Computer Architecture

Control Unit Operation

Md. Biplob Hosen
Lecturer, IIT-JU

Reference Books

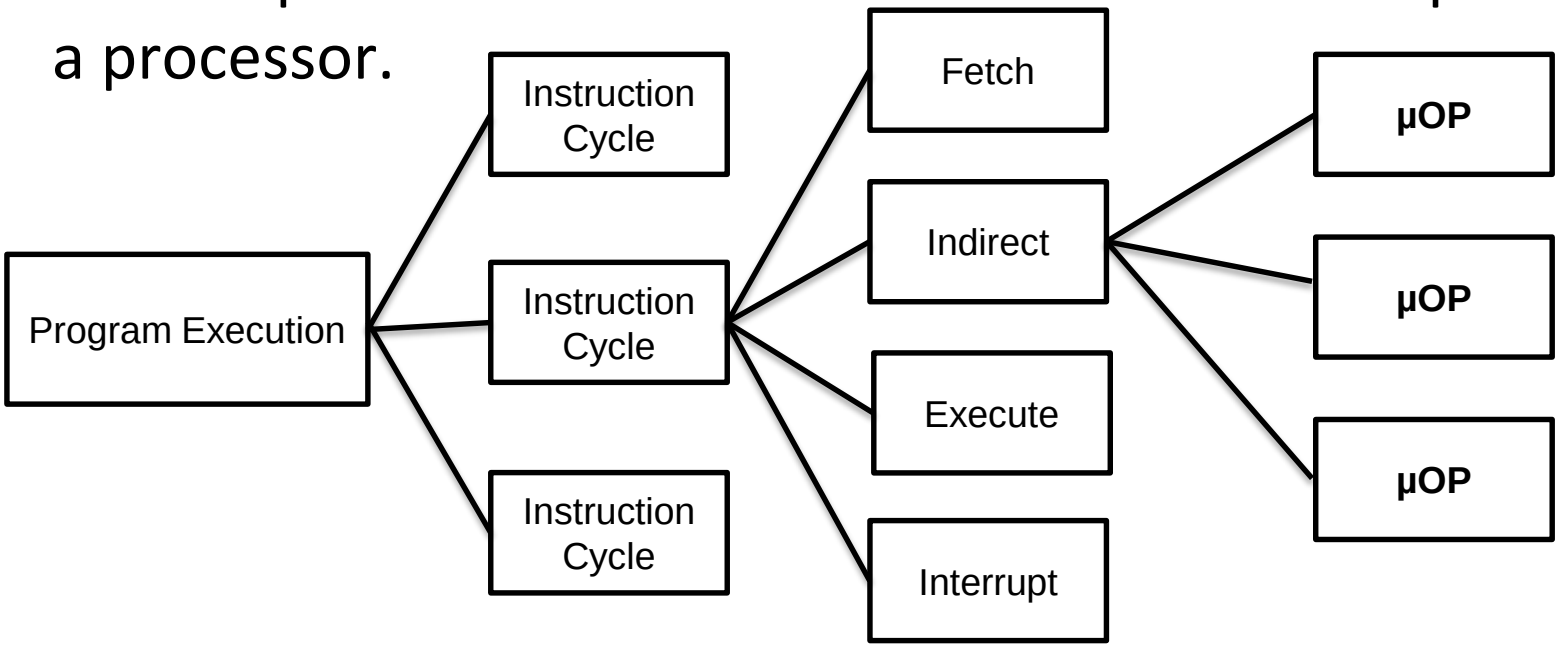
- Computer Organization and Architecture: Designing for Performance- William Stallings (8^{th} Edition) (CH-15)
  - Any later edition is fine

Overview

- The control unit of a processor performs two tasks:
  - It causes the processor to step through a series of **micro-operations** in the proper sequence, based on the program being executed.
  - It generates the **control signals** that cause each micro-operations to be executed.
- The control signals generated by the control unit cause the opening and closing of logic gates, resulting the transfer of data to and from registers and the operation of the ALU.
- One technique for implementing a control unit is referred to as **hardwired implementation**, in which the control unit is a combinatorial circuit. Its input logic signals, governed by the current machine instruction, are transferred into a set of output control signals.

