

# Introduction

## ■ Computer Registers

### ◆ Basic computer registers and memory :

- Data Register(**DR**) (16) : hold the operand(Data) read from memory
- Accumulator Register(**AC**) (16) : general purpose processing register
- Instruction Register(**IR**) (16) : hold the instruction read from memory
- Program Counter PC (12) : Holds address of instruction
- Temporary Register(**TR**) (16) : hold a temporary data during processing
- Address Register(**AR**) : hold a memory address, 12 bit width
- INPR/OUTR (8): Holds input/output data

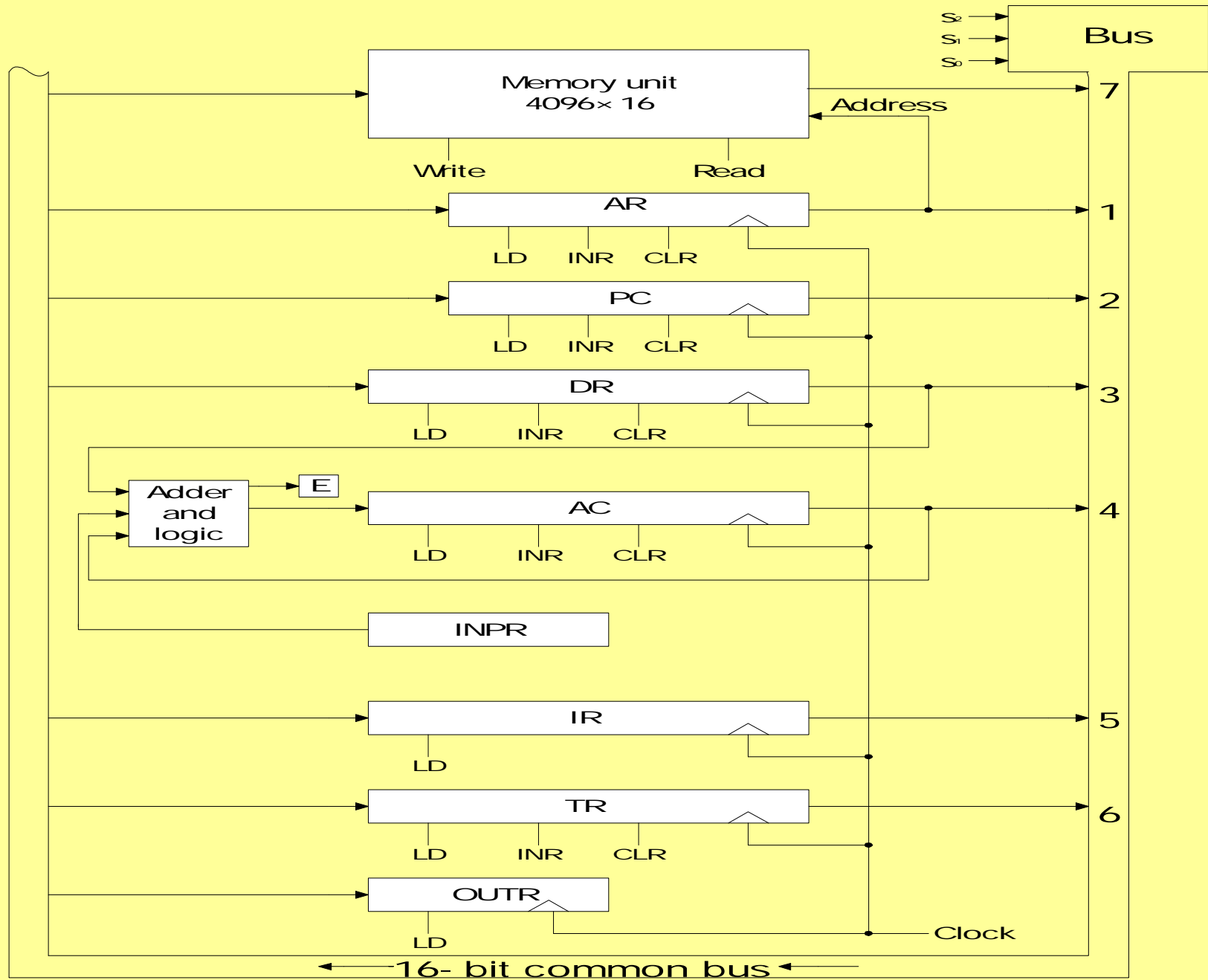
- Program Counter(*PC*) :
  - » hold the address of the next instruction to be read from memory after the current instruction is executed
  - » Instruction words are read and executed in sequence unless a branch instruction is encountered
  - » A branch instruction calls for a transfer to a nonconsecutive instruction in the program
  - » The address part of a branch instruction is transferred to PC to become the address of the next instruction
  - » To read instruction, memory read cycle is initiated, and PC is incremented by one(next instruction fetch)

