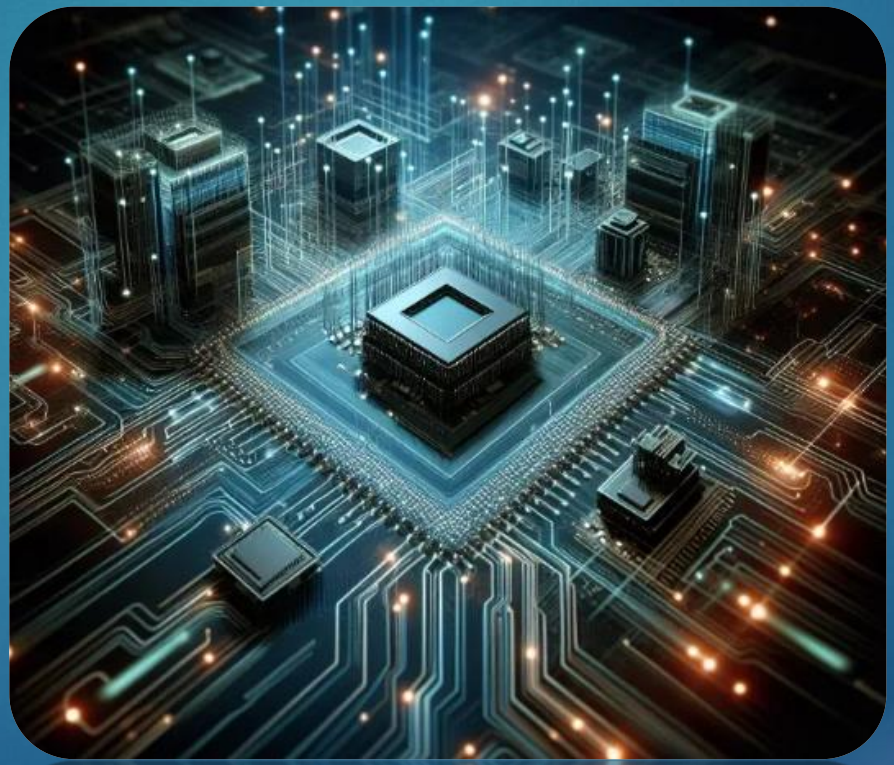
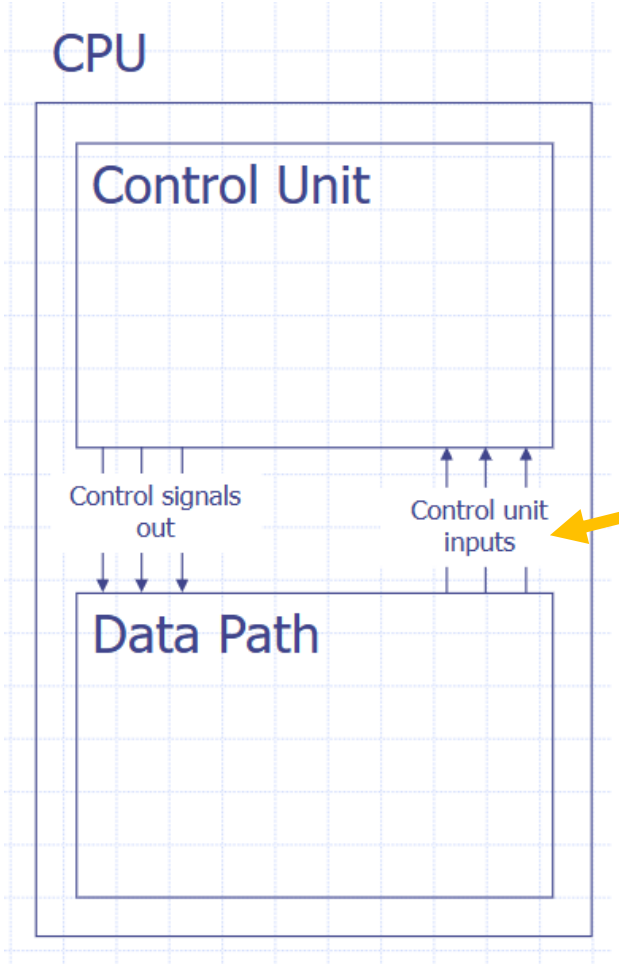
In the name of God

Computer Architecture

S.Ali.Zendehbad



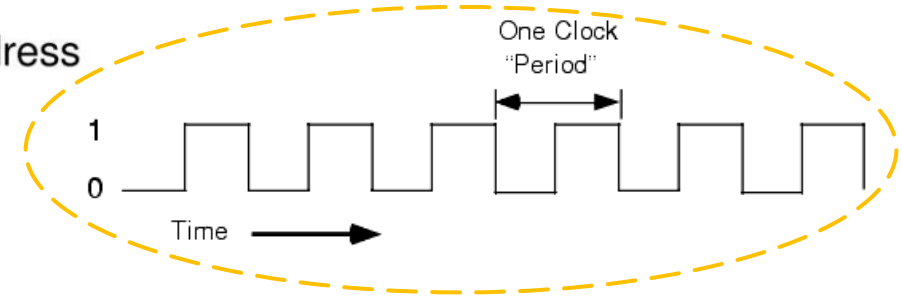
Processor Design



RTL

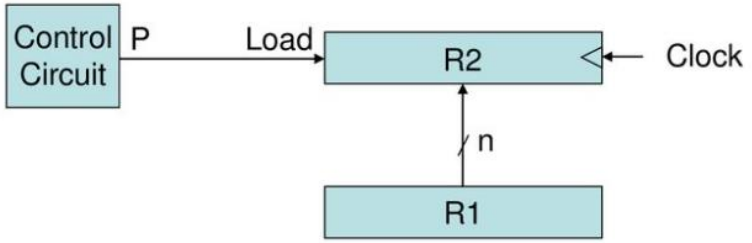
RTL

- R1: processor register
- MAR: Memory Address Register (holds an address for a memory unit)
- PC: Program Counter
- IR: Instruction Register
- SR: Status Register

