



Chapter – 2

Basic Computer Organization

- 2.1 Instruction Codes
- 2.2 Computer Registers
- 2.3 Computer Instructions
- 2.4 Timing and Control
- 2.5 Instruction Cycle
- 2.6 Memory Reference Instructions
- 2.7 Input-Output and Interrupt
- 2.8 Complete Computer Description

2.1 Instruction Codes

Ref. Book Name : Computer System Architecture, M. Morris Mano

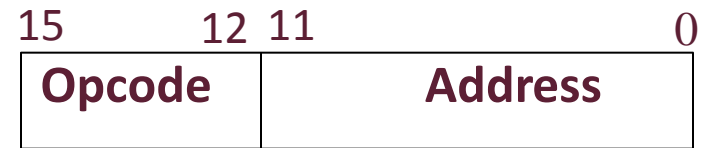
Instruction:

- A computer instruction is a binary code that specifies a sequence of micro-operations for the computer.

Instruction code:

- A group of bits that instruct the computer to perform a specific operation.
- Instruction is divided into two parts

1. **Operation code:** group of bits that define operation.



Instruction Format

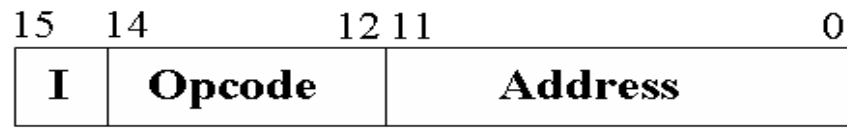
2. **Address:** Memory words can be specified in instruction code by their address.



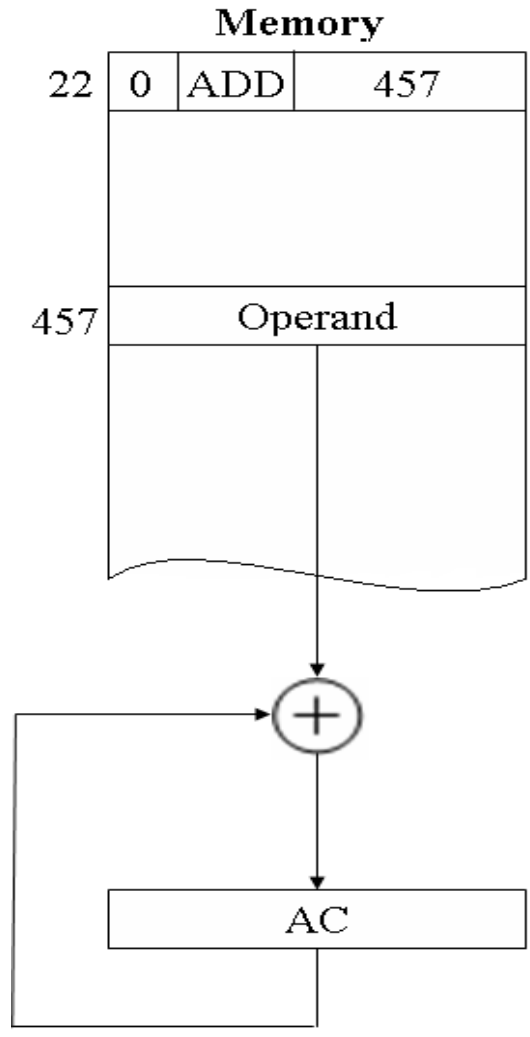


Indirect Address

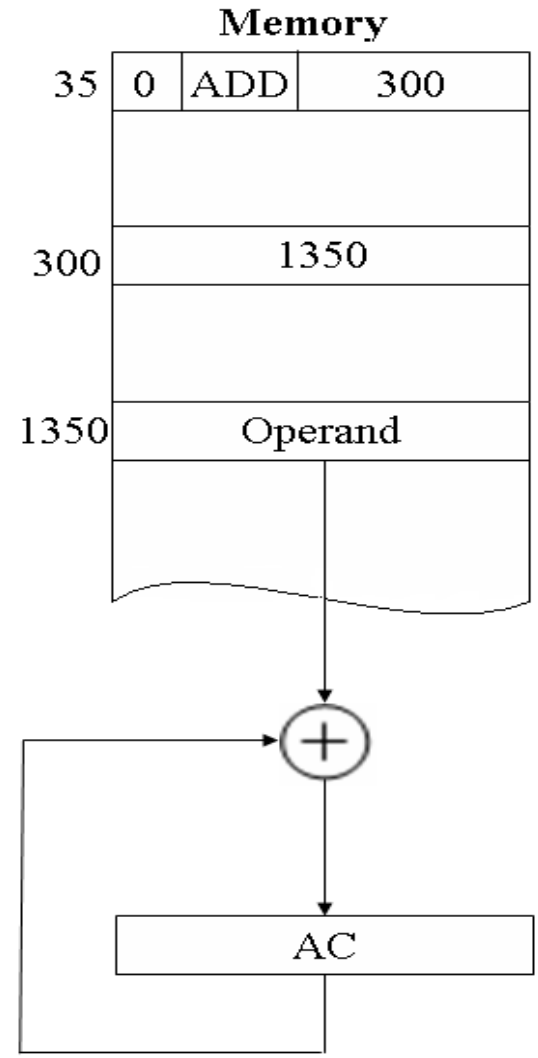
- ❑ When the address part of instruction code specifies the operand, the instruction is said to have **immediate operand**.
- ❑ When the address part specifies the address of a operand the instruction is said to have **direct address**.
- ❑ When the bits in the address part of the instruction designate an address of a memory word in which the address of the operand is found, is called **indirect address**.
- ❑ To distinguish between direct and indirect address, I bit is used.
 - If I=0, instruction is in direct addressing mode.
 - If I=1, instruction is in indirect addressing mode.



