Content

- Instruction Execution Cycle (Fetch-Decode-Execute Cycle)
- TYPE OF INSTRUCTION
  - REGISTER REFERENCE INSTRUCTIONS
  - MEMORY REFERENCE INSTRUCTIONS
  - Input-Output Instructions

INSTRUCTION CYCLE

- In Basic Computer, a machine instruction is executed in the following cycle:
  1. Fetch an instruction from memory
  2. Decode the instruction
  3. Read the effective address from memory if the instruction has an indirect address
  4. Execute the instruction
- After an instruction is executed, the cycle starts again at step 1, for the next instruction
- *Note*: Every different processor has its own (different) instruction cycle

Fetch-Decode-Execute Cycle

- During the **fetch** part, the CPU fetches the next instruction from the address contained in the **Program Counter** and places the instruction in the **Instruction Register**.
  - When a program starts, the program counter contains 0, so the instruction at address 0 is fetched.
- As soon as an instruction is fetched, the CPU adds **1** word to the contents of the **Program Counter**, so that it will contain the address of the next **sequential** instruction.

Fetch-Decode-Execute Cycle

- The decode unit within the CPU deciphers the instruction in the Instruction Register and determines what operations need to be done and what type of operands will be used (decode part).
- During the execution part, the specified operation is performed (add, compare, etc).
- After execution of the instruction has been completed the cycle starts all over again (unless the instruction terminates the program).