Micro-Operations

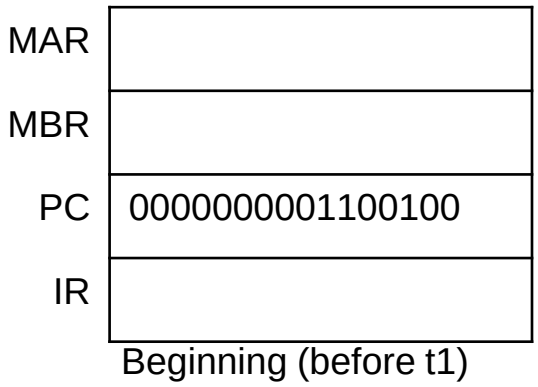
- The execution of a program consists of a sequential execution of instructions.
- Each instruction is executed during an instruction cycle made up of shorter sub-cycles (fetch, execute, interrupt etc.).
- The execution of each sub-cycle involves one or more shorter operations, that is micro-operations.
- Micro-operations are the functional or atomic operations of a processor.



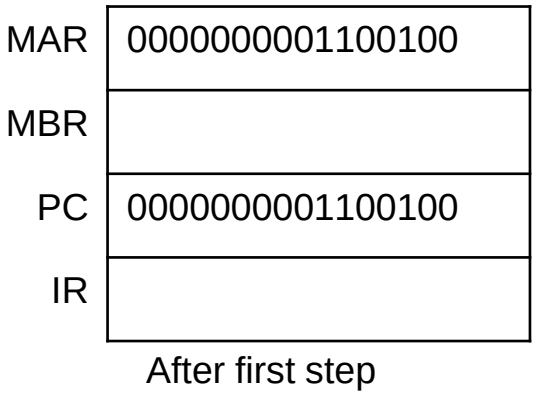
Constituent Elements of a Program Execution

The Fetch Cycle

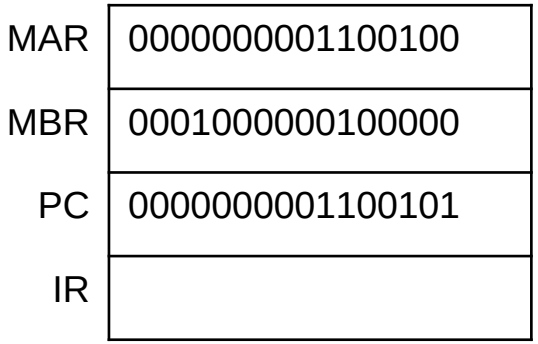
- Instruction to be fetched from memory at the beginning of each instruction cycle.
- Four registers are used: MAR, MBR, PC, IR.
- Following figure shows the sequence of events for the fetch cycle from the point of view of its effect on the processor registers.



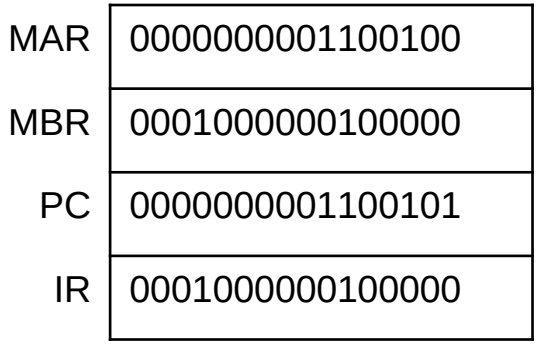
Beginning (before t1)