(a) Instruction Format



(b) Direct address



(c) Indirect address

2.2 Computer Registers

Ref. Book Name : Computer System Architecture, M. Morris Mano

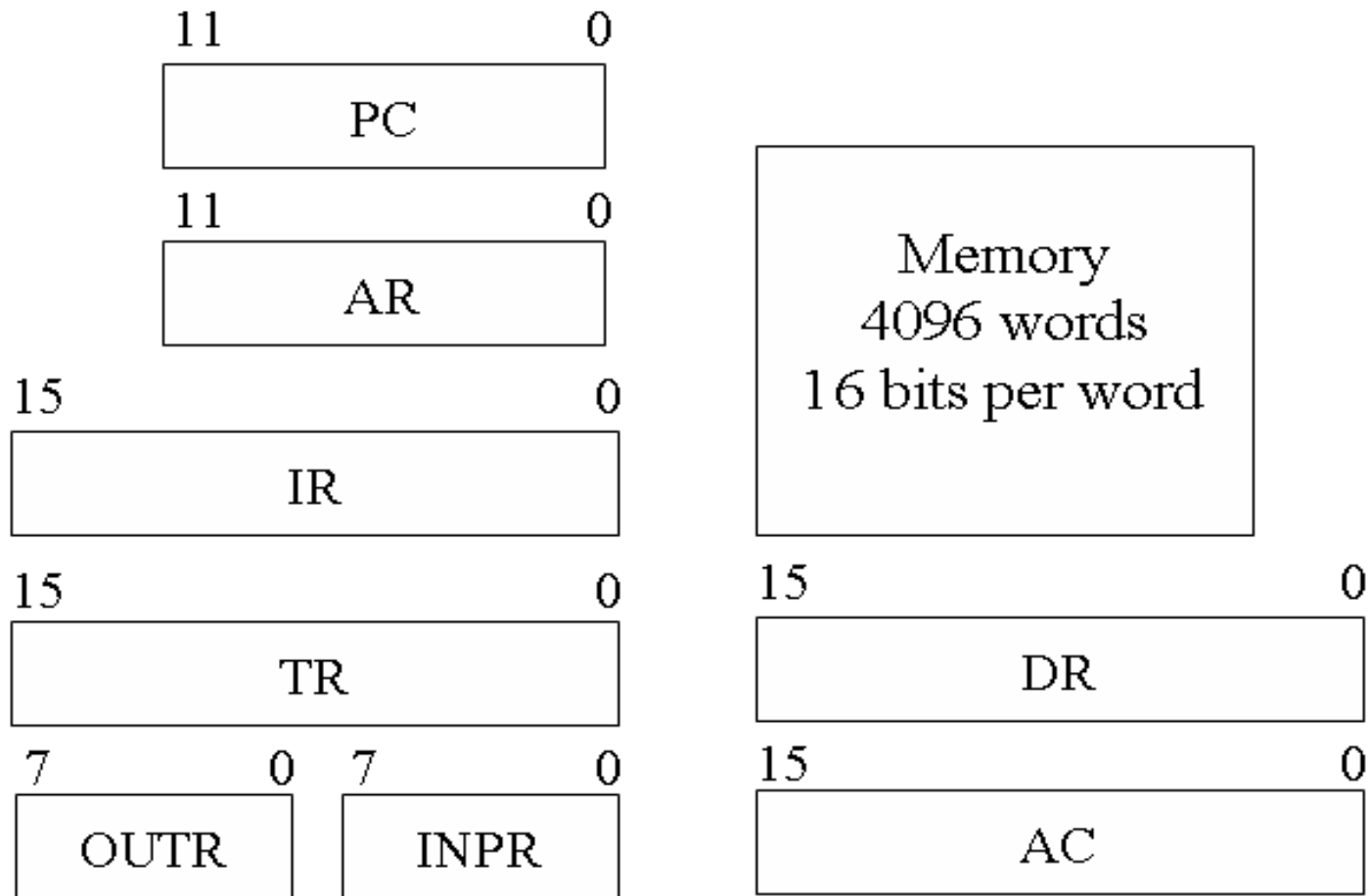


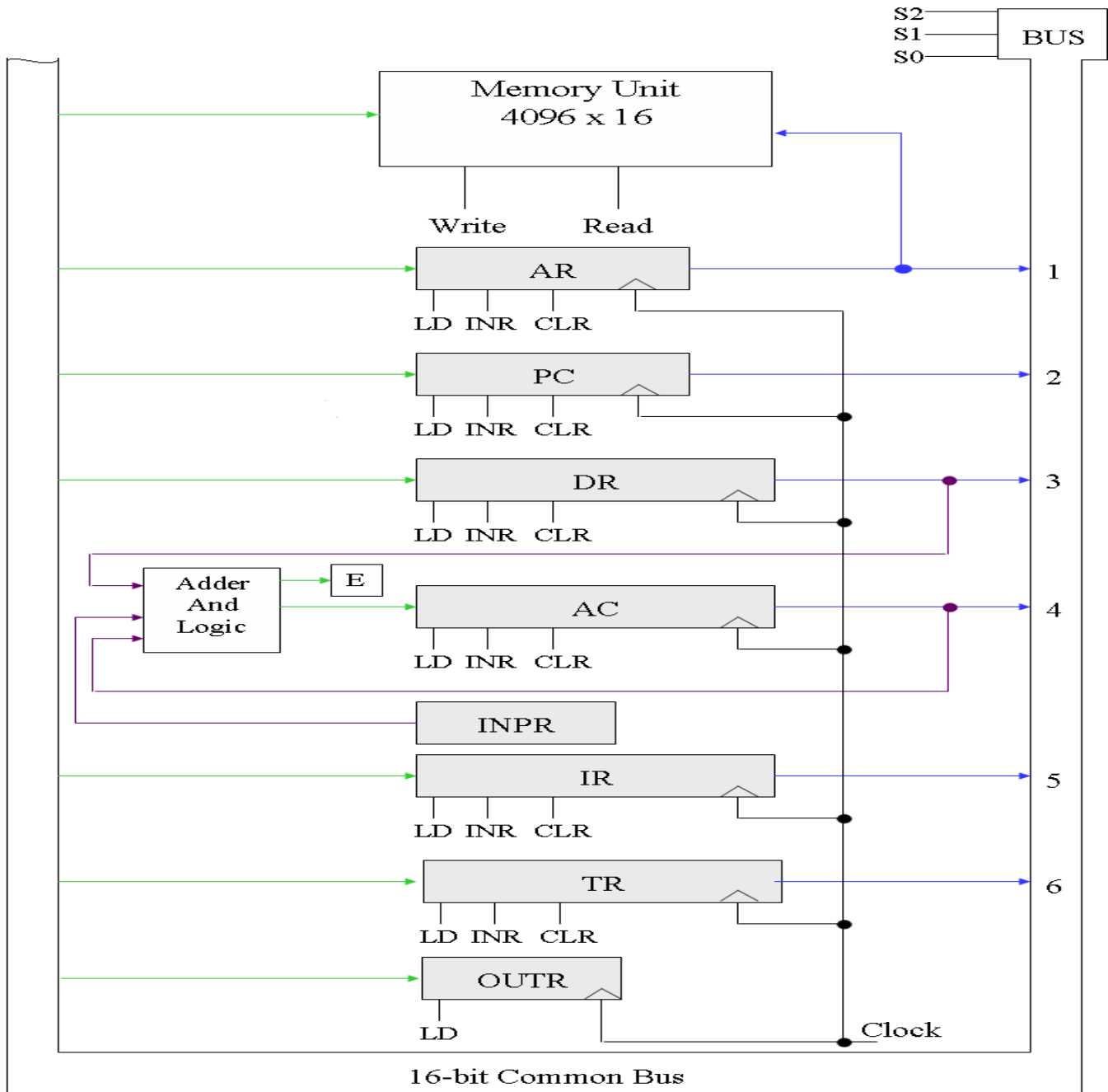
Figure: Basic Computer Registers and Memory

List of Registers for basic Computer

Register Symbol	No. of bits	Register Name	Function
DR	16	Data Register	Holds memory operand
AR	12	Address Register	Holds address for memory
AC	16	Accumulator	Processor Register
IR	16	Instruction Register	Holds instruction code
PC	12	Program Counter	Holds address of instruction
TR	16	Temporary Register	Holds temporary data
INPR	8	Input Register	Holds input character
OUTR	8	Output Register	Holds output character



Common Bus System



16-bit Common Bus

2.3 Computer Instructions

Ref. Book Name : Computer System Architecture, M. Morris Mano

- ❑ The basic computer has three instruction code format. Each format has 16 bits.
- ❑ The operation code (opcode) part of the instruction contains 3 bits and remaining 13 bits depends on opcode.

