به نام خدا

سازمان داخلی کامپیوتر پایه

واحد کنترل (Hardwired)

Dr. Aref Karimiafshar
A.karimiafshar@iut.ac.ir

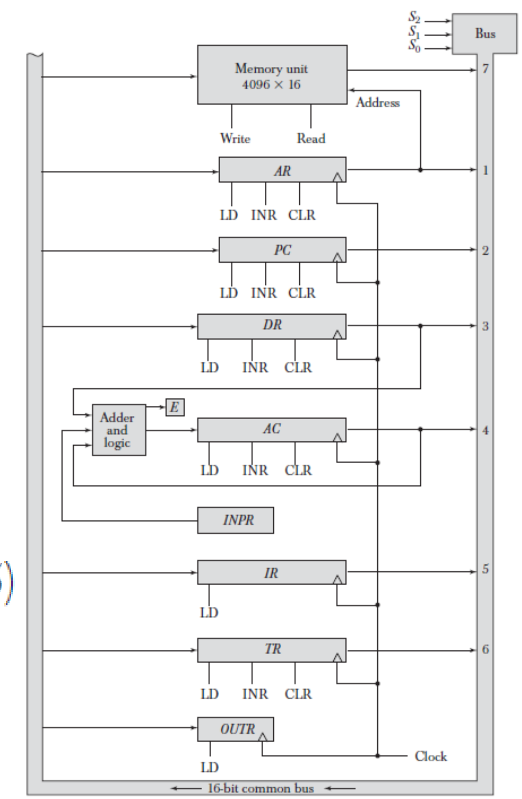
Control unit of basic computer

- A program residing in the memory unit of the computer consists of a sequence of instructions
  - The program is executed in the computer by going through a cycle for each instruction
    - Each instruction cycle in turn is subdivided into a sequence of subcycles or phases
- In the basic computer each instruction cycle consists of the following phases:
  - Fetch an instruction from memory
  - Decode the instruction
  - Read the effective address from memory if the instruction has an indirect address
  - Execute the instruction

Upon the completion of step 4, the control goes back to step 1 to fetch, decode, and execute the next instruction. This process continues indefinitely unless a HALT instruction is encountered

Fetch and Decode

Initially, the program counter PC is loaded with the address of the first instruction in the program. The sequence counter SC is cleared to 0, providing a decoded timing signal T_0. After each clock pulse, SC is incremented by one, so that the timing signals go through a sequence T_0, T_1, T_2, and so on. The microoperations for the fetch and decode phases can be specified by the following register transfer statements.