- Input Register(*INPR*) : receive an 8-bit character from an input device
- Output Register(*OUTR*) : hold an 8-bit character for an output device

### ◆ Common Bus System

- The basic computer has eight registers, a memory unit, and a control unit
- Paths must be provided to transfer information from one register to another and between memory and registers
- A more efficient scheme for transferring information in a system with many registers is to use a common bus

- The connection of the registers and memory of the basic computer to a common bus system :
  - » The outputs of seven registers and memory are connected to the common bus
  - » The specific output is selected by mux(S0, S1, S2) :
    - Memory(7), AR(1), PC(2), DR(3), AC(4), IR(5), TR(6)
    - Device AC INPR OUTR
    - mux memory register bus
    - When LD(Load Input) is enable, the particular register receives the data from the bus
  - » Control Input : LD, INR, CLR, Write, Read
  - » Address Register : Address bus Bus address data )
    - AC DR memory read (*p. 146, LDA*)
    - Memory write  $\equiv$  AC write (*p. 147, STA*)



» Operations on common bus:

- 1) Register Microoperation : clear AC, shift AC,...
- 2) Data Register : **add DR to AC, and DR to AC** ( AC End carry bit set/reset), memory READ(DR)
- 3) INPR : Device(*Adder & Logic*)

» Note) Two microoperations can be executed at the same time

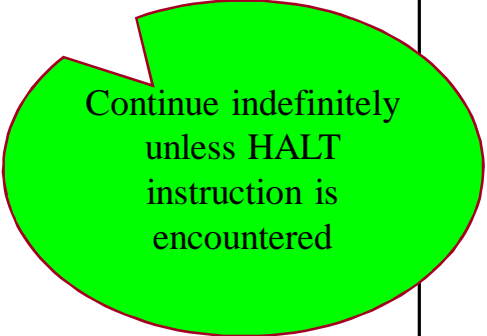
$DR \leftarrow AC : s_2s_1s_0 = 100(4), DR(load)$

$AC \leftarrow DR : DR \rightarrow Adder \& Logic \rightarrow AC(load)$

## ■ 5-5 Instruction Cycle

### ◆ Instruction Cycle

- 1) Instruction Fetch from Memory
- 2) Instruction Decode
- 3) Read Effective Address(if indirect addressing mode)
- 4) Instruction Execution
- 5) Go to step 1) : Next Instruction[PC + 1]