❑ Memory Reference Instruction



I = 0 : direct addressing mode

I = 1 : indirect addressing mode

2.3 Computer Instructions

Register Reference Instruction

- Instructions which process data stored in processor register, accumulator, E-sign bit, over flow register are called register-reference instructions.



Input Output Instruction

- Input and output from peripheral devices are controlled by those type of instructions.



2.4 Timing and Control

Ref. Book Name : Computer System Architecture, M. Morris Mano

- ❑ The timing for all registers in the basic computer is controlled by a master clock generator.
- ❑ Control signals are generated in the control unit and provide control inputs for multiplexers in common bus, in processor registers and micro-operations for accumulator.
- ❑ There are two types of control organization:
 1. Hard-wired control
 2. Micro programmed control



1. Hardwired control

- Control logic is implemented with gates, flip-flops, decoders and other digital circuits.
- It can be optimized to produce a fast mode of operation.
- It requires changes in wiring among the various components, if the design has to be modified.

2. Micro-programmed control

- Control information is stored in control memory.
- Control memory is programmed to initiate the required sequence of micro-operation.
- Any required changes can be done by updating the micro-program in control memory.

2.5 Control Unit

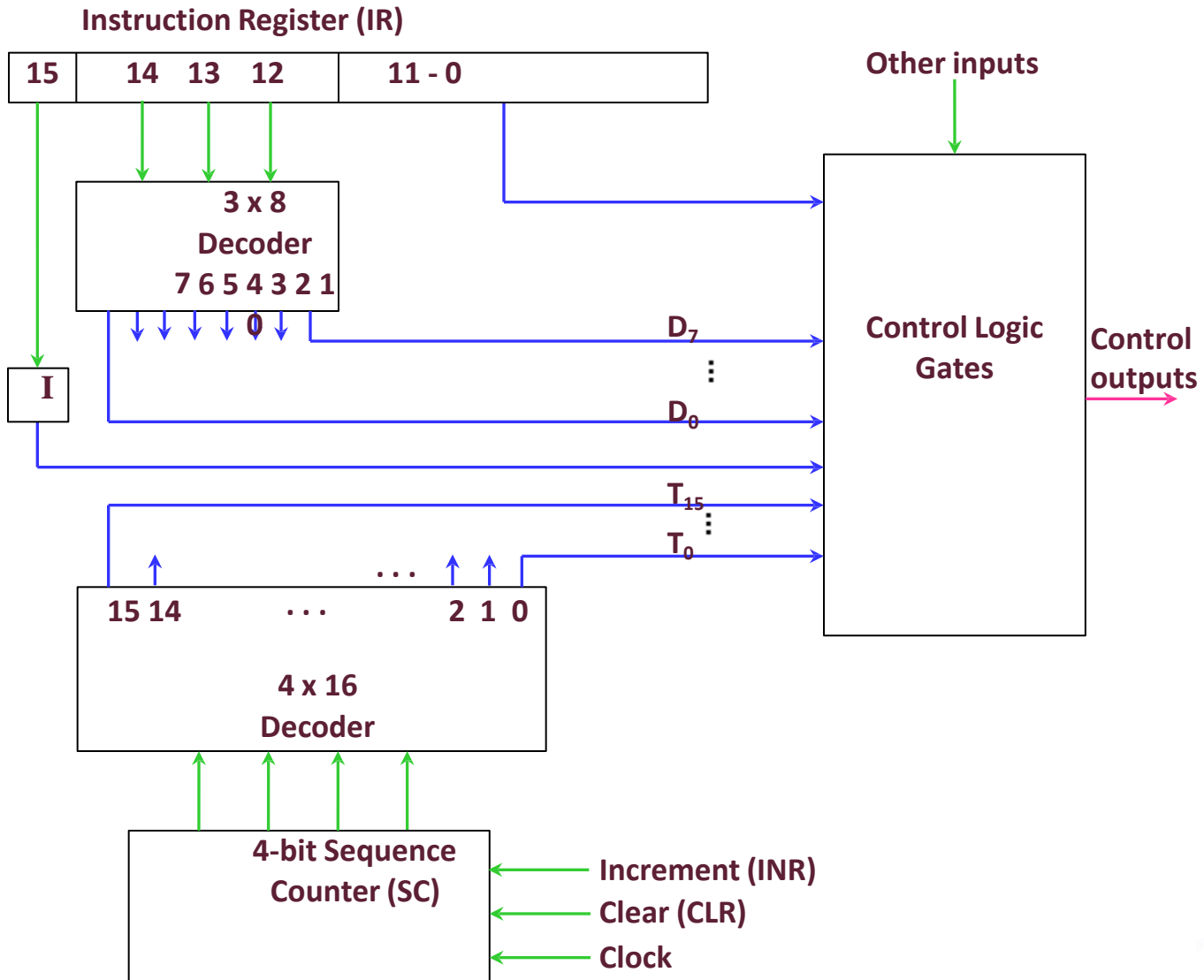


Figure: Control Unit of basic computer

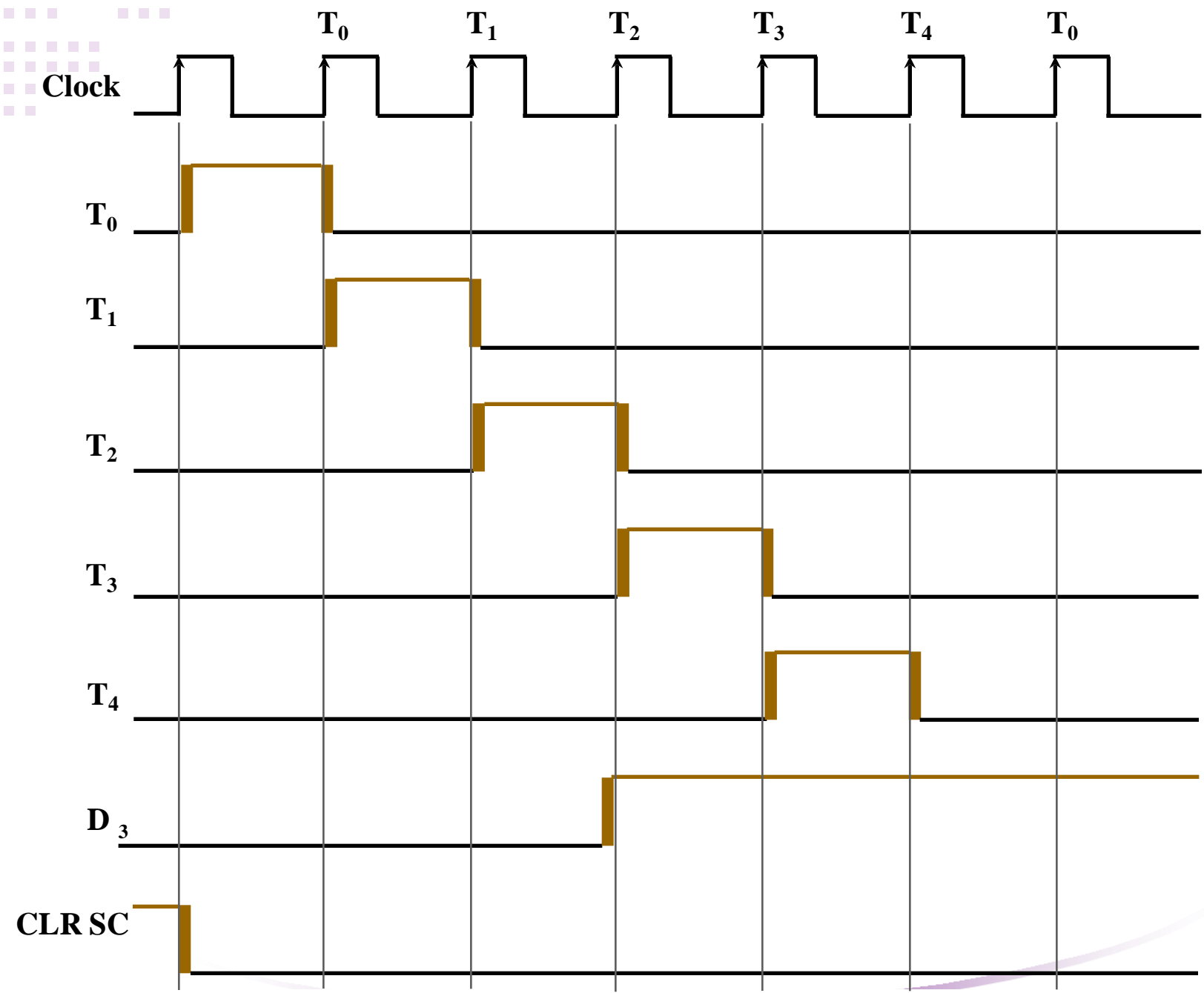


Figure: Example of control timing signals

2.6 Instruction cycle

Ref. Book Name : Computer System Architecture, M. Morris Mano

- ❑ Program residing in memory unit consists of a sequence of micro-operations.
- ❑ The program is executed in the computer by going through a cycle for each instruction.
- ❑ In basic computer, instruction cycle consists of the following phases:
 1. Fetch instruction from memory
 2. Decode the instruction
 3. Read effective address from memory if the instruction has an indirect address
 4. Execute instruction

2.6 Instruction cycle

Fetch and Decode

- The micro-operations for fetch and decode phases can be specified by the following register transfer statements:

$$T_0: AR \leftarrow PC$$
$$T_1: IR \leftarrow M[AR], PC \leftarrow PC + 1$$
$$T_2: D_0, \dots, D_7 \leftarrow \text{Decode } IR(12-14),$$
$$AR \leftarrow IR(0-11),$$
$$I \leftarrow IR(15)$$

