AMRITA VISHWA VIDYAPEETHAM

AHEAD Online

Addressing Modes

Ms. Prathibha Prakash
Department of Computer Science
Amrita Vishwa Vidyapeetham

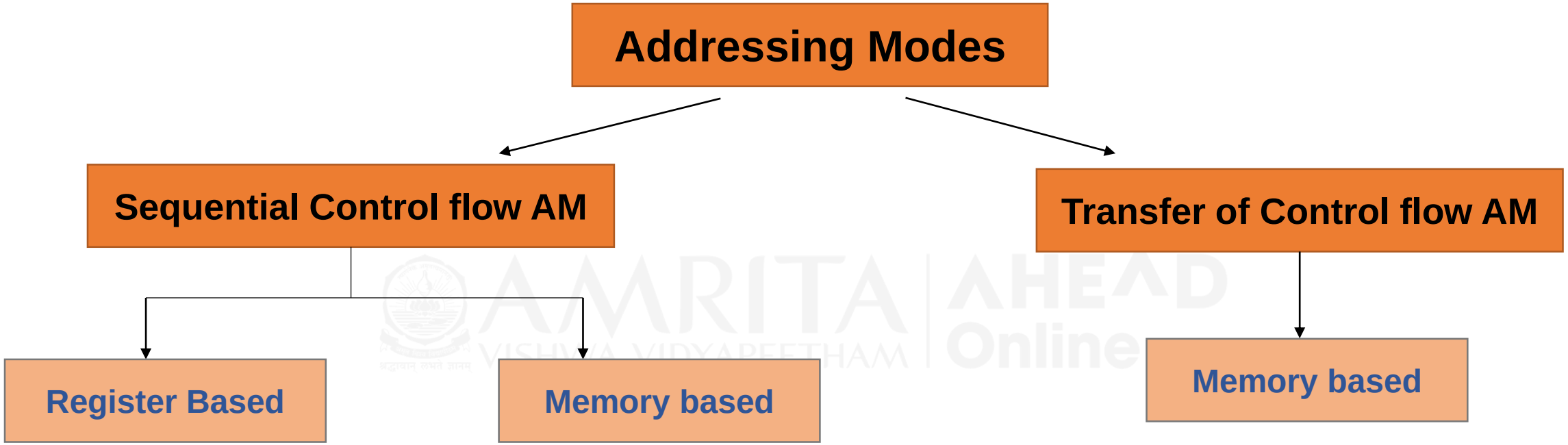
Objective

- Addressing Modes in detail

Introduction

- The opcode (operation field) of an instruction specifies the operation to be performed.
- Operation is performed on the operands(data) stored either in register file or main memory.
- Operands chosen during program execution depends on the addressing mode of the instruction.
- Addressing mode is all about how we address the operands

Addressing Modes



Definitions

- Effective Address (EA) : Final address where the data we want resides.
- Determined by adding any combination of three address elements: displacement, base and index.
- Displacement : Immediate value given in the instruction.
- Base : Content of base register BX or BP
- Index : Content of index register SI or DI

Register based AM

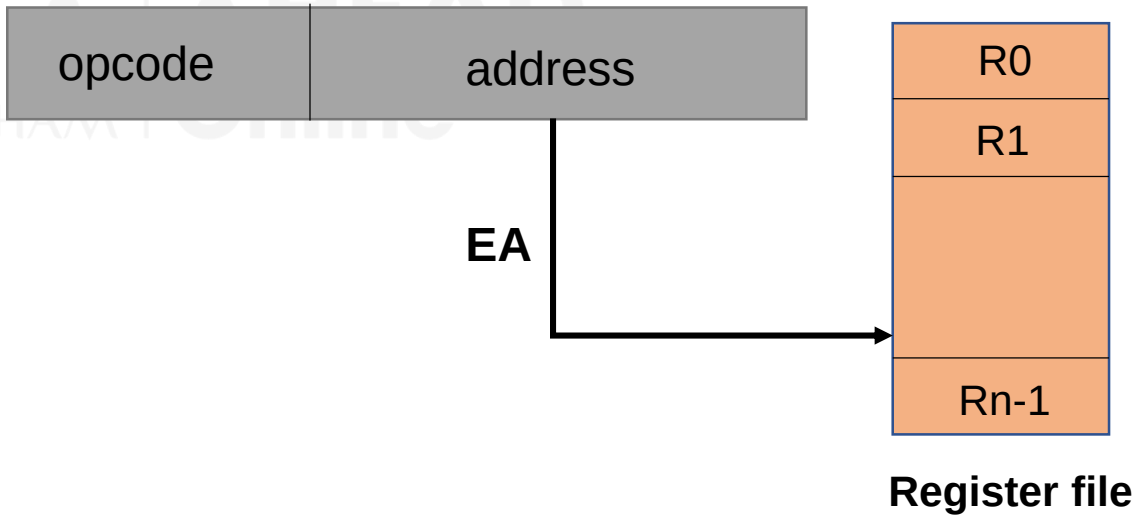
- The operands are placed in one of the general-purpose register.
- **ADD R0, R1** (opcode followed by an address in the register file)
- R0 ← R0 + R1

Write **Read** **Read**

Read R0 and R1 from register file to ALU

Perform the arithmetic operation

Write the result back to R0



Implied /Implicit AM

- In this mode, the data information is present in the opcode itself
- Zero address instructions (only opcode no address) are designed with the implied AM.
- Stack CPU organization :Address is implied, take the operands needed to perform the operation from the stack and place the result back to stack.
- ADD, STC , CLC

Immediate AM

- The instruction has operand field rather than address field.
- The operand field contain the actual operand(data) to be used.
- MOV R0 , $(23)_{16}$ (R0 gets the value 23)

data

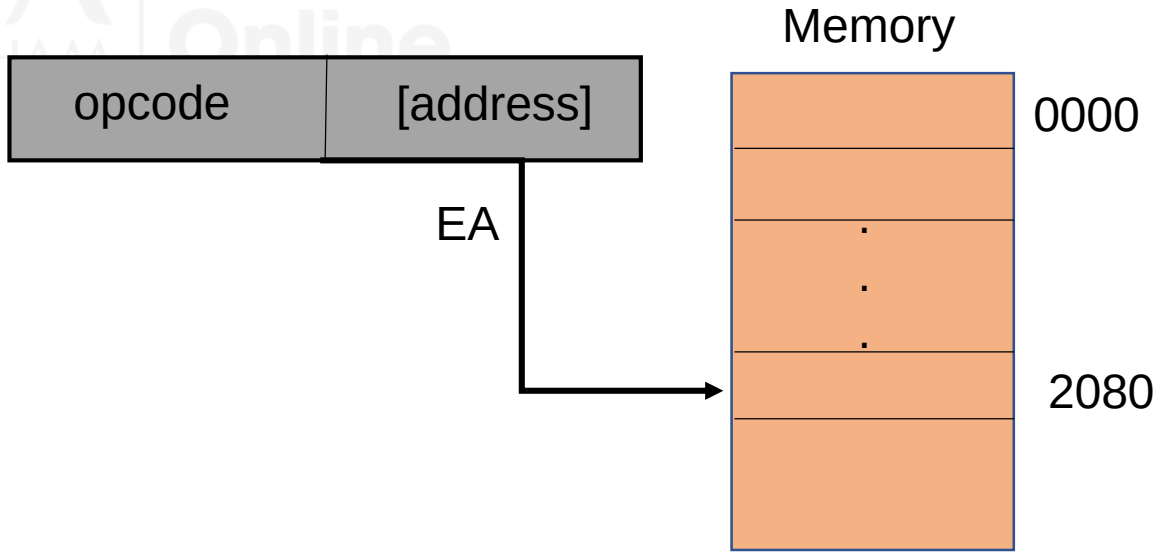
- Whatever we have in instruction is the data/constant
- Can immediately access data without any register or memory access

Direct/Absolute AM

- The operand resides in memory and its address is given directly by the address field of instruction.
- Address field itself is the effective address
- ADD R0, [2080]

Direct AM

R0 ⟵ R0 + M[2080]



Summary

- Addressing Modes

Reference

- M Morris Mano - Computer System Architecture - PHI - Third Edition
- P Pal Chaudhuri - Computer Organization and Design - PHI - Second Edition
- Thomas C Bartee - Digital Computer Fundamentals - Tata Mc Graw Hill - Sixth Edition

Addressing Modes-II

Ms. Prathibha Prakash
Department of Computer Science
Amrita Vishwa Vidyapeetham

Objective

- Addressing Modes

Indirect AM

- The address field of instruction gives the address where the EA is stored in memory.
- Control fetches the instruction from memory and uses its address part to again read the EA.
- Within indirect AM, there are 2 broad subcategories
- Register Indirect AM
- Memory indirect AM