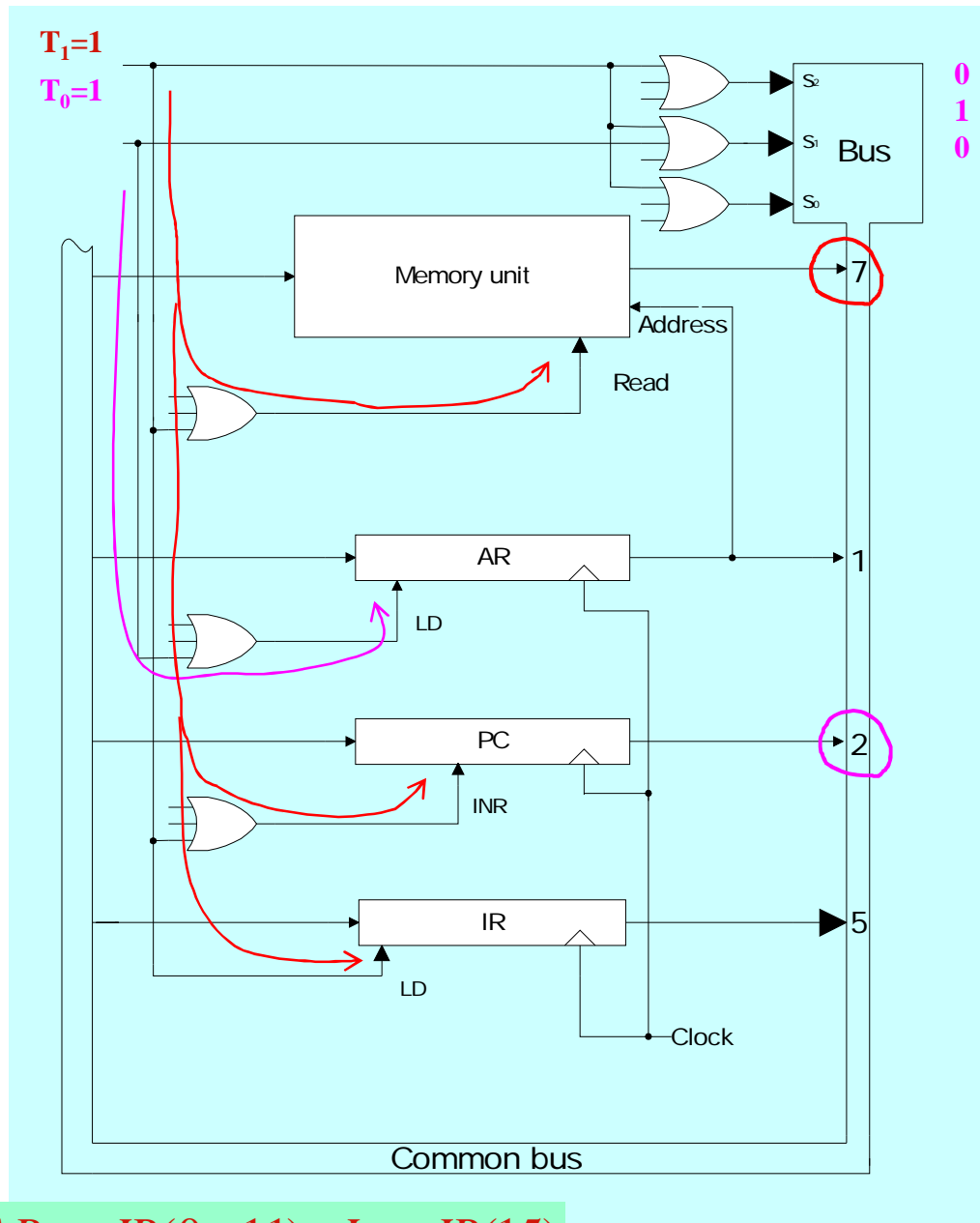


Continue indefinitely  
unless HALT  
instruction is  
encountered

Instruction Fetch : T0, T1  
 Initially the program Counter PC is loaded with the first instruction of program. The sequence counter SC is cleared to 0 providing a decoded timing signal T0. After each clock