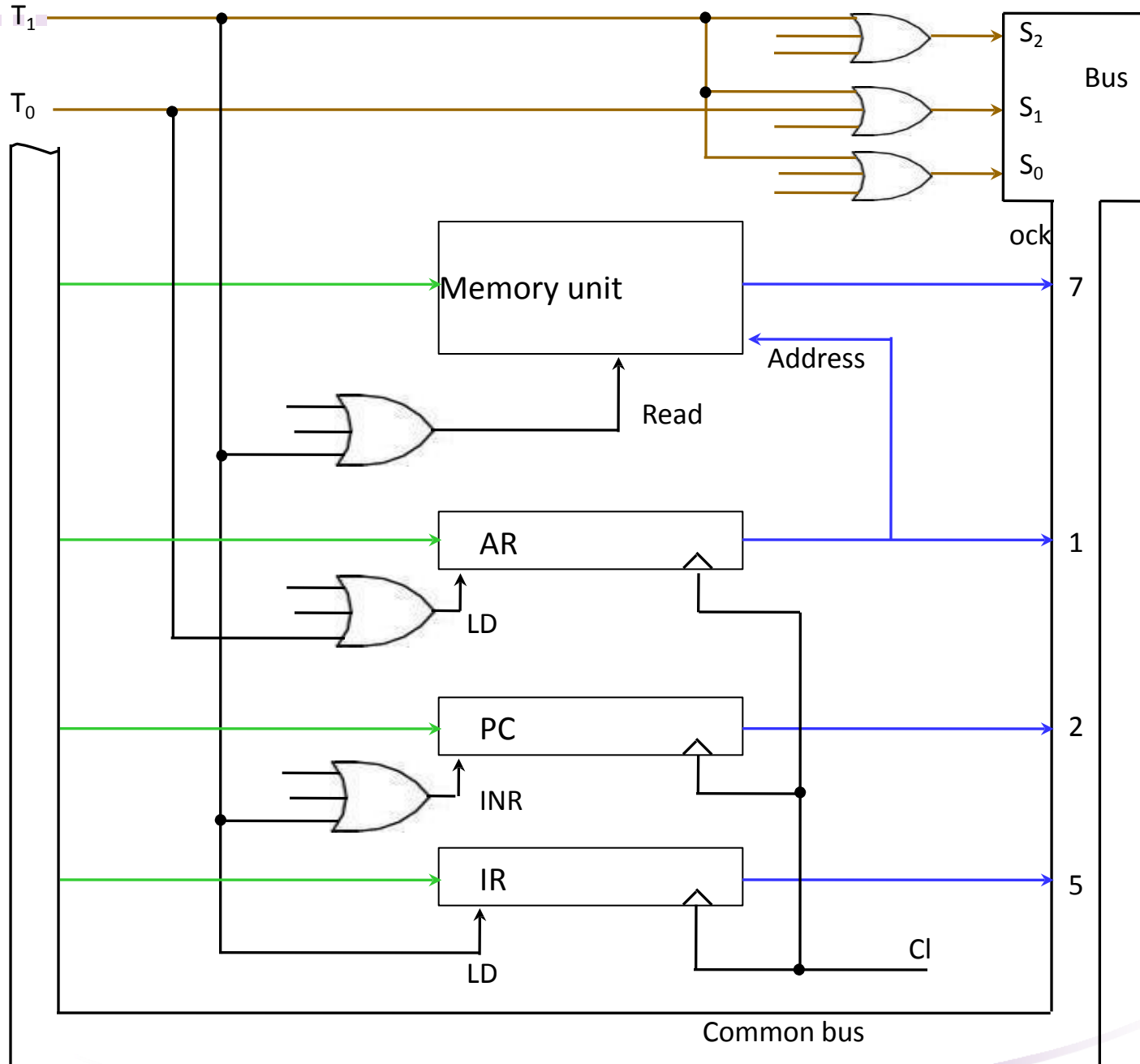


Figure: Register transfers for the fetch phase

Determine the type of instruction

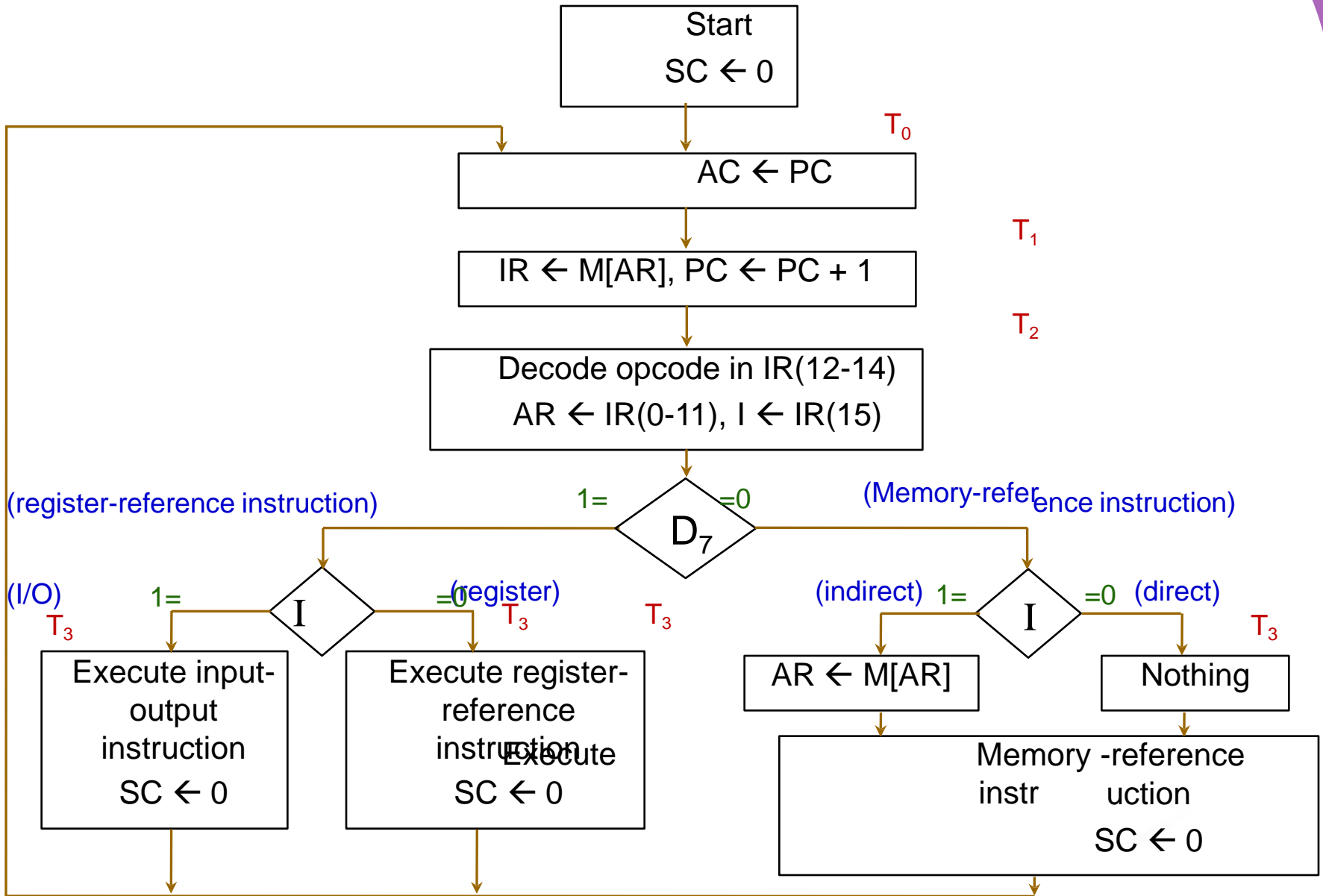



Figure: Flow chart of instruction cycle

- 
- ❑ The **three** instruction types are subdivided into four different paths:
 - ❑ Selected operation is activated at timing signal T_3 .

$D_7'I T_3 : AR \leftarrow M[AR]$

$D_7'I' T_3 : \text{Nothing}$

$D_7I' T_3 : \text{Execute register reference instruction}$

$D_7I T_3 : \text{Execute input-output instruction}$



Register Reference Instructions

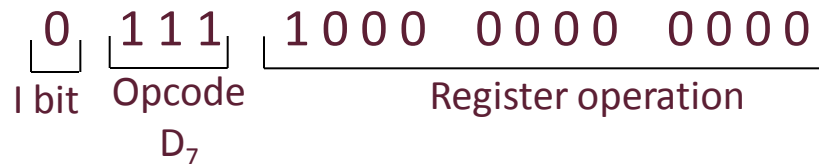
Register-reference instructions are recognized by the control when $D_7 = 1$ and $I = 0$.

These instructions are executed with the clock transition associated with timing variable T_3

Each control function needs a Boolean relation, $D_7 I' T_3$ which is designated by r .

$$D_7 I' T_3 = r$$

For example, instruction CLA has hexadecimal code 7800, which gives a binary equivalent



Execution of Register reference Instructions

D7I'T3 = r (Common to all register reference instructions)

IR(i) = Bi (bit in IR(0-11) that specifies the operation)

CLA	$rB_{11} :$	$AC \leftarrow 0$	Clear AC
CLE	$rB_{10} :$	$E \leftarrow 0$	Clear E
CMA	$rB_9 :$	$AC \leftarrow \overline{AC}$	Complement AC
CME	$rB_8 :$	$E \leftarrow \overline{E}$	Complement E
CIR	$rB_7 :$	$AC \leftarrow \text{shr } AC,$ $E \leftarrow AC(0),$ $AC(15) \leftarrow E$	Circular Right
CIL	$rB_6 :$	$AC \leftarrow \text{shl } AC$ $E \leftarrow AC(15) AC(0)$ $\leftarrow E$	Circular Left