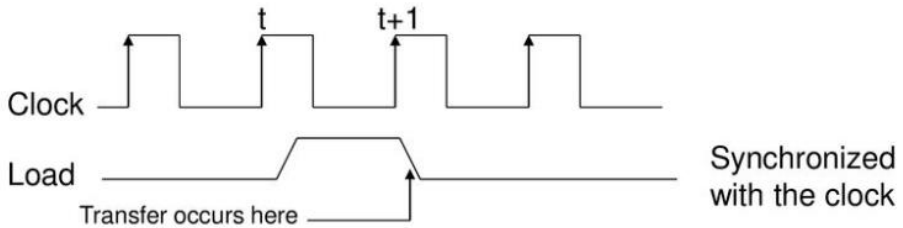


Hardware implementation of a controlled transfer: P: R2 ← R1

Block diagram:

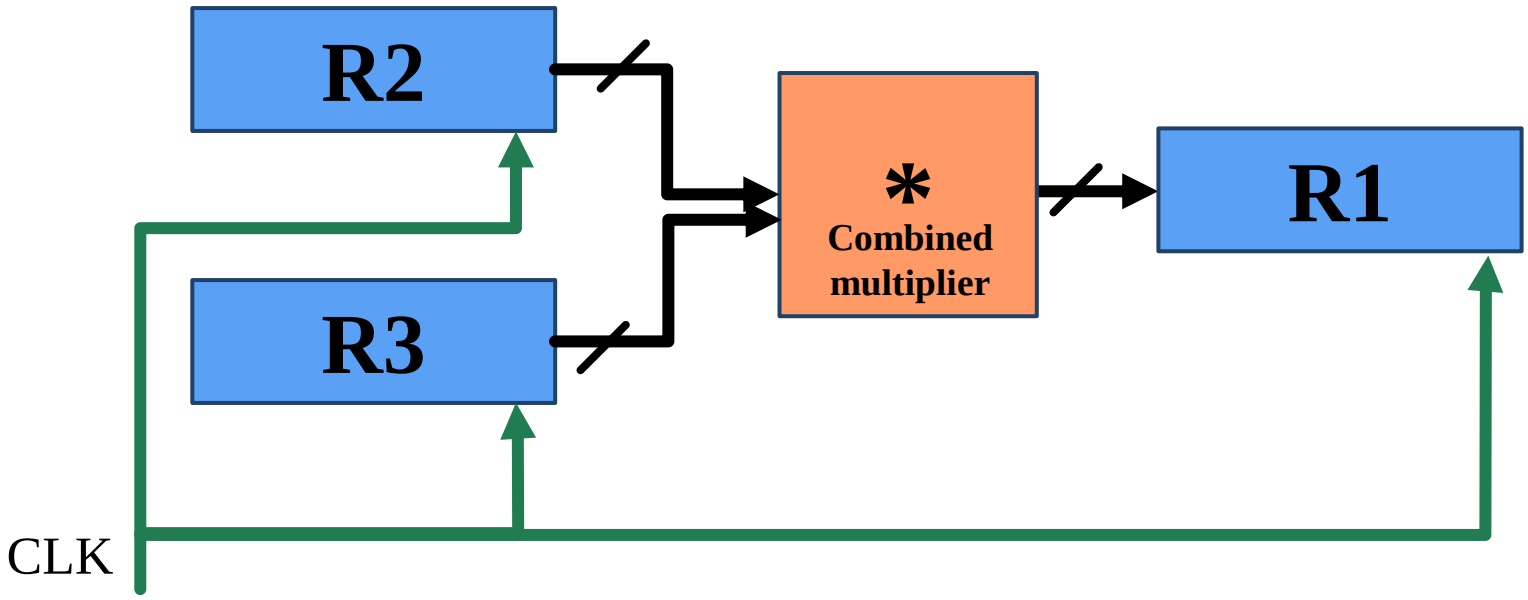


Timing diagram

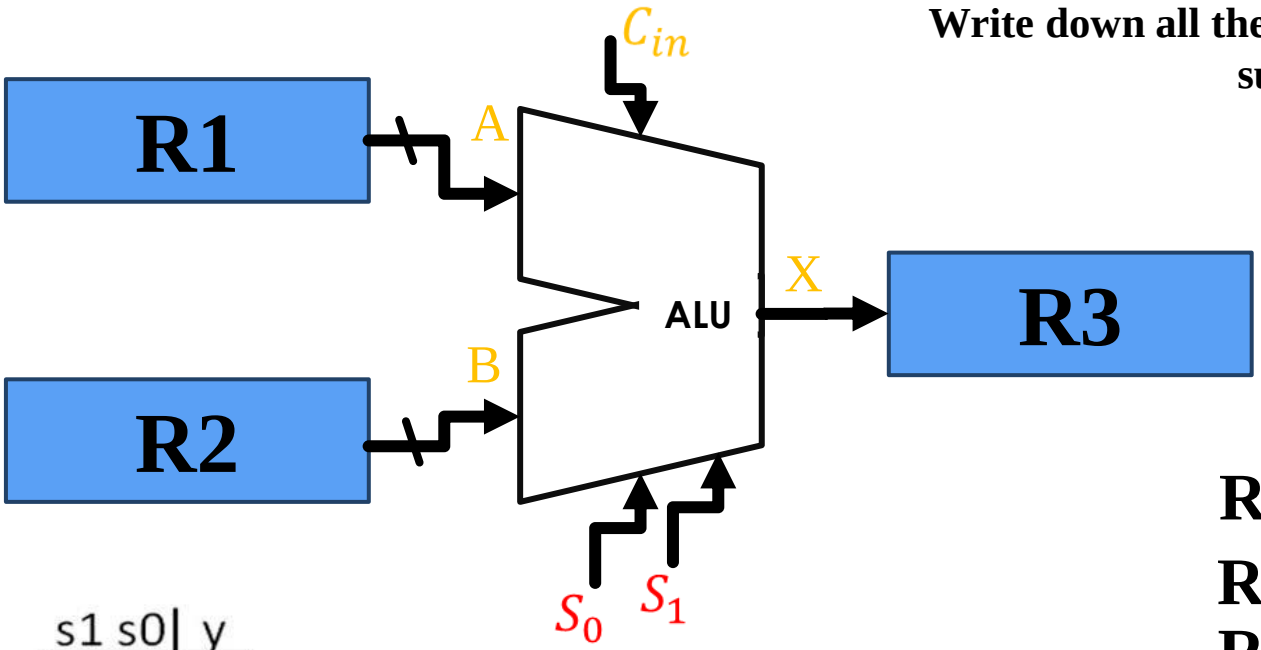


MOP

R1←R2*R3



Example



s1	s0	y	
0	0	D0	A
0	1	D1	B
1	0	D2	A+B
1	1	D3	A+B+ C_{in}

Write down all the MOP that this hardware supports?

