K. Geetha-COCA-U3-L1.1

SASTRA DEEMED TO BE UNIVERSITY	**COMPUTER ORGANIZATION AND ARCHITECTURE** **Course Code: Semester : III / CSBS** **Dr. K. Geetha/SAP/CSE/SOC/SASTRA**

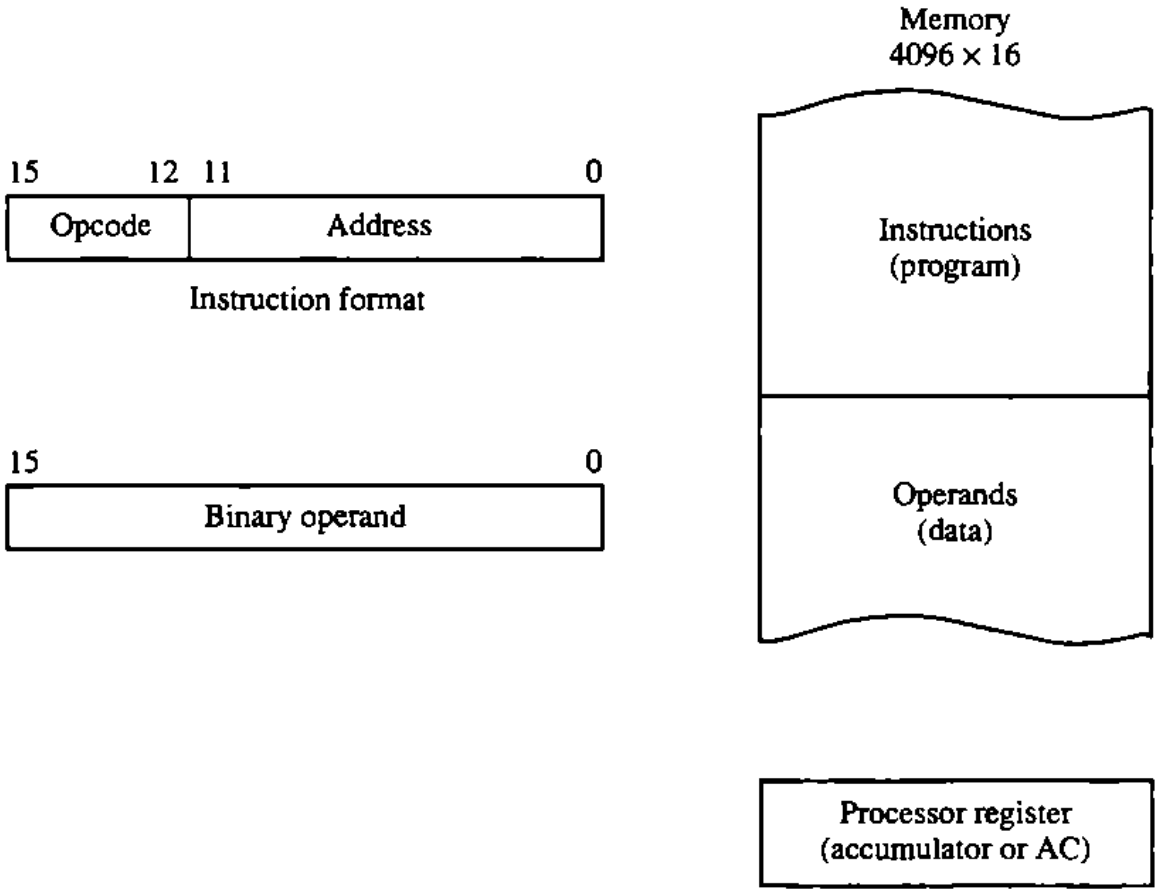
Instruction Code

Program - Set of Instructions that specify the operations, operands and the sequence of execution
Instruction - Binary code that specifies sequence of micro-operations for the computer
Instruction code and data are stored in memory
Computer reads instruction from memory and stores in control register
Control Unit decodes instructions and issues micro operations.
Each computer has its own Instruction set.

Stored Program Concept- Ability to store and execute instructions

Instruction Code - Group of bits that instruct the computer to perform specific operation.

Figure 5-1 Stored program organization.



All operations are done on data stored in Memory or in Accumulator.
Some instructions may refer operands from accumulator alone
For e.g. Clear AC, Complement AC, Increment AC, Decrement AC.
For such instructions the operand part of an instruction is not requires
And can be used to specify other operations.

Immediate Instruction : If the address part of the instruction is not having an address but the actual operand. Second part Specifies the operand itself rather than the address.

Direct Address: Second part of the instruction gives the address Of the operand.

Indirect Address: Second part of the instruction designates an Address of memory word that contains the address of the operand.

In the instruction if I = 0, Direct address else Indirect address (I =1).