INC	$rB_5 :$	$AC \leftarrow AC + 1$	Increment AC

Execution of Register reference Instructions

SPA	rB_4 :	If $(AC(15) = 0)$ then $PC \leftarrow PC + 1$	Skip if positive
SNA	rB_3 :	If $(AC(15) = 1)$ then $PC \leftarrow PC + 1$	Skip if negative
SZA	rB_2 :	If $(AC = 0)$ then $PC \leftarrow PC + 1$	Skip if AC is 0
SZE	rB_1 :	If $(E = 0)$ then $PC \leftarrow PC + 1$	Skip if E is 0
HLT	rB_0 :	$S \leftarrow 0$	Halt Computer

2.6 Memory Reference Instructions

- Decoded output D_i for $i = 0,1,2,3,4,5,6$ from the operation decoder belongs to each instruction.
- The execution starts with timing signal T4.

Symbol	Operation Decoder	Symbolic description
AND	D_0	$AC \leftarrow AC \wedge M[AR]$
ADD	D_1	$AC \leftarrow AC + M[AR], E \leftarrow Cout$
LDA	D_2	$AC \leftarrow M[AR]$
STA	D_3	$M[AR] \leftarrow AC$
BUN	D_4	$PC \leftarrow AR$
BSA	D_5	$M[AR] \leftarrow PC, PC \leftarrow AR + 1$
ISZ	D_6	$M[AR] \leftarrow M[AR] + 1,$ If $(M[AR] + 1)$ then $PC \leftarrow PC + 1$



AND to AC

- This instruction performs AND logic operation on AC and memory word.
- The result is transferred to AC.

$$D_0T_4: DR \leftarrow M[AR]$$
$$D_0T_5: AC \leftarrow AC \wedge Dr, SC \leftarrow 0$$

ADD to AC

- This instruction adds the content of memory word specified by the effective address to the value of AC.
- The sum is transferred to AC and output carry Cout is to E flip-flop.

$$D_1T_4: DR \leftarrow M[AR]$$
$$D_1T_5: AC \leftarrow AC + DR, E \leftarrow Cout, SC \leftarrow 0$$