Fetch-Decode-Execute Diagram

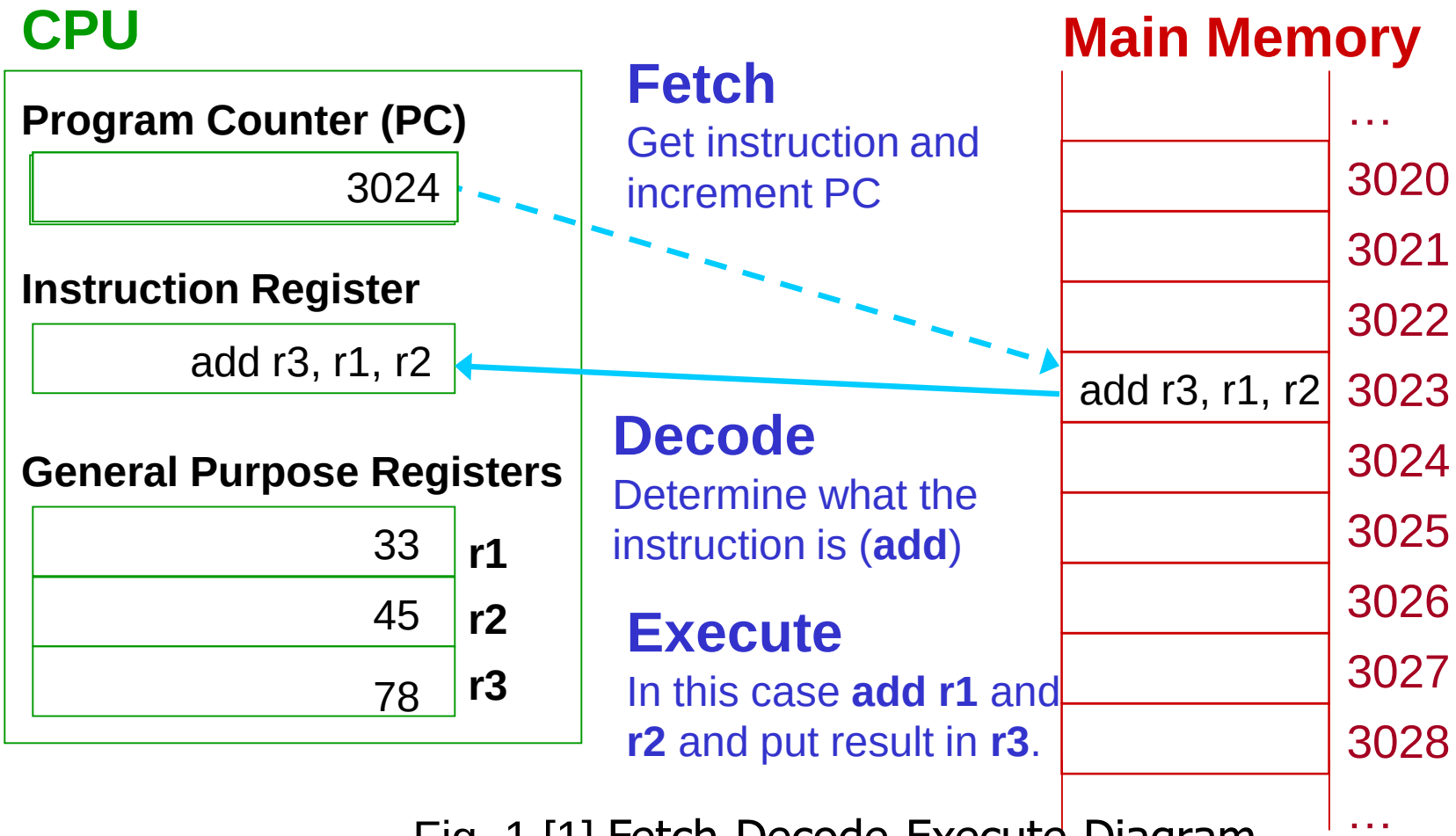


Fig. 1 [1] Fetch-Decode-Execute Diagram

FETCH and DECODE

- **T0: AR ← PC ($S_0S_1S_2$=010, T0=1)**
- **T1: IR ← M [AR], PC ← PC + 1 (S0S1S2=111, T1=1)**
- **T2: D0, . . . , D7 ← Decode IR(12-14), AR ← IR(0-11), I ← IR(15)**

To provide the data path for the transfer of PC to AR we must apply timing signal T0 to achieve the following connection:

1. Place the content of PC onto the bus by making the bus selection inputs s2s1s0 equal to 010.
2. Transfer the content of the bus to AR by enabling the LD input of AR .

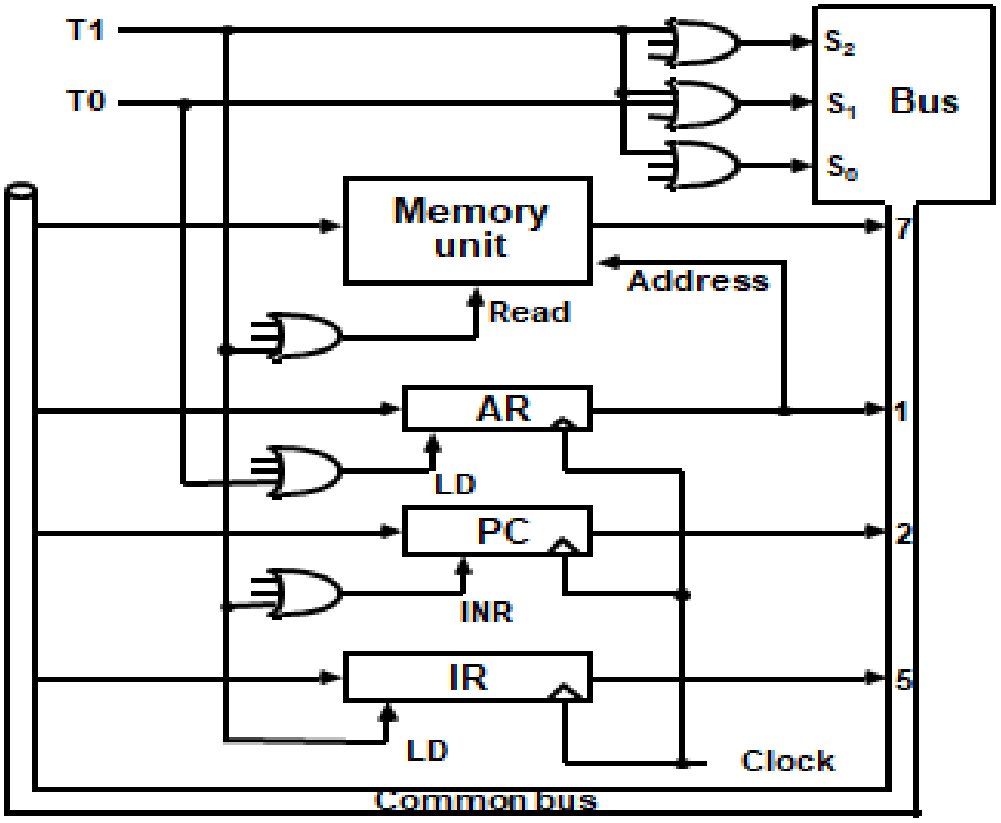


Fig. 2 [2] Register transfer for fetch phase

FETCH and DECODE

Timing signal T1 provide the following connections in the bus system.

