Determine the Type of Instructions

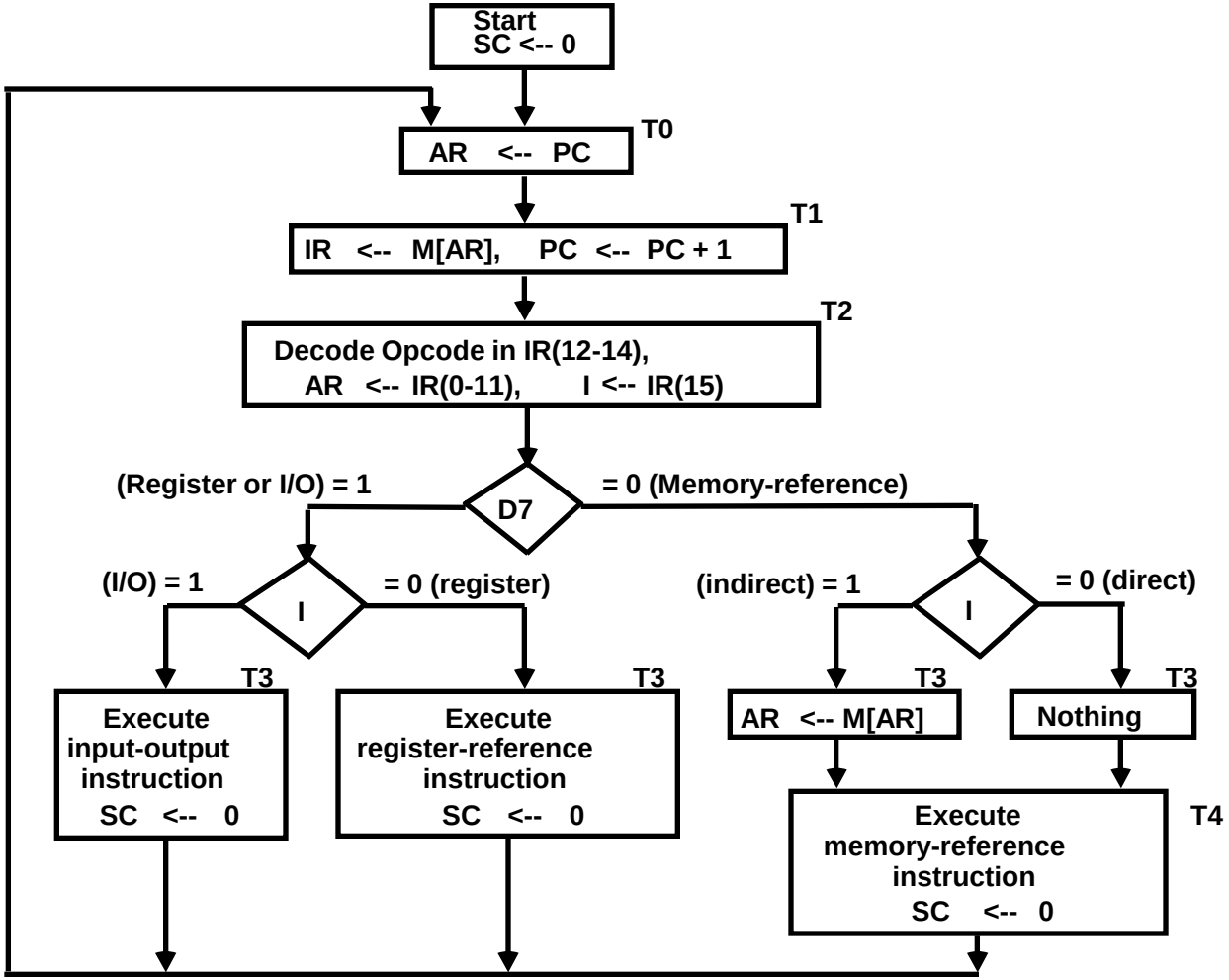


Fig : Flow chart for Instruction Cycle

Determining Type of Instruction

- **D'_7IT_3: AR ← M[AR]**
- **$D'_7I'T_3$: Nothing**
- **$D_7I'T_3$: Execute a register-reference instr.**
- **D_7IT_3: Execute an input-output instr.**

Register Reference Instruction

Register Reference Instructions are identified when

- $D_7 = 1,\ I = 0$
- Register Ref. Instr. is specified in b_0 ~ 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

e.g. rB_{11}=CLA

	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)

In case of Register Reference Instructions, CLA Stands for …….