Register Indirect AM

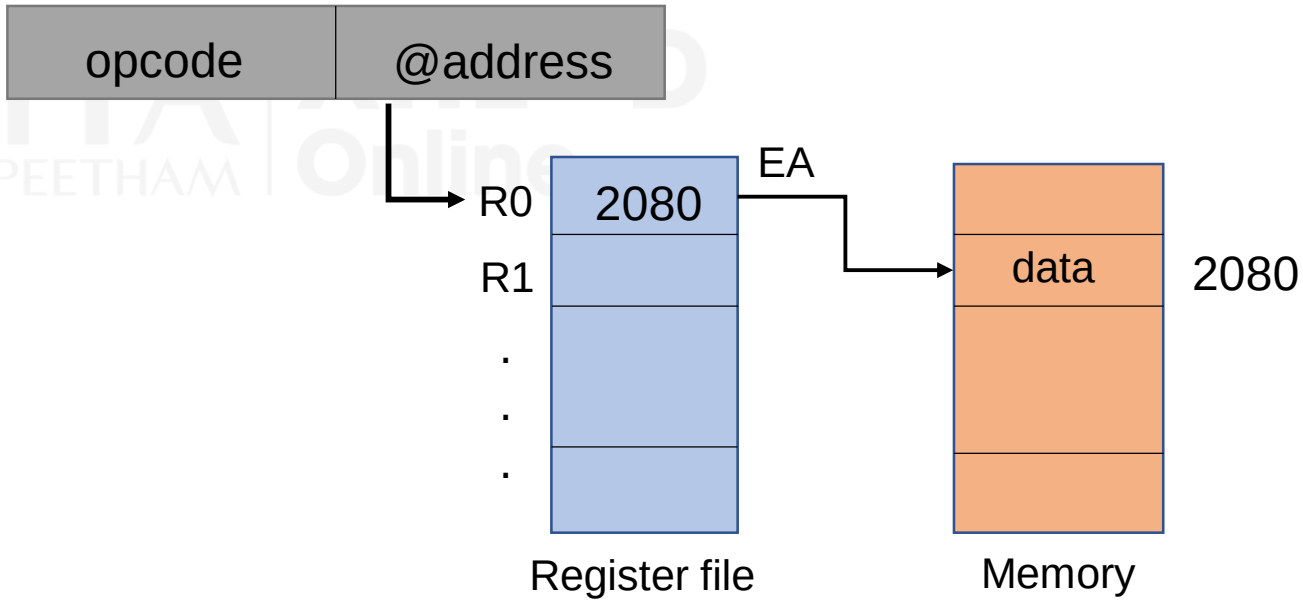
- Way of differentiation is using @ symbol.
- ADD @R0 , R1 (@R0 means R0 contains the EA where actual data resides)

Register Indirect AM

- Let R0 contain the EA 2080

  Read the data from 2080 in memory

  M[2080] ⟵ M[2080] + R1



Memory Indirect AM

- ADD @2080 , R1

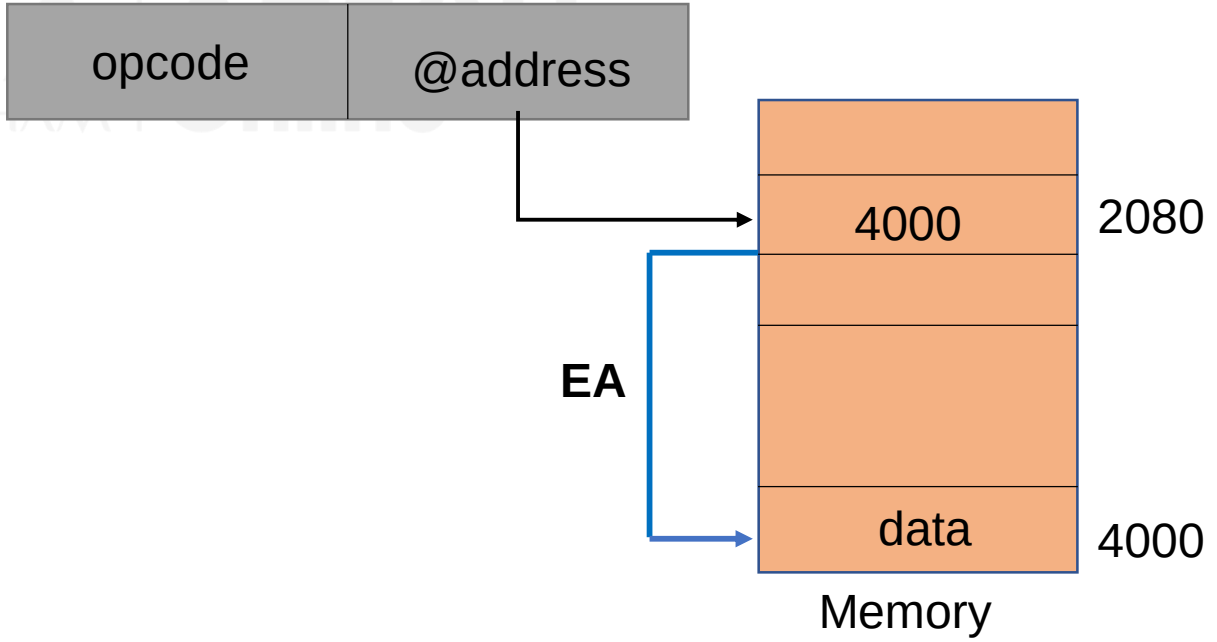
  Memory Indirect AM

  Register AM

- Control Unit access main memory location 2080 where EA is stored , go to EA and read the data.

- M[4000] ← M[4000] + R1

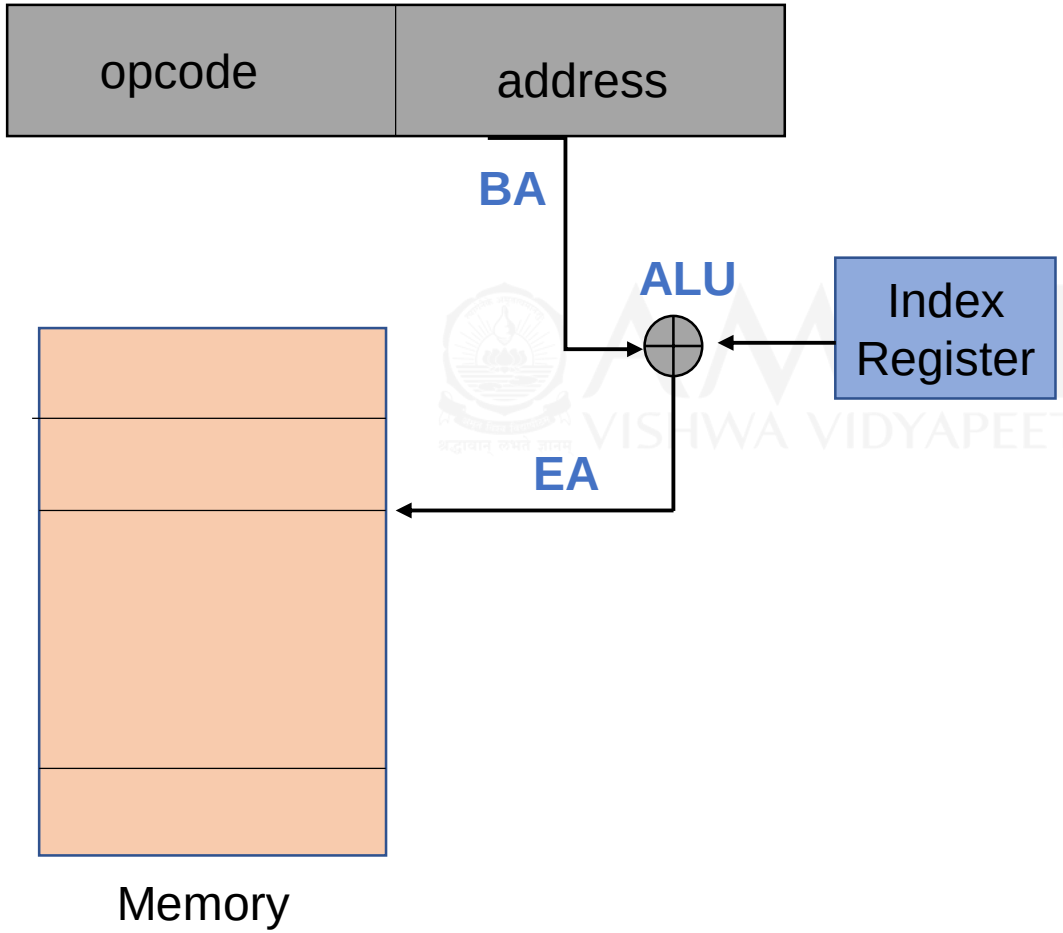
  EA where the actual data resides

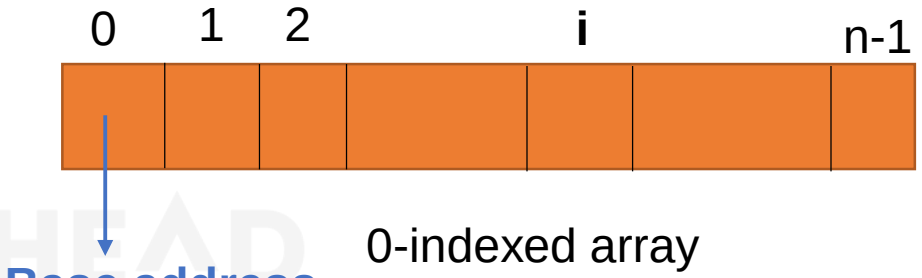


Indexed AM

- The content of an index register is added to the address part of instruction to obtain the EA.
- The index register is a special CPU register that contains an index value.
- Address field of instruction defines the beginning address of a data array in memory.
- This is useful when we want to randomly access any element of array.