LDA : Load AC

- This instruction transfers the memory word specified by the effective address to AC

$$D_2T_4 : DR \leftarrow M[AR]$$
$$D_2T_4 : AC \leftarrow DR, SC \leftarrow 0$$

STA : Store AC

- This instruction stores the content of AC into memory word specified by the effective address.

$$D_3T_4 : M[AR] \leftarrow AC, SC \leftarrow 0$$

BUN : Branch Unconditionally

- This instruction transfers the program to the instruction specified by the effective address

$$D_4T_4 : PC \leftarrow AR, SC \leftarrow 0$$



BSA : Branch and Save Return Address

- This instruction is useful for branching to a portion of the program called subroutine or procedure.

$$M[AR] \leftarrow PC, PC \leftarrow AR + 1$$

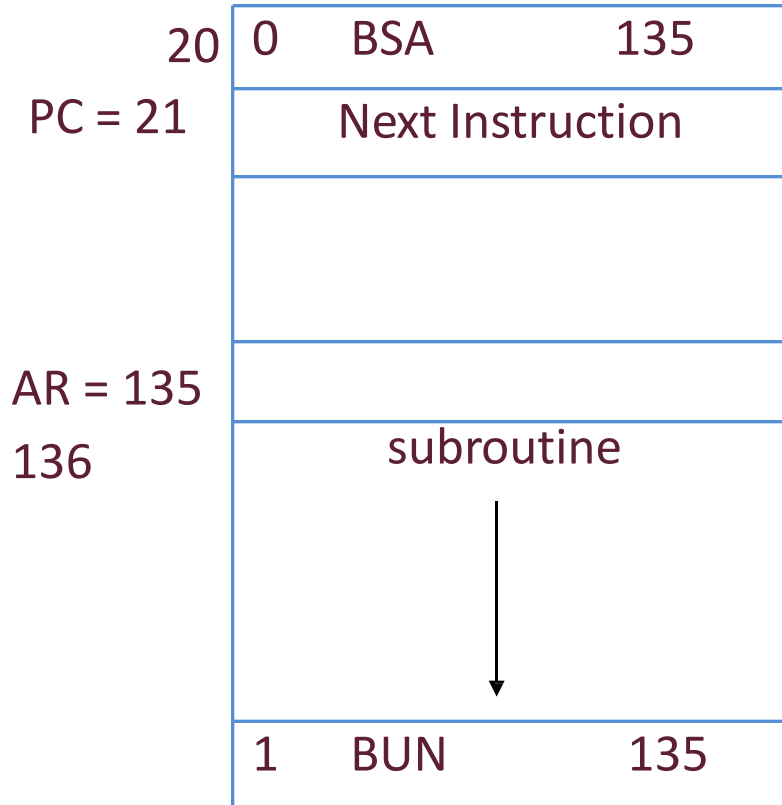
- BSA instruction performs the function usually referred to as subroutine call.

$$D_5T_4: M[AR] \leftarrow PC, AR \leftarrow AR + 1$$

$$D_5T_5: PC \leftarrow AR, SC \leftarrow 0$$

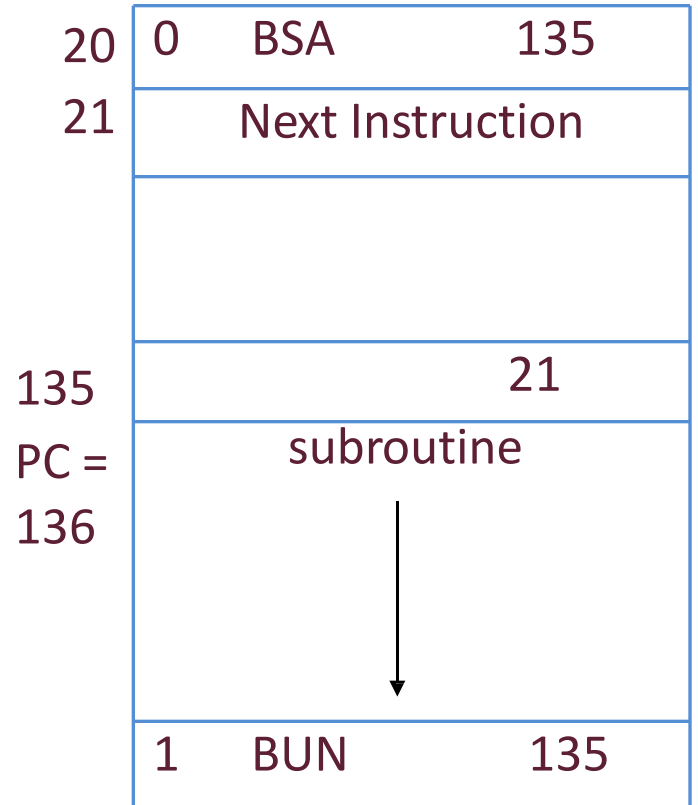


Memory



(a) Memory; PC and AR at time T_4

Memory



(b) Memory; PC and AR after execution



ISZ : Increment and Skip if zero

- This instruction increments the word specified by the effective address, and if incremented value is 0, PC is incremented by 1.
- Since it is not possible to increment a word inside the memory, it is necessary to read the word into DR, increment Dr and store the word back into memory.