Fetch

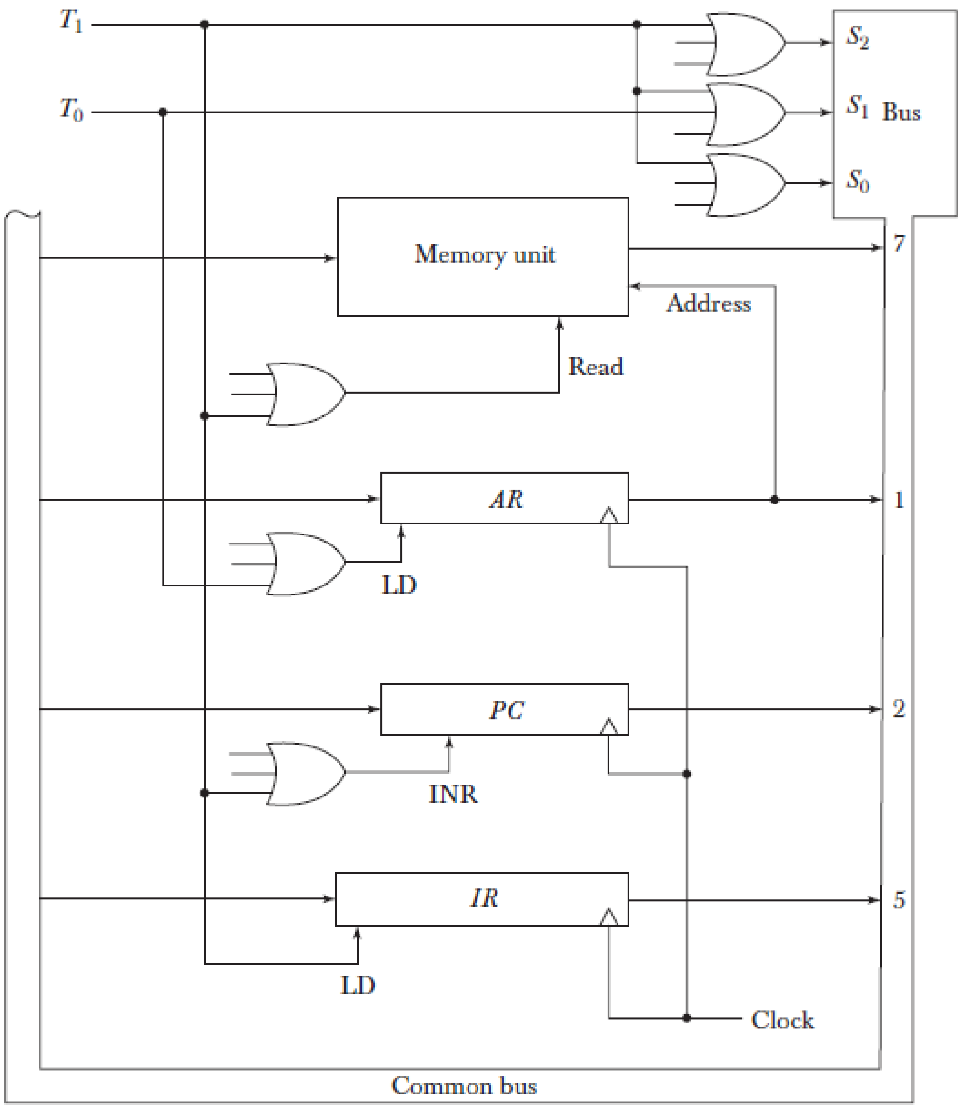
T_0: $AR \leftarrow PC$

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

Decode

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

Fetch and Decode



$$T_0: \quad AR \leftarrow PC$$

1. Place the content of PC onto the bus by making the bus selection inputs $S_2S_1S_0$ equal to 010.
2. Transfer the content of the bus to AR by enabling the LD input of AR.

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

1. Enable the read input of 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 enabling the LD input of IR.
