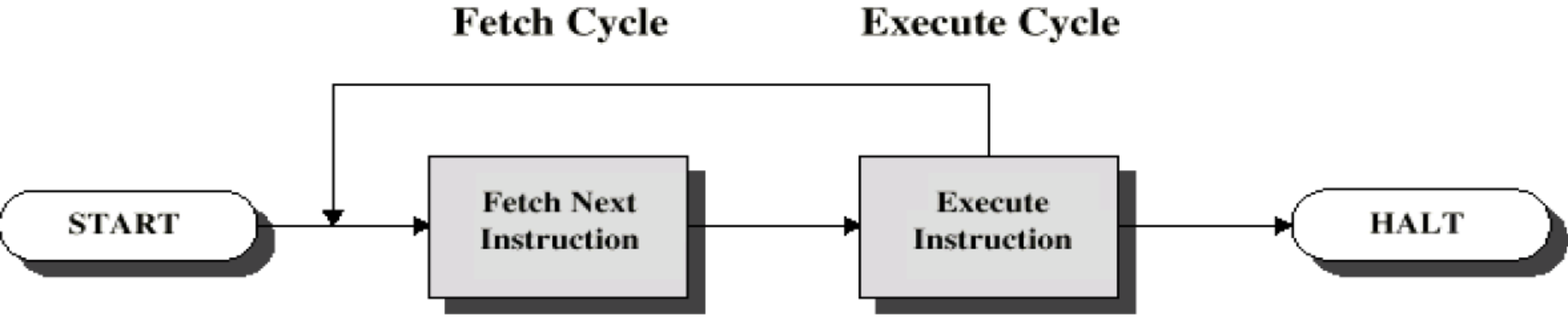
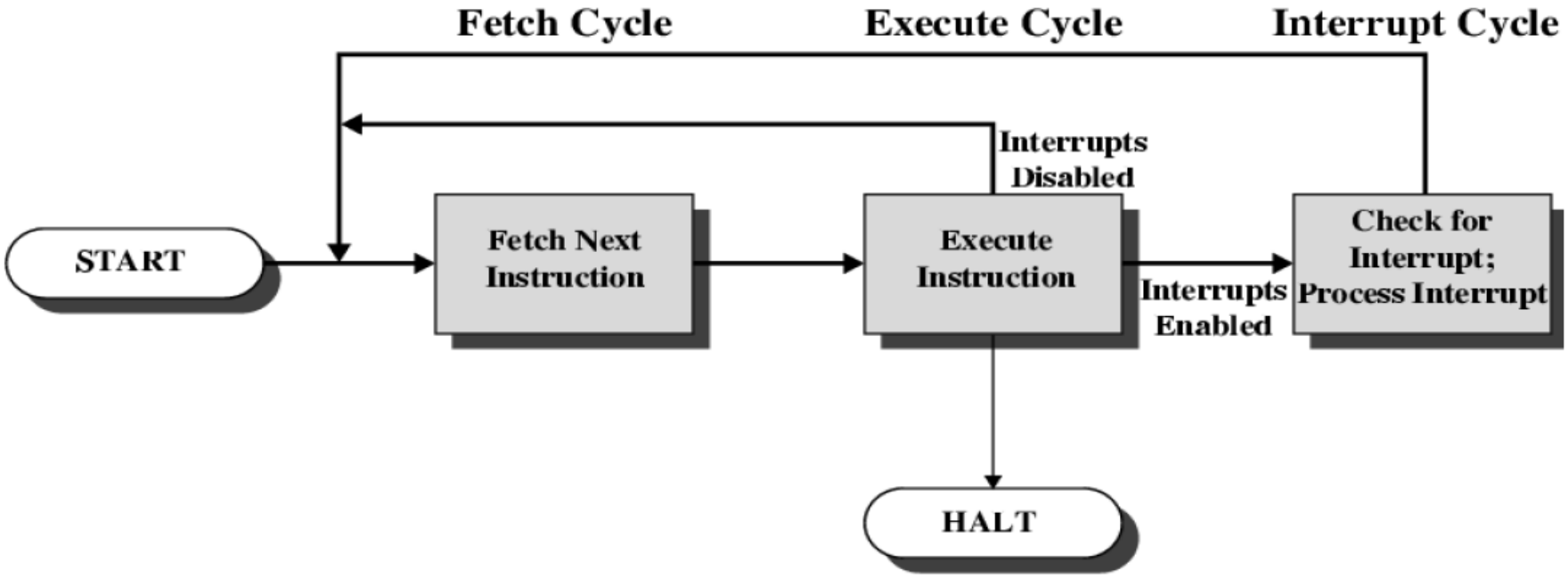
Instruction Cycle