- Enable the read input of memory.
- Place the content of memory onto the bus by making S2S1S0 = 111.
- Transfer the content of the bus to IR by enabling the LD input of IR .
- Increment PC by enabling the INR input of PC .

DETERMINE THE TYPE OF INSTRUCTION

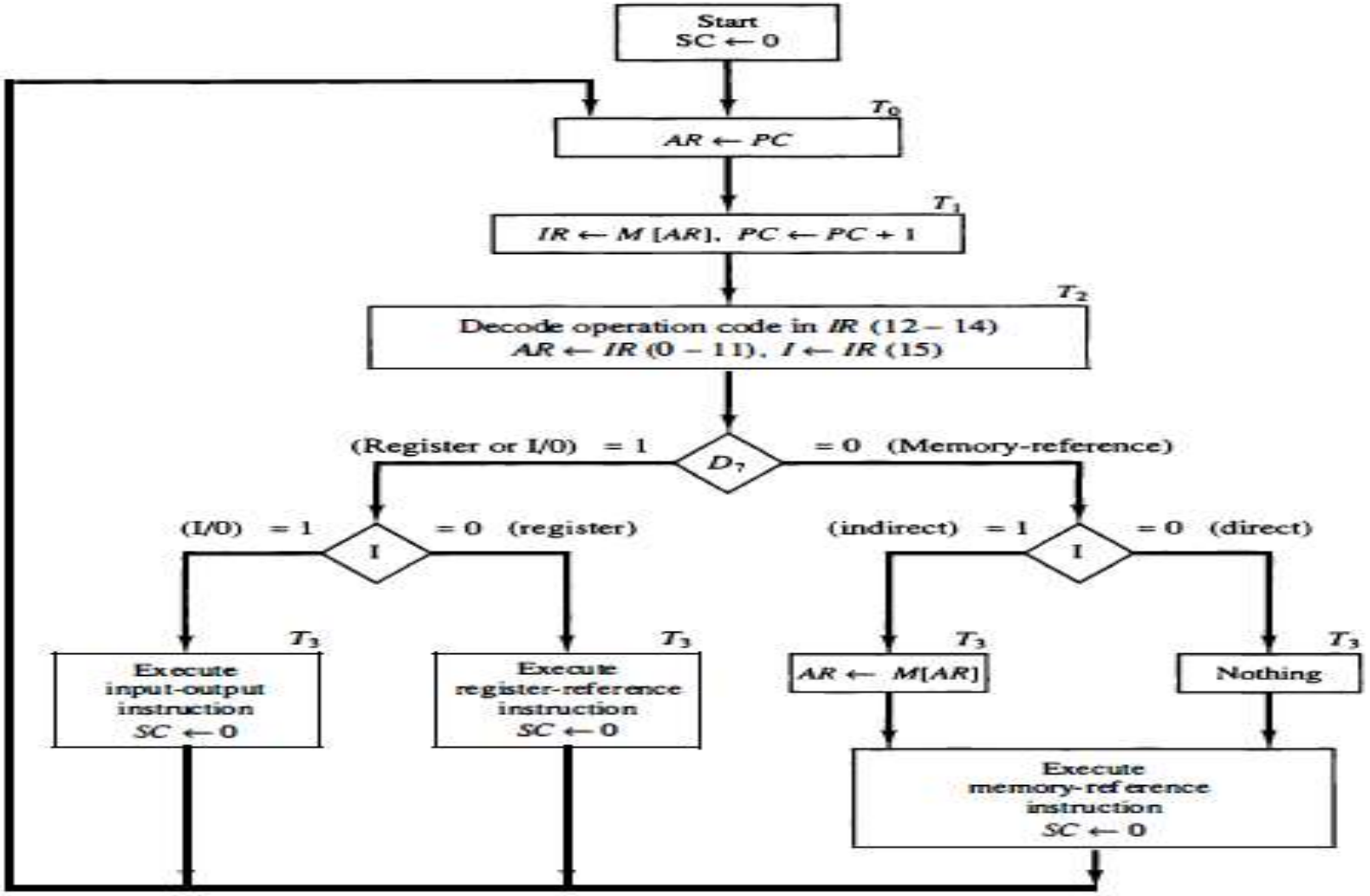


Fig. 3 [2] Register transfer for fetch phase

DETERMINE THE TYPE OF INSTRUCTION

- The three instruction types are subdivided into four separate paths.
- The selected operation is activated with the clock transition associated with timing signal T3.

D7’ I T3: AR <-- M [AR]
D7’ I' T3: Nothing
D7 I' T3: Execute a register-reference instruction
D7IT3: Execute an input-output instruction

REGISTER REFERENCE INSTRUCTIONS

Register Reference Instructions are identified when

- $D_7 = 1$, $I = 0$
- Register Ref. Instr. is specified in $b_0 \sim b_{11}$ of IR
- Execution starts with timing signal T_3

$r = D_7 I'T_3$ => Register Reference Instruction

$B_i = IR(i)$, i=0,1,2,...,11

	r:	SC ← 0
CLA	rB_{11}:	AC ← 0
CLE	rB_{10}:	E ← 0
CMA	rB_9:	AC ← AC'
CME	rB_8:	E ← E'
CIR	rB_7:	AC ← shr AC, AC(15) ← E, E ← AC(0)
CIL	rB_6:	AC ← shl AC, AC(0) ← E, E ← AC(15)
INC	rB_5:	AC ← AC + 1
SPA	rB_4:	if (AC(15) = 0) then (PC ← PC+1)
SNA	rB_3:	if (AC(15) = 1) then (PC ← PC+1)
SZA	rB_2:	if (AC = 0) then (PC ← PC+1)
SZE	rB_1:	if (E = 0) then (PC ← PC+1)
HLT	rB_0:	S ← 0 (S is a start-stop flip-flop)

MEMORY REFERENCE INSTRUCTIONS

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

Fig. 4 [2] Register transfer for fetch phase

- The effective address of the instruction is in AR and was placed there during timing signal T_2 when I = 0, or during timing signal T_3 when I = 1
- Memory cycle is assumed to be short enough to complete in a CPU cycle
- The execution of MR instruction starts with T_4

AND to AC

D_0T_4: $DR \leftarrow M[AR]$ Read operand

D_0T_5: $AC \leftarrow AC \wedge DR$, $SC \leftarrow 0$ AND with AC

ADD to AC

D_1T_4: $DR \leftarrow M[AR]$ Read operand

D_1T_5: $AC \leftarrow AC + DR$, $E \leftarrow C_{out}$, $SC \leftarrow 0$ Add to AC and store carry in E

MEMORY REFERENCE INSTRUCTIONS

- LDA: Load to AC

  D_2T_4: DR $\leftarrow$ M[AR]