4. Increment PC by enabling the INR input of PC.

Type of Instruction

Basic Computer Instructions

	Hexadecimal code		
Symbol	$I = 0$	$I = 1$	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
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
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

15	14 – 12	11 – 0	
I	Opcode	Address	(Opcode = 000 through 110)

(a) Memory – reference instruction

15 – 12	11 – 0	
0 1 1 1	Register operation	(Opcode = 111, $I = 0$)

(b) Register – reference instruction

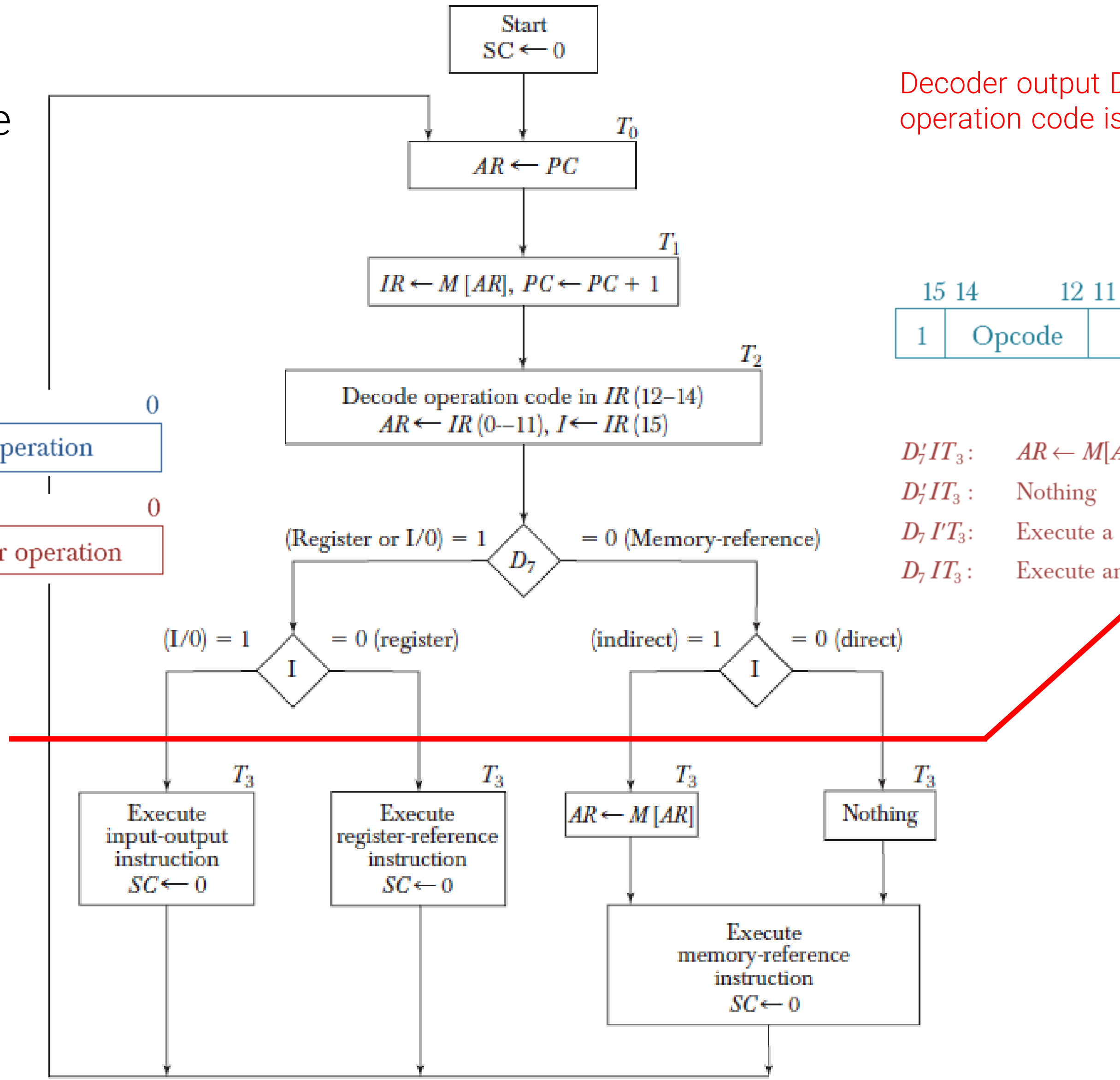
15 – 12	11 – 0	
1 1 1 1	I/0 operation	(Opcode = 111, $I = 1$)