Instruction Cycle

- Two steps:
  - Fetch
  - Execute



Instruction Cycle with Interrupts



Instruction Cycle

- It is the time in which a single instruction is fetched from memory, decoded, and executed
- An **Instruction Cycle** requires the following subcycle:
  - **FETCH**
  - **EXECUTE**
  - **INDIRECT**
  - **INTERRUPT**

Instruction Cycle

- **Fetch**

  Read next instruction from memory into the processor

- **Indirect Cycle (Decode Cycle)**

  May require memory access to fetch operands, therefore more memory accesses.

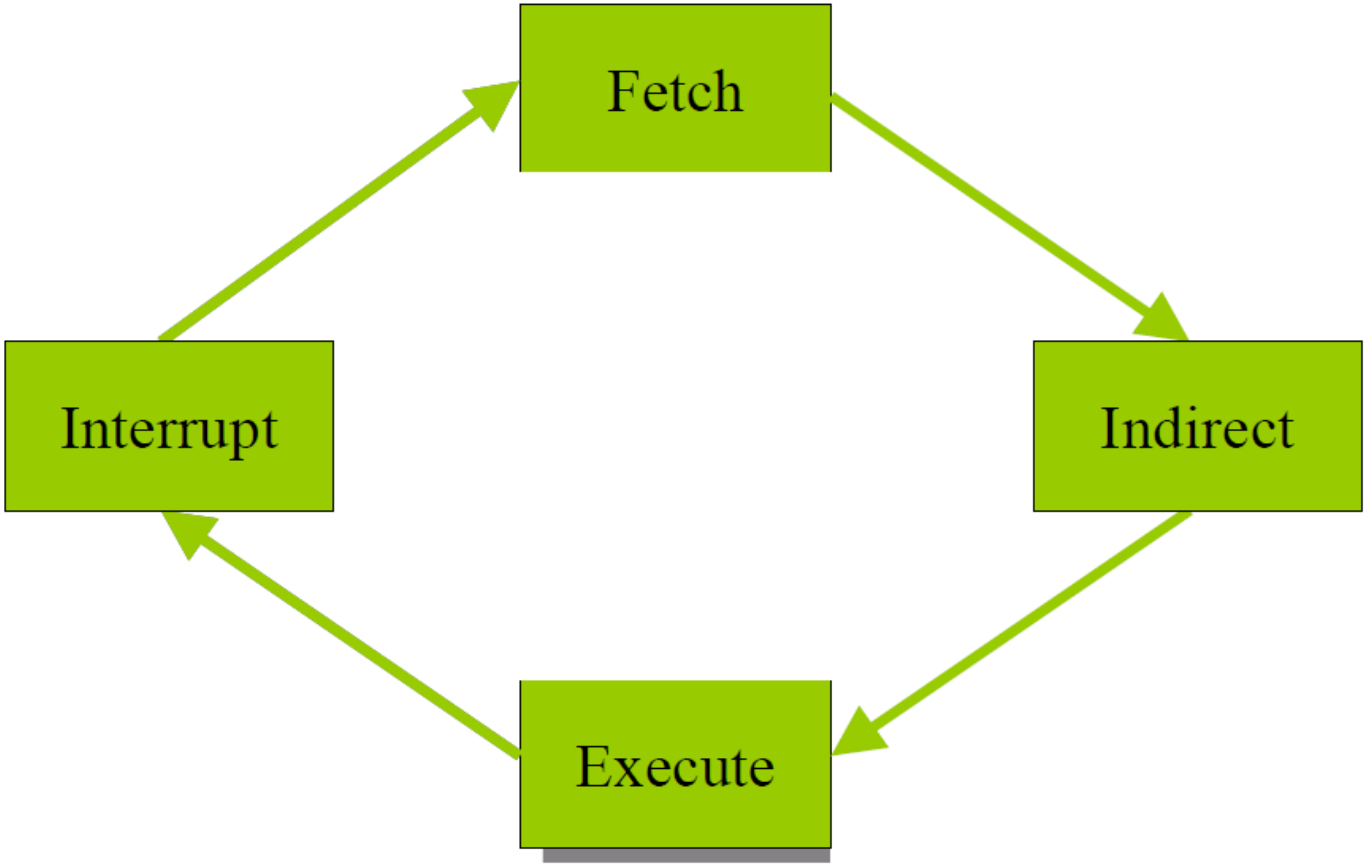