Indexed AM





MOV R1 , BX , IDX

Indexed AM

R1 ⟵ M [BX + IDX]

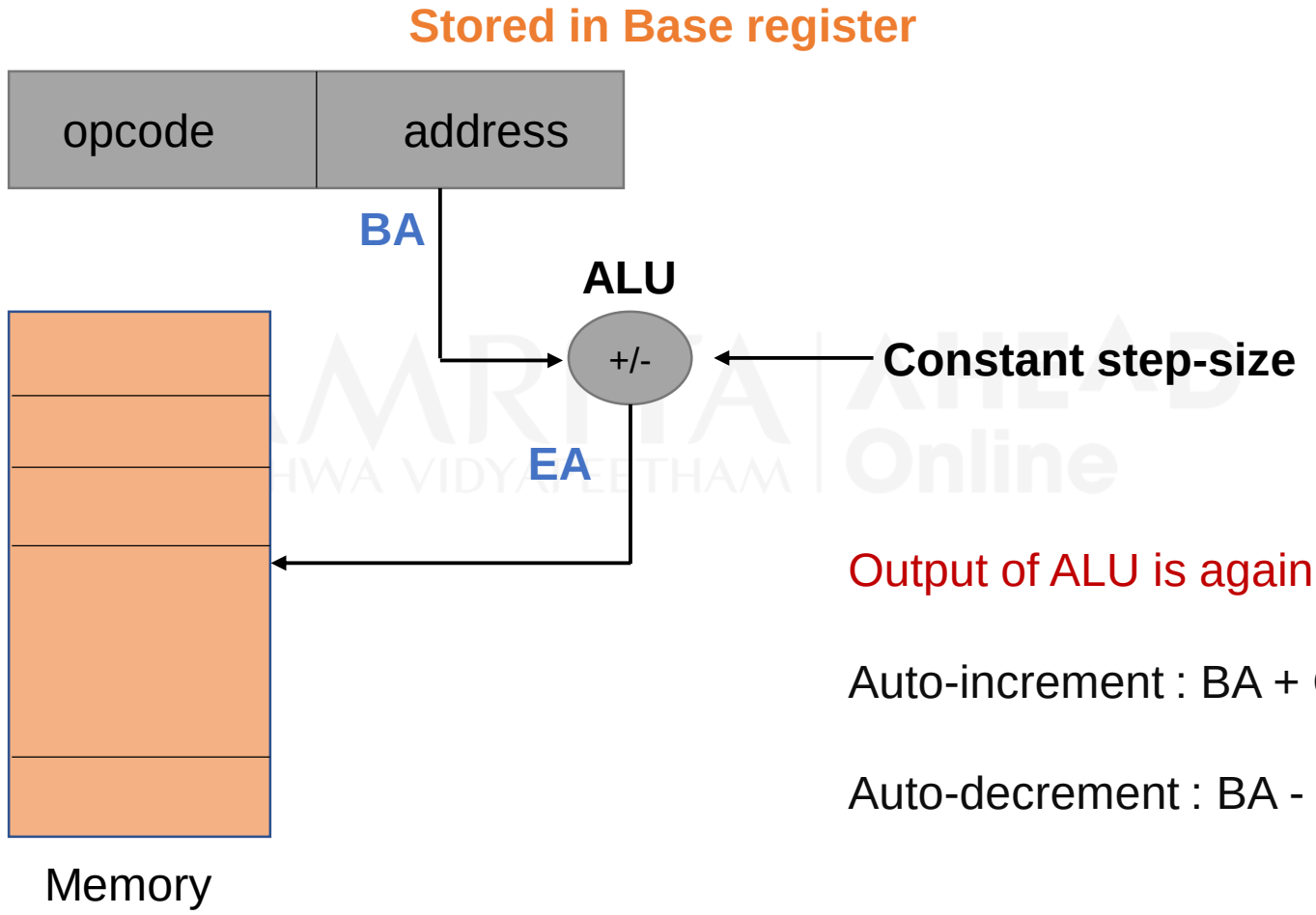
Auto Indexed AM

- The register is incremented or decremented after (or before) its value is used to access memory.
- Auto indexed AM only consist of base address with a constant value.
- Of two types :

  Auto increment AM : add constant step-size

  Auto decrement AM : subtract constant step-size

Auto Indexed AM



Output of ALU is again stored back in BR

Auto-increment : BA + Constant = EA

Auto-decrement : BA - Constant = EA

Summary

Addressing Modes

Reference

- M Morris Mano - Computer System Architecture - PHI - Third Edition
- P Pal Chaudhuri - Computer Organization and Design - PHI - Second Edition
- Thomas C Bartee - Digital Computer Fundamentals - Tata Mc Graw Hill - Sixth Edition

Transfer of control AM

Ms. Prathibha Prakash
Department of Computer Science
Amrita Vishwa Vidyapeetham

Objective

Transfer of Control addressing Modes

Transfer of Control AM

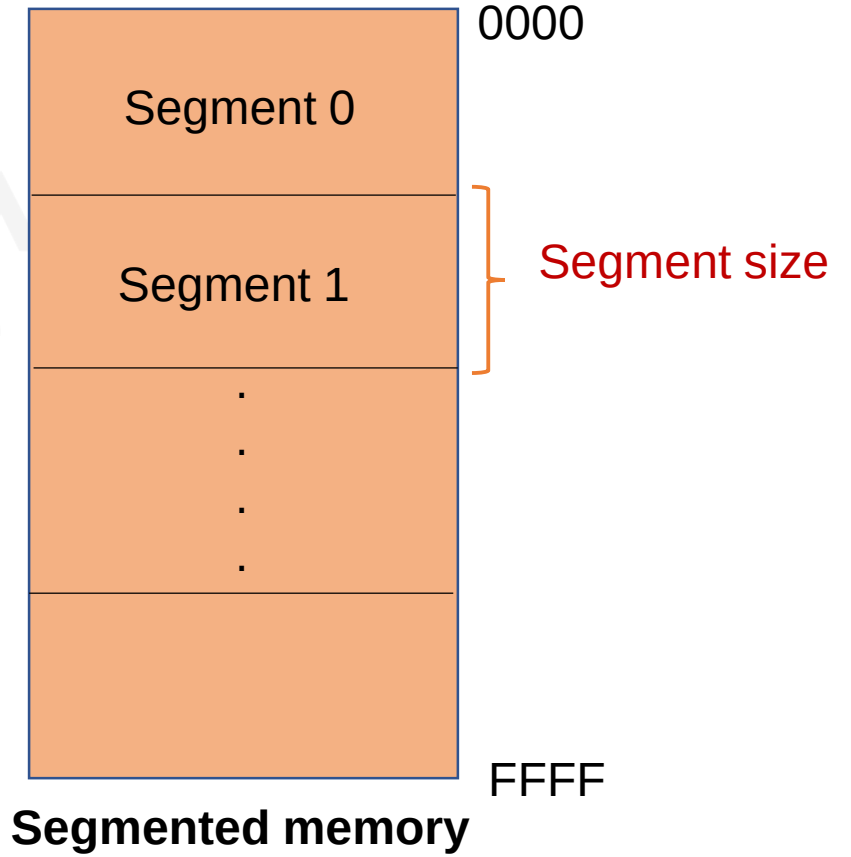
- To transfer the control, after executing an instruction next instruction to be executed is not in sequential manner.
- Transfer the control of execution to some other location.
- Using jump, branch or goto …. statements
- 2 types of AM :

