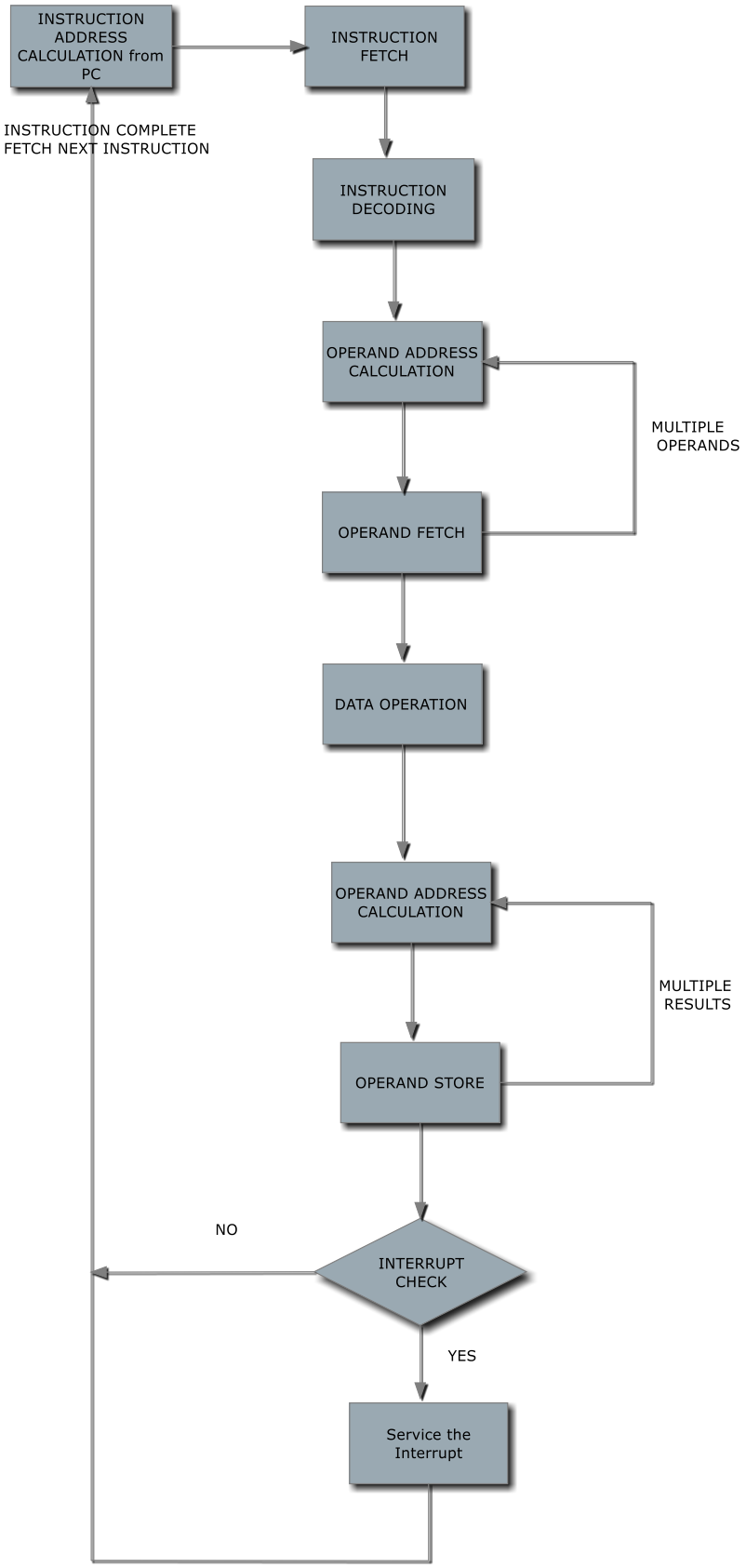
a) Step1: Save the value of PC in a stack (after servicing the ISR the current program should start from the instruction before which interrupt had arrived)
b) Step2: PC <=ISR address (Now the instructions from memory locations corresponding to the ISR will executed)
c) Step 4: CPU execute the instructions in ISR and then return.
d) PC <= Pop the return address from the stack (then execution of the last program continues)

Question 2: Briefly explain using an example how to provide the CPU services an I/O device with interrupt. Give the instruction cycle with interrupt state diagram.