- **Interrupt**

  Save current instruction and service the interrupt

- **Execute**

  Interpret the opcode and perform the indicated operation

Instruction Cycle



Instruction Cycle State Diagram

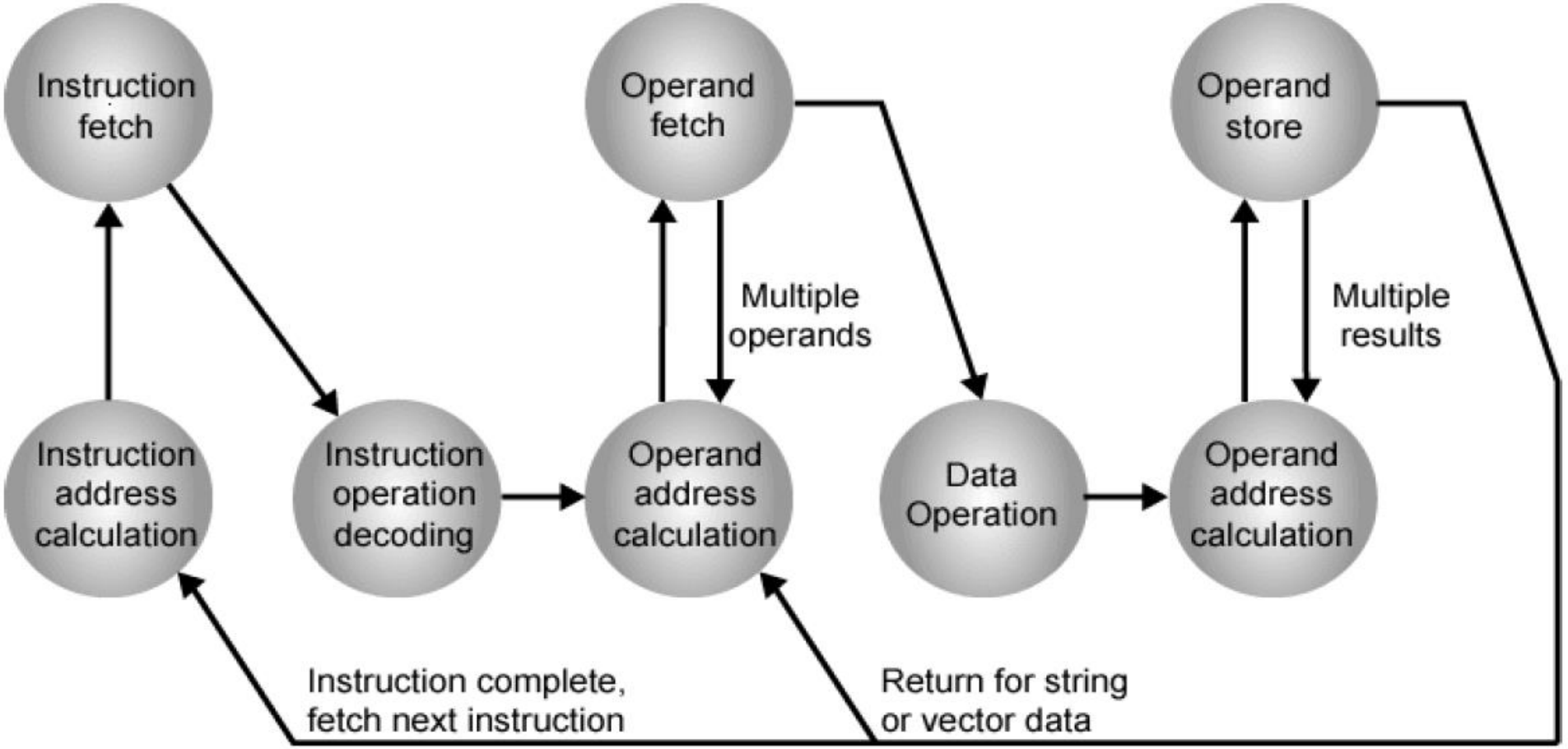


Figure 10.1 Instruction Cycle State Diagram

Instruction Cycle State Diagram

- This illustrates more correctly the nature of the instruction cycle.
- Once an instruction is fetched, its operand specifiers must be identified.
- Each input operand in memory is then fetched, and this process may require indirect addressing.