  PC relative AM

  Based/ base register AM

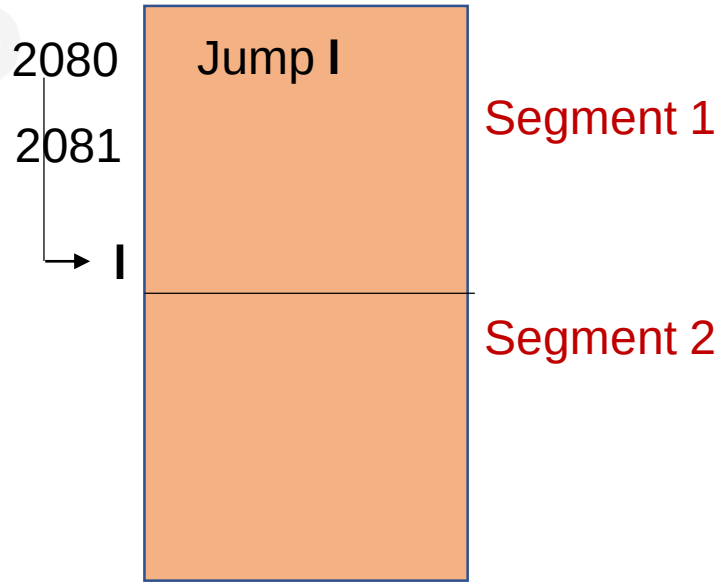
Segmented memory

- Divide the memory into equal sized segments
- Any segmented memory have segment number and segment offset



PC relative AM

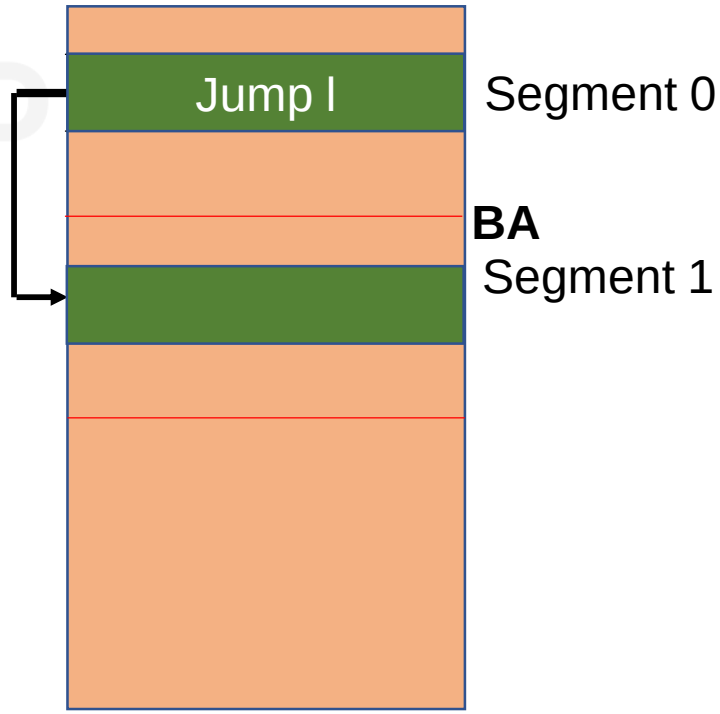
- To implement intra segment transfer of control.
- Jump to the address location of same segment we use PC relative AM.
- EA is obtained by adding displacement to PC.
- EA = PC + Address field value



Base address AM

- To implement inter segment transfer of control.
- Jump from one address to another address in another segment.
- EA = Base register + address field value

Base address of segment
to which we want to jump



Summary

Different types of Transfer of Control Addressing Modes

Reference

- M Morris Mano - Computer System Architecture - PHI - Third Edition
- P Pal Chaudhuri - Computer Organization and Design - PHI - Second Edition
- Thomas C Bartee - Digital Computer Fundamentals - Tata Mc Graw Hill - Sixth Edition

Instruction cycle

Ms. Prathibha Prakash
Department of Computer Science
Amrita Vishwa Vidyapeetham

Objective

Instruction Cycle

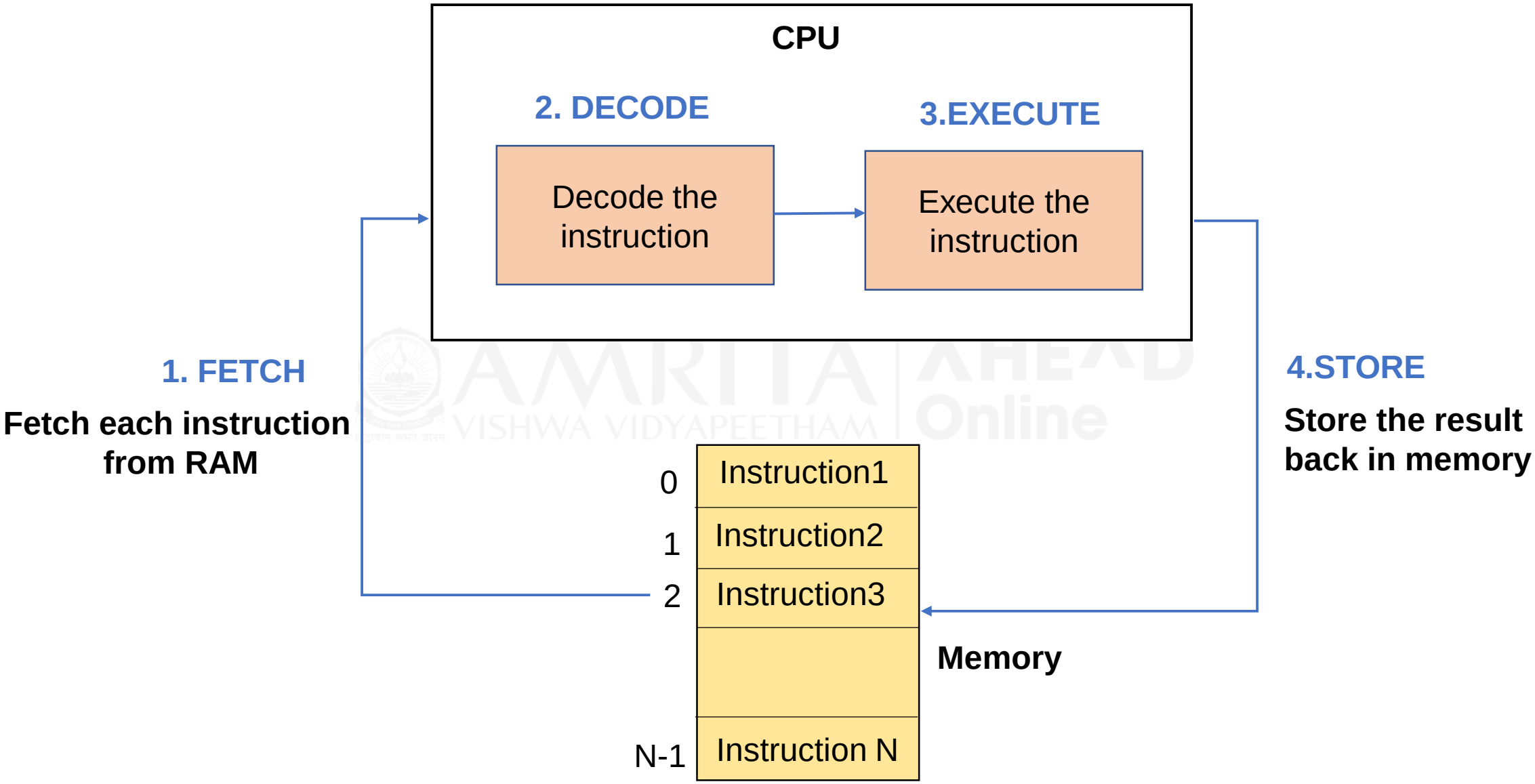
Introduction

- The program residing in the memory unit is a sequence of instructions.
- Program is executed in computer through a cycle for each instruction.
- **Instruction cycle** is a sequence of steps performed by the CPU to execute one instruction.
- The basic operation of CPU is to repeatedly perform the instruction cycle.

Clock Speed

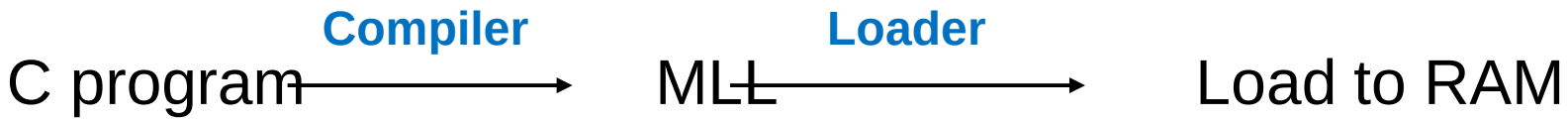
- Operations performed by the processor are controlled by a **system clock**.
- All operations begin with the pulse of clock.
- The speed of processor is measured in cycles per second.
- Most instructions on processor require multiple clock cycles to complete its execution.

