Basic Computer Organization and Design

- INSTRUCTION CODES
- COMPUTER REGISTERS
- COMPUTER INSTRUCTIONS
- TIMING AND CONTROL
- INSTRUCTION CYCLE
- INPUT OUTPUT AND INTERRUPT

Prepared By: Bednidhi Rijal

Instruction codes

- Instruction code
  - A group of bits that instructs the computer to perform a specific operation.
- Usually divided into parts--- each having its own particular interpretation
  - Operation code (op-code) (Most basic part)
    - a group of bits that defines operations as add, subtract, multiply, shift, complement etc.
  - Other part of an instruction may specifies the register or memory where the operands are to be found, as well as the register or memory where the operands are to be stored
- *Note: Number of operation codes for any machine are finite. So the number of bits assigned to an op-code is also finite. The op-code must consist of at least n bits for a given 2^n (or less).*

Stored Program Organization (Basic computer)

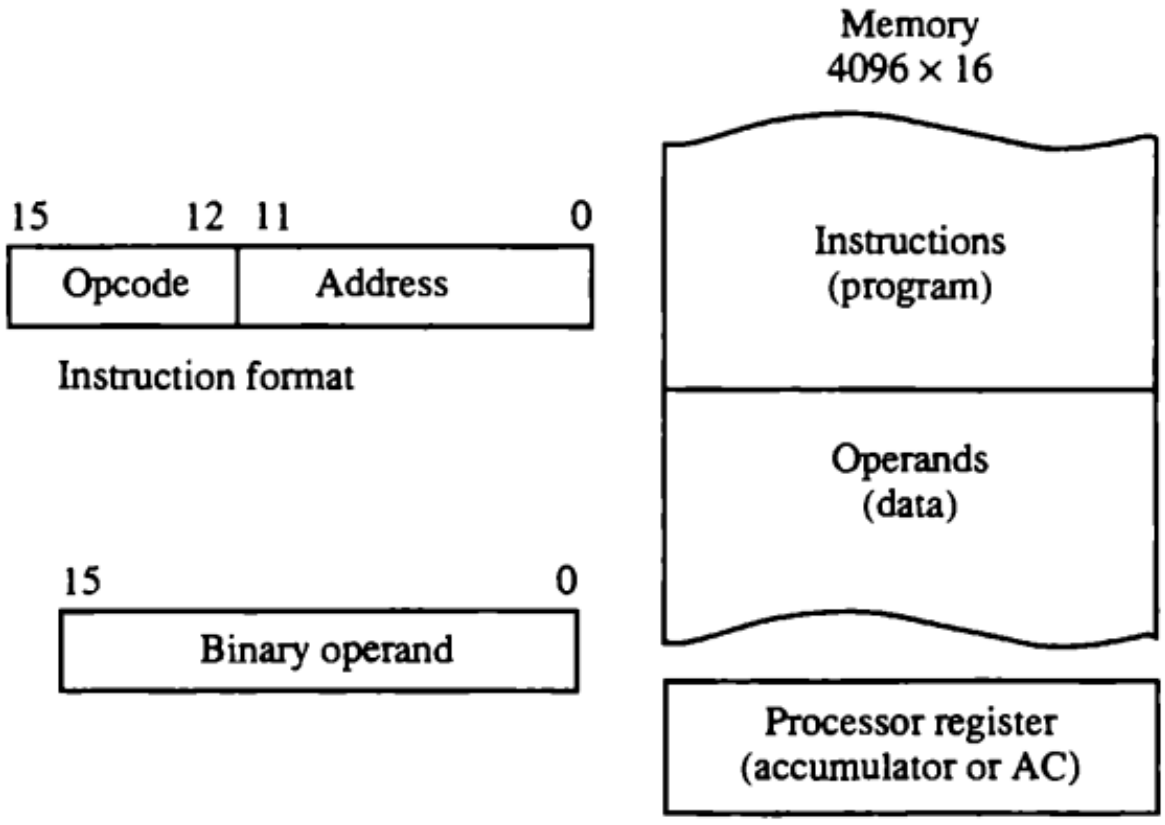
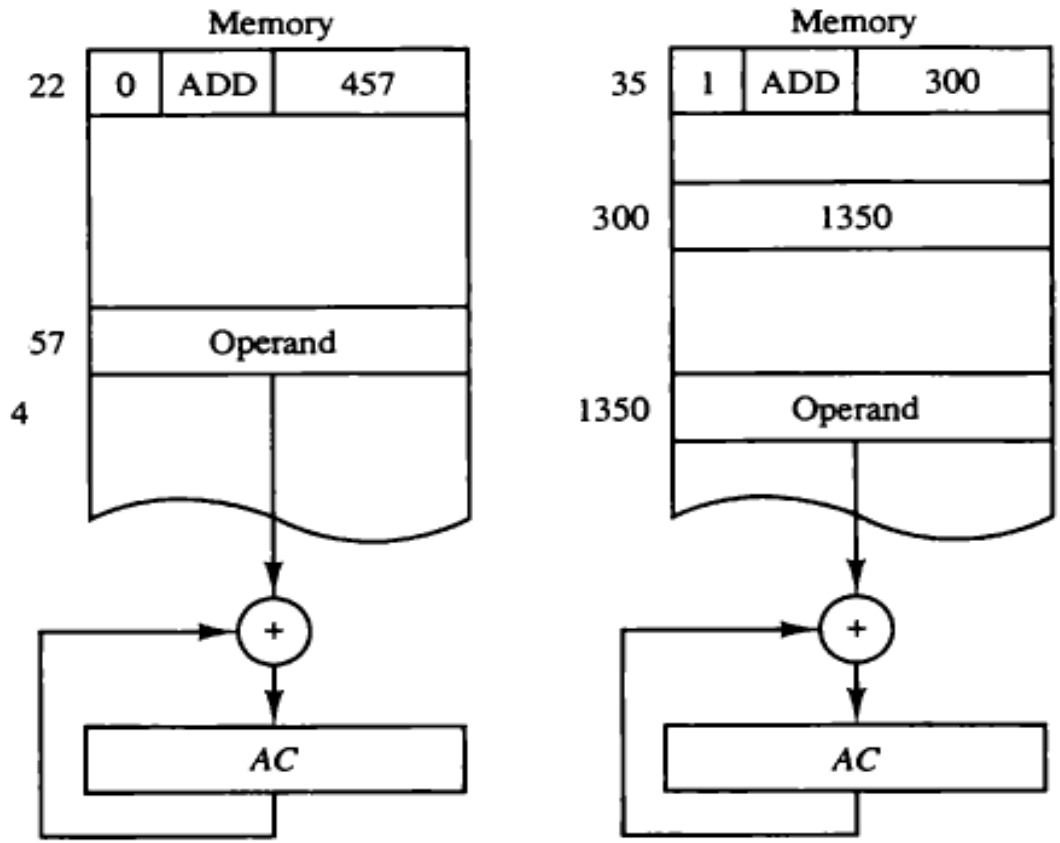


Figure Stored program organization.