- Register-based operands need not be fetched.
- Once the opcode is executed, a similar process may be needed to store the result in main memory.

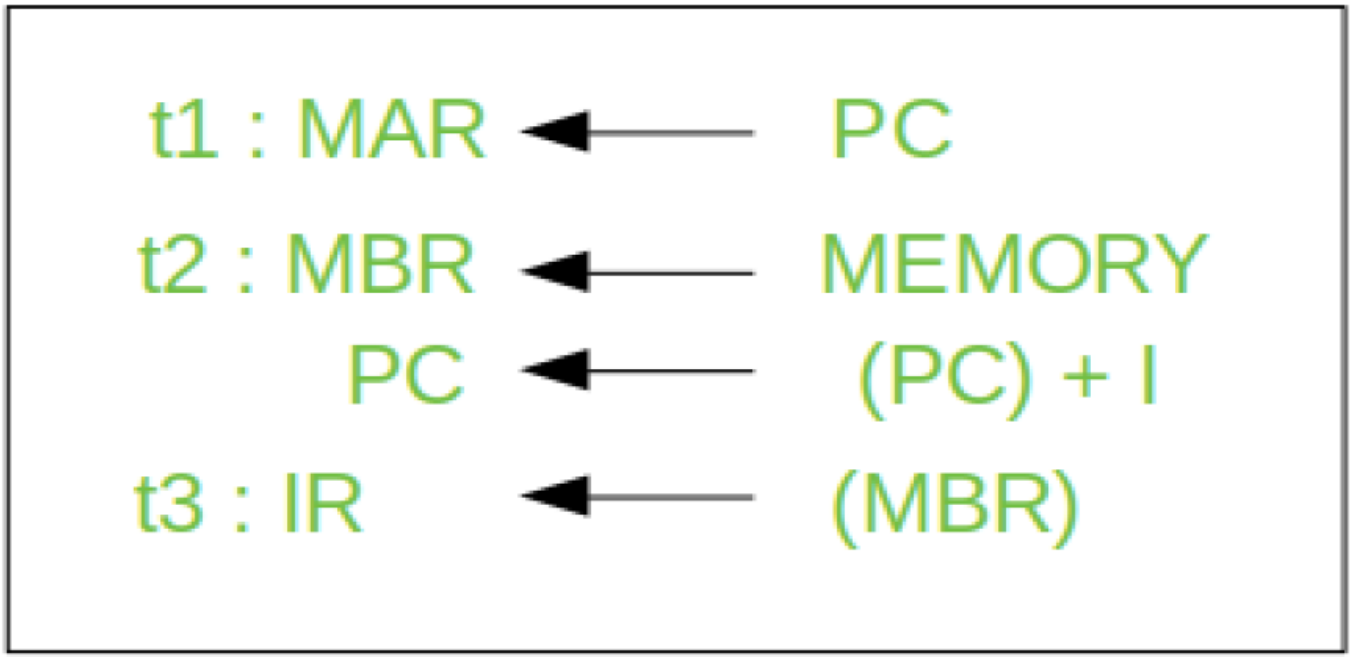
Registers

- **Memory Address Register (MAR)**
  - Connected to address bus
  - Specifies address for read or write op
- **Memory Buffer Register (MBR)**
  - Connected to data bus
  - Holds data to write or last data read
- **Program Counter (PC)**
  - Holds address of next instruction to be fetched
- **Instruction Register (IR)**
  - Holds last instruction fetched/current instruction being executed