(c) Input – output instruction

Determine the Type of Instruction

Instruction Cycle

Decoder output D7 is equal to 1 if the operation code is equal to binary 111.



15	14	12	11	0
1	Opcode		Address	

15			12	11 0
1	1	1	1	I/0 operation

15			12	11 0
0	1	1	1	Register operation

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

$D_7' I' T_3$: Nothing

$D_7 I' T_3$: Execute a register-reference instruction

$D_7 I T_3$: Execute an input–output instruction

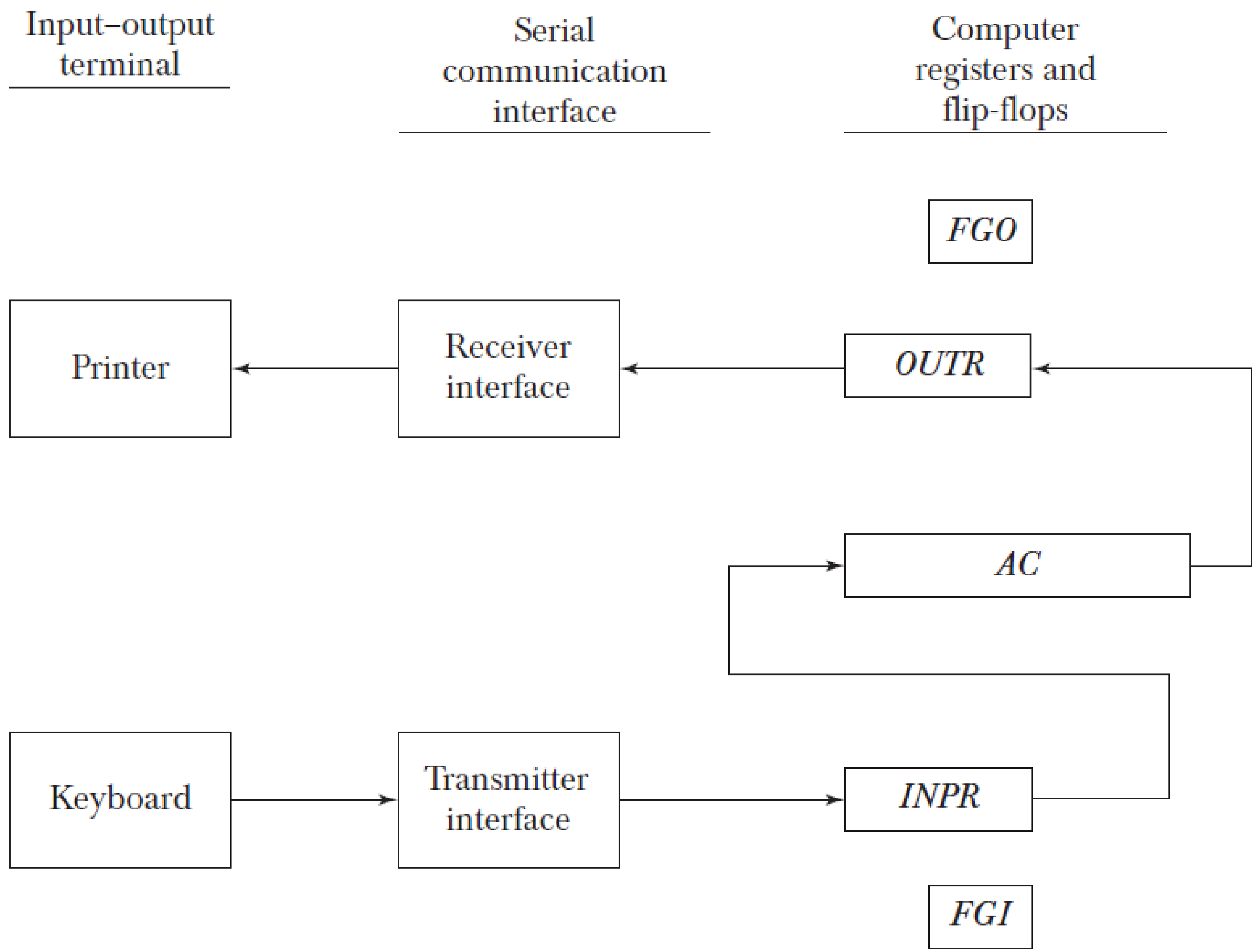
Register-Reference Instructions

Execution of Register-Reference Instructions

$D_7 I' T_3 = r$ (common to all registcr-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:	$AC \leftarrow \text{shr } AC, AC(15) \leftarrow E, E \leftarrow AC(0)$	Circulate right
CIL	rB_6:	$AC \leftarrow \text{shl } AC, AC(0) \leftarrow E, E \leftarrow AC(15)$	Circulate left
INC	rB_5:	$AC^* \rightarrow AC + 1$	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

Input–Output



Input–Output Instructions

Input–Output Instructions

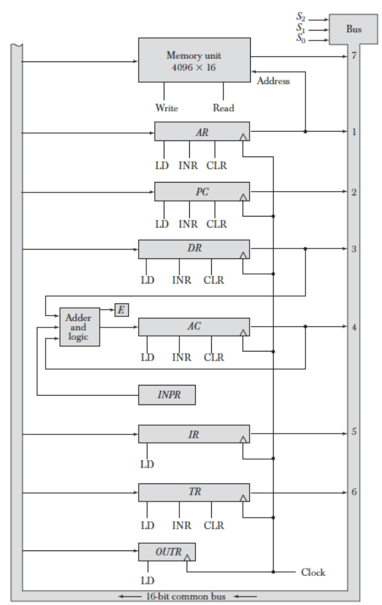
$D_7IT_3 = p$ (common to all input–output instructions)
$IR(i) = B_i$ [bit in $IR(6\text{–}11)$ that specifies the instruction]

	p:	$SC \leftarrow 0$	Clear SC
INP	pB_{11}:	$AC(0\text{–}7) \leftarrow INPR, \quad FGI \leftarrow 0$	Input character
OUT	pB_{10}:	$OUTR \leftarrow AC(0\text{–}7), \quad FGO \leftarrow 0$	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

Memory-Reference Instructions

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 $PC \leftarrow PC + 1$



Memory-Reference Instructions

AND to *AC*

$$D_0T_4: \quad DR \leftarrow M[AR]$$

$$D_0T_5: \quad AC \leftarrow AC \wedge DR, \quad SC \leftarrow 0$$

ADD to *AC*

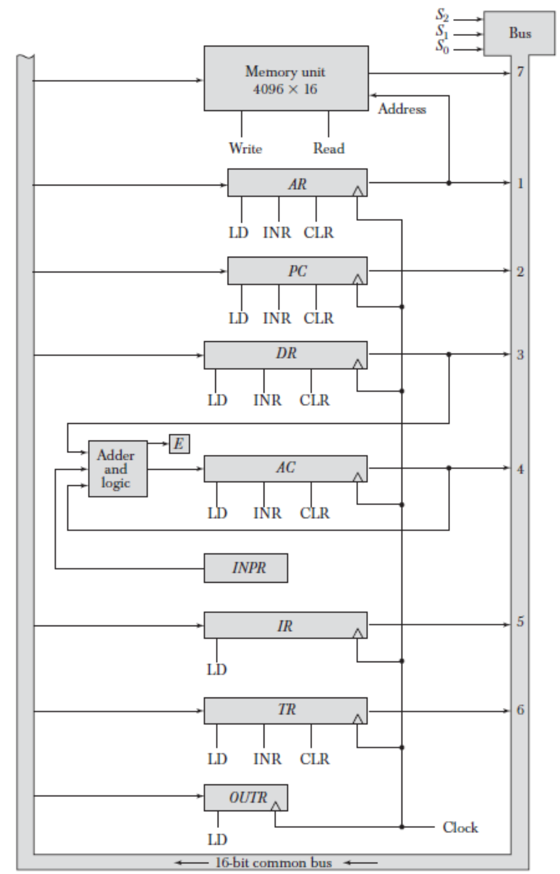
$$D_1T_4: \quad DR \leftarrow M[AR]$$

$$D_1T_5: \quad AC \leftarrow AC + DR, \quad E \leftarrow C_{out}, \quad SC \leftarrow 0$$