Solution 2:

Interrupt is a technique to primarily improve efficiency of CPU. In modern times, multi tasking is generally followed, where several processes compete for the resources. For example, say a printing task and a computation task are ready to be executed. Generally, I/O devices are much slower than any computational task. Suppose that the processor is transferring data to a printer and the computational task is waiting for the CPU to be free. After each block of data transfer operation by the CPU, the processor must pause and remain idle until the printer finishes the printing and requests for the next block of data. During this idle time the computational task needs to wait. The length of this wait may be avoided if we use interrupt. With interrupts, the processor can be engaged in executing the computational task while the printing operation is in progress. Once the printing of a given block of data is over, the printer sends an interrupt. The CPU after completing the current instruction of the computational process checks the interrupt. As interrupt has arrived, it is served by an Interrupt Service Routine (ISR) and in this case next block of data is transferred. After interrupt is serviced, next instruction of the computational process is executed by the CPU.

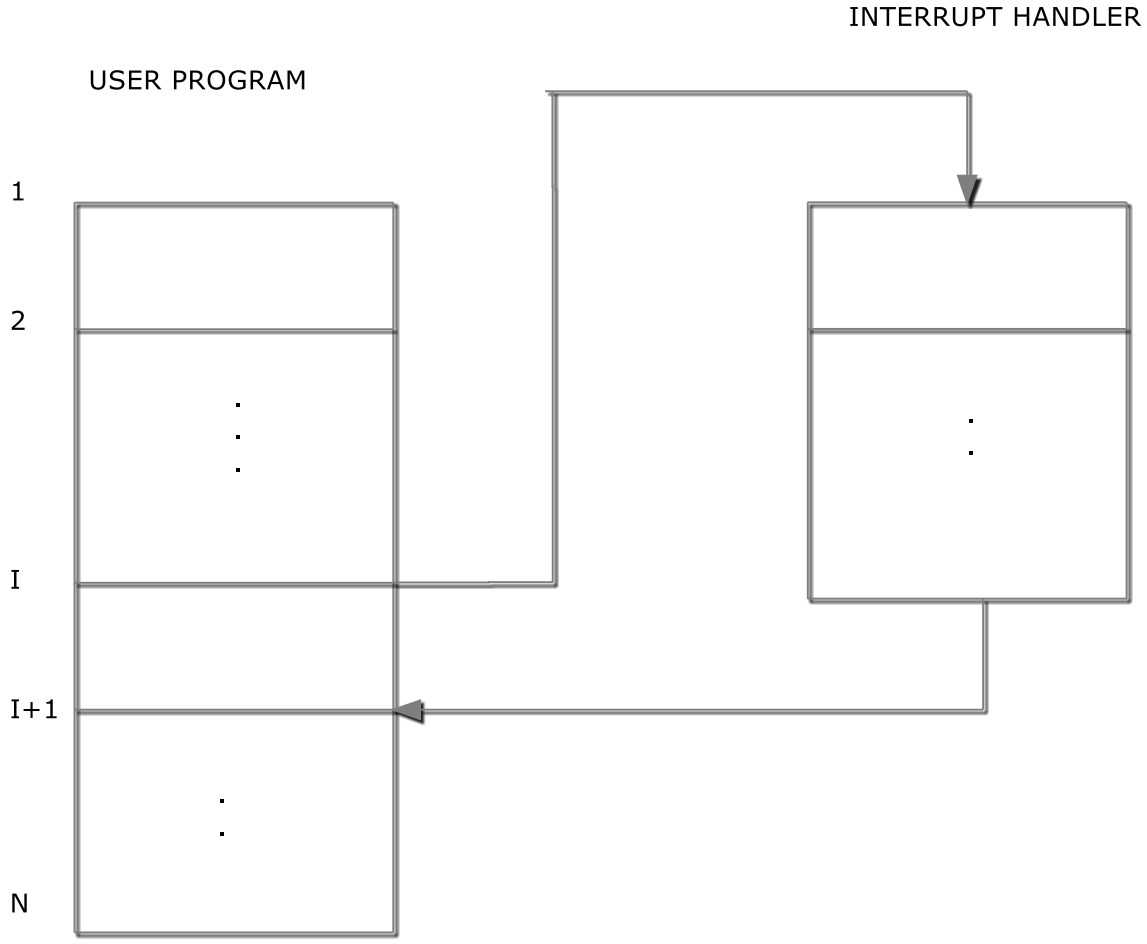
The instruction cycle with interrupt state diagram is given below.