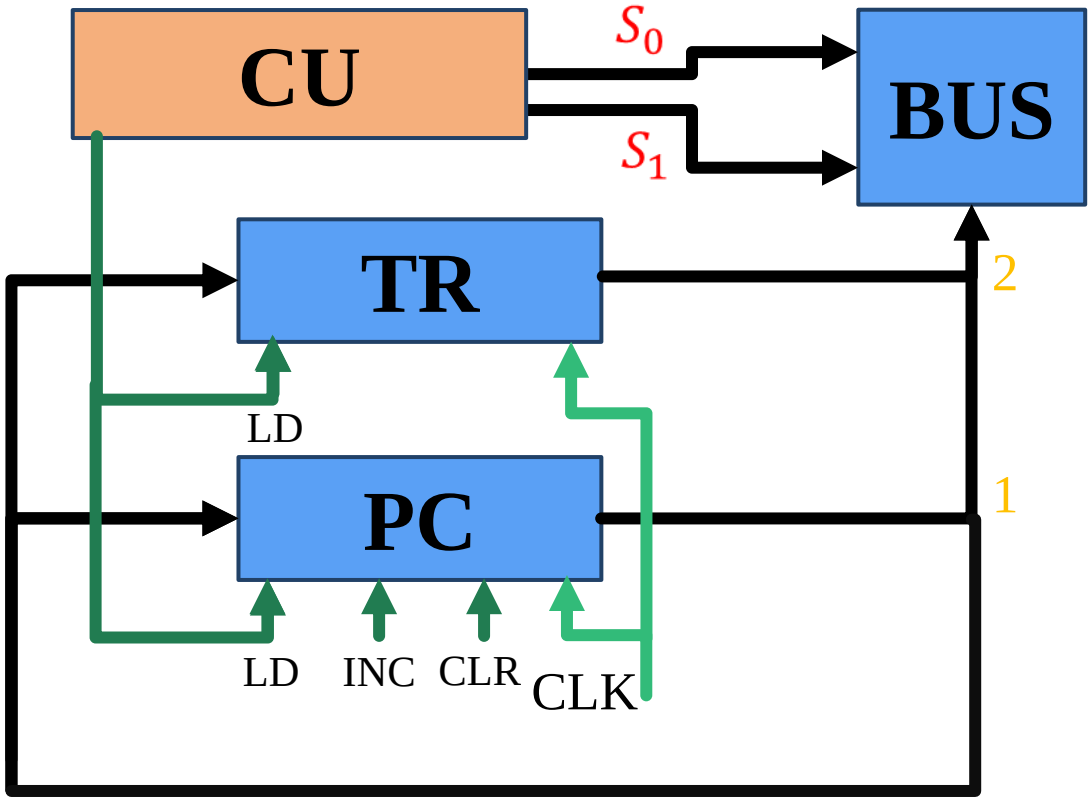
R3←R1

R3←R2

R3←R1+R2

R3←R1+R2+ C_{in}

Example



PC←TR

TR←PC

PC←PC+1

PC←0

TR←PC+1

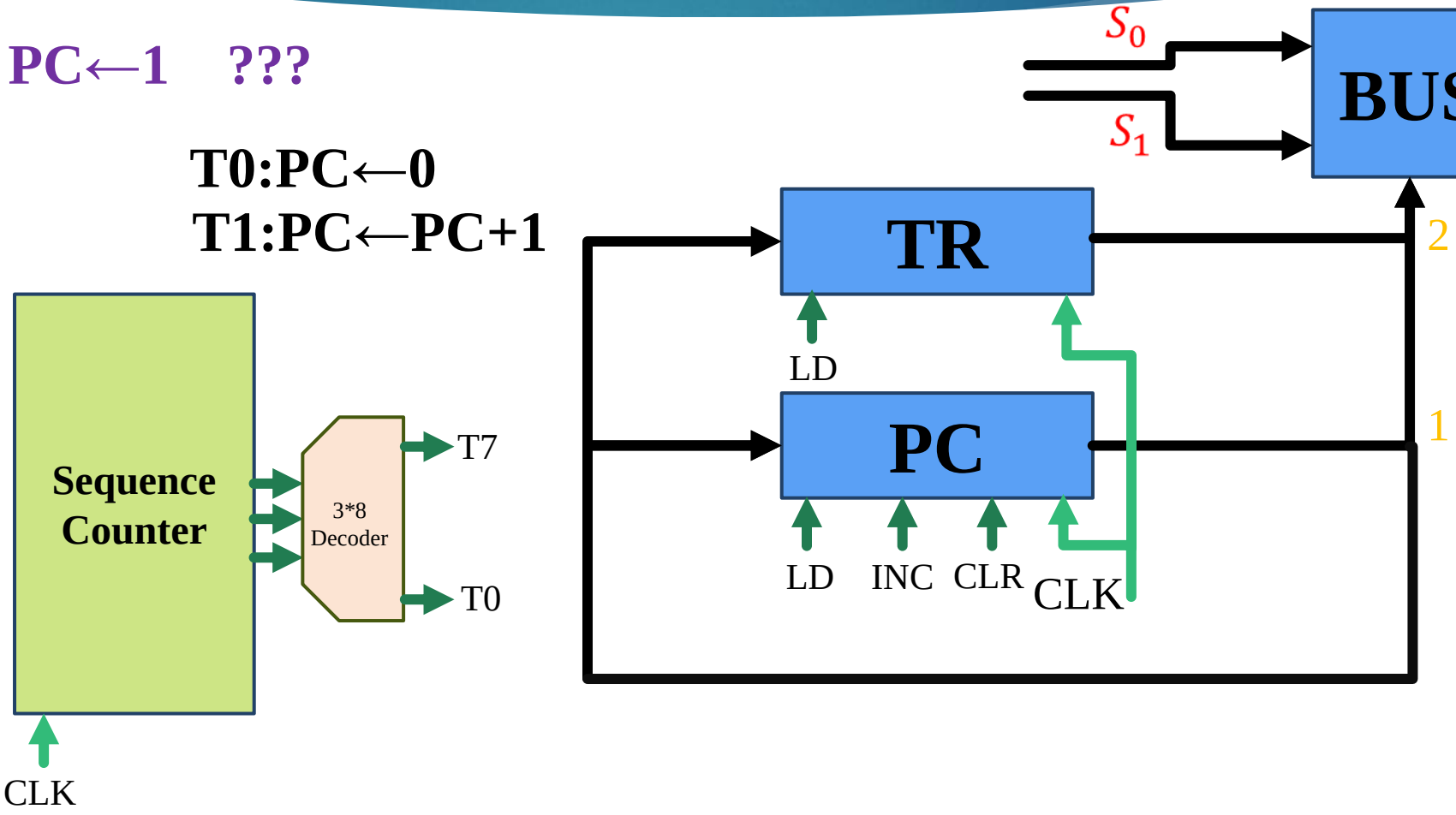
PC←1 ???