Direct and Indirect Address

Effective address: An address of the operand in a computation type instruction or target address in a branch type instruction

Immediate instruction: Instruction in which second part specifies operand



(b) Direct address (c) Indirect address

Figure Demonstration of direct and indirect address.

Computer Register

TABLE : List of Registers for the Basic Computer

Register symbol	Number 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

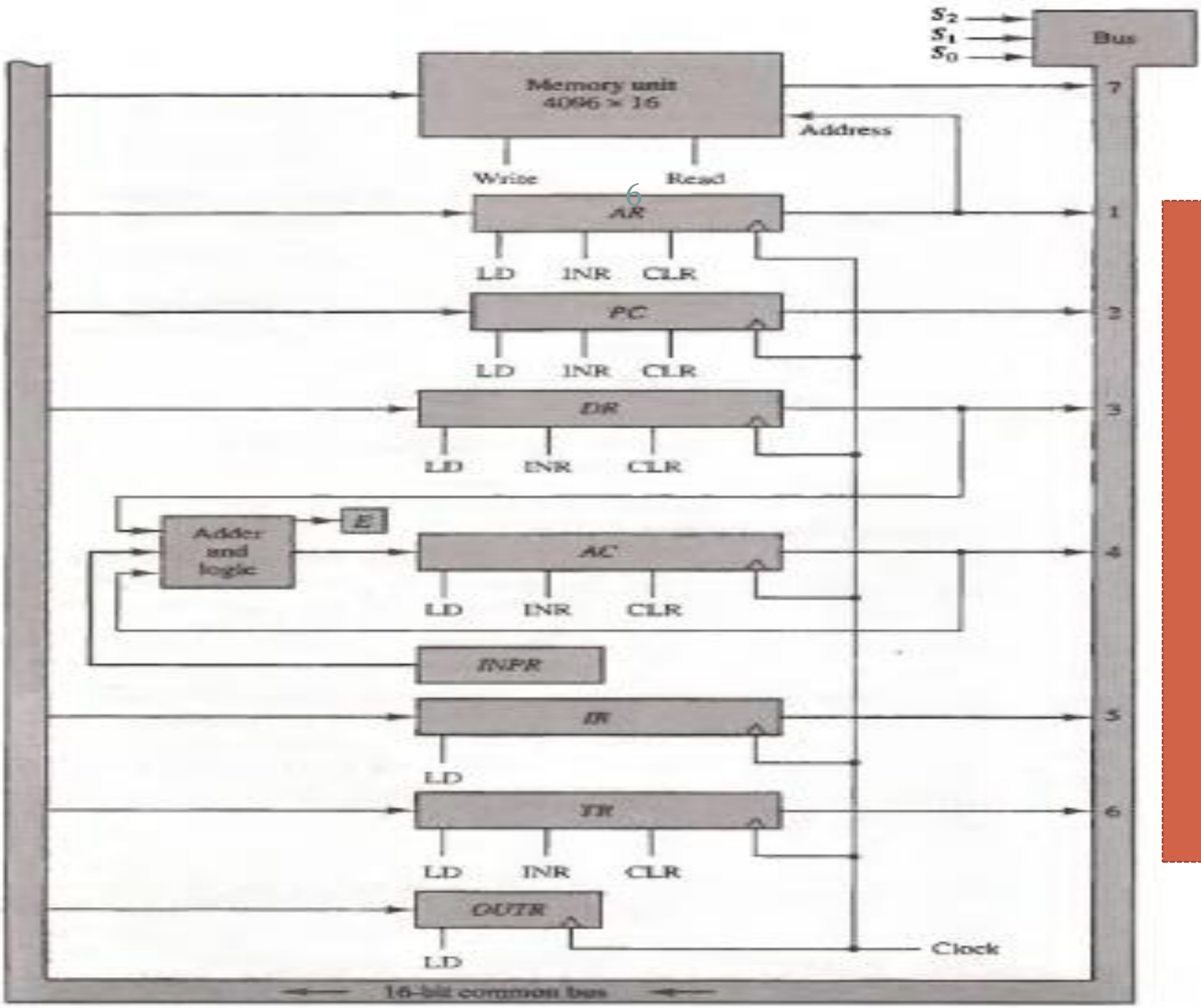


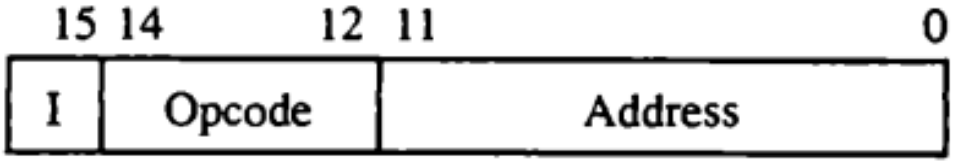
Figure Basic computer registers connected to a common bus.