pulse SC is incremented by one, so the timing can be denoted by T0, T1, T2 .....

$$T_0 : AR \leftarrow PC$$

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



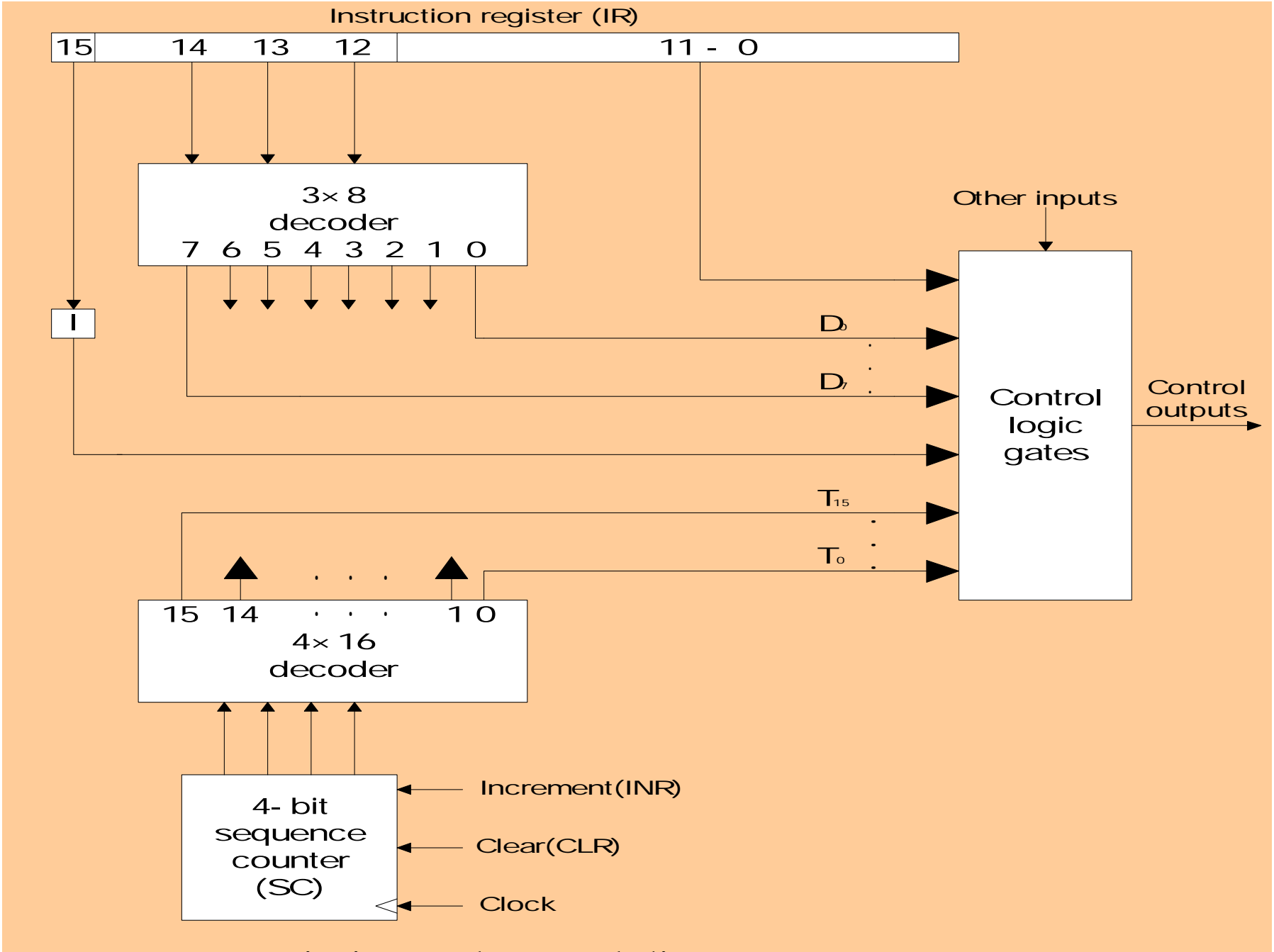
$$T_2 : D_0, \dots, D_7 \leftarrow \text{Decode } IR(12-14), AR \leftarrow IR(0-11), I \leftarrow IR(15)$$