Instruction Cycle

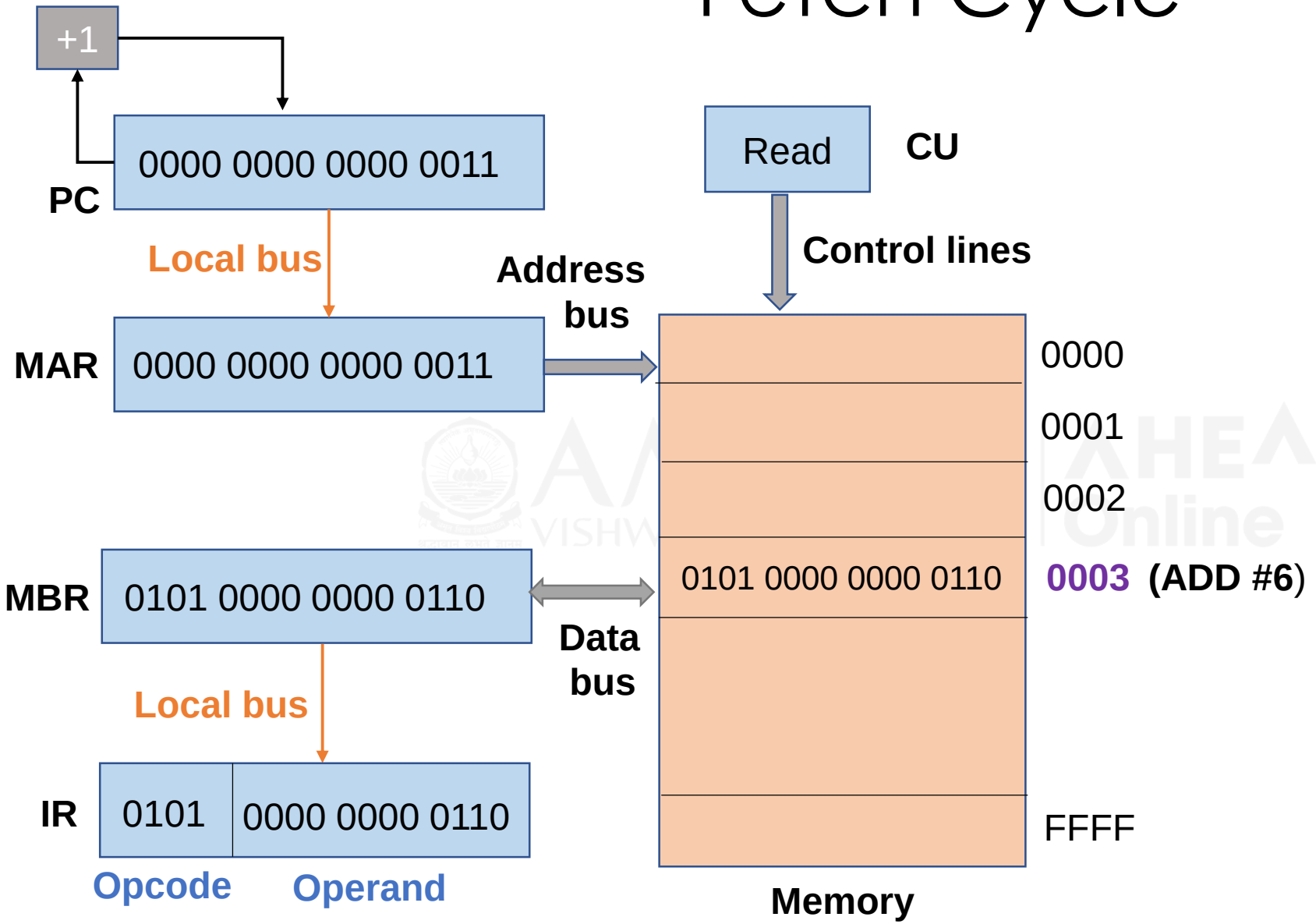


Fetch cycle

- First step in instruction cycle is to **fetch the instruction** stored in memory for execution.
- Processor fetches one instruction at a time and performs the operation specified.
- Program should be loaded into the memory to execute it by the processor.



Fetch Cycle



Microprogram

T1 : [PC] ⟶ MAR

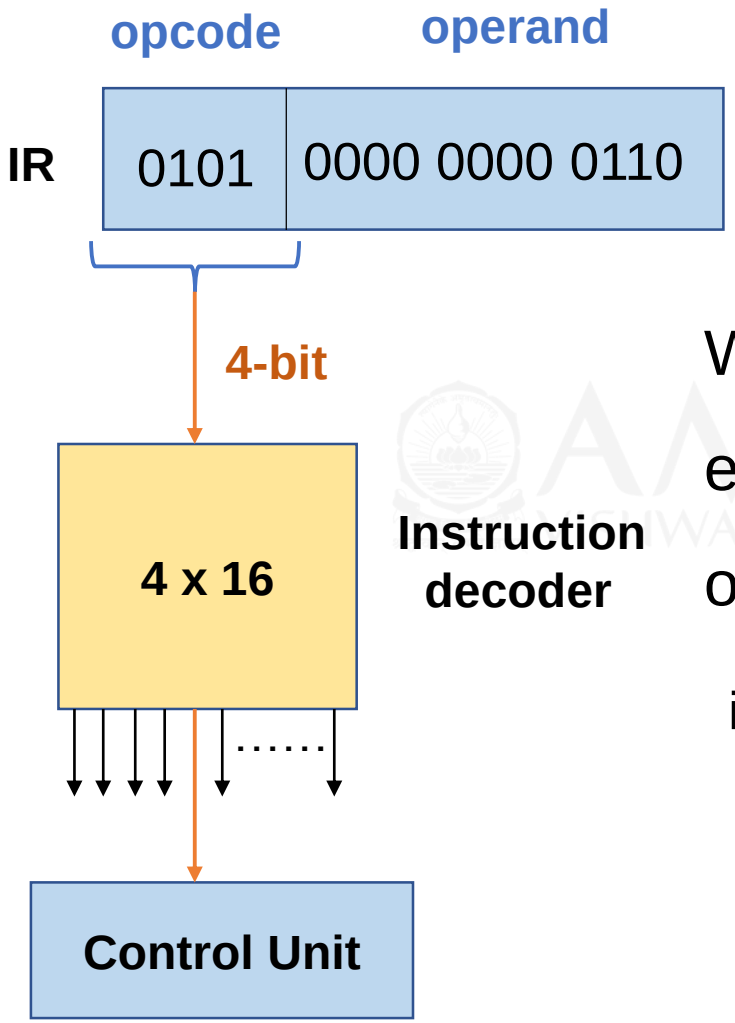
T2 : M [MAR] ⟶ MBR
[PC] + IL ⟶ PC

T3 : MBR ⟶ IR

Execute cycle : Instruction decode

- After fetch stage the Instruction Register contains the fetched instruction from memory.
- The opcode is sent through the Instruction decoder and tell what operation to be performed to control unit
- If opcode is 4-bit then 4 x 16 decoder i.e., 4 input and 16 output.
- In our instruction opcode is 0101 - ADD

Instruction decode



Which binary encoding corresponds to which operation

either load operation or arithmetic operations or logical operations

i.e., 0000 = load operation

0001 = mul

0010 = sub

0101 = add

Summary

Instruction Cycle

Reference

- M Morris Mano - Computer System Architecture - PHI - Third Edition
- P Pal Chaudhuri - Computer Organization and Design - PHI - Second Edition
- Thomas C Bartee - Digital Computer Fundamentals - Tata Mc Graw Hill - Sixth Edition