	s1	s0	y
2	1	0	BUS←TR
1	0	1	BUS←PC

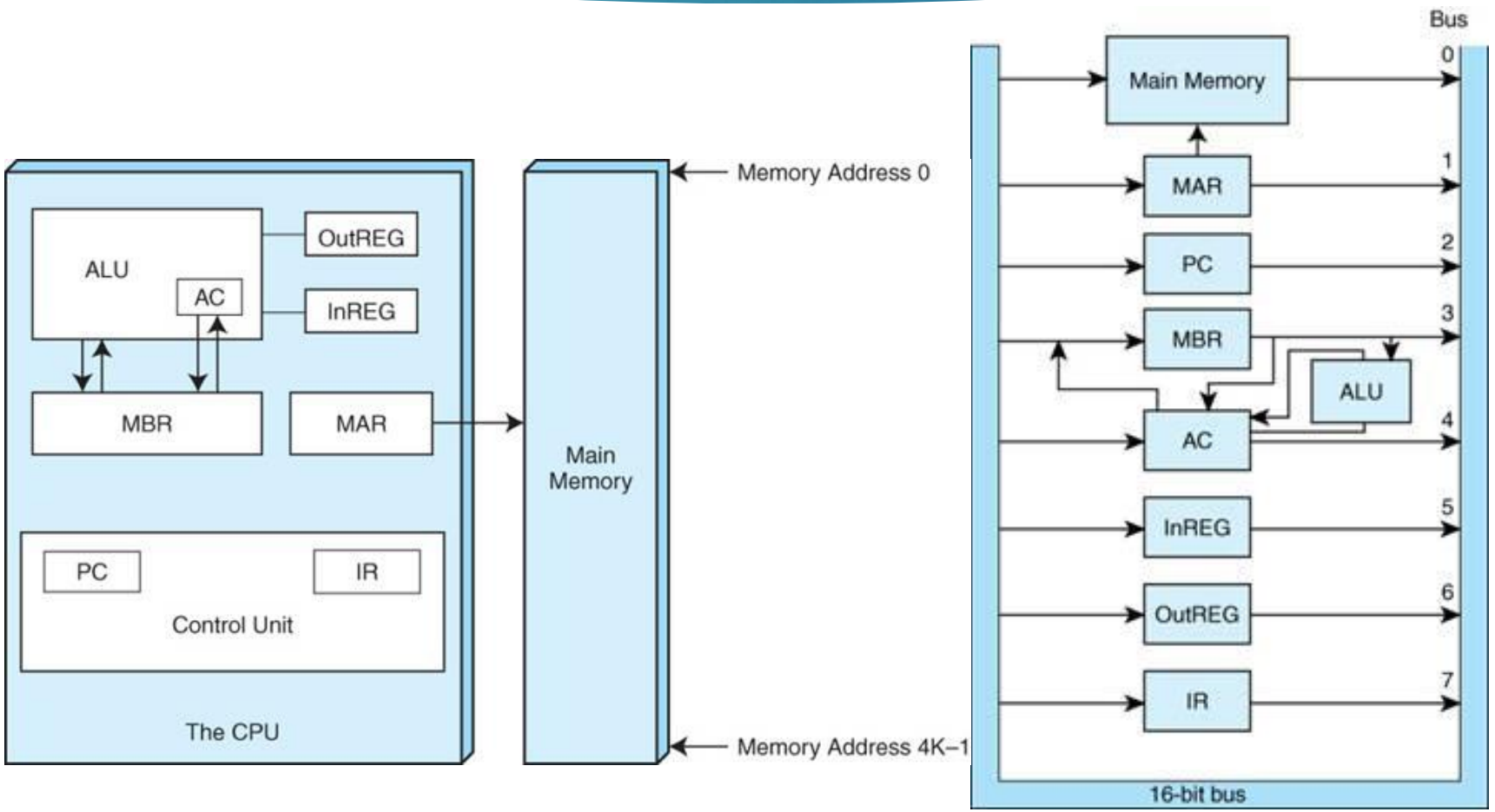
Example

PC←1 ???

T0:PC←0
T1:PC←PC+1



Processor Design

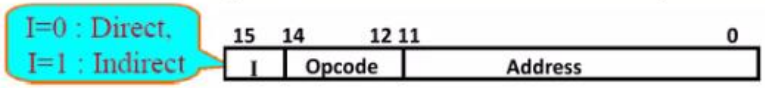


Processor Design

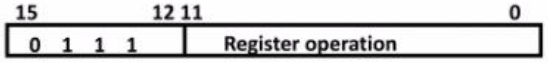
Instruction(16 bit):

- Memory access
- Register access
- I/O management

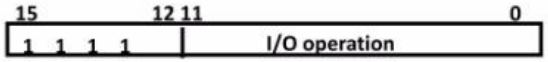
1. Memory-Reference Instructions **(OP-code = 000 ~ 110)**