  D_2T_5: AC $\leftarrow$ DR, SC $\leftarrow$ 0
- STA: Store AC

  D_3T_4: M[AR] $\leftarrow$ AC, SC $\leftarrow$ 0
- BUN: Branch Unconditionally

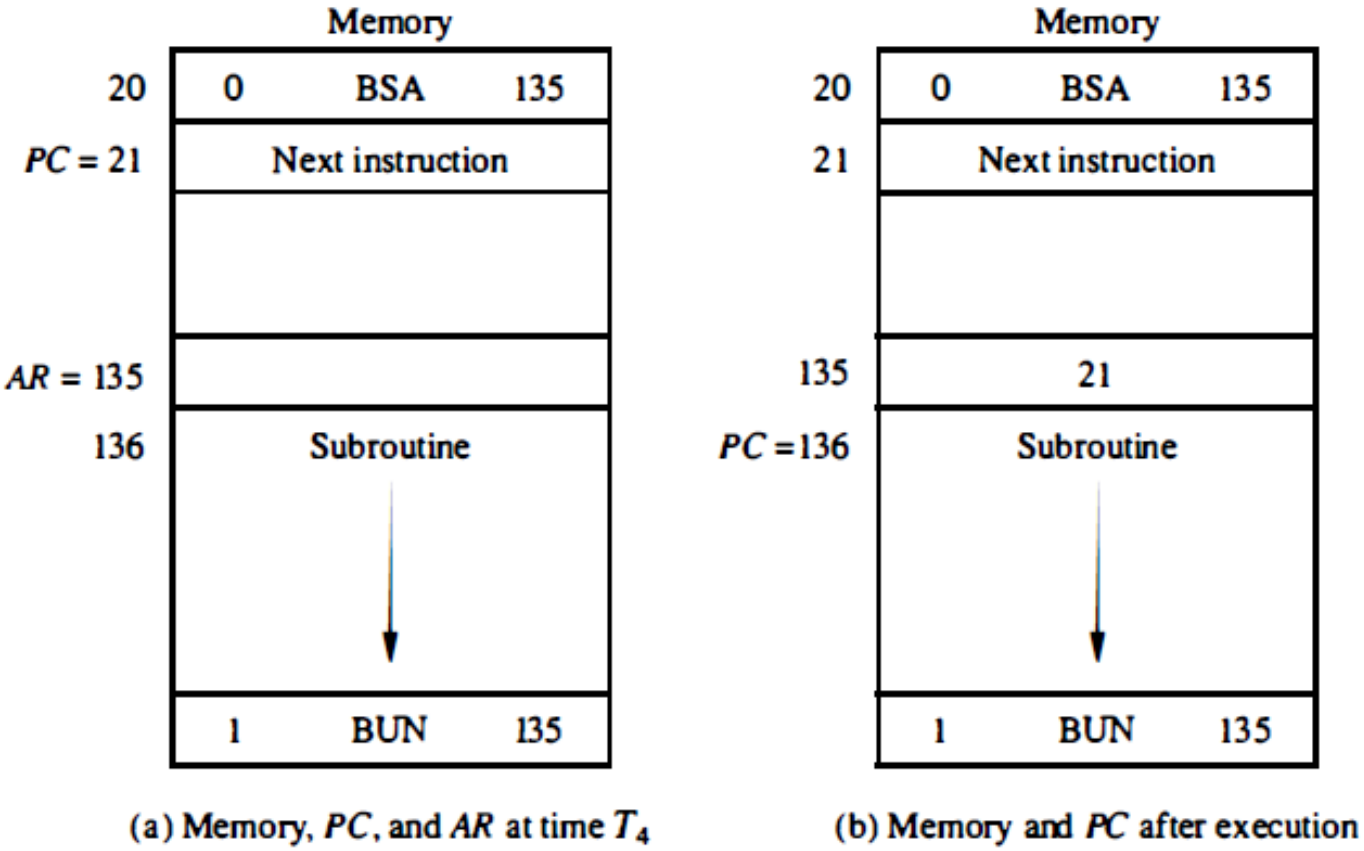
  D_4T_4: PC $\leftarrow$ AR, SC $\leftarrow$ 0

MEMORY REFERENCE INSTRUCTIONS

BSA: Branch and Save Return Address

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

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



Example of BSA instruction execution[1].

- ISZ: Increment and Skip-if-Zero
- D_6T_4: DR $\leftarrow$ M[AR]
- D_6T_5: DR $\leftarrow$ DR + 1
- D_6T_4: M[AR] $\leftarrow$ DR, if (DR = 0) then (PC $\leftarrow$ PC + 1), SC $\leftarrow$ 0