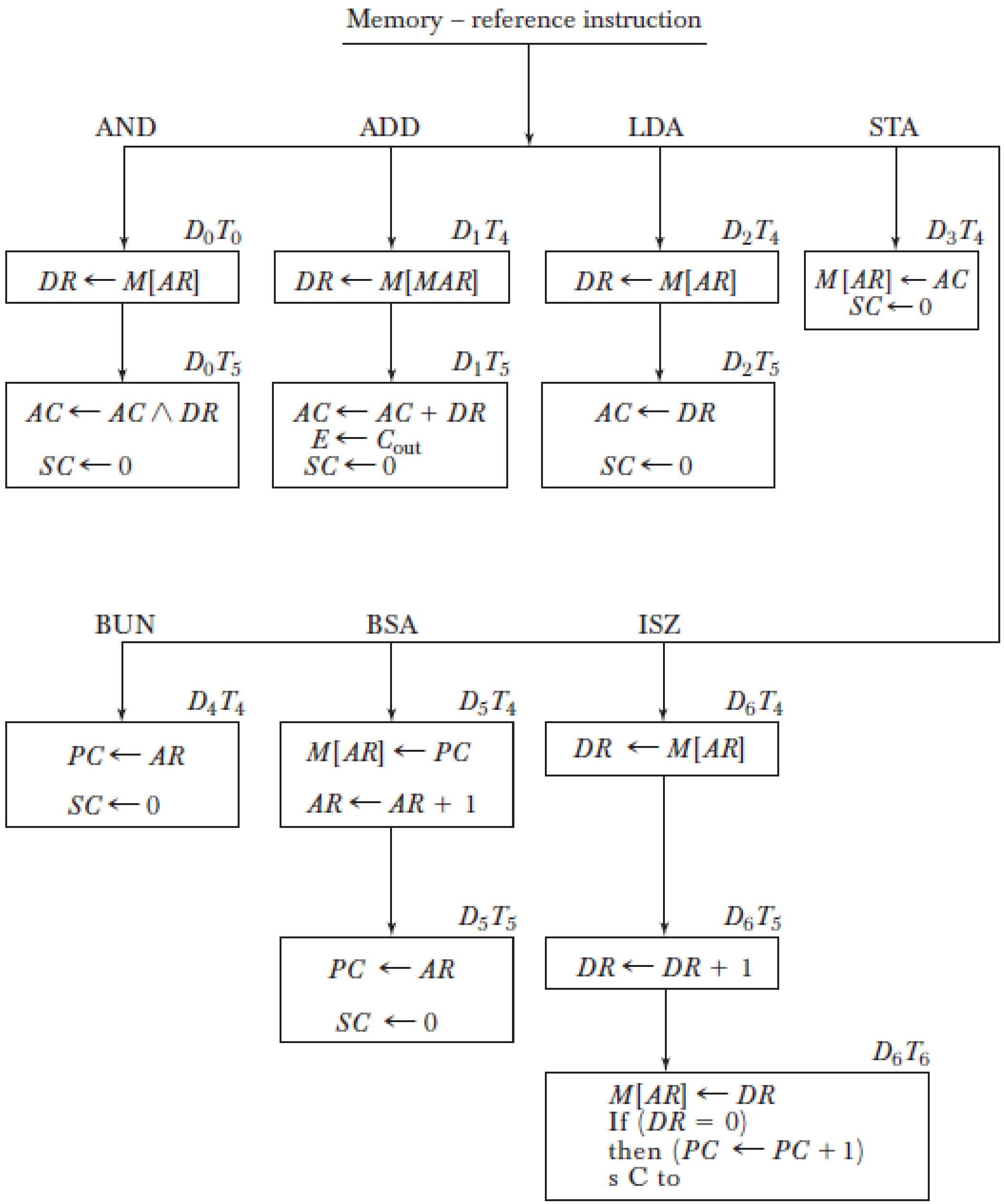
LDA: Load to AC

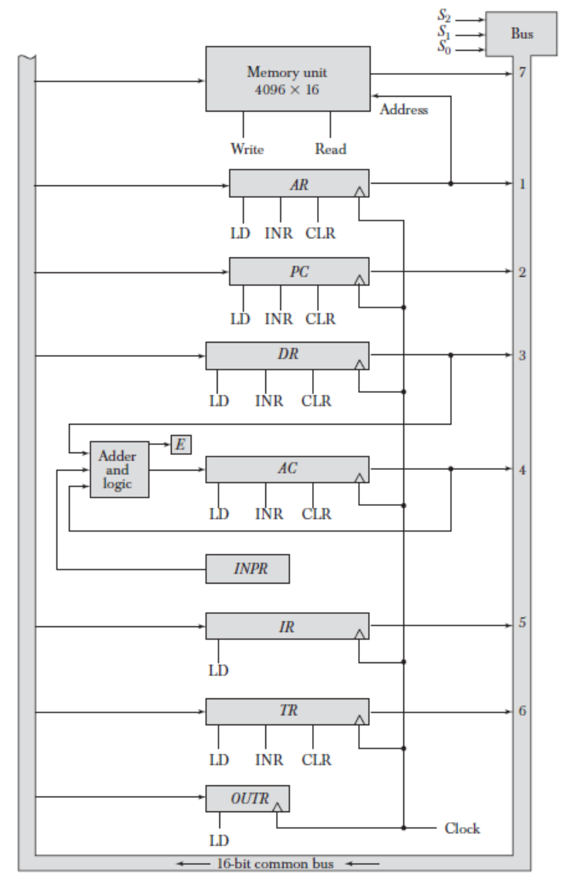
$$D_2T_4: \quad DR \leftarrow M[AR]$$

$$D_2T_5: \quad AC \leftarrow DR, \quad SC \leftarrow 0$$



Memory-Reference Instructions



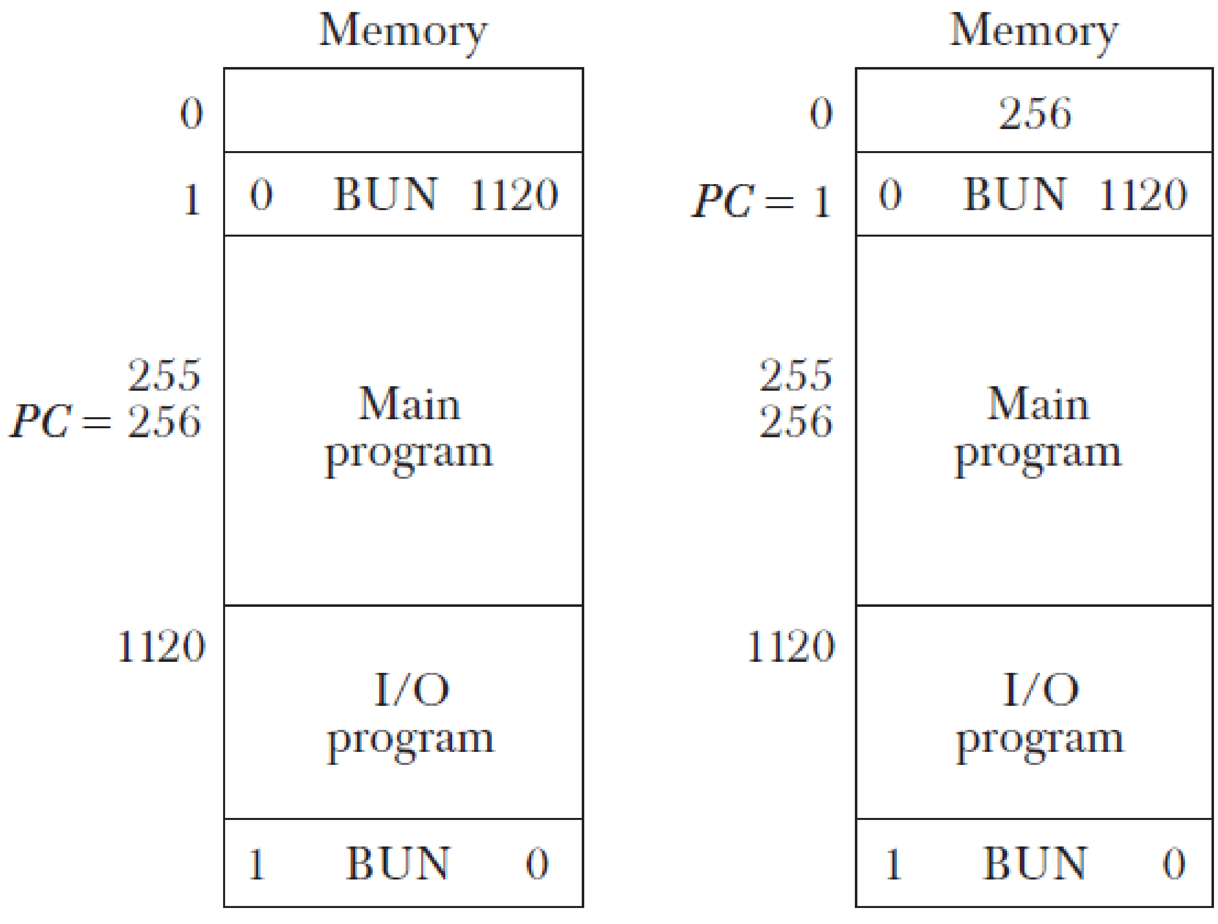


Program Interrupt

- Programmed control transfer
  - Computer keeps checking the flag bit, and when it finds it set, it initiates an information transfer
  - The difference of information flow rate between the computer and that of the input–output device makes this type of transfer inefficient
  - Consider a computer that can go through an instruction cycle in 1μs
  - Assume that the input–output device can transfer information at a maximum rate of 10 characters per second. This is equivalent to one character every 100,000 μs
    - Two instructions are executed when the computer checks the flag bit and decides not to transfer the information
    - This means that at the maximum rate, the computer will check the flag 50,000 times between each transfer. The computer is wasting time while checking the flag instead of doing some other useful processing task.