After first step



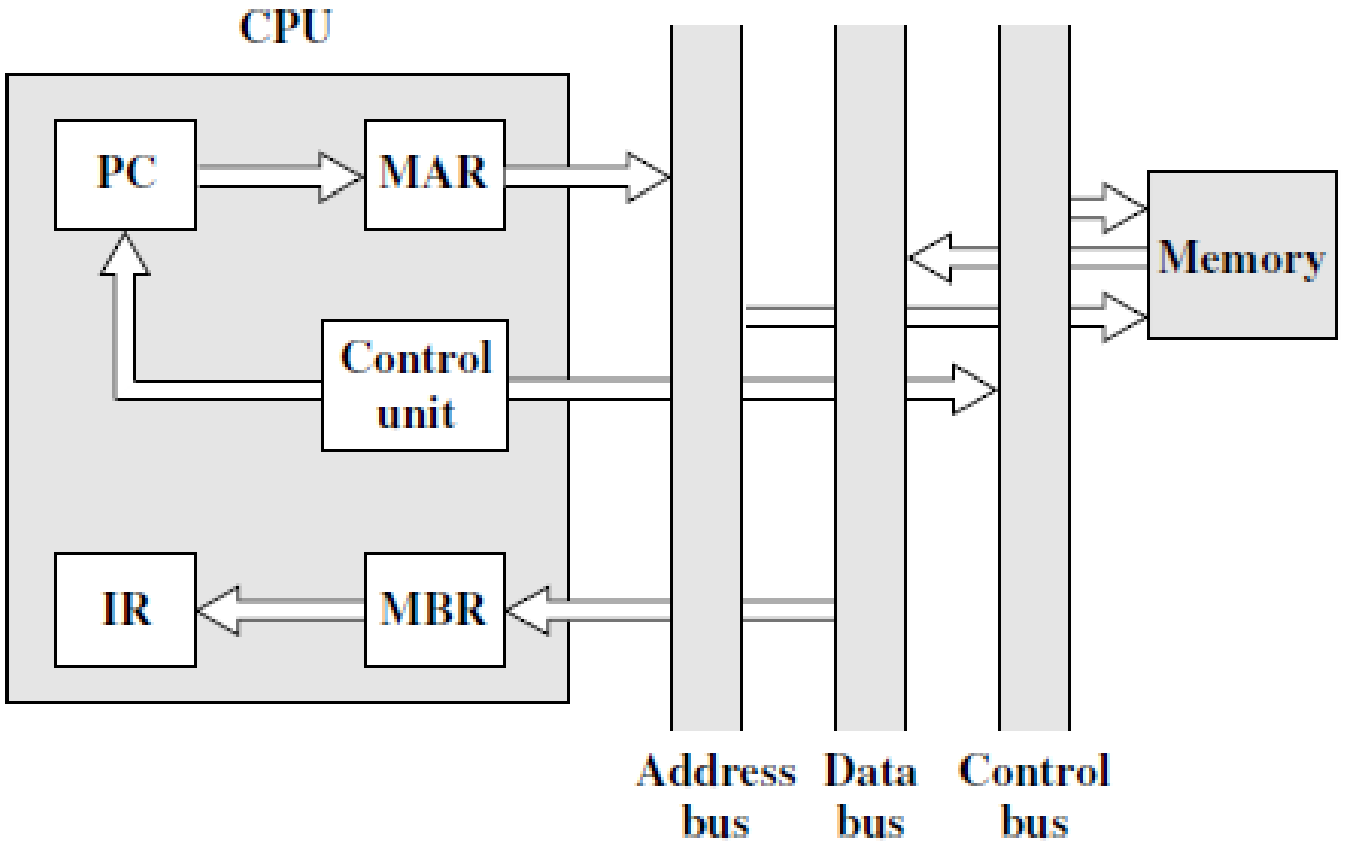
After second step



After third step

Continue…

Figure: Data Flow, Fetch Cycle



MBR = Memory buffer register
MAR = Memory address register
IR = Instruction register
PC = Program counter