Instruction cycle - II

Ms. Prathibha Prakash
Department of Computer Science
Amrita Vishwa Vidyapeetham

Objective

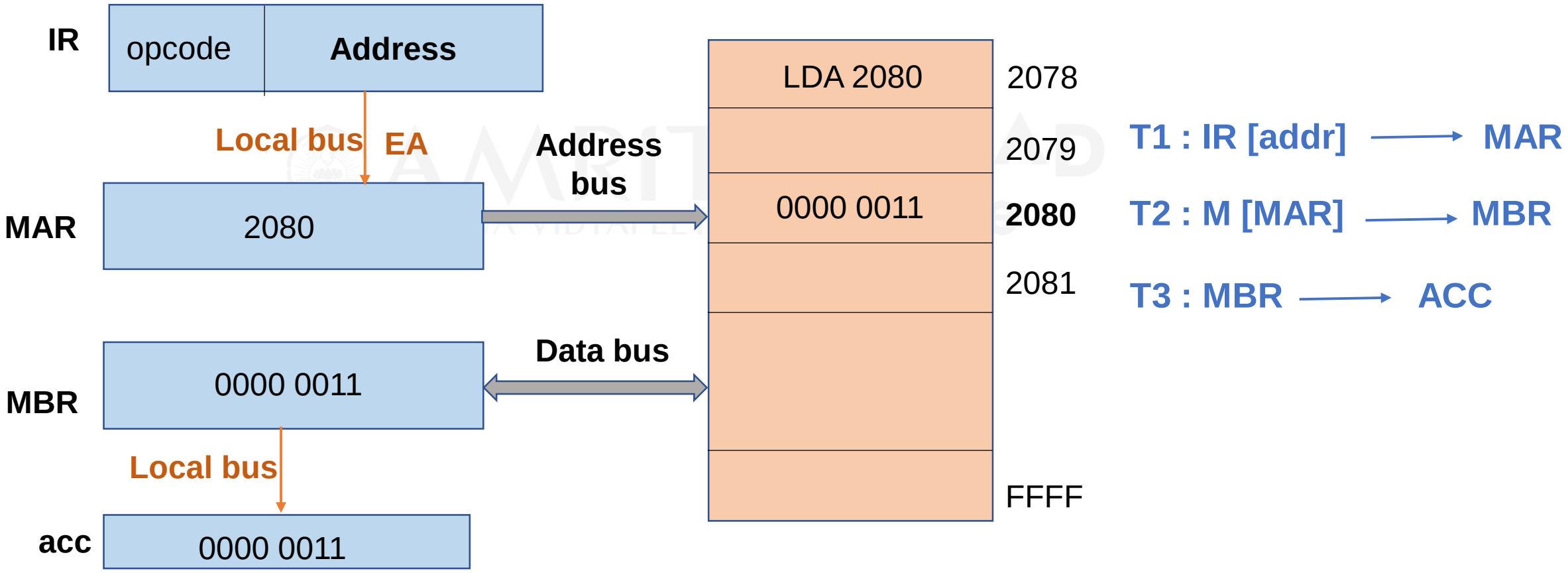
Instruction Cycle

Execute cycle : Operand fetch

- Fetched the instruction from memory, decoded the instruction opcode.
- Now control unit knows it is an add operation, so need to obtain the data stored in corresponding address.
- Operand fetch can be two types : direct mode & indirect mode

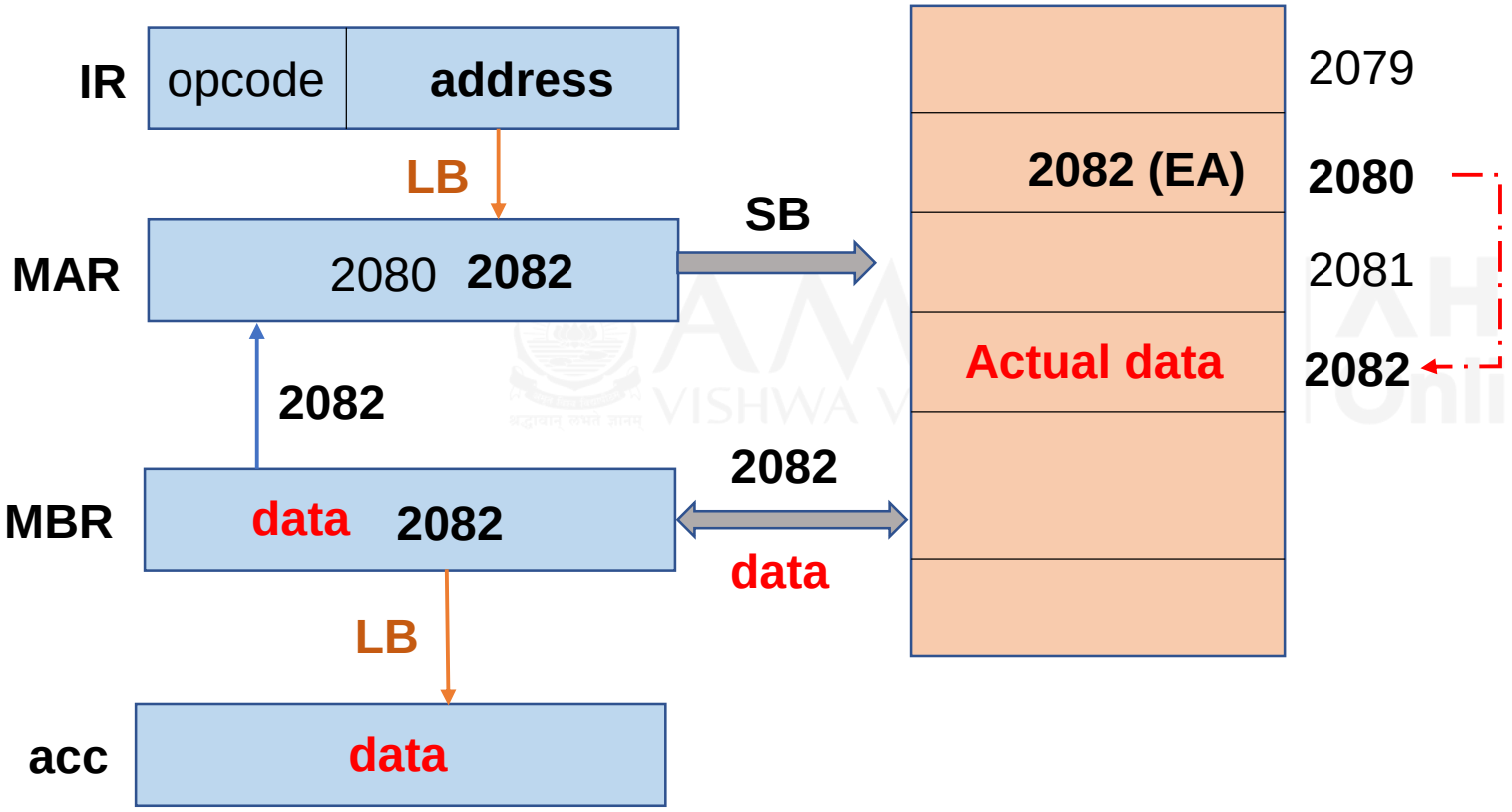
Operand Fetch

- **Direct mode** : LDA [2080] (Effective address =2080)



Operand Fetch

- **Indirect mode** : LDA @2080



T1: IR [ADDR] ⟶ MAR

T2 : M [MAR] ⟶ MBR

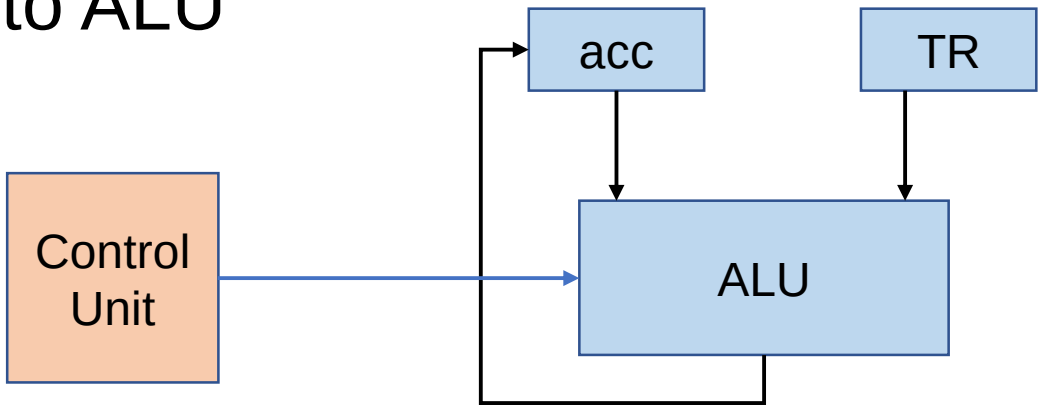
T3 : MBR ⟶ MAR

T4 : M [MAR] ⟶ MBR

T5 : MBR ⟶ ACC

Execute stage

- Control unit takes the instruction from location in PC and loads to IR .
- Decodes the opcode and tells CU what operation to be done.
- If an arithmetic or logic operation, fetch the operand, CU gives the control to ALU