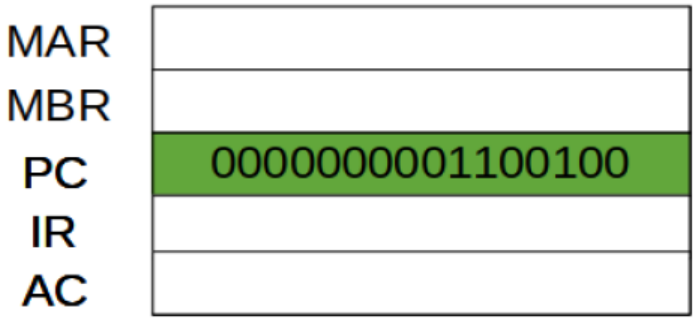
Fetch Cycle

- **Program Counter (PC)** holds address of next instruction to be fetched
- Processor fetches instruction from memory location pointed to by PC
- Increment PC
  - Unless told otherwise
- Instruction loaded into Instruction Register (IR)
- Processor interprets instruction and performs required actions

Fetch Sequence (symbolic)



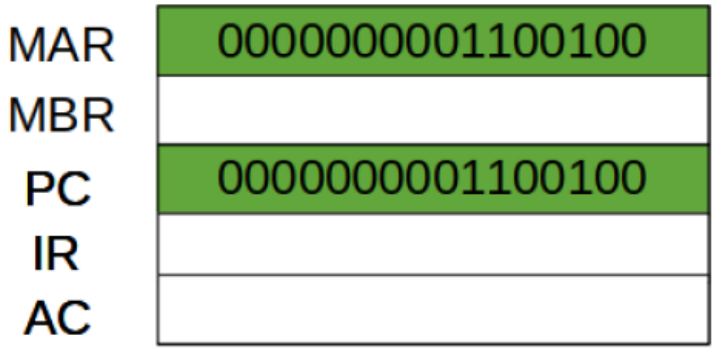
- **The Fetch Cycle –**
  At the beginning of the fetch cycle, the address of the next instruction to be executed is in the *Program Counter*(PC).



BEGINNING

Step 1:

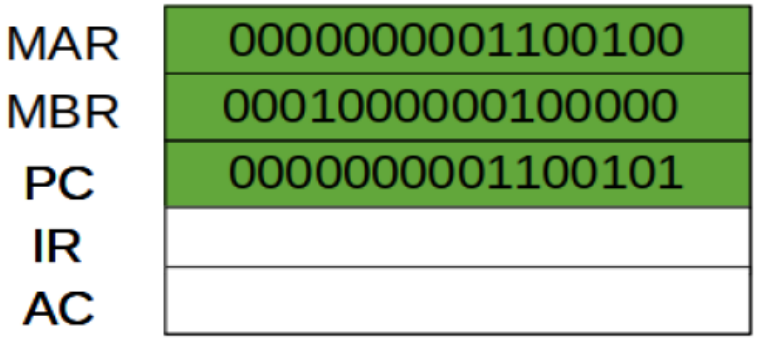
- The address in the program counter is moved to the memory address register(MAR), as this is the only register which is connected to address lines of the system bus.



FIRST STEP

Step 2:

- The address in MAR is placed on the address bus, now the control unit issues a READ command on the control bus, and the result appears on the data bus and is then copied into the memory buffer register(MBR).
- Program counter is incremented by one, to get ready for the next instruction.(These two action can be performed simultaneously to save time)



SECOND STEP

Step 3:

- The content of the MBR is moved to the instruction register(IR)

MAR	0000000001100100
MBR	0001000000100000
PC	0000000001100100
IR	0001000000100000
AC	

THIRD STEP

Indirect Cycle

Step 1:

The address field of the instruction is transferred to the MAR.

This is used to fetch the address of the operand.