2. Register-Reference Instructions **(OP-code = 111, I = 0)**

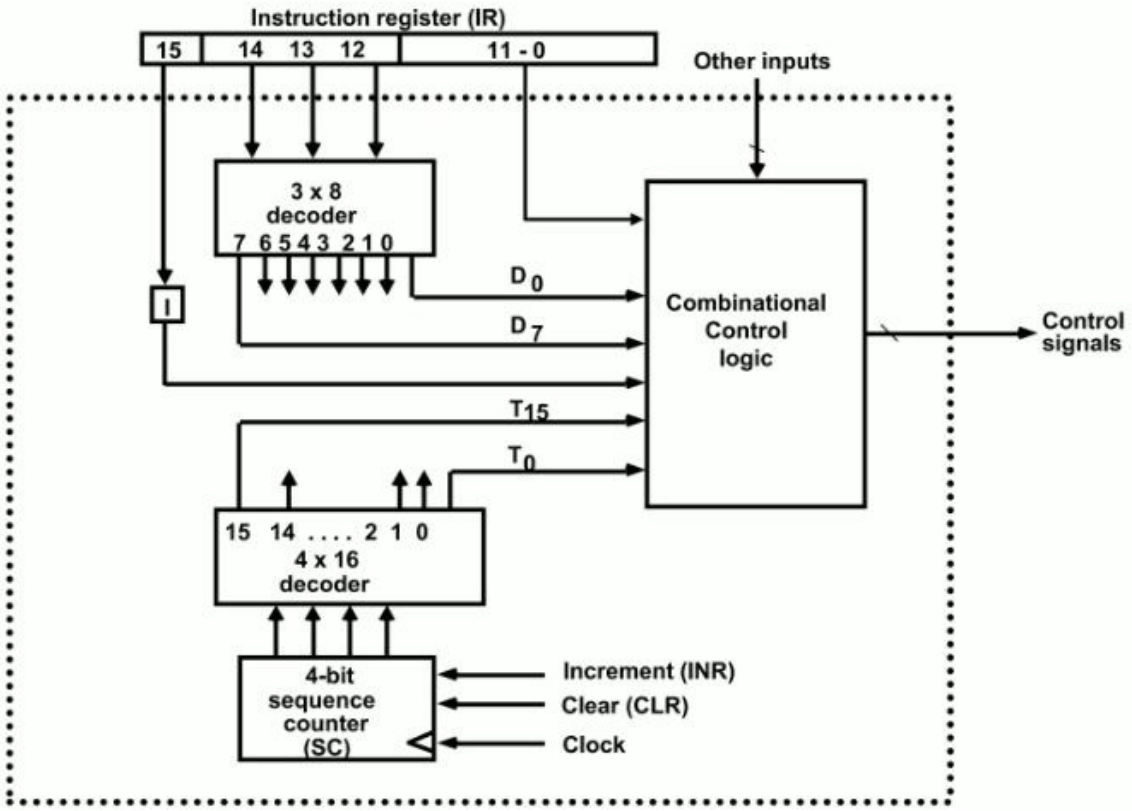


3. Input-Output Instructions **(OP-code =111, I = 1)**



Processor Design

Control unit of Basic Computer



Control Unit

- Control unit (CU) of a processor translates from machine instructions to the control signals for the microoperations that implement them