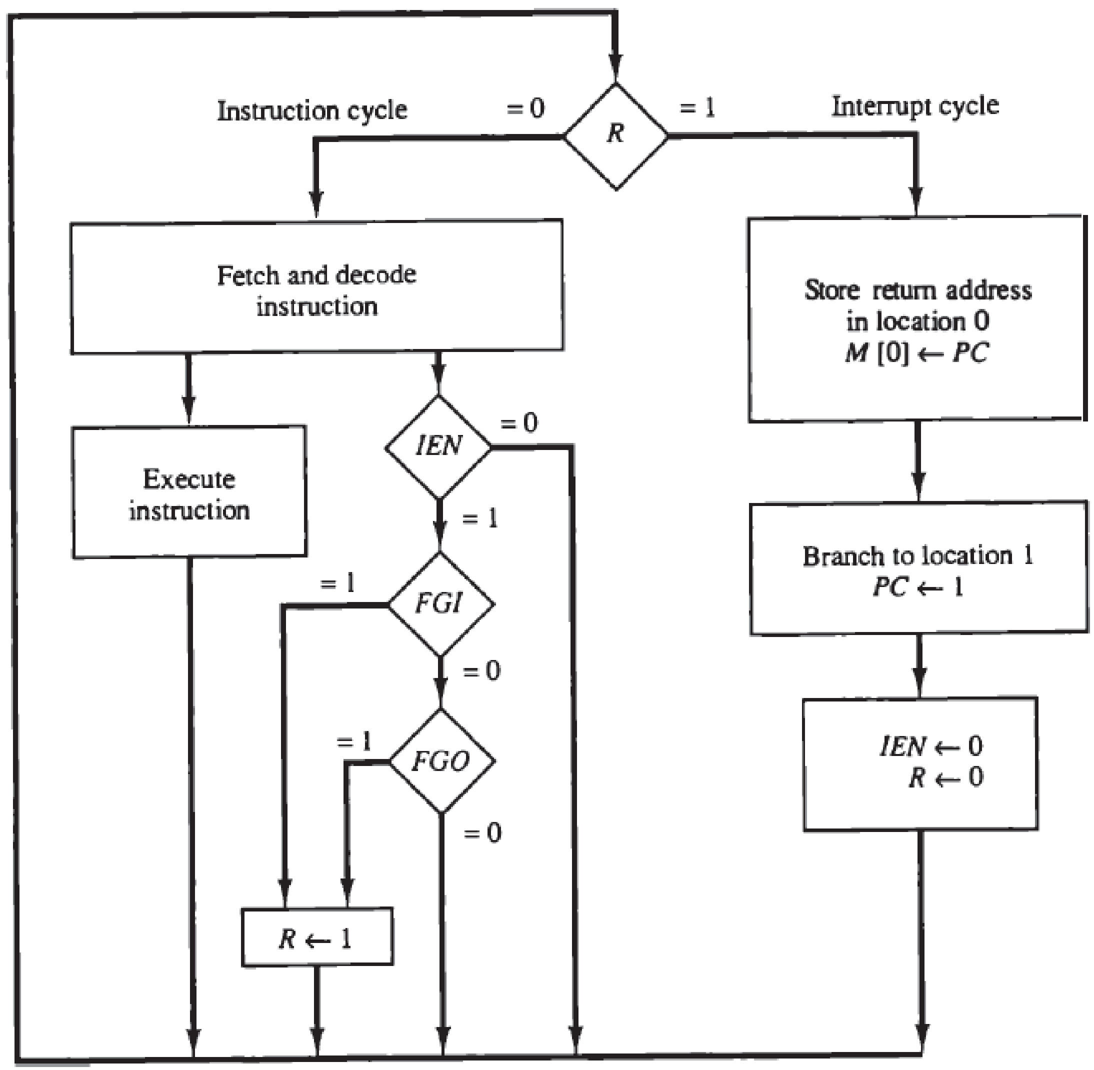
Demonstration of the interrupt cycle

The interrupt cycle is a hardware implementation of a branch and save return address operation