FLOWCHART FOR MEMORY REFERENCE INSTRUCTIONS

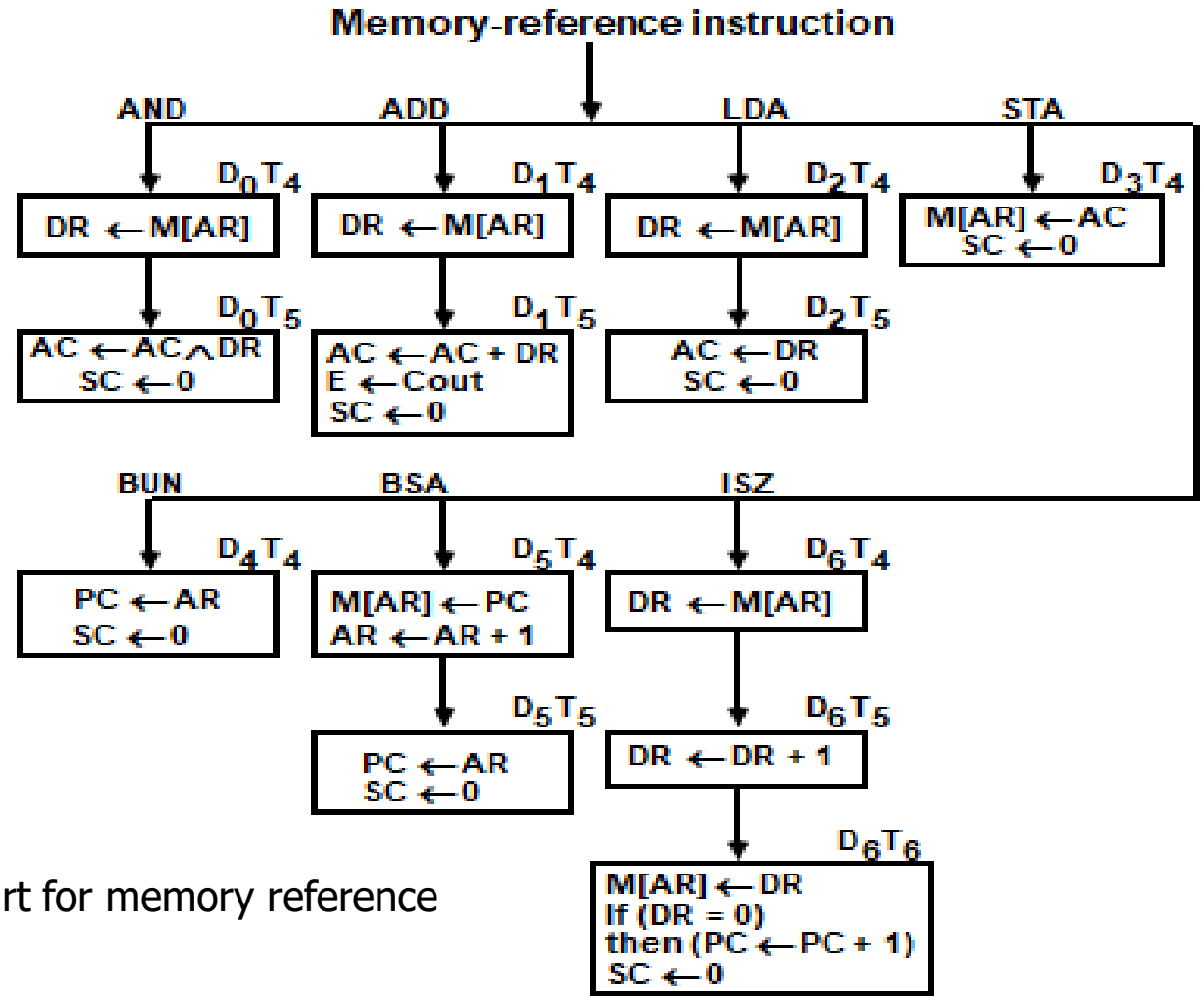


Fig. 5 [2] Flowchart for memory reference instructions

Input-Output Instructions

TABLE 5-5 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]

	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

Fig. 6 [2] Input output instructions

References

- [1]. Tanenbaum & Austin, Structured Computer Organization, 6th Edition, Pearson Education
- [2]. M. Morris Mano, Computer System Architecture, Prentice Hall of India Pvt Ltd, 3rd Edition (upda ted) , 30 June 2017.
- [3]. William Stallings, Computer Organization and Architecture– Designing for Performance, Ninth Edition, Pearson Education, 2013.