- Control units are implemented in one of two ways

  Hardwired Control

  CU is made up of sequential and combinational circuits to generate the control signals

  Advantage : optimized to provide fast mode of operations

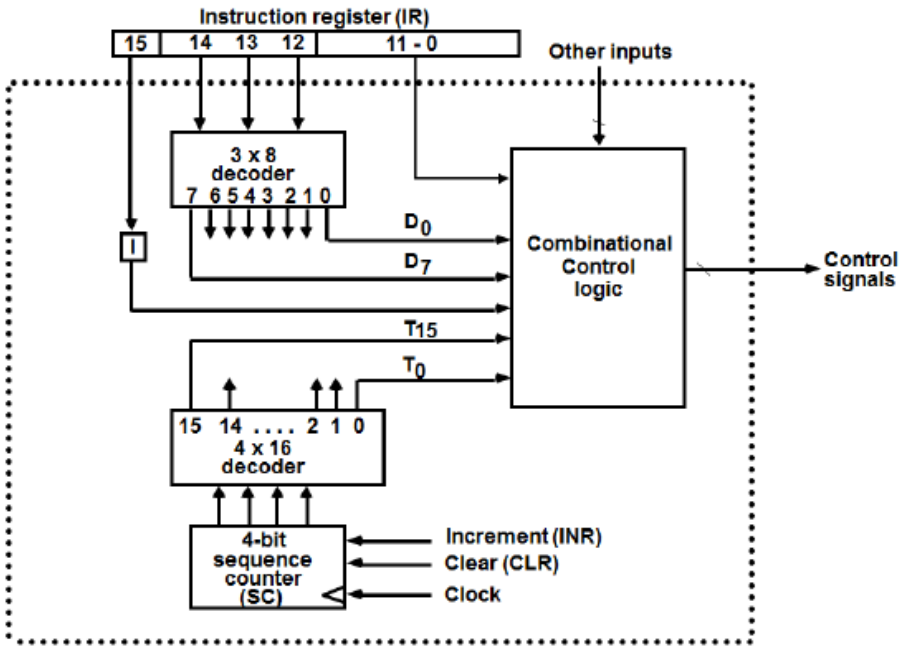
  Disadvantage : requires changes in wiring if design has been modified

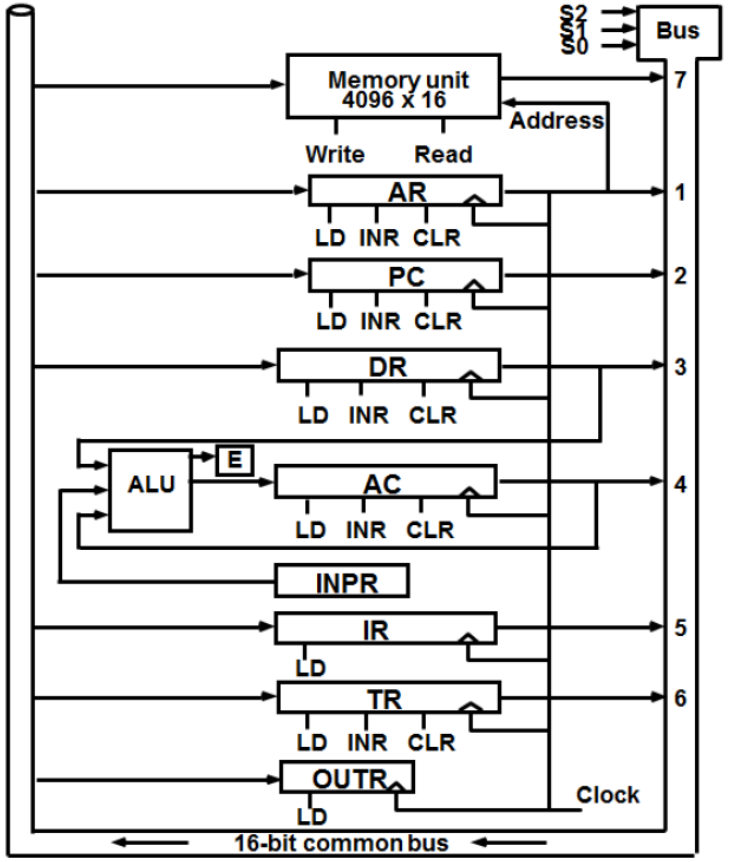
  Microprogrammed Control (ROM)