Continue…

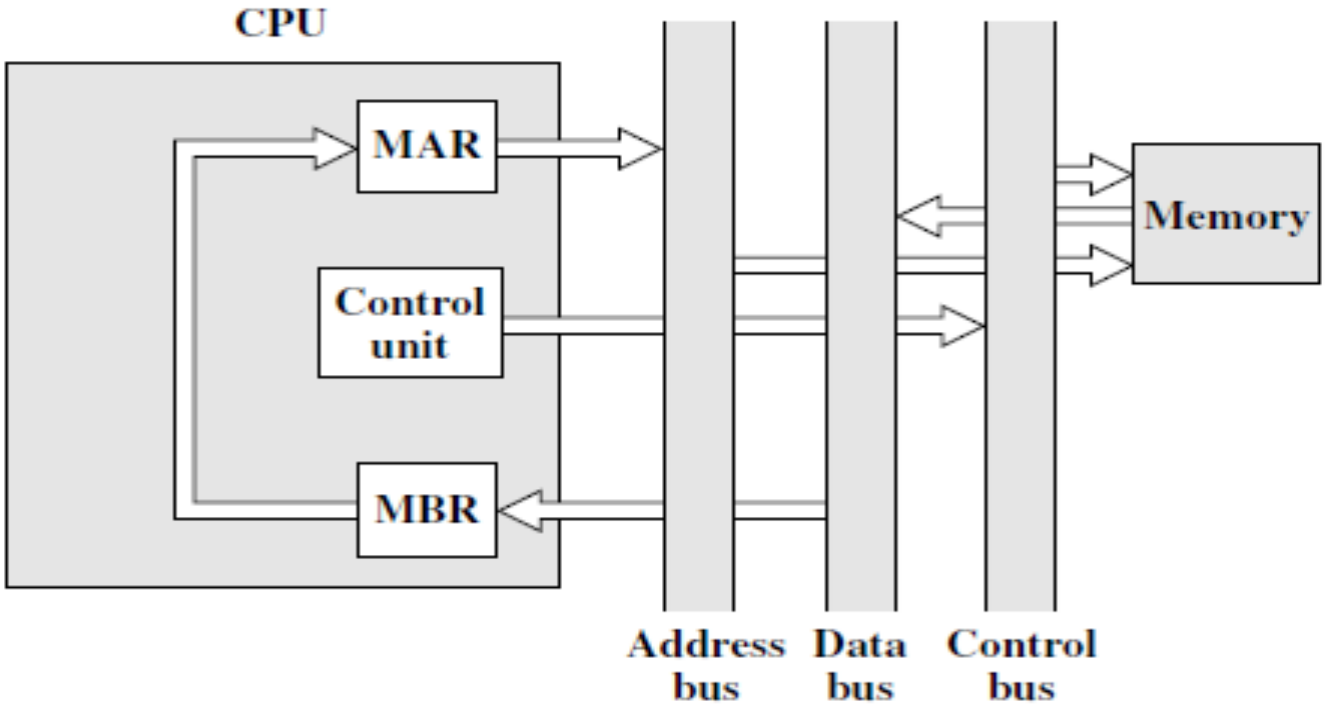
- Here , the simple fetch cycle consists of three steps and four micro-operations.
- Each micro-operation involves the movement of data into or out of a register.
- So long as the movements do not interfere with another, several of them take place during one step, saving time.
- Symbolically we can write this sequence of events as follows:
  - T1: MAR ← (PC)
  - T2: MBR ← Memory
    PC ← (PC) + I **Group**
  - T3: IR ← (MBR)
- Where I is the instruction length, T1, T2, T3 indicates time-unit.

Groupings of micro-operations follows two rules:

- Follow proper sequence: **MAR ← (PC)** before **MBR ← Memory**.
- Avoid conflicts. Read & write from same register – not same time.
  - **MBR ← Memory** and **IR ← MBR** not in same time unit.