Step 2:

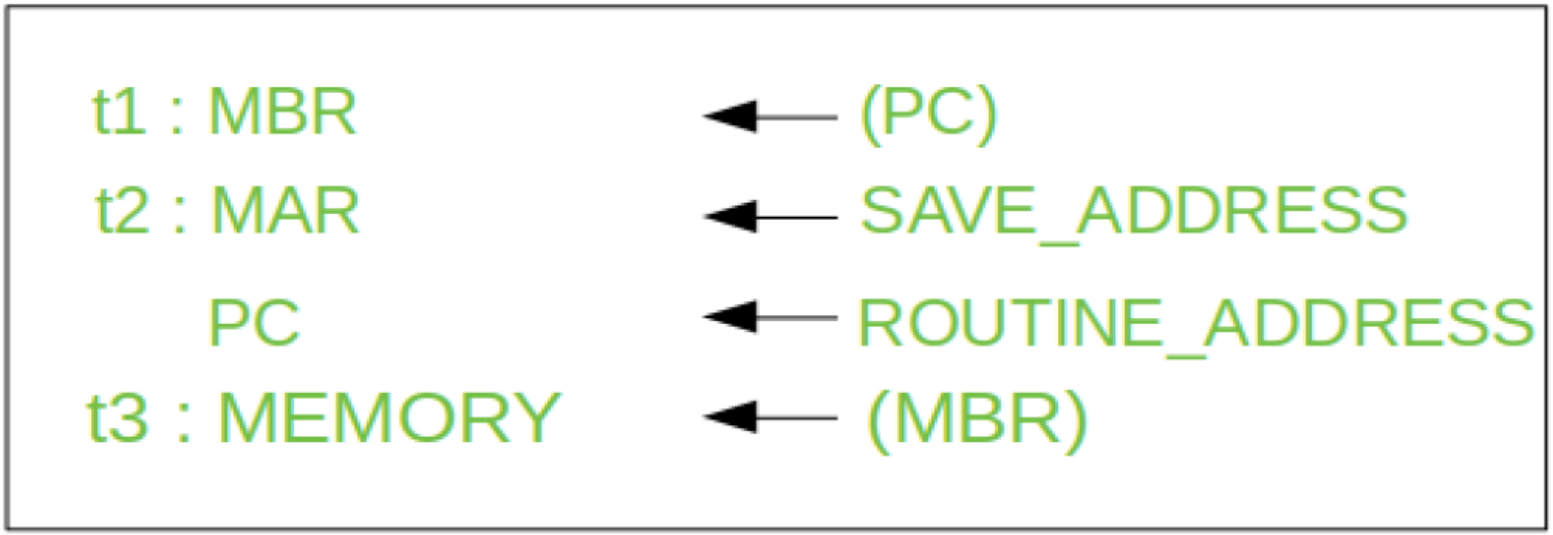
The address field of the IR is updated from the MBR.(So that it now contains a direct addressing rather than indirect addressing)

Step 3:

The IR is now in the state, as if indirect addressing has not been occurred.

Interrupt Cycle

- At the completion of the Execute Cycle, a test is made to determine whether any enabled interrupt has occurred or not.
- If an enabled interrupt has occurred then Interrupt Cycle occurs.
- The nature of this cycle varies greatly from one machine to another.


t1 : MBR ← (PC)
t2 : MAR ← SAVE_ADDRESS
PC ← ROUTINE_ADDRESS
t3 : MEMORY ← (MBR)

- Step 1: Contents of the PC is transferred to the MBR, so that they can be saved for return.

  Step 2: MAR is loaded with the address at which the contents of the PC are to be saved.

  PC is loaded with the address of the start of the interrupt-processing routine.

  Step 3: MBR, containing the old value of PC, is stored in memory.