  A control memory on the processor contains microprograms that activate the necessary control signals

- We will consider a hardwired implementation of the control unit for the Basic Computer

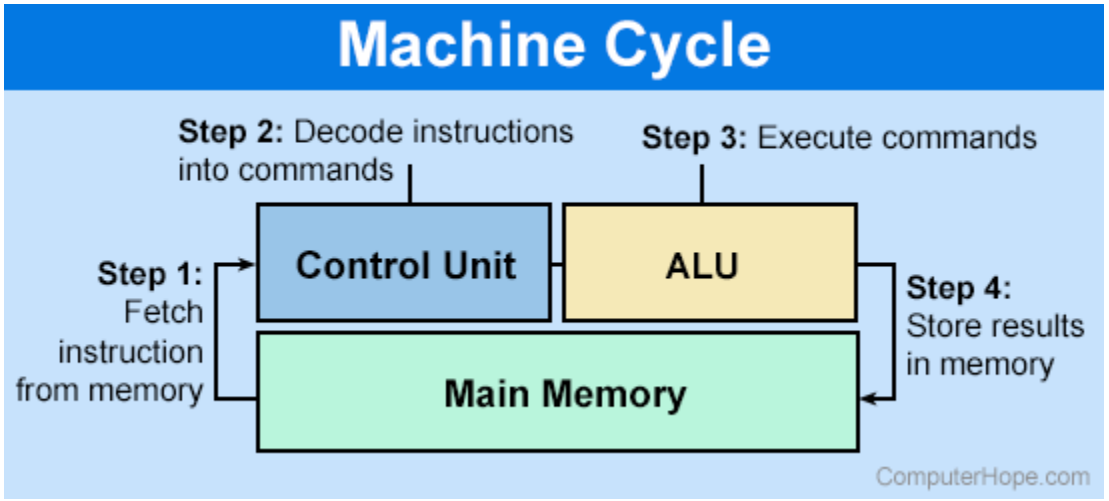
Processor Design





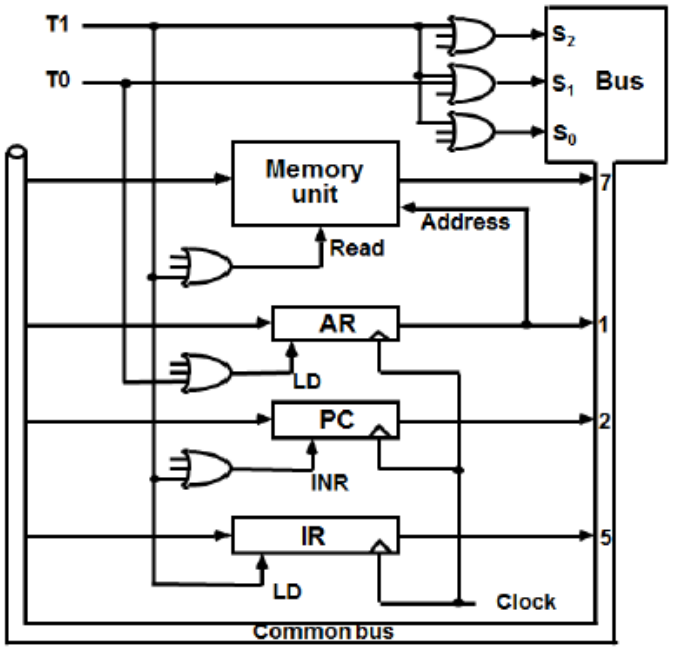
Instruction Cycle

- In Basic Computer, a machine instruction is executed in the following 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.
  - 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