COMMON BUS SYSTEM

Computer Instruction Format

- Basic computer has three instruction code format:
  - Memory reference instruction
  - Register reference instruction
  - I/O reference instruction

Memory Reference Instruction

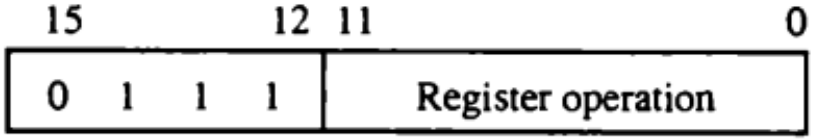


(Opcode = 000 through 110)

(a) Memory – reference instruction

Symbol	Hexadecimal code $I = 0$	Hexadecimal code $I = 1$	Description
AND	0xxx	8xxx	AND memory word to *AC*
ADD	1xxx	9xxx	Add memory word to *AC*
LDA	2xxx	Axxx	Load memory word to *AC*
STA	3xxx	Bxxx	Store content of *AC* in memory
BUN	4xxx	Cxxx	Branch unconditionally
BSA	5xxx	Dxxx	Branch and save return address
ISZ	6xxx	Exxx	Increment and skip if zero

Register Reference Instruction

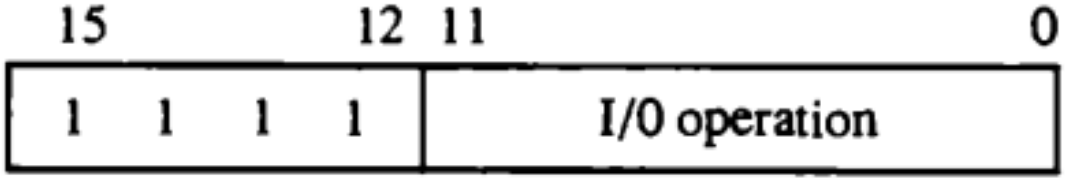


(Opcode = 111, $I = 0$)

Register – reference instruction

Symbol	Hex code	Description
CLA	7800	Clear *AC*
CLE	7400	Clear *E*
CMA	7200	Complement *AC*
CME	7100	Complement *E*
CIR	7080	Circulate right *AC* and *E*
CIL	7040	Circulate left *AC* and *E*
INC	7020	Increment *AC*
SPA	7010	Skip next instruction if *AC* positive
SNA	7008	Skip next instruction if *AC* negative
SZA	7004	Skip next instruction if *AC* zero
SZE	7002	Skip next instruction if *E* is 0
HLT	7001	Halt computer

Input – Output Instruction



(Opcode = 111, $I = 1$)

Input – output instruction

Symbol	Hex code	Description
INP	F800	Input character to *AC*
OUT	F400	Output character from *AC*
SKI	F200	Skip on input flag
SKO	F100	Skip on output flag
ION	F080	Interrupt on
IOF	F040	Interrupt off

Instruction Set Completeness

- The set of instructions are said to be complete if the computer incudes a sufficient number of instructions in each of the following categories:
  - Arithmetic, logical, and shift instructions
  - Memory transfer instruction
  - Program control instruction
  - Input and Output instructions

Control unit

- Control unit generates control signal and provides control inputs for the multiplexers in the common bus, control inputs in processor registers, and micro-operations for the accumulator
- There are two types of control organization:
  - Hardwired Control
    - CU is made up of sequential and combinational circuits to generate the control signals.
    - If logic is changed we need to change the whole circuitry
    - Expensive
    - Fast
  - Micro-programmed Control
    - A control memory on the processor contains micro-programs that activate the necessary control signals
    - If logic is changed we only need to change the micro-program
    - Cheap
    - Slow

Block diagram of hardwired CU

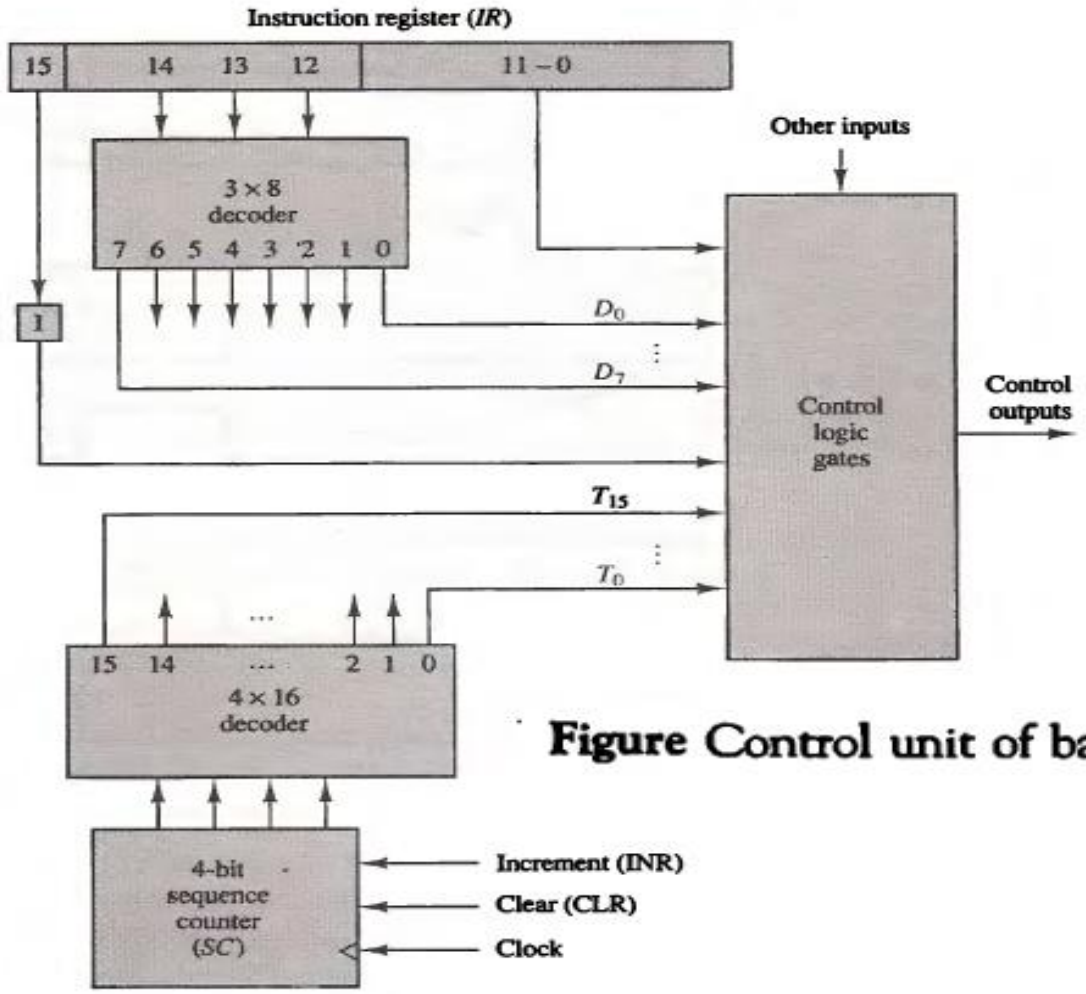
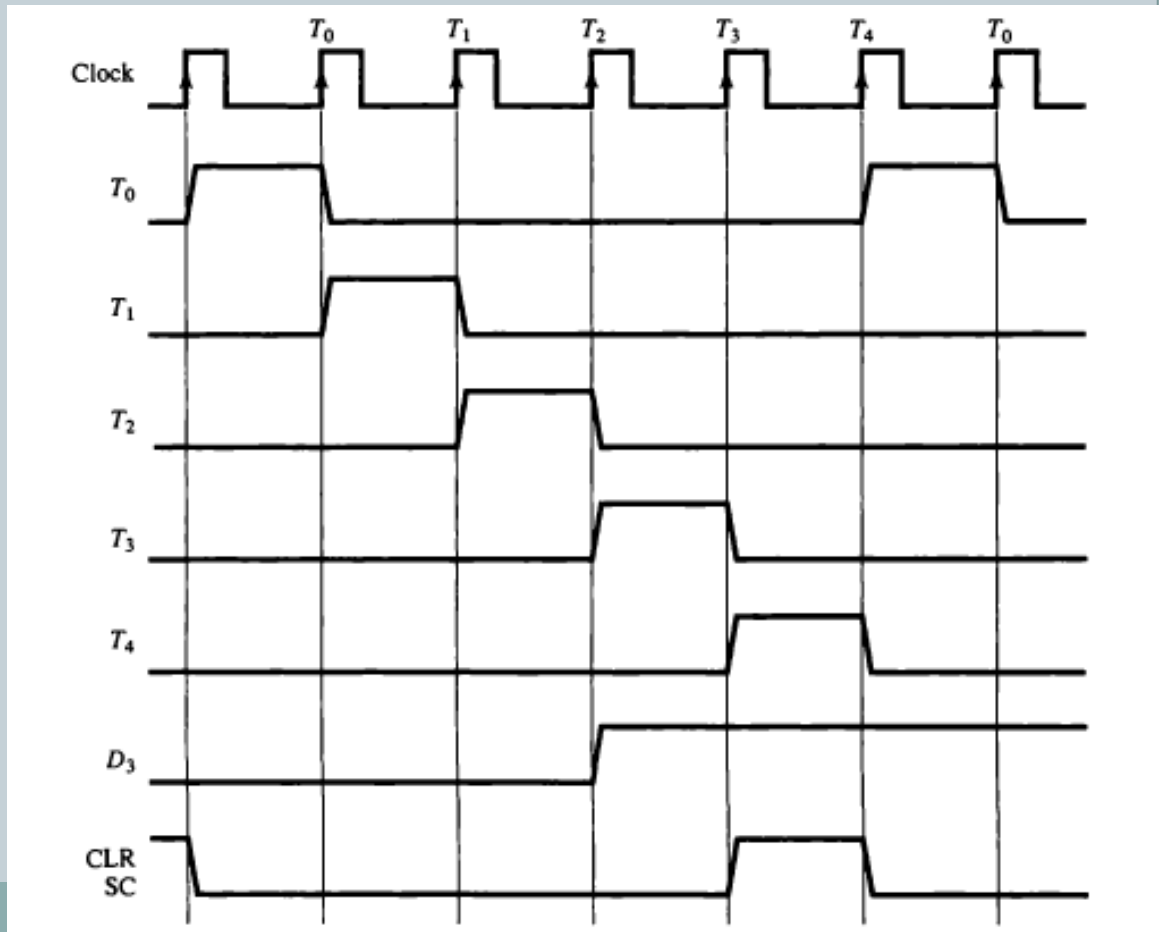