  - **Note:** In step 2, two actions are implemented as one micro-operation. However, most processor provide multiple types of interrupts, it may take one or more micro-operation to obtain the save_address and the routine_address before they are transferred to the MAR and PC respectively.a

Execute Cycle (ADD)

Different for each instruction

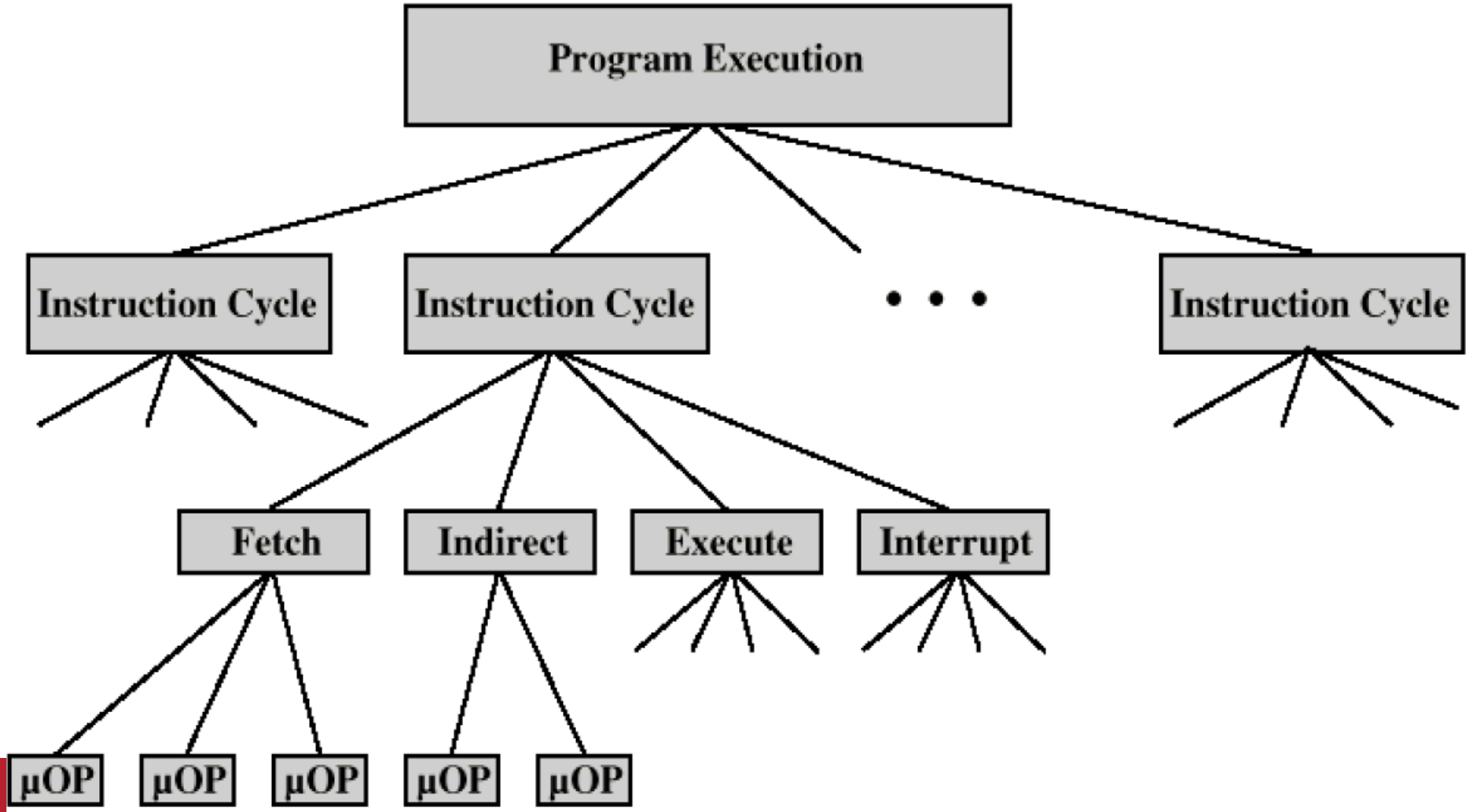
e.g. ADD R1,X - add the contents of location X to Register 1 , result in R1

$$t_1 : IR \rightarrow MAR$$

$$t_2 : MEMORY \rightarrow MDR$$

$$t_3 : MDR + R1 \rightarrow R1$$

Constituent Elements of Program Execution



Micro-Operations

- A computer executes a program
- Fetch/execute cycle
- Each cycle has a number of steps
- Called micro-operations
- Each step does very little
- Atomic operation of CPU

Types of Micro-operation

- Transfer data between registers
- Transfer data from register to external
- Transfer data from external to register
- Perform arithmetic or logical ops