D6T4 : $DR \leftarrow M[AR]$

D6T5 : $DR \leftarrow DR + 1$

D6T6 : $M[AR] \leftarrow DR,$

if $(DR = 0)$ then $(PC \leftarrow PC + 1), SC \leftarrow 0$

Control Flowchart

Memory – reference instruction

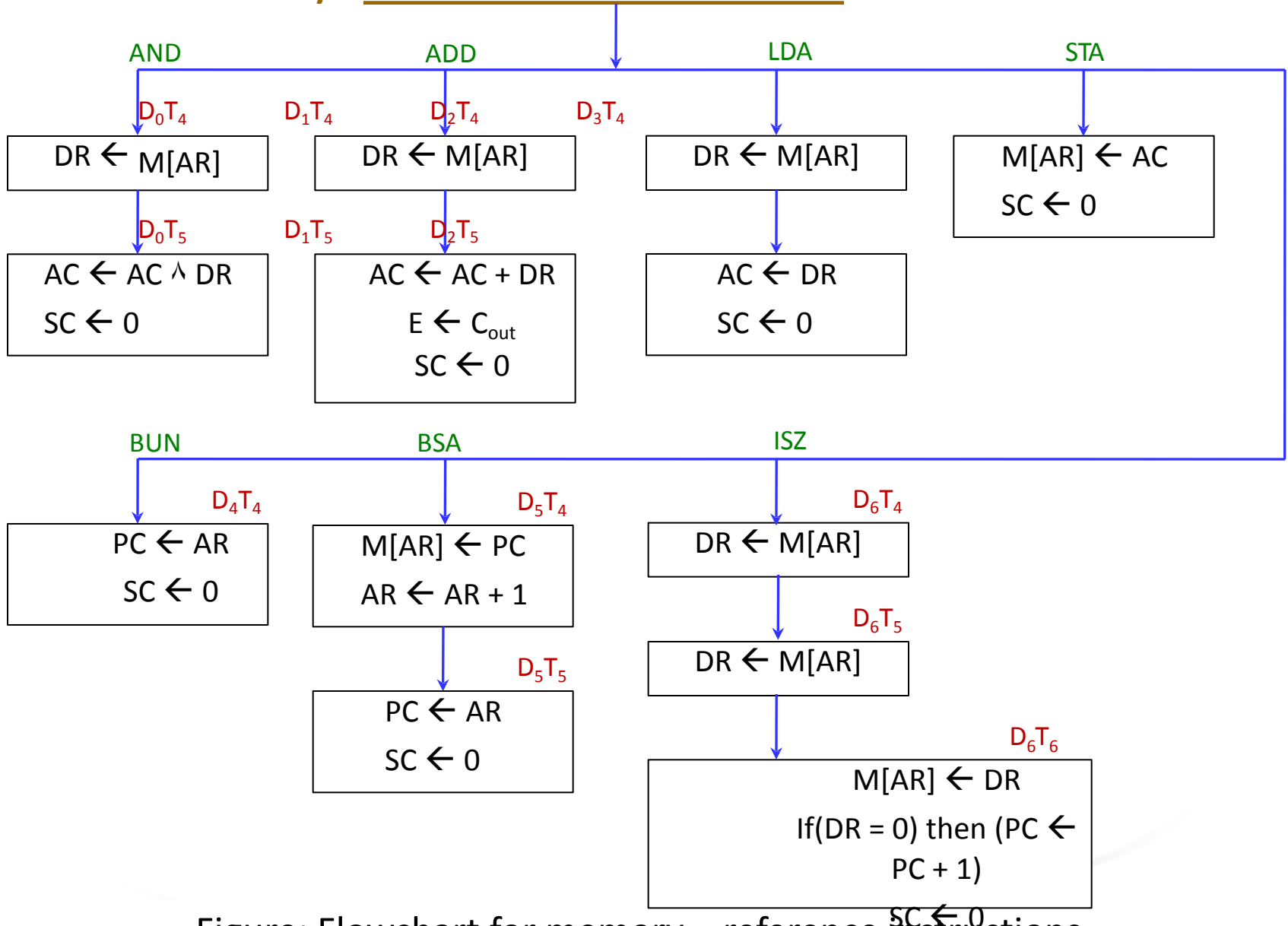


Figure: Flowchart for memory – reference instructions

2.7 Input Output and Interrupt

Ref. Book Name : Computer System Architecture, M. Morris Mano

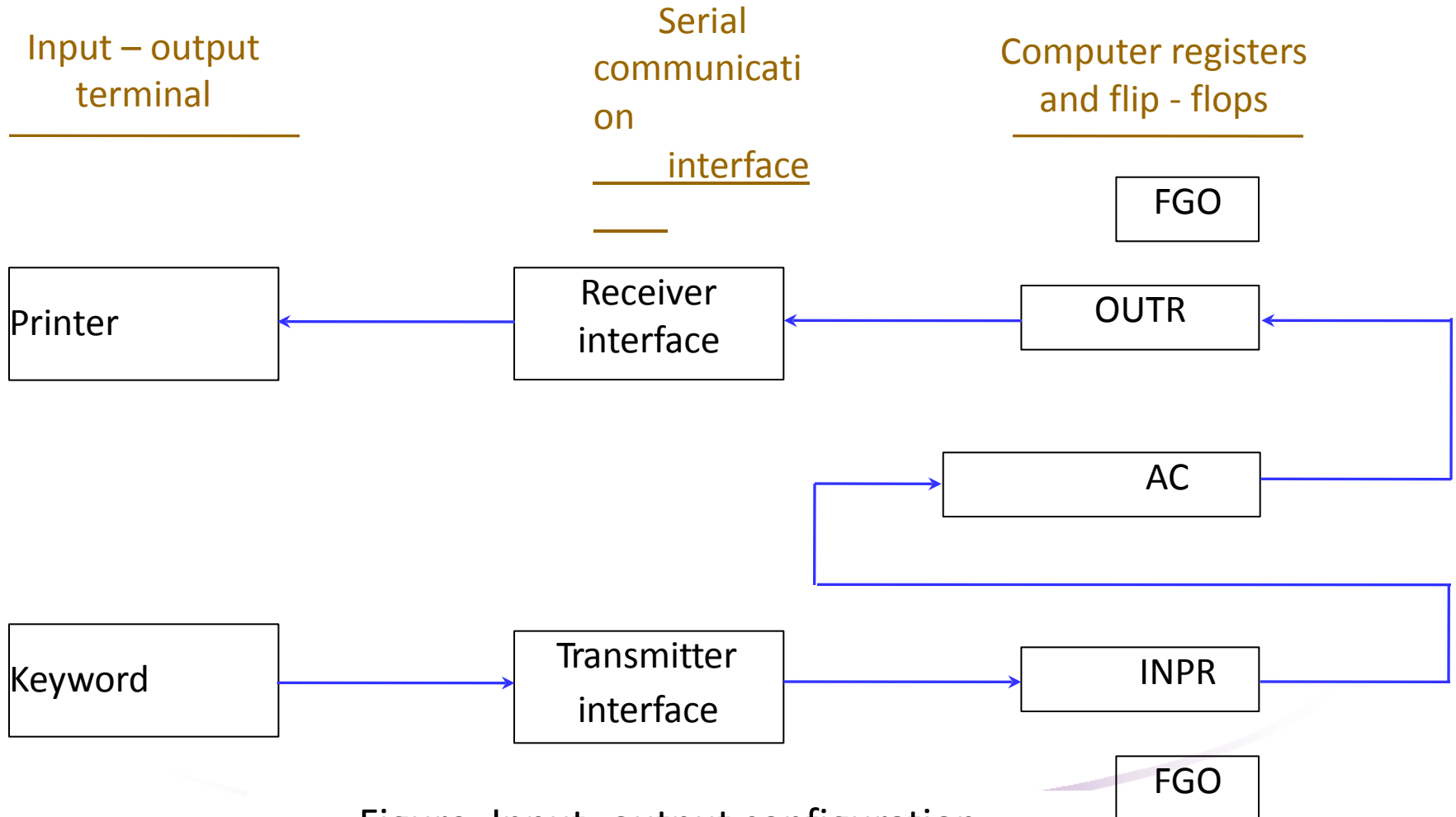


Figure: Input- output configuration

Input-Output Instructions

$D7_1T_3 = p$ (common to all i/I instruction)

$IR(i) = B_i$ (bit in $IR(6-11)$ that specifies the instruction)

INP	$pB_{11} :$	$AC(0-7) \leftarrow INPR, FGI \leftarrow 0$	Input character
OUT	$pB_{10} :$	$OUTR \leftarrow AC(0-7),$ $FGI \leftarrow 0$	Output character
SKI	$pB_9 :$	If $(FGI = 1)$ then $PC \leftarrow PC + 1$	Skip on input flag
SKO	$pB_8 :$	If $(FGO = 1)$ then $PC \leftarrow PC + 1$	Skip on output flag
ION	$pB_7 :$	$IEN \leftarrow 1$	Interrupt enable ON.
IOF	$pB_6 :$	$IEN \leftarrow 0$	Interrupt enable Off.