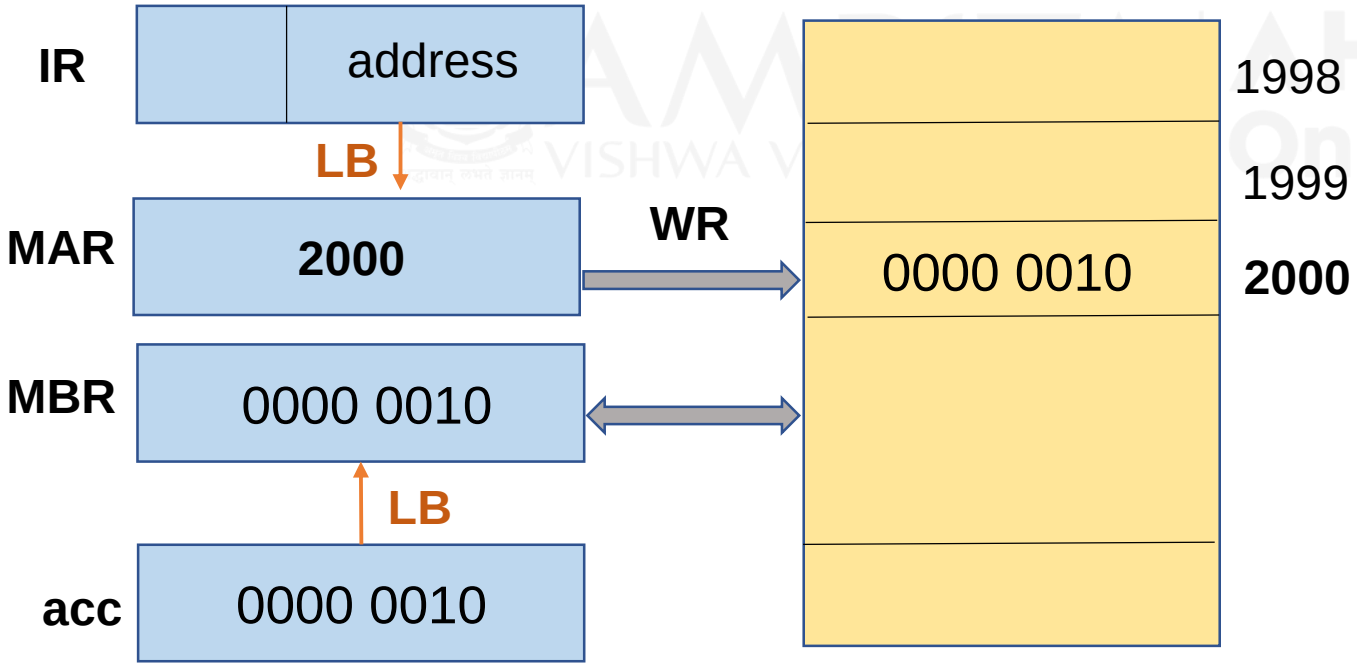
1st operand from accumulator,

2nd operand from Temporary register

Output is stored back to accumulator

Execute stage: Writeback

- After executing the operations writeback the result to memory.
- STA [2000] (store from accumulator to address 2000 in memory)

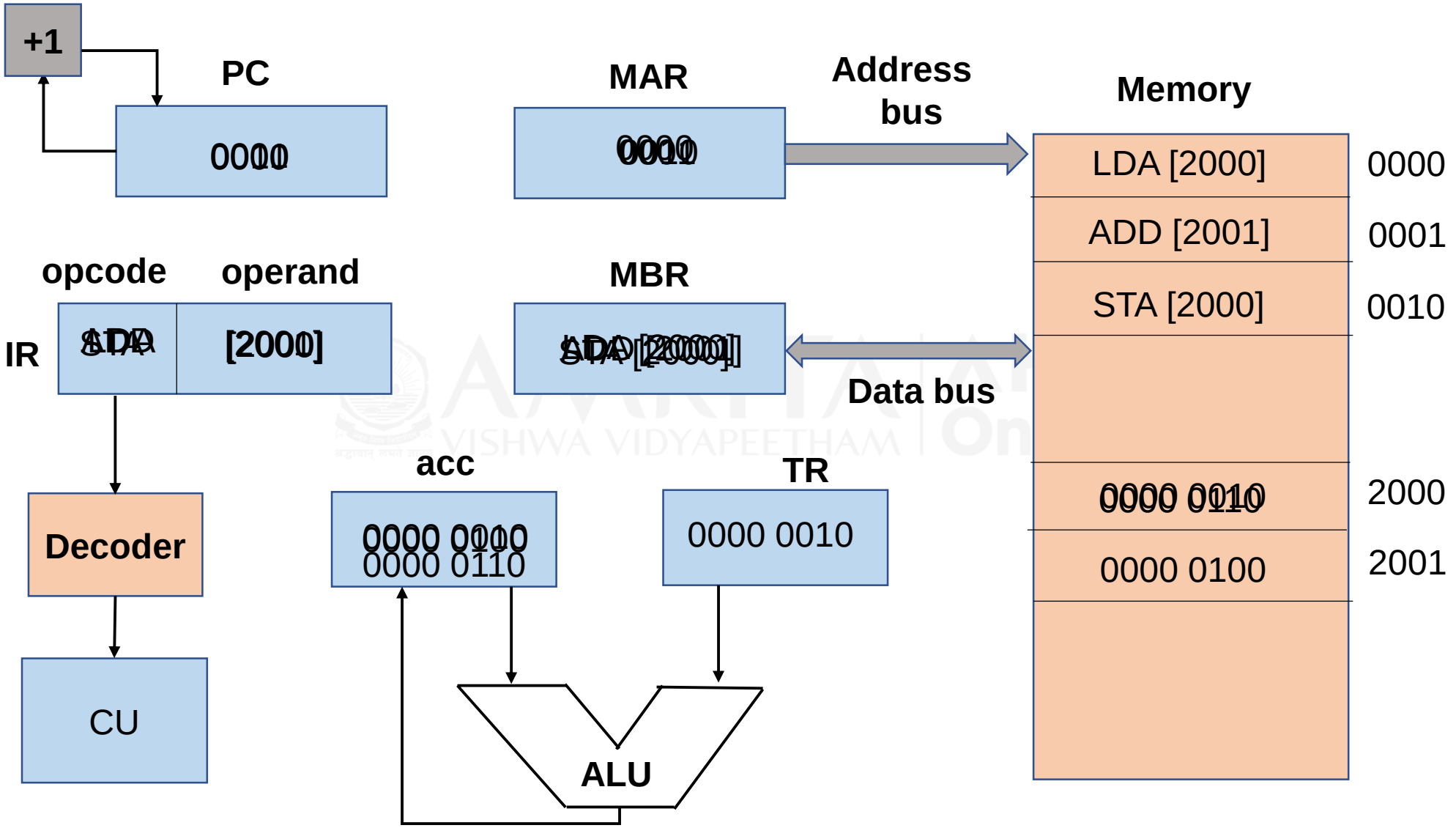


T1 : ACC ⟶ MBR

T2 : IR [ADDR] ⟶ MAR

T3 : MBR ⟶ M [MAR]

Instruction cycle



Summary

Instruction Cycle and its phases

Reference

- M Morris Mano - Computer System Architecture - PHI - Third Edition
- P Pal Chaudhuri - Computer Organization and Design - PHI - Second Edition
- Thomas C Bartee - Digital Computer Fundamentals - Tata Mc Graw Hill - Sixth Edition