Figure Control unit of basic computer.

Example of Control Timing Signals

- SC is incremented to provide timing signal T0,T1,T2,T3,T4 in sequence. At Time T4 SC is cleared to 0 if decoder output D3 is active.

  i.e. D3T4: SC ← 0



Instruction Cycle

- Instruction cycle is defined as the time required for completing the execution of an instruction.
- Phases of Instruction cycle:
  - Fetch an instruction from memory
  - Decode the instruction
  - Read the effective address from memory if the instruction has an indirect address
  - Execute the instruction

Fetch and Decode

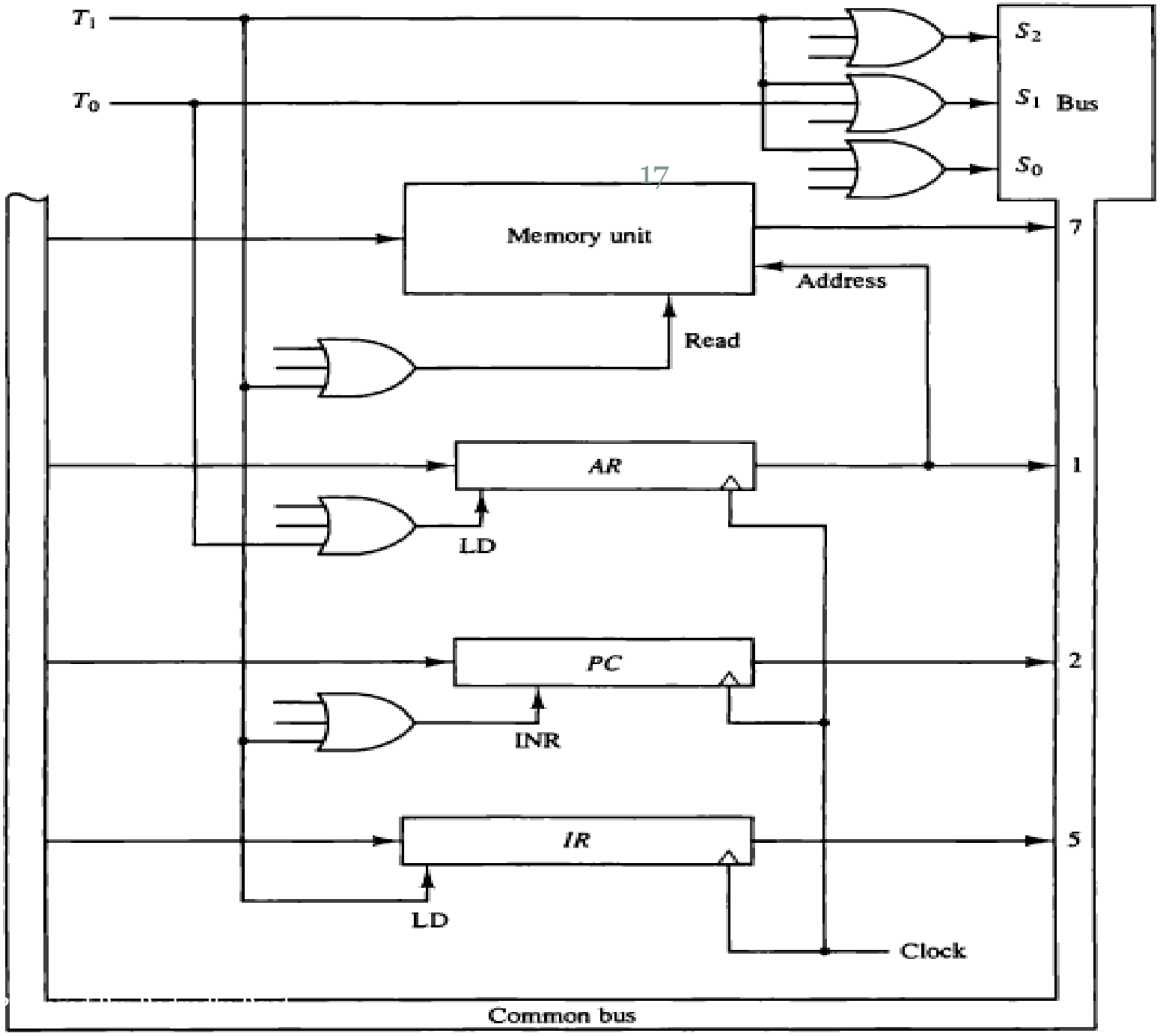
- The micro-operations for the fetch and decode phases can be specified by:

T_0: $AR \leftarrow PC$

T_1: $IR \leftarrow M[AR]$, $PC \leftarrow PC + 1$

T_2: $D_0, \ldots, D_7 \leftarrow$ Decode $IR(12\text{–}14)$, $AR \leftarrow IR(0\text{–}11)$, $I \leftarrow IR(15)$

Register Transfers for the Fetch Phase



Determine the type of instruction

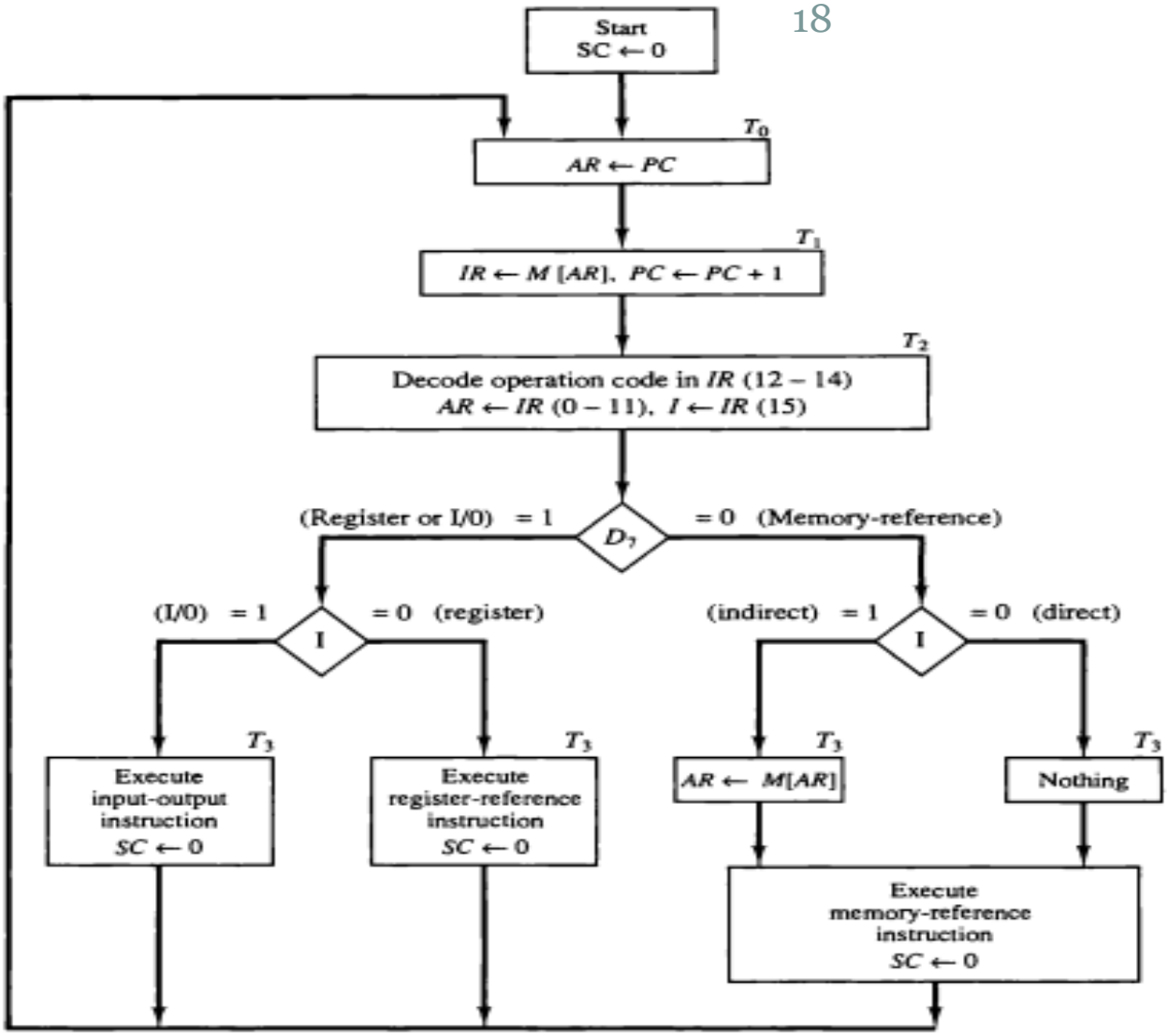


Figure Flowchart for instruction cycle (initial configuration).

Flow Chart for Instruction Cycle

Contd..