INSTRUCTION ADDRESS CALCULATION from PC
INSTRUCTION FETCH
INSTRUCTION COMPLETE
FETCH NEXT INSTRUCTION
INSTRUCTION DECODING
OPERAND ADDRESS CALCULATION
MULTIPLE OPERANDS
OPERAND FETCH
DATA OPERATION
OPERAND ADDRESS CALCULATION
MULTIPLE RESULTS
OPERAND STORE
NO
INTERRUPT CHECK
YES
Service the Interrupt

The functions of most of the blocks in the diagram have been explained in answer to Question-1 of this unit. The functions of the block “Service the Interrupt” is as follows.

- Suspend execution of the current process and saves its context. This means saving the address of the next instruction to be executed (current contents of the PC) and other data (saved in variables) relevant to the processor’s current state.
- PC is set to the starting address of the ISR.
- The CPU now executes all instructions of the ISR (as in the case of a normal program). When the ISR is completed, the CPU resumes execution of the user program at the point of interruption; this is achieved by retrieving the value of the PC that has been saved.

The figure given below explains this process.



Question 3.

Give a scheme to identify the four phases in an instruction (fetch, Decode and operators fetch, execute and interrupt).

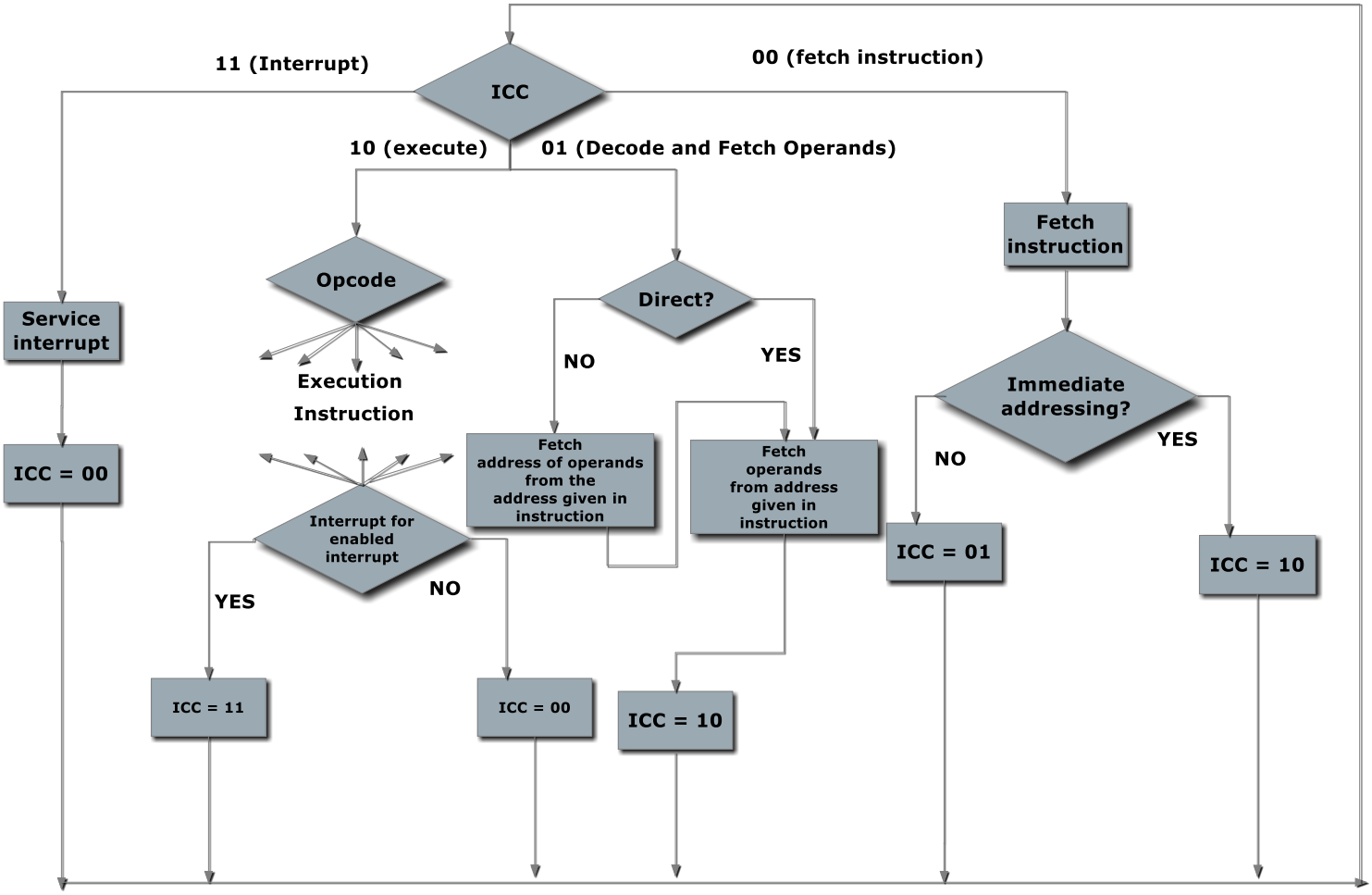
Solution 3:

The four phases can be identified using a code called Instruction Cycle Code (ICC). ICC is a two bit code and designates which part of the phase the CPU is in.

The codes assigned to each phase is as follows

00: Fetch phase
01: Decode and Operand phase
10: Execute phase
11: Interrupt phase

In the first phase the ICC is 00. After the instruction is fetched ICC is made to 01, when instruction is decoded and fetching of operands is accomplished. Based on the type of instruction, operand may be immediately available (in the instruction), or its location is available in the instruction, or the instruction has the address of the location that has the operand (indirect). After all operands are fetched ICC is made 10 and instruction is executed. Next ICC is11 and it is checked in there is any interrupt. In case an interrupt has come it is serviced and then ICC is made 00. If there is not interrupt ICC is directly made 00.
The figure below illustrates this scheme.


ICC
11 (Interrupt)
00 (fetch instruction)
10 (execute)
01 (Decode and Fetch Operands)
Service interrupt
ICC = 00
Opcode
Execution
Instruction
Interrupt for enabled interrupt
YES
NO
ICC = 11
ICC = 00
Direct?
NO
YES
Fetch address of operands from the address given in instruction
Fetch operands from address given in instruction
ICC = 10
Fetch instruction
Immediate addressing?
NO
YES
ICC = 01
ICC = 10

Question 4. Explain using an example how a simple assembly language program is executed.

Solution 4:

As both the data and program is stored in a memory, execution of an instruction involves the following steps:

1. Instruction Fetch
   This cycle basically involves bringing the current instruction from the corresponding memory location to the Instruction register.

   To elaborate it involves the following sub-steps:

   - The Program Counter (PC) has the address of the instruction to be executed. So, value of PC is loaded into the Memory Address Register (MAR) and the control signal to the memory is Read.
   - The content of the memory location (specified in the MAR) is read into Memory Buffer Register (MBR).
   - The data from the MBR is transferred to the Instruction Register (IR)

   Following these data transfers, the PC is incremented by 1, so that it now contains the address of the next instruction to be executed.

   It may be noted that if the current instruction is a Branch or Jump then the PC gets the value of the address specified in the Jump or Branch instruction.

2. Instruction Decode

   After the instruction is fetched in the IR, it has to be decoded to that it can be executed. Decoding involves determining what operation to be done and accordingly fetching the operands accordingly.

   This also it involves three sub-steps:

   - MAR is loaded with the address of the operand present in the IR.
   - MBR is loaded with the value of the operand
   - AC (Accumulator) is loaded with the operand that is present in MBR.

   Note: The above discussion of the decode cycle is for an instruction that uses direct addressing mode. In case of immediate addressing the operand is available in the IR and is copies into AC.