A) Clear Address Register
B) Clear Accumulator Register
C) Clear Account Register
D) None of the above

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 $PC \leftarrow PC+1$

- 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 //performs AND logic with AC and memory word specified by EA

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

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

Memory Reference Instructions

ADD to AC // add content of memory word specified by EA to value of AC
sum is transferred to AC and Carry to E (Extended Accumulator)

D_1T_4: DR ← M[AR] Read operand
D_1T_5: AC ← AC + DR, E ← C_{out}, SC ← 0 Add to AC and store carry in E

LDA: Load to AC // Transfers memory word specified by memory address to AC
D_2T_4: DR ← M[AR]
D_2T_5: AC ← DR, SC ← 0

STA: Store AC // Stores the content of AC into memory specified by EA
D_3T_4: M[AR] ← AC, SC ← 0

BUN: Branch Unconditionally // Transfer program to instruction specified by EA
D_4T_4: PC ← AR, SC ← 0

Memory Reference Instructions

BSA: Branch and Save Return Address // 1. stores address of next instruction in sequence (PC) into address specified by EA 2. EA+1 transfer to PC serve as 1st inst. In subroutine

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

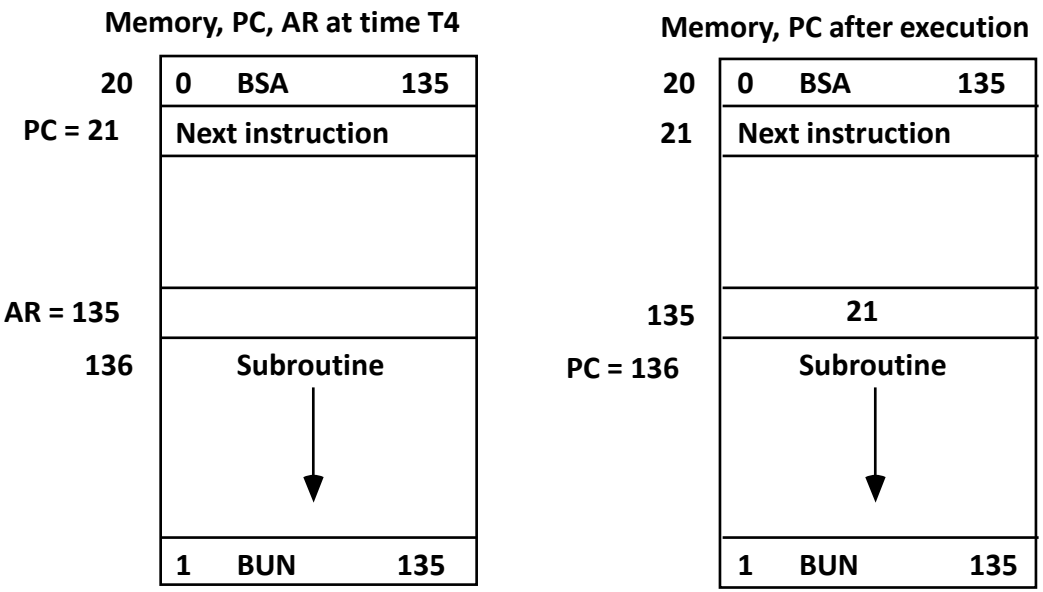
BSA:

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

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

BSA: Example

$M[135] \leftarrow 21, PC \leftarrow 135 + 1 = 136$



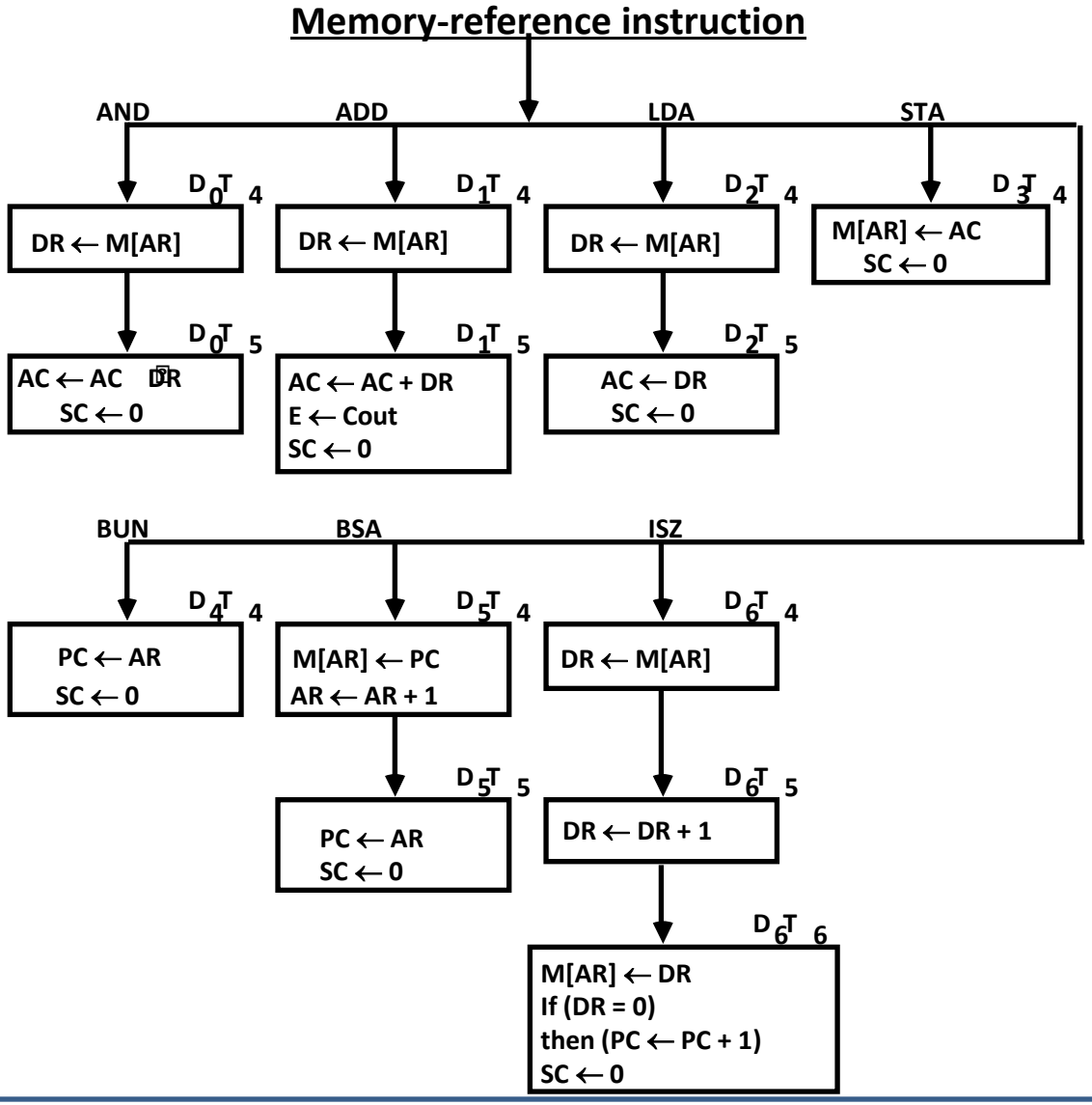
Memory Reference Instructions

ISZ: Increment and Skip-if-Zero

// increments the word specified by effective address, and if incremented value=0 , PC incremented by 1