Program Interrupt

- ❑ Open communication only when some data has to be passed.
- ❑ When interface finds that I/O device is ready for data transfer, it generates interrupt request to CPU.
- ❑ Upon detecting interrupt:
 - CPU stops momentarily the task it is doing
 - Branches to service routine to process data transfer
 - And returns to task it was performing

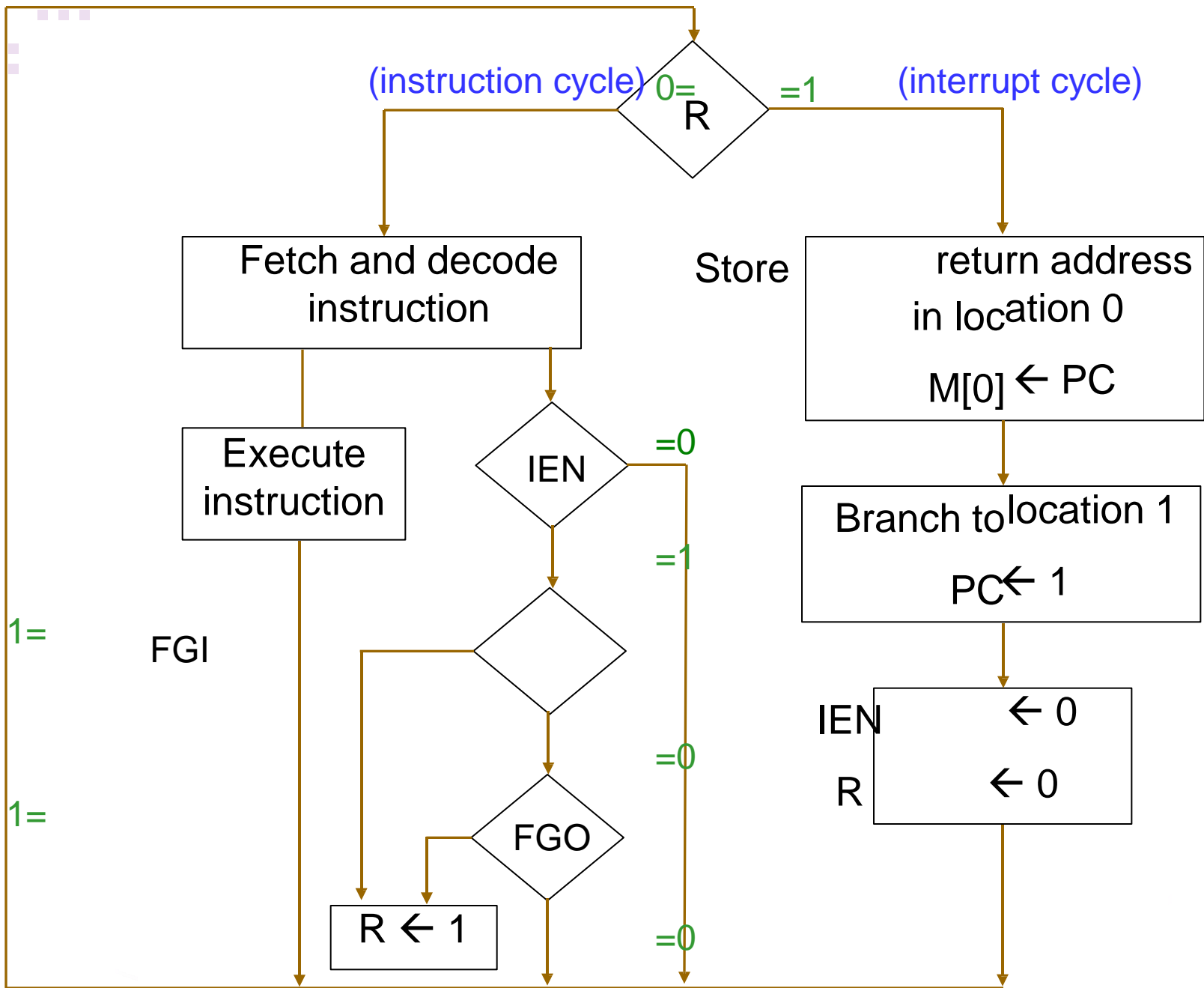
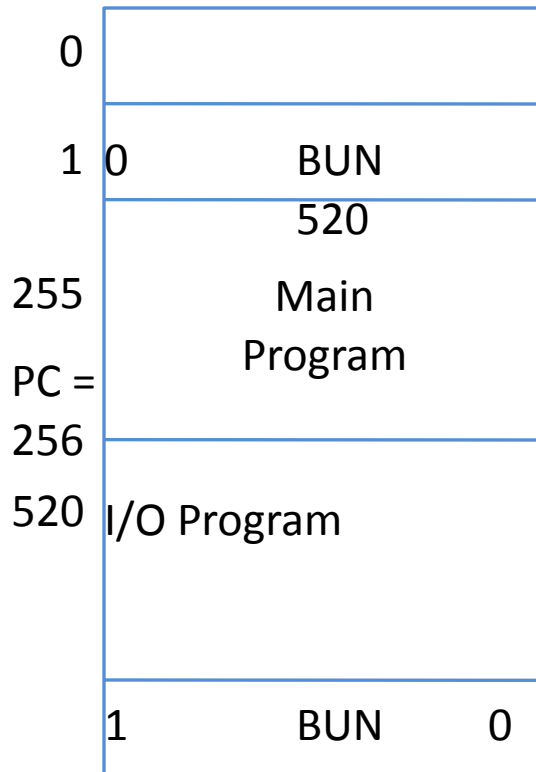


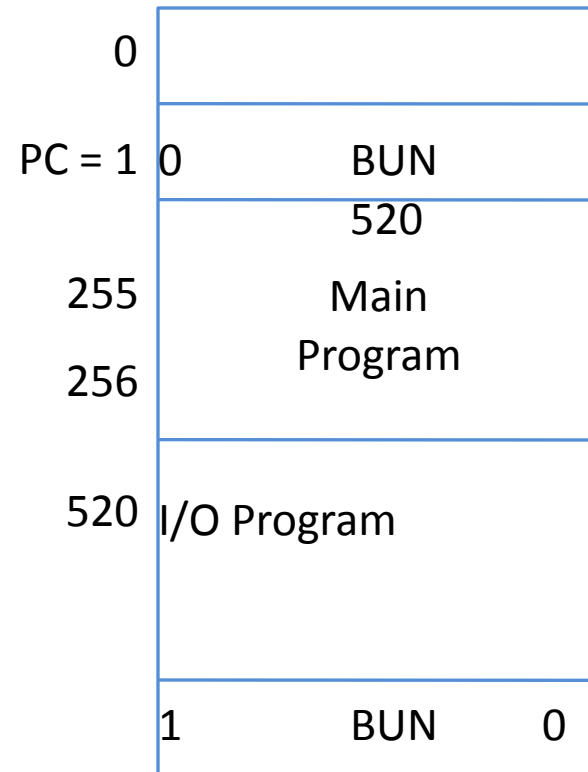
Figure: Flow chart for interrupt cycle

Interrupt Cycle

- It is hardware implementation of a branch and save return address operation.



(a) Before interrupt



(a) After interrupt

Figure: Demonstration of interrupt cycle



Complete Computer Description