3. Instruction Execute
   As the name suggests, this step involves simply executing the instructions (that have been fetched and decoded) on the operands. Based on the type of instruction, execution may involve data transfer between the CPU and the I/O module. Also, arithmetic and logical operations may be performed on the data, as well as some instructions such as jumping to another location involve updating the PC to the address of the jumping location.

The example given below illustrates the execution of a simple code.

We need to add two numbers that are in memory locations FF0 and FF1, and finally store the result in memory location FF2.

To perform the addition following operations have to be accomplished.

1. Contents of memory location FF0 has to be loaded into accumulator. Let us assume that 5 is present in FF0.
2. Contents of memory location FF1 have to be read and should be added to value in accumulator. Result of the addition should be stored back in accumulator. Let us assume that 7 is present in FF1. So, 5 and 7 have to be added and the result should be stored in accumulator.
3. The result of the addition which is stored in accumulator must be written into memory location FF2. In this example, result of addition (12) have to be written into memory location FF2.

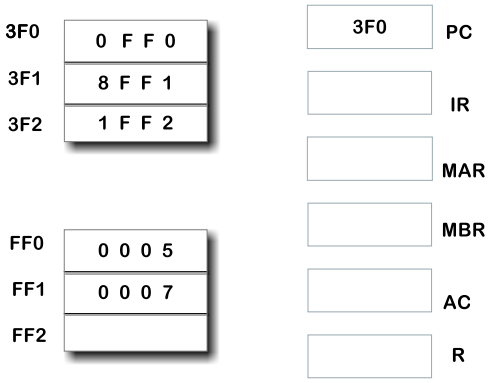
The assembly language program is as follows:

1st Instruction **LDA FF0**:
The contents in memory location FF0 are loaded into accumulator. After the instruction is executed accumulator stores value 5.

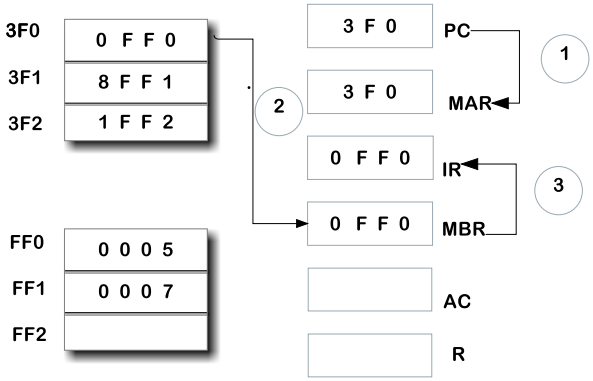
2nd Instruction **ADD FF1**:
The contents in memory location FF1 is added to accumulator. The final result is stored in accumulator. So 5+7 addition is performed and result 12 is stored in accumulator.

3rd Instruction **STA FF2**:
This instruction stores the contents of accumulator in memory location FF2.

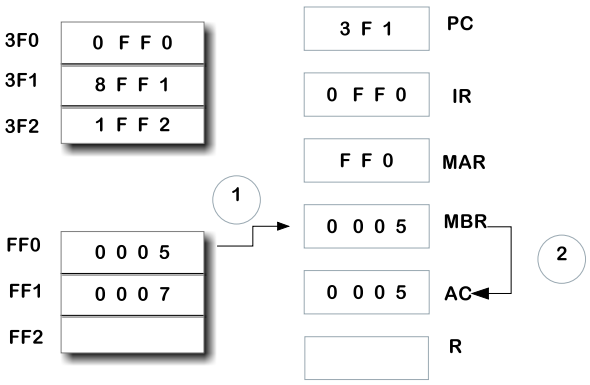
If we assume that the first instruction is store in location 3F0, the following diagram demonstrates step by step execution of this code.



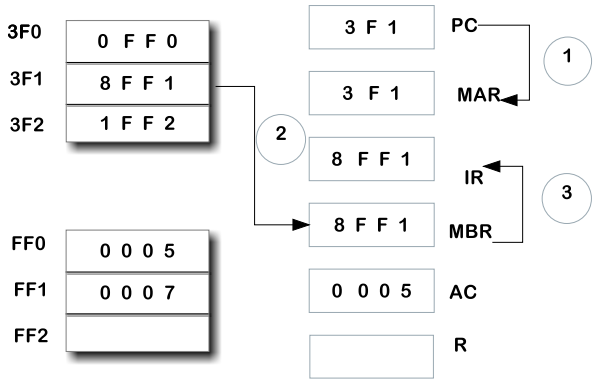
This is the initial status of registers in CPU. Program counter contains address 3F0.