Since only AR is connected to the address input of memory, so it is necessary to transfer the address from PC to AR during the clock associated with timing signal T0.

The instruction read from memory is then placed in the instruction register IR with the clock transition associated with timing T1. At the same time the PC is incremented by one to prepare it for next instruction. At the same time T2, the operation code in IR is decoded and address part of instruction is transferred to AR.

SC is incremented each time after each clock pulse to produce the sequence of T0, T1 and T2.



Timing and control diagram

T0 = 1

$T_0 : AR \leftarrow PC$

- Place the content of PC onto the bus by making the bus selection inputs  $S_2S_1S_0=010$
- Transfer the content of the bus to AR by enabling the LD input of AR

• T1 = 1      $T_1 : IR \leftarrow M[AR], PC \leftarrow PC + 1$

- » 1) Enable the read input memory
- » 2) Place the content of memory onto the bus by making  $S_2S_1S_0= 111$
- » 3) Transfer the content of the bus to IR by enable the LD input of IR
- » 4) Increment PC by enabling the INR input of PC

◆ Instruction Decode : T2

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

Op.code      Address      Di/Indirect

◆ IR(12-14) Fig. 5-6 D0 - D7

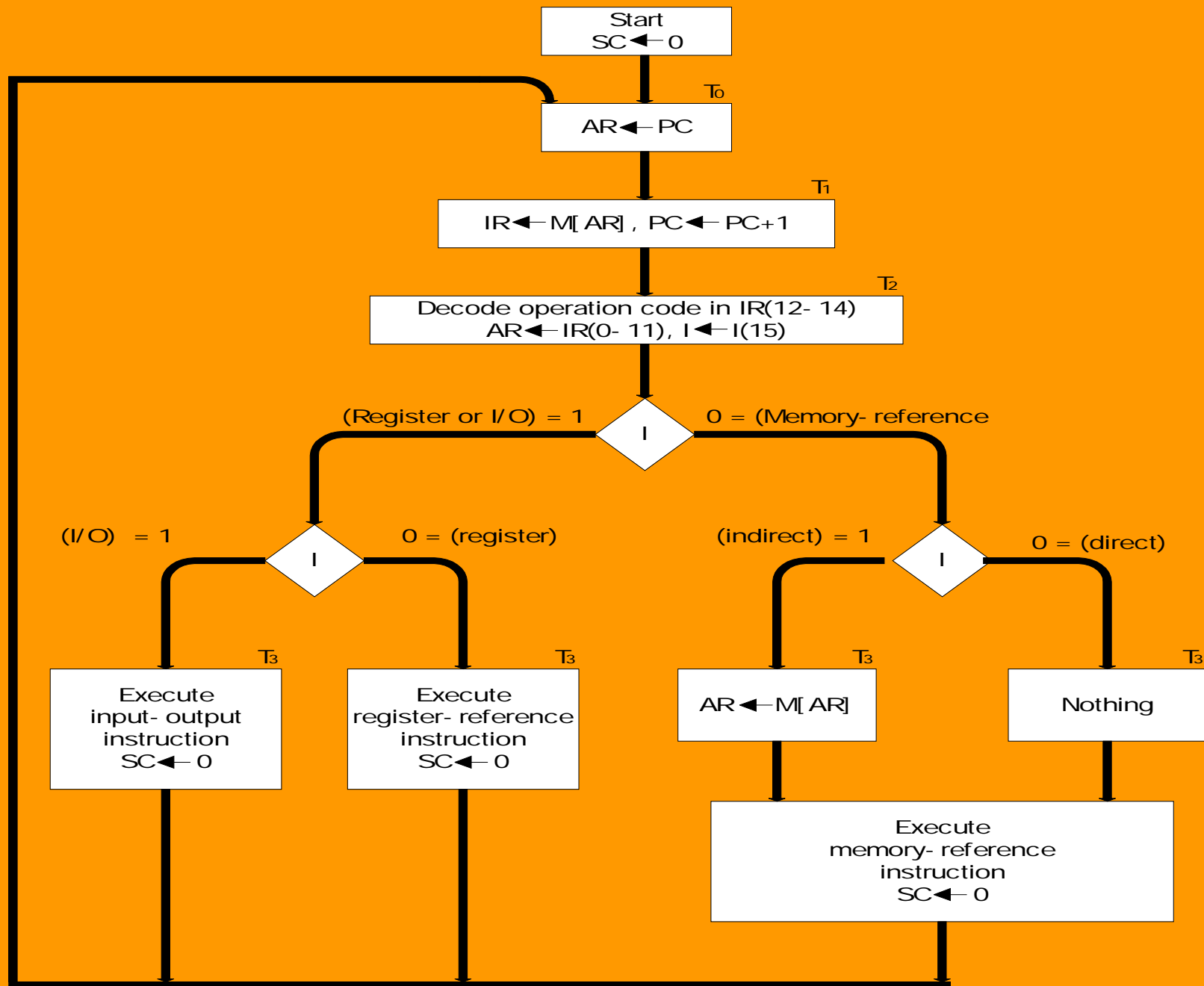
◆ Instruction Execution : T3, T4, T5, T6

$IR(12-14)$   
= 111

$D_7=1$  { Register (I=0)  $\rightarrow D_7 I T_3$  (Execute)      Read effective Address  
           { I/O      (I=1)  $\rightarrow D_7 I T_3$  (Execute)  
 $D_7=0$  : Memory Ref. { Indirect (I=1)  $\rightarrow D_7 I T_3 (AR \leftarrow M[AR])$   
                               { Direct (I=0)  $\rightarrow$  nothing in  $T_3$

◆ Register I/O T3 Memory Ref. T3 Operand effective address

◆ Memory Ref. T4, T5, T6 : Fig. 5-11



# 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]

	$r$ :	$SC \leftarrow 0$	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	$rB_7$ :	$AC \leftarrow shr\ AC, AC(15) \leftarrow E, E \leftarrow AC(0)$	Circulate right
CIL	$rB_6$ :	$AC \leftarrow shl\ AC, AC(0) \leftarrow E, E \leftarrow AC(15)$	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

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 $PC \leftarrow PC + 1$

◆ STA : memory write

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

◆ BUN : branch unconditionally

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

◆ BSA : branch and save return address

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

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

- Return Address : save return address ( 135 ← 21 )

- Subroutine Call : **Fig. 5-10**

◆ ISZ : increment and skip if zero

$D_6T_4 : DR \leftarrow M[AR]$

$D_6T_5 : DR \leftarrow DR + 1$

$D_6T_6 : M[AR] \leftarrow DR, \text{if } (DR = 0) \text{ then } (PC \leftarrow PC + 1), SC \leftarrow 0$

◆ Control Flowchart : **Fig. 5-11**

- Flowchart for the 7 memory reference instruction

- » The longest instruction : ISZ(T6)

- » 3 bit Sequence Counter 4 )

*Fig. 5-10 Example of BSA*

PC = 10	0	BSA 135
PC = 21	next instruction	
135	21(return address)	
PC = 136	Subroutine	
	1	BUN 135

$D_5T_4 : M[135] \leftarrow 21(PC), 136(AR) \leftarrow 135 + 1$   
 $D_5T_5 : 136(PC) \leftarrow 136(AR), SC \leftarrow 0$