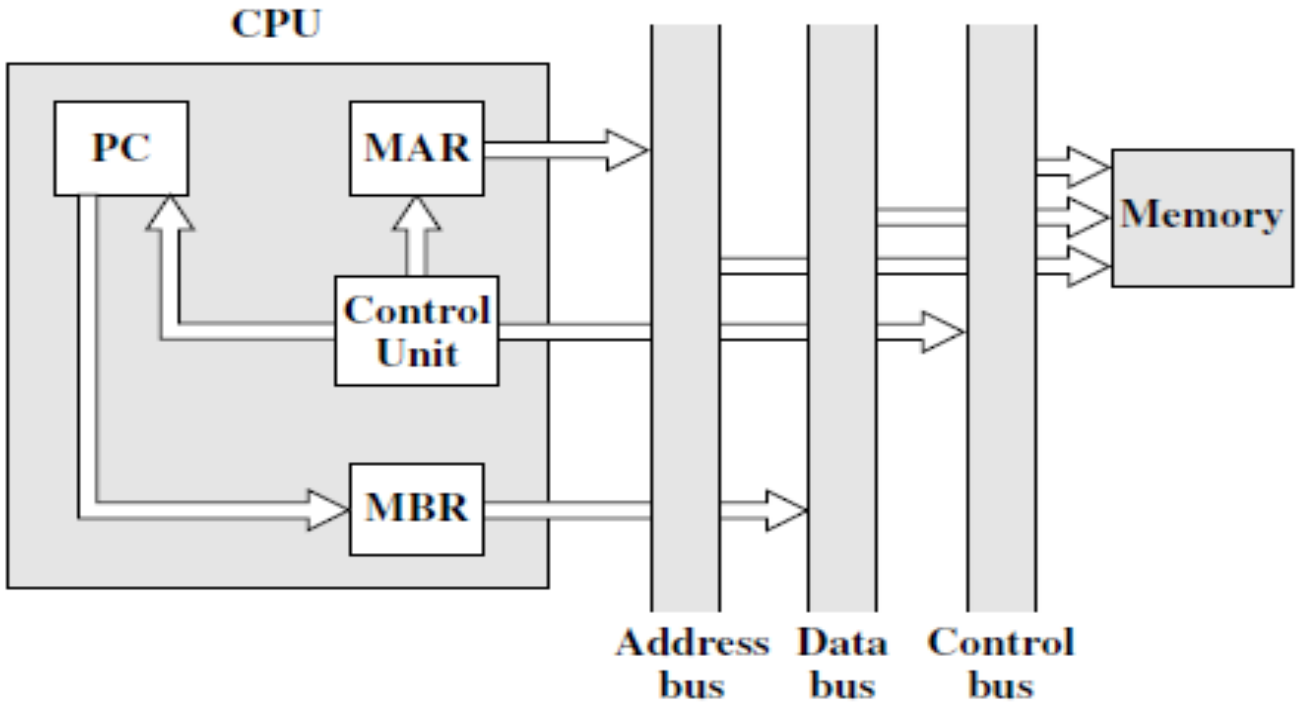
The Indirect Cycle

- Once an instruction is fetched, next step is to fetch source operands.
- If indirect addressing is used, an indirect cycle must precede the execute cycle.
  - T1: MAR ← (IR (Address))
  - T2: MBR ← Memory
  - T3: IR (address) ← (MBR (Address))



The Interrupt Cycle

- At the completion of the execute cycle, a test is made to determine whether any interrupt has occurred.
- If so, the interrupt cycle occurs.
  - T1: MBR ← (PC)
  - T2: MAR ← Save_Address
    PC ← Routine_Address
  - T3: Memory ← (MBR)



The Execute Cycle

- The fetch, indirect and interrupt cycles are simple and predictable.
- Each involves a small, fixed sequence of micro-operations and, in case, the same micro-operations are repeated each.
- This is not true of the execute cycle. Because of the variety Opcodes, there are a number of different sequences of micro-operations that can occur.
- For an add instruction:

  Add **R1, X** [Adds the contents of location X to register R1].

  - T1: MAR ← (IR (Address))
  - T2: MBR ← Memory
  - T3: R1 ← (R1) + (MBR)

Continue…

- Another example: **Conditional**

  ISZ **X** [Increment the content of X by 1. Skip next instruction if result is 0].
  - T1: MAR ← (IR (Address))
  - T2: MBR ← Memory
  - T3: MBR ← (MBR)+1
  - T4: Memory ← (MBR)
    If ((MBR) = 0) then (PC ← (PC)+1)