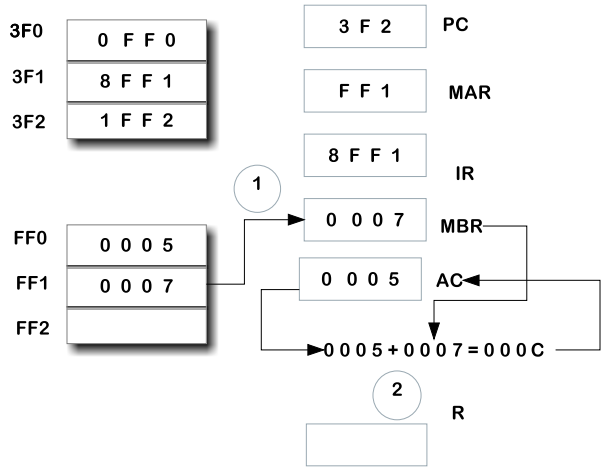
1) The address present in PC is copied into MAR.
2) In second clock cycle. data from memory is copied into MBR. simultaneously,PC is incremented.
3)In third clock cycle. Instruction present in MBR is copied into IR.



After the completion of fetch cycle,the instruction is going to be executed.
As per the instruction,Address FF0 is copied into MAR. Data in corresponding memory location is being read from memory into MBR
The data is then copied into AC.



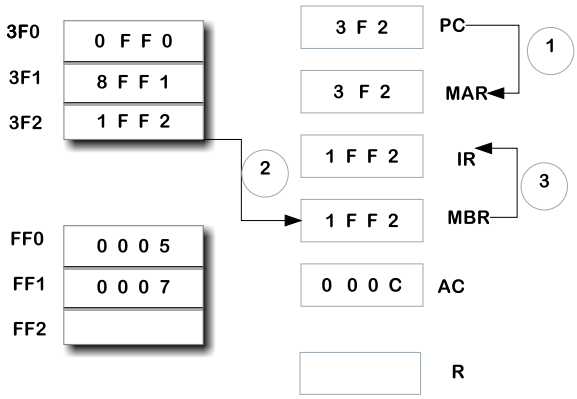
1)To fetch next instruction ,the address present in the PC is copied into MAR.
2)In second clock cycle,Data from corresponding memory location is copied into MBR. simultaneously,PC is incremented.
3)In third clock cycle,Instruction present in MBR is copied into IR.



As per the instruction ,
1)Address FF1 is copied into MAR.
Data in corresponding memory location is copied into MBR.

2)Value in MBR is added value in AC.

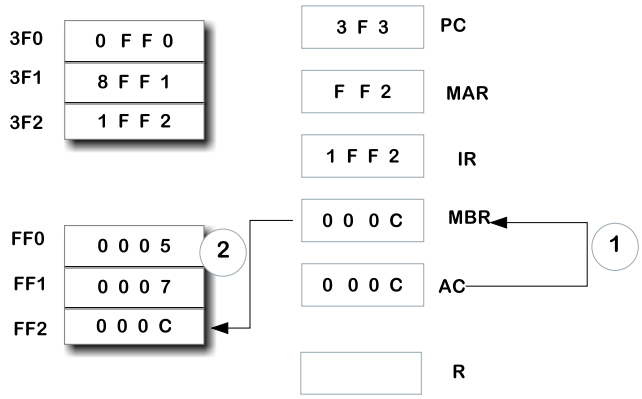
3)result is stored in ac.



To fetch next instruction,
1)The address present in PC is copied into MAR.

2)In second clock cycle ,Data from corresponding memory location is copied into MBR.
Simultaneously,PC is incremented.

3)In third clock cycle ,instruction present in MBR is copied into IR.



As per the instruction ,
1)Address FF2 is copied into MAR.
value present in AC is copied to MBR.

2)Value present in MBR is Copied to memory location FF2.