- The three instruction types are divided into four separates paths
- The selected operation is activated with the clock transition associated with timing signal T3.
- This can be symbolized as:

$D_7' I T_3$: $AR \leftarrow M[AR]$
$D_7' I' T_3$: Nothing
$D_7 I' T_3$: Execute a register-reference instruction
$D_7 I T_3$: Execute an input–output instruction

Register reference instruction

TABLE Execution of Register-Reference Instructions

$D_7I'T_3 = r$ (common to all register-reference instructions)
$IR(i) = B_i$ [bit in $IR(0–11)$ that specifies the operation]

	Control function	Micro-operations	
	r:	$SC \leftarrow 0$	Clear SC
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:	$AC \leftarrow \text{shr } AC, AC(15) \leftarrow E, E \leftarrow AC(0)$	Circulate right
CIL	rB_6:	$AC \leftarrow \text{shl } AC, AC(0) \leftarrow E, E \leftarrow AC(15)$	Circulate left
INC	rB_5:	$AC \leftarrow AC + 1$	Increment AC
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 zero
SZE	rB_1:	If $(E = 0)$ then $(PC \leftarrow PC + 1)$	Skip if E zero
HLT	rB_0:	$S \leftarrow 0$ (S is a start–stop flip-flop)	Halt computer

Memory Reference instructions

TABLE Memory-Reference Instructions

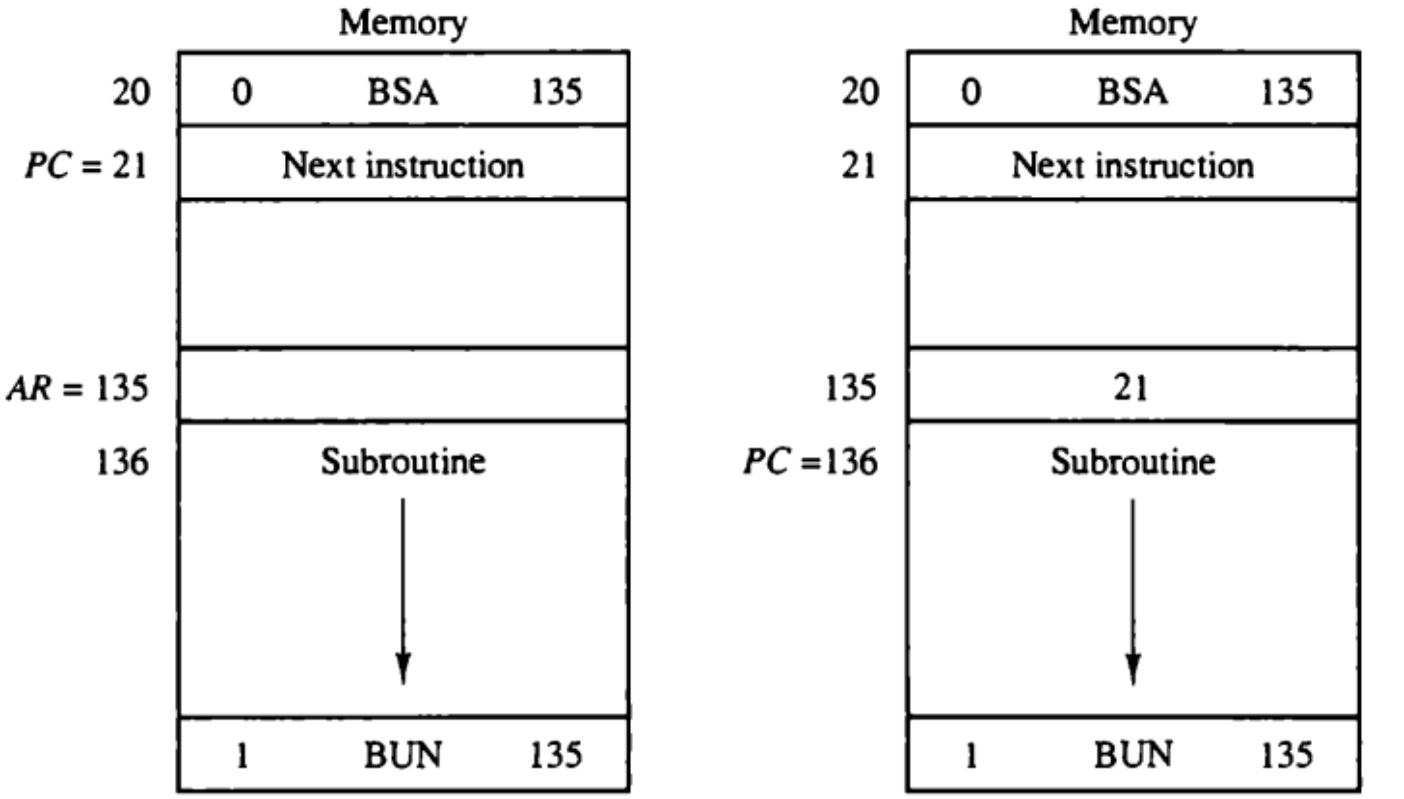
Symbol	Operation decoder	Symbolic description
AND	D_0	$AC \leftarrow AC \wedge M[AR]$
ADD	D_1	$AC \leftarrow AC + M[AR], \quad E \leftarrow C_{out}$
LDA	D_2	$AC \leftarrow M[AR]$
STA	D_3	$M[AR] \leftarrow AC$
BUN	D_4	$PC \leftarrow AR$
BSA	D_5	$M[AR] \leftarrow PC, \quad PC \leftarrow AR + 1$
ISZ	D_6	$M[AR] \leftarrow M[AR] + 1,$ If $M[AR] + 1 = 0$ then $PC \leftarrow PC + 1$

(micro-operation that executes memory ref. instruction) Contd..

AND	D_0T_4: $DR \leftarrow M[AR]$ D_0T_5: $AC \leftarrow AC \wedge DR,\quad SC \leftarrow 0$
ADD	D_1T_4: $DR \leftarrow M[AR]$ D_1T_5: $AC \leftarrow AC + DR,\quad E \leftarrow C_{out},\quad SC \leftarrow 0$
LDA	D_2T_4: $DR \leftarrow M[AR]$ D_2T_5: $AC \leftarrow DR,\quad SC \leftarrow 0$
STA	D_3T_4: $M[AR] \leftarrow AC,\quad SC \leftarrow 0$
BUN	D_4T_4: $PC \leftarrow AR,\quad SC \leftarrow 0$
BSA	$M[AR] \leftarrow PC,\quad PC \leftarrow AR + 1$
ISZ	D_6T_4: $DR \leftarrow M[AR]$ D_6T_5: $DR \leftarrow DR + 1$ D_6T_6: $M[AR] \leftarrow DR,\quad$ if $(DR = 0)$ then $(PC \leftarrow PC + 1),\quad SC \leftarrow 0$

Example of BSA instruction execution

Figure Example of BSA instruction execution.