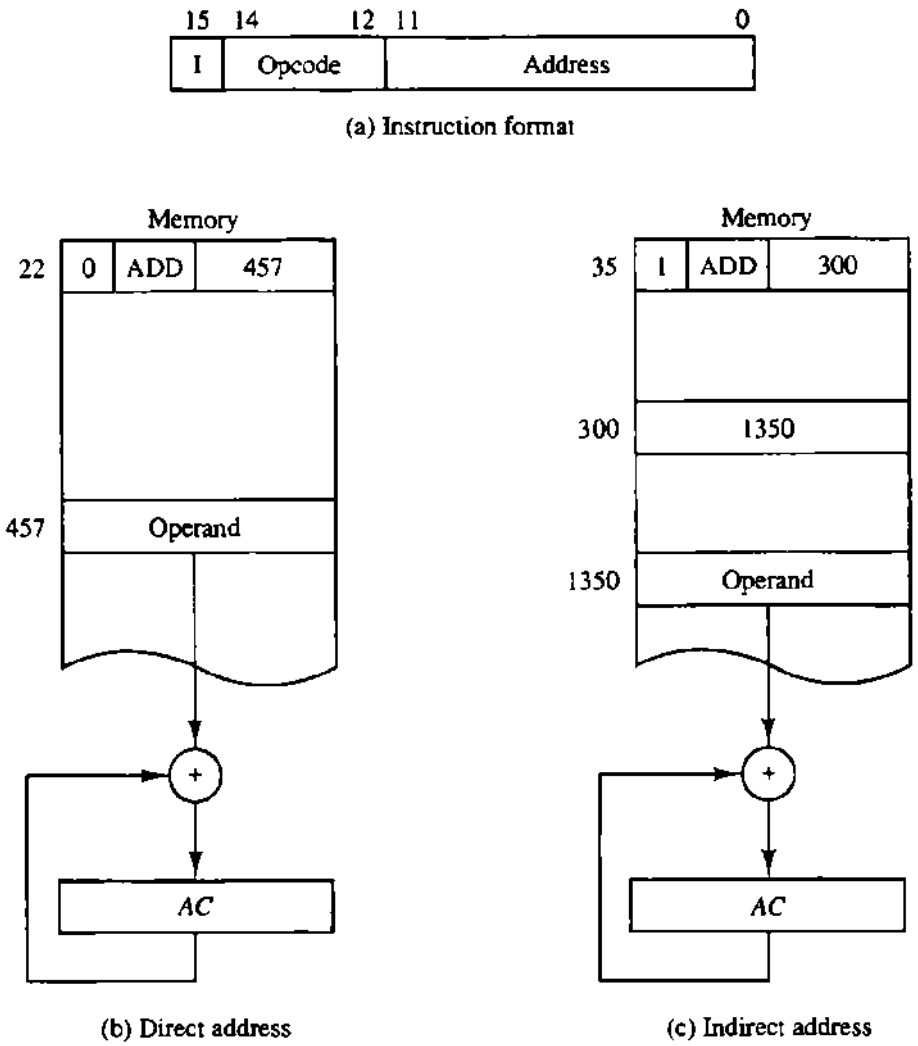


Figure 5-2 Demonstration of direct and indirect address.

Computer Registers

TABLE 5-1 List of Registers for the Basic Computer

Register symbol	Number of bits	Register name	Function
DR	16	Data register	Holds memory operand
AR	12	Address register	Holds address for memory
AC	16	Accumulator	Processor register
IR	16	Instruction register	Holds instruction code
PC	12	Program counter	Holds address of instruction
TR	16	Temporary register	Holds temporary data
INPR	8	Input register	Holds input character
OUTR	8	Output register	Holds output character