D_6T_4: DR ← M[AR]
D_6T_5: DR ← DR + 1
D_6T_4: M[AR] ← DR, if (DR = 0) then (PC ← PC + 1), SC ← 0

Flow Chart - Memory Reference Instructions



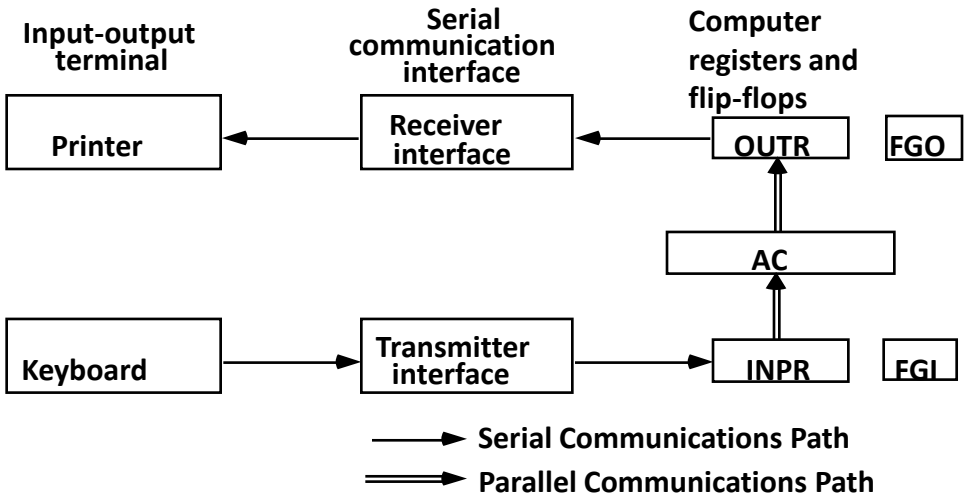
Which of the following instructions is used to save the return address?

A) BUN
B) BSA
C) ISZ
D) None of the above

Input/Output and Interrupt

A Terminal with a keyboard and a Printer

Input-Output Configuration



INPR	Input register - 8 bits
OUTR	Output register - 8 bits
FGI	Input flag - 1 bit
FGO	Output flag - 1 bit
IEN	Interrupt enable - 1 bit

- The terminal sends and receives serial information
- The serial info. from the keyboard is shifted into INPR
- The serial info. for the printer is stored in the OUTR
- INPR and OUTR communicate with the communication interface serially and with the AC in parallel.
- The flags are needed to synchronize the timing difference between I/O device and the computer

Determining Type of Instruction

- **FGI =1 when new information available at input device, and cleared to 0 when information accepted by computer**
- **Initially FGI=0, new key pressed , 8 bit alphanumeric shifted to INPR and FGI=1, Computer checks flag if 1 then transfer content to AC and clear FGI to 0.**
- **Initially FGO=1,**
  - **computer checks flag bit if 1, then OUTR ← AC and clears FGO=0**
  - **O/P device accepts information prints character and finally sets FGO=1.**

Input/Output Instructions

I/O instructions are needed for transferring info to and from AC register, for checking the flag bits and for controlling interrupt facility

$D_7IT_3 = p$
$IR(i) = B_i$, i = 6, …, 11

	p:	SC ← 0	Clear SC
INP	pB_{11}:	AC(0-7) ← INPR, FGI ← 0	Input char. to AC
OUT	pB_{10}:	OUTR ← AC(0-7), FGO ← 0	Output char. from AC
SKI	pB_9:	if(FGI = 1) then (PC ← PC + 1)	Skip on input flag
SKO	pB_8:	if(FGO = 1) then (PC ← PC + 1)	Skip on output flag
ION	pB_7:	IEN ← 1	Interrupt enable on
IOF	pB_6:	IEN ← 0	Interrupt enable off