Input-Output Configuration

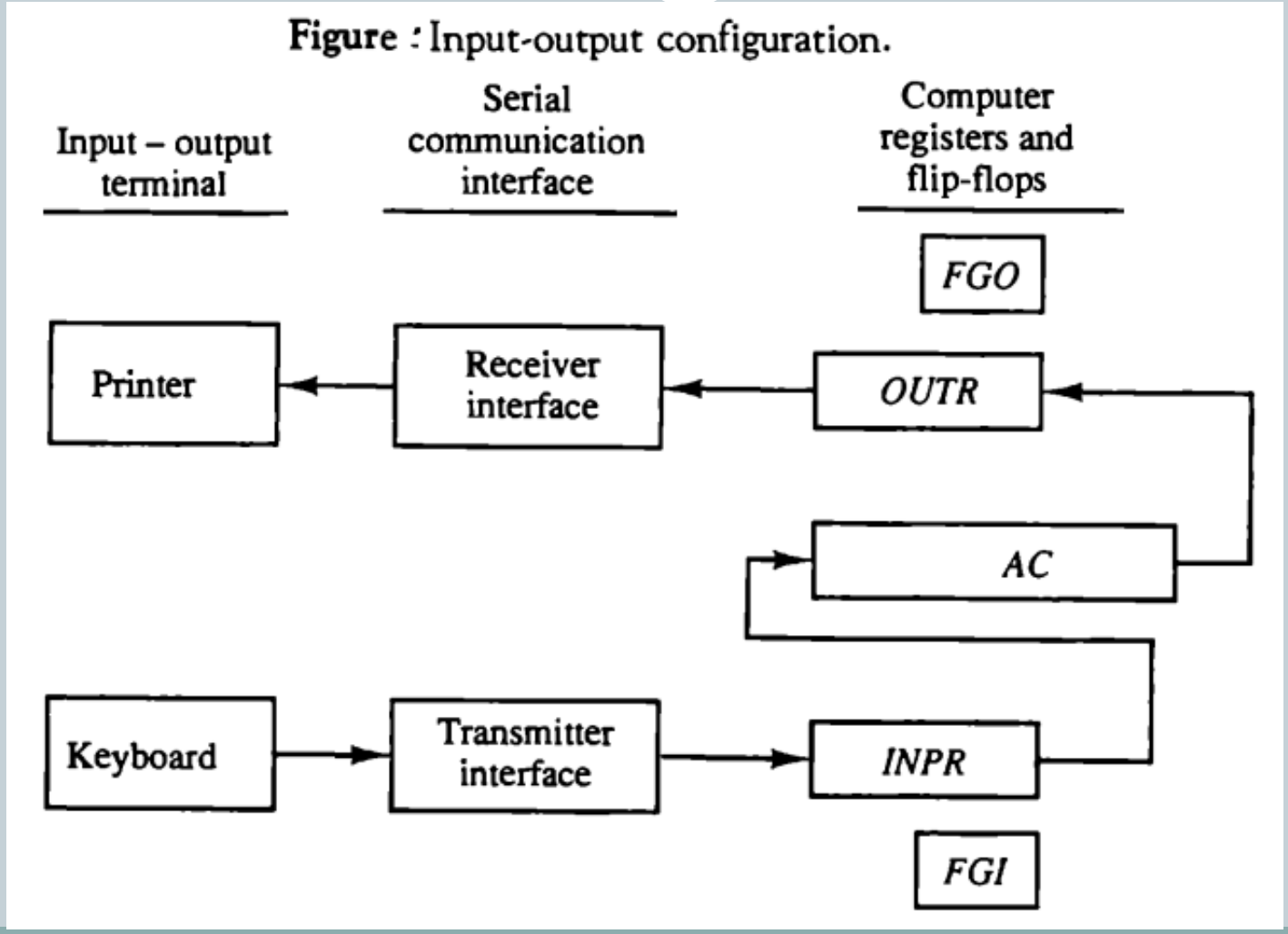


Figure : Input-output configuration.

Input-Output instructions

TABLE Input-Output Instructions

$D_7IT_3 = p$ (common to all input–output instructions)
$IR(i) = B_i$ [bit in $IR(6–11)$ that specifies the instruction]

	Control Fucntion	Micro-Operations	
	p:	$SC \leftarrow 0$	Clear SC
INP	pB_{11}:	$AC(0–7) \leftarrow INPR$, $FGI \leftarrow 0$	Input character
OUT	pB_{10}:	$OUTR \leftarrow AC(0–7)$, $FGO \leftarrow 0$	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