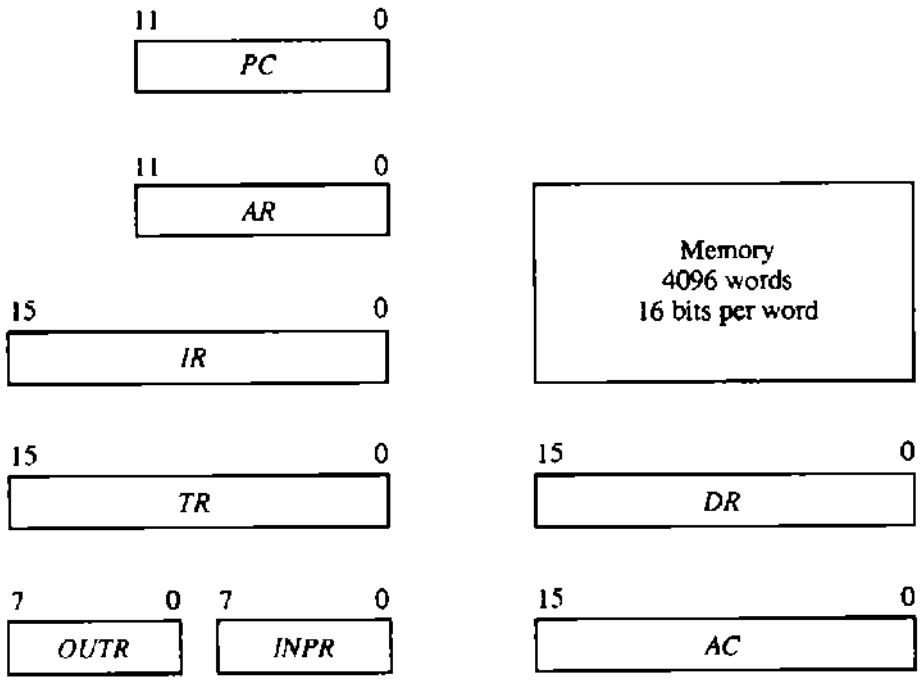


Figure 5-3 Basic computer registers and memory.

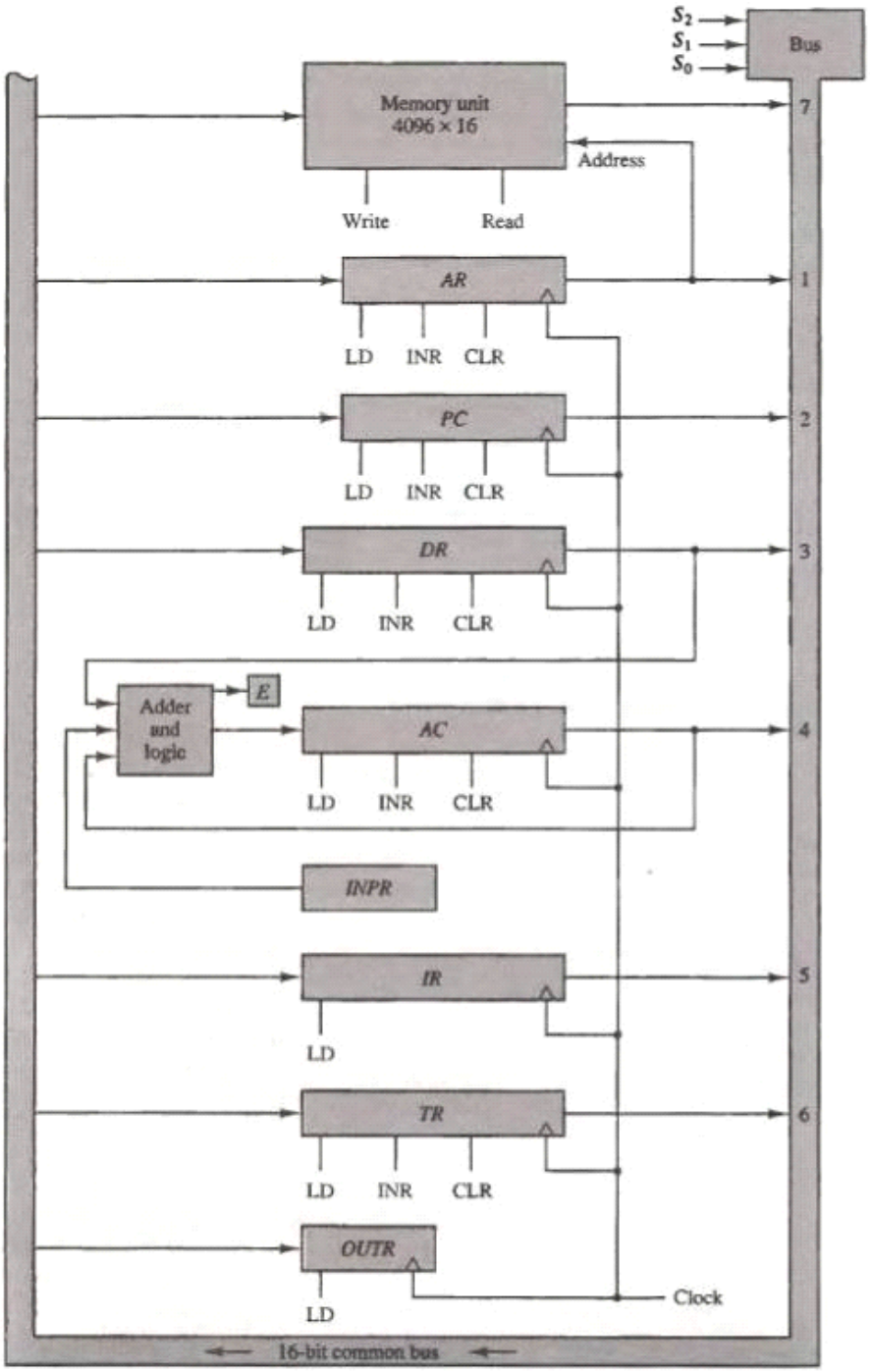


Figure 5-4 Basic computer registers connected to a common bus.

Computer Instructions

Instruction Format

Figure 5-5 Basic computer instruction formats.