- Another example: **Subroutine call instruction**

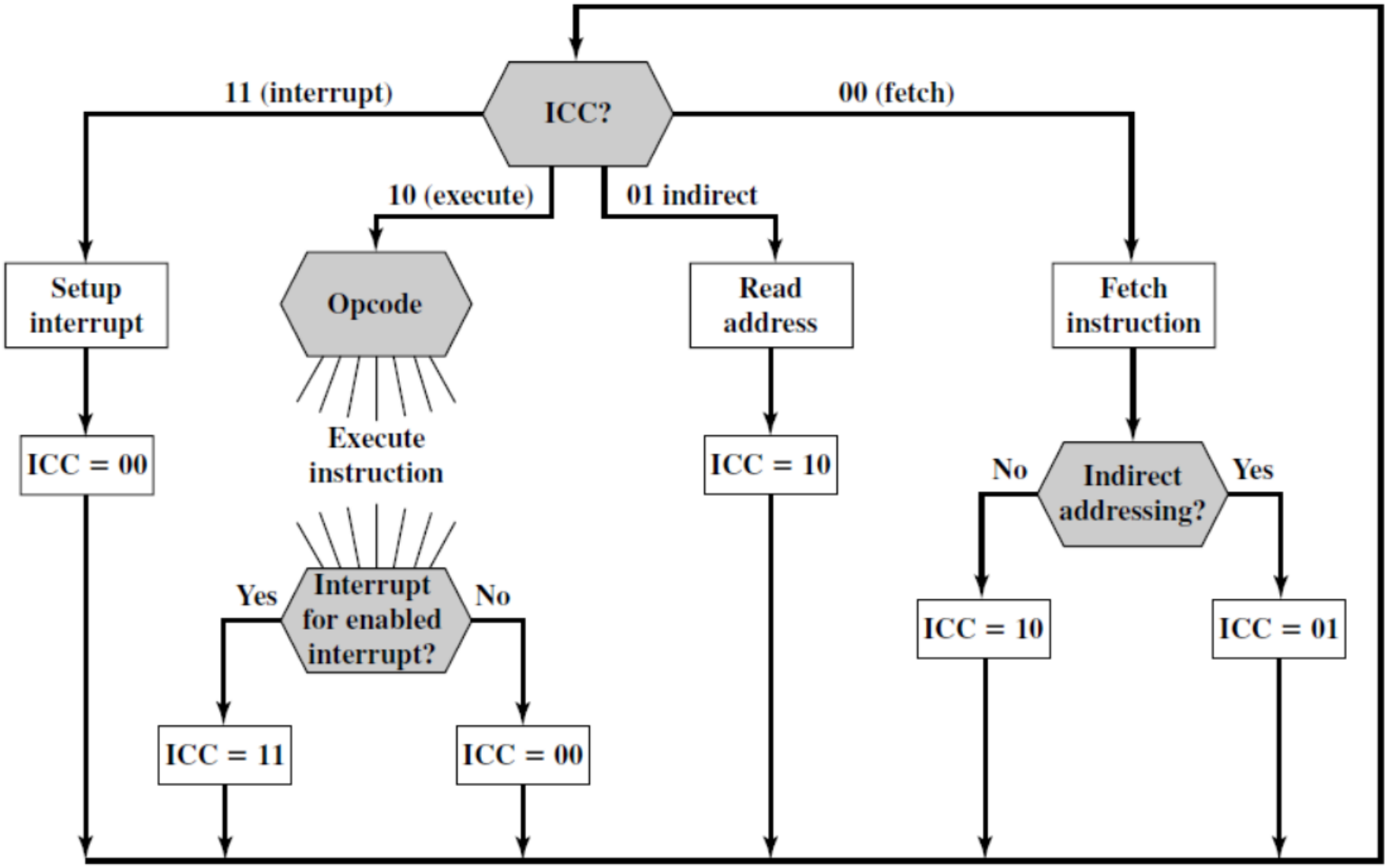
  BSA **X** [Branch and save address to location X in memory].
  - T1: MAR ← (IR (Address))
    MBR ← (PC)
  - T2: PC ← (IR (Address))
    Memory ← (MBR)
  - T3: PC ← (PC) + I

The instruction Cycle

- Consists of Fetch cycle, indirect cycle, execute cycle and interrupt cycles together.
- Here, we use a new 2-bit register called the instruction cycle code (ICC) which designates the state of the processor in terms of which portion of the cycle it is in:
  - 00: **Fetch**; 01: **Indirect**;
  - 10: **Execute**; 11: **Interrupt**.
- At the end of each of four cycles, the ICC is set appropriately.
- The indirect cycle is always followed by execute cycle.
- The interrupt cycle is always followed by fetch cycle.
- For both fetch cycle and execute cycle, the next cycle depends on the state of the system.
- Following flowchart defines the complex sequence of micro-operations depending only on the instruction sequence and the interrupt pattern.

Continue…

Flowchart for Instruction Cycle:



Thank you!