Program Interrupt

Ms. Prathibha Prakash
Department of Computer Science
Amrita Vishwa Vidyapeetham

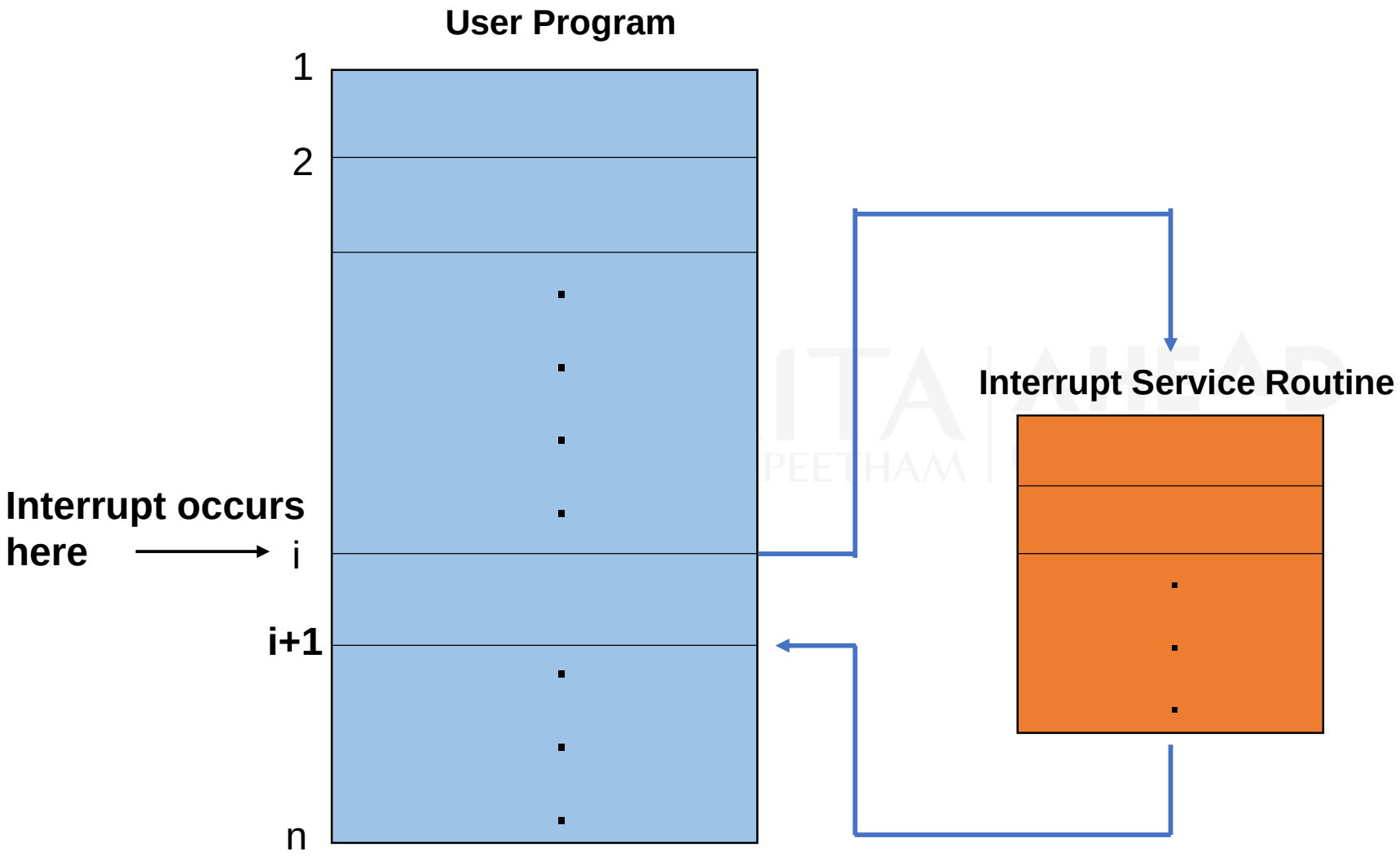
Objective

- Instruction cycle with interrupts

Introduction

- Interrupt is a signal to the processor emitted by the hardware/ software indicating an event that need immediate attention.
- The processor suspend its current execution and service the occurred interrupt.
- To service the interrupt the processor executes the corresponding interrupt service routine(ISR).
- After the execution of ISR the processor resumes the execution of the suspended program.

Interrupt


User Program
1
2
i
i+1
n
Interrupt occurs here
Interrupt Service Routine

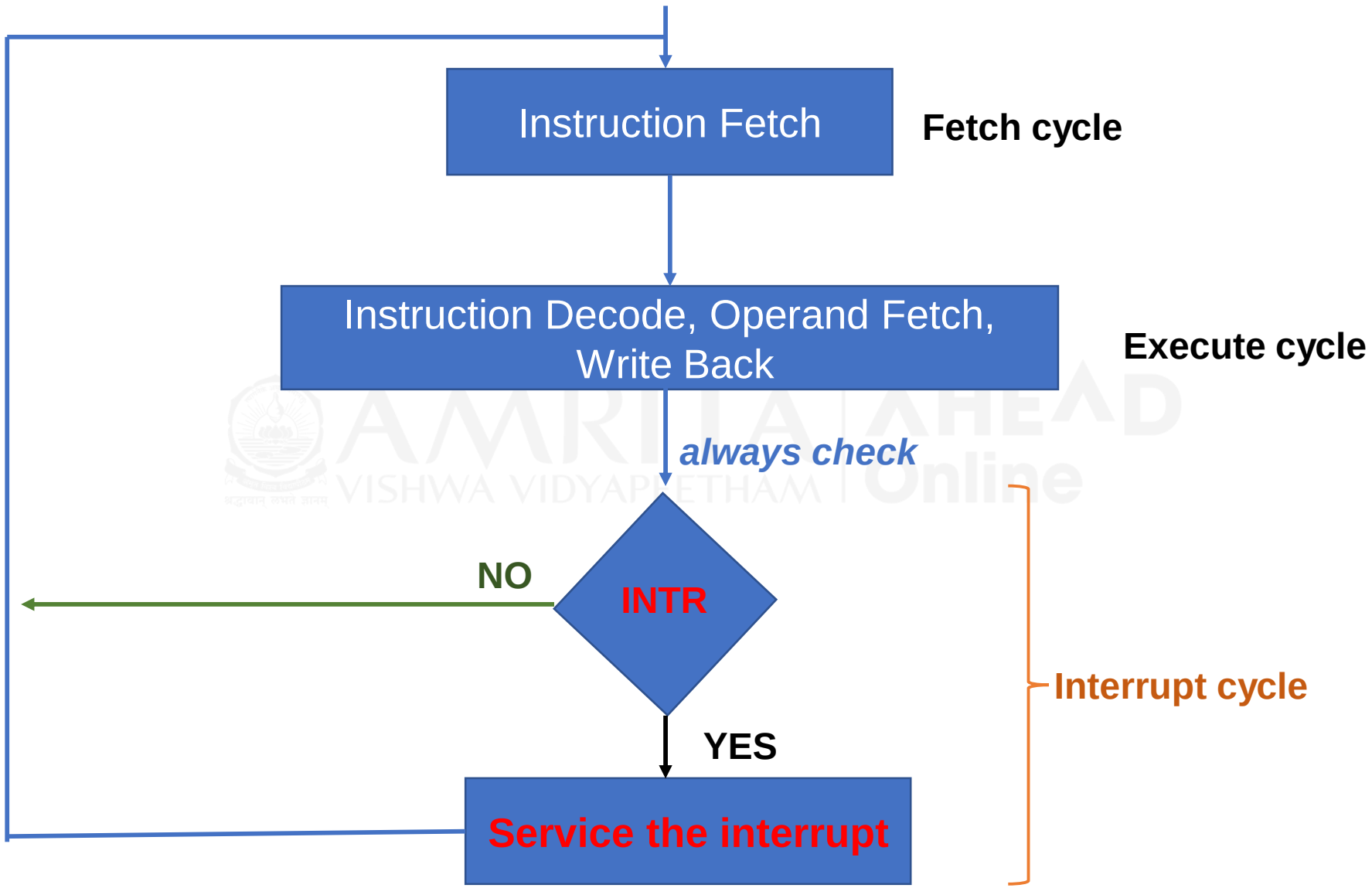
Instruction Cycle with Interrupts

- Interrupt can occur at any point while executing an instruction.
- A normal instruction cycle starts with the instruction fetch and execute.
- To accommodate the occurrence of interrupts, the interrupt cycle is added to the normal instruction cycle.
- After the execution of current instruction, the processor verifies any interrupt signal is pending or not.
- If no pending interrupts the processor proceeds to fetch the next instruction in the sequence.

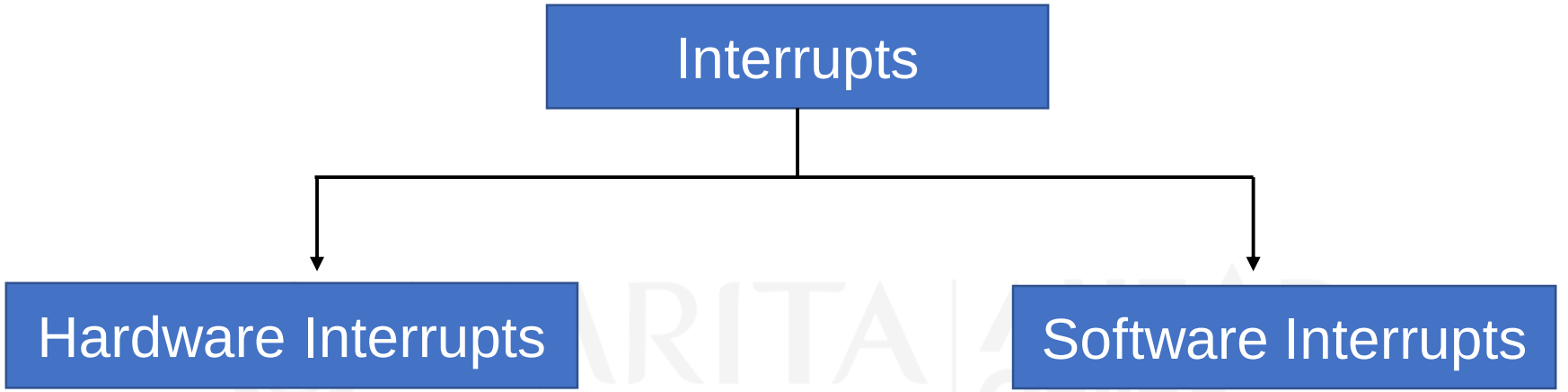
Interrupt cycle

- If any pending interrupts, the processor suspends the execution of the current program.
- Processor saves the address of the next instruction before handling the ISR.
- Update the program counter with the starting address of interrupt service routine to service the occurred interrupt.
- After servicing the interrupts completely the processor resumes the execution of suspended program.

Interrupt cycle



Types of interrupts



If a processor receives the interrupt request from an external I/O device

The interrupts that occur when a condition is met or a system call occurs

Summary

- Discussed about Program interrupts and how to service it.

Reference

- M Morris Mano - Computer System Architecture - PHI - Third Edition
- P Pal Chaudhuri - Computer Organization and Design - PHI - Second Edition
- Thomas C Bartee - Digital Computer Fundamentals - Tata Mc Graw Hill - Sixth Edition