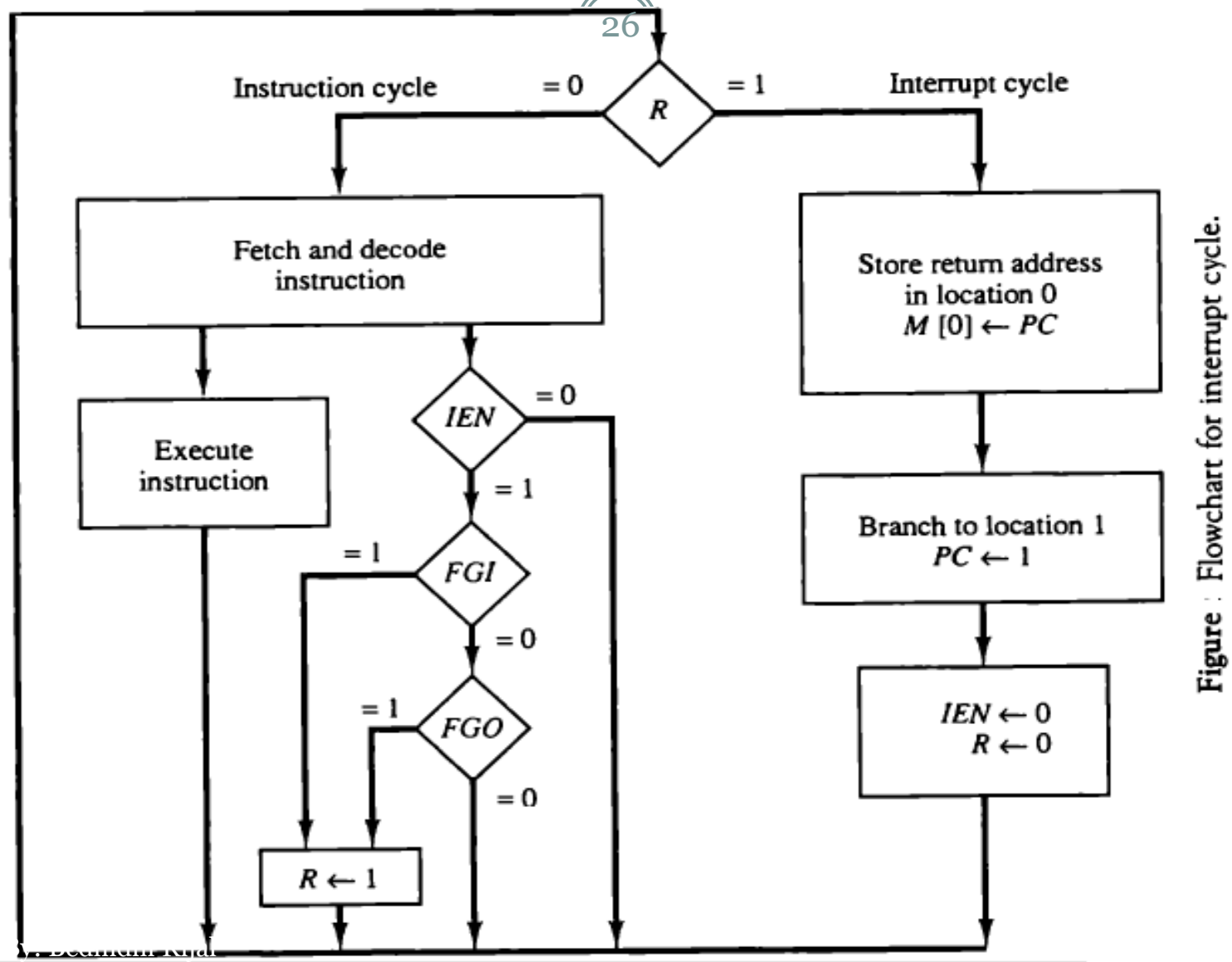
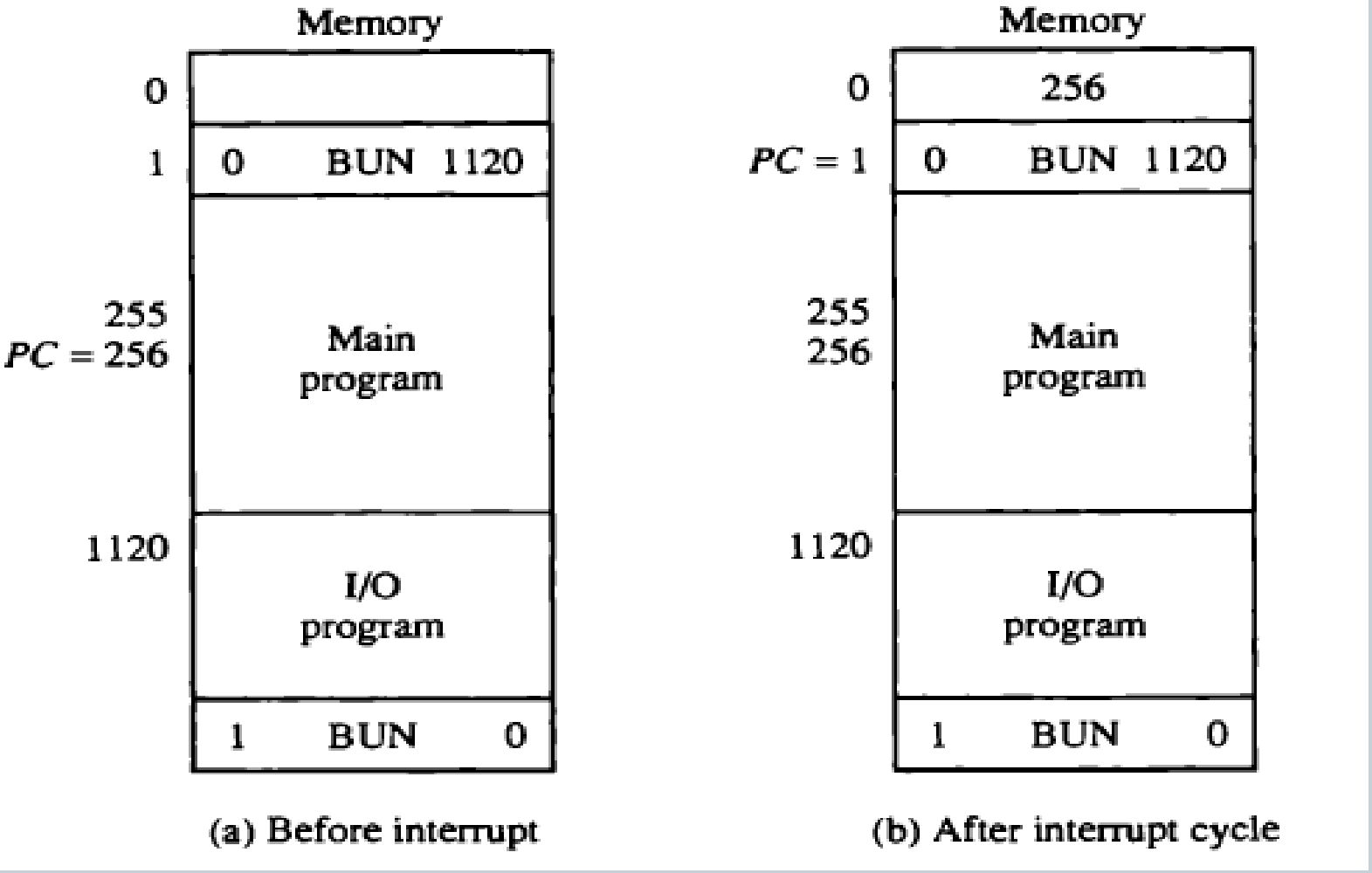


Figure : Flowchart for interrupt cycle.

Demonstration of Interrupt cycle



Assignment

- Differentiate between programmed controlled transfer and interrupt transfer.
- Explain the flowchart for interrupt cycle.