(a) Before interrupt (b) After interrupt cycle

Interrupt Cycle



Interrupt Cycle

The interrupt cycle is initiated after the last execute phase if the interrupt flip-flop R is equal to 1. This flip-flop is set to 1 if $IEN = 1$ and either FGI or FGO are equal to 1. This can happen with any clock transition except when timing signals T_0, T_1, or T_2 are active. The condition for setting flip-flop R to 1 can be expressed with the following register transfer statement:

$$T_0'T_1'T_2'(IEN)(FGI + FGO): \quad R \leftarrow 1$$

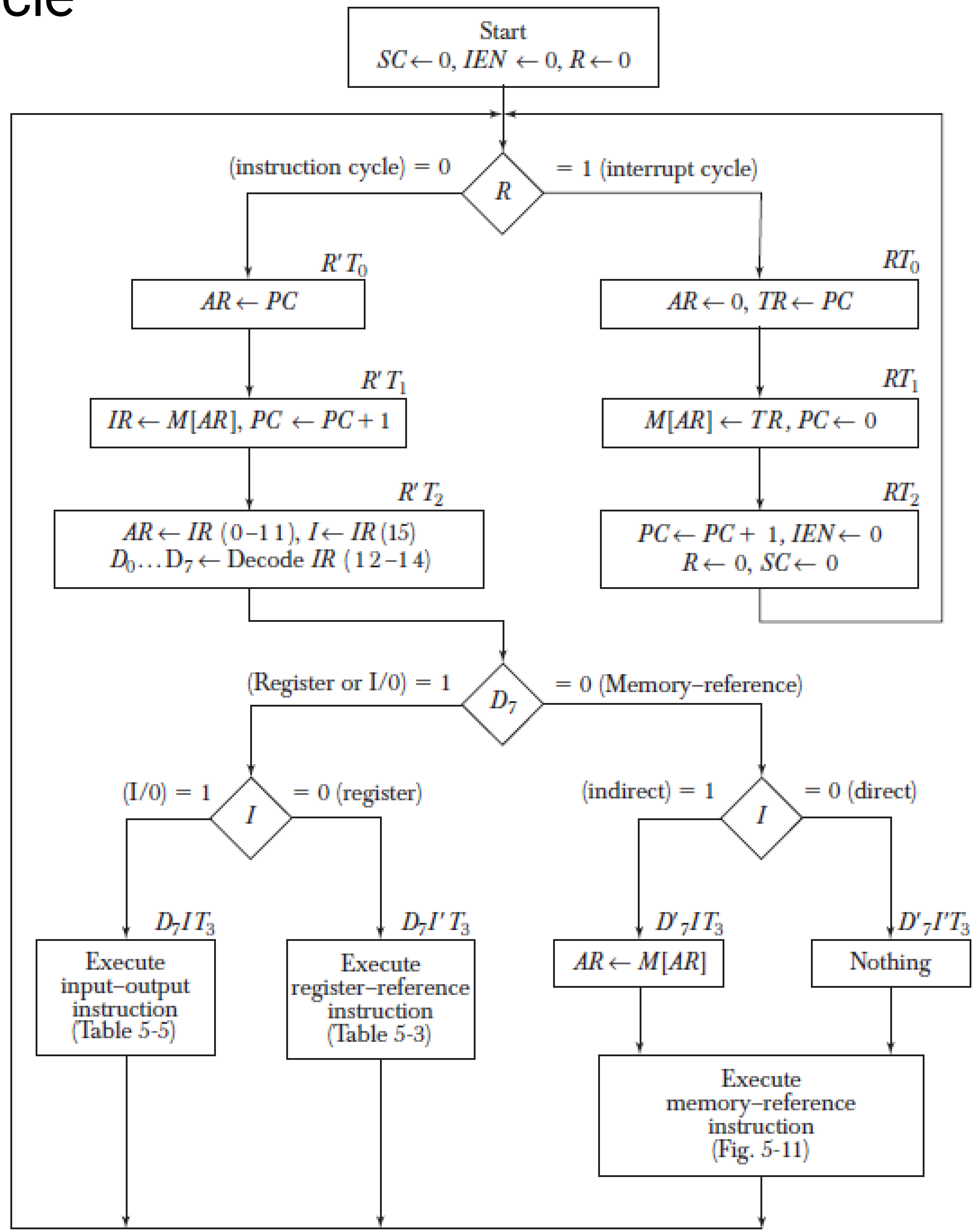
Instead of using only timing signals T_0, T_1, and T_2 (as shown in Fig. 5-9) we will AND the three timing signals with R' so that the fetch and decode phases will be recognized from the three control functions $R'T_0$, $R'T_1$, and $R'T_2$. The reason for this is that after the instruction is executed and SC is cleared to 0, the control will go through a fetch phase only if $R = 0$. Otherwise, if $R = 1$, the control will go through an interrupt cycle. The interrupt cycle stores the return address (available in PC) into memory location 0, branches to memory location 1, and clears IEN, R, and SC to 0. This can be done with the following sequence of microoperations:

$$RT_0: \quad AR \leftarrow 0, \quad TR \leftarrow PC$$

$$RT_1: \quad M[AR] \leftarrow TR, \quad PC \leftarrow 0$$

$$RT_2: \quad PC \leftarrow PC + 1, \quad IEN \leftarrow 0, \quad R \leftarrow 0, \quad SC \leftarrow 0$$

Instruction Cycle



پایان

موفق و پیروز باشید