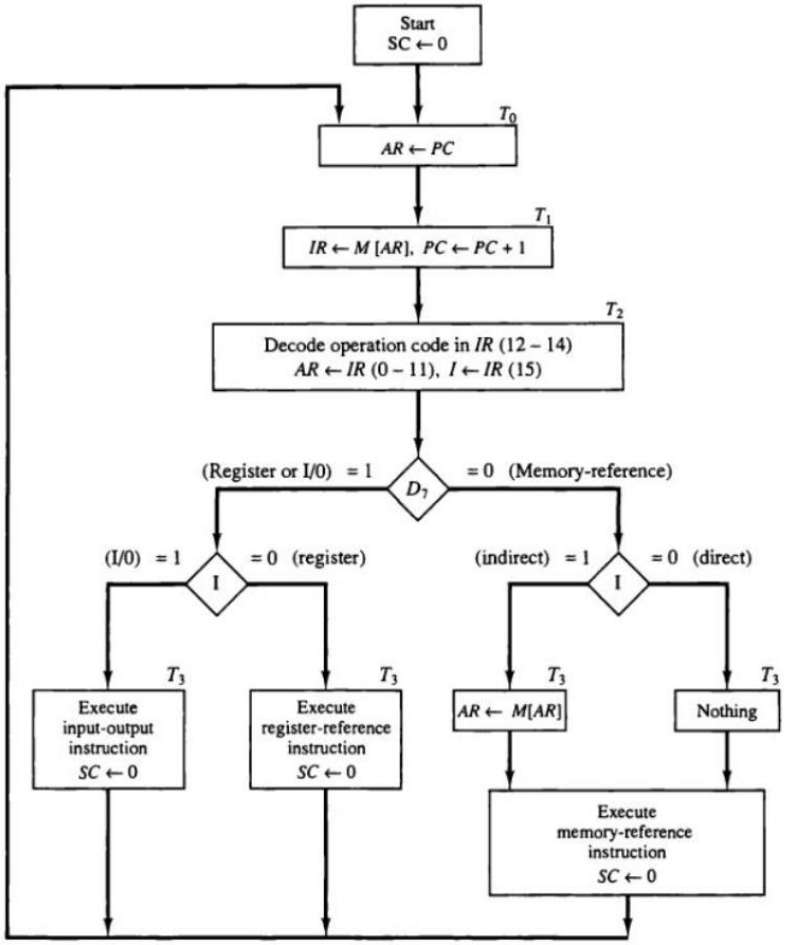


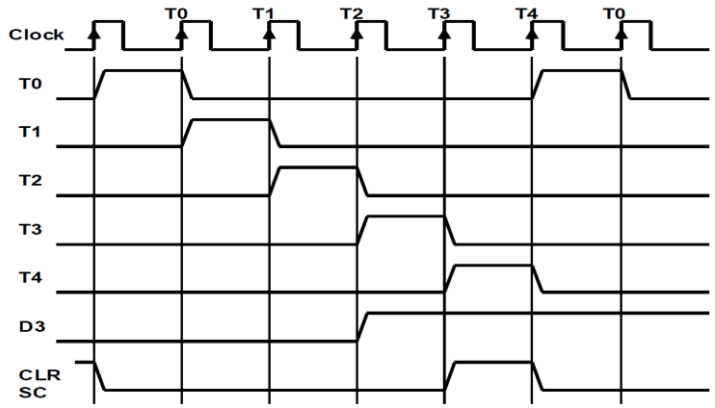
Fetching & Decode

T_0: $AR \leftarrow PC$
T_1: $IR \leftarrow M[AR]$, $PC \leftarrow PC + 1$
T_2: $D_0, \ldots, D_7 \leftarrow$ Decode $IR(12\text{–}14)$, $AR \leftarrow IR(0\text{–}11)$, $I \leftarrow IR(15)$



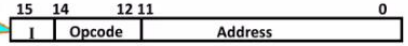
Instruction Cycle



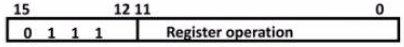


1. Memory-Reference Instructions **(OP-code = 000 ~ 110)**

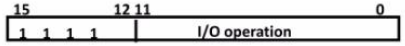
I=0 : Direct,
I=1 : Indirect



2. Register-Reference Instructions **(OP-code = 111, I = 0)**



3. Input-Output Instructions **(OP-code =111, I = 1)**



Register Reference Instructions

r = D_7 I'T_3 => Register Reference Instruction
B_i = IR(i) , i=0,1,2,...,11

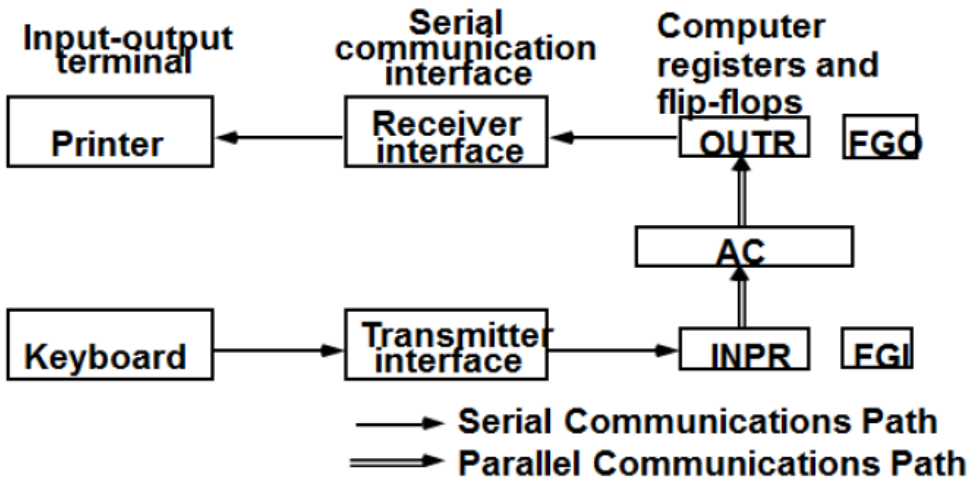
$D_7I'T_3 = r$ (common to all register-reference instructions)
$IR(i) = B_i$ [bit in IR(0–11) that specifies the operation]

		r:	$SC \leftarrow 0$	Clear SC
D7I'T3IR(11)	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

I/O 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



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 $PC \leftarrow PC + 1$

contact us

E-mail :
Ali.zendebad@gmail.com

Homepage:
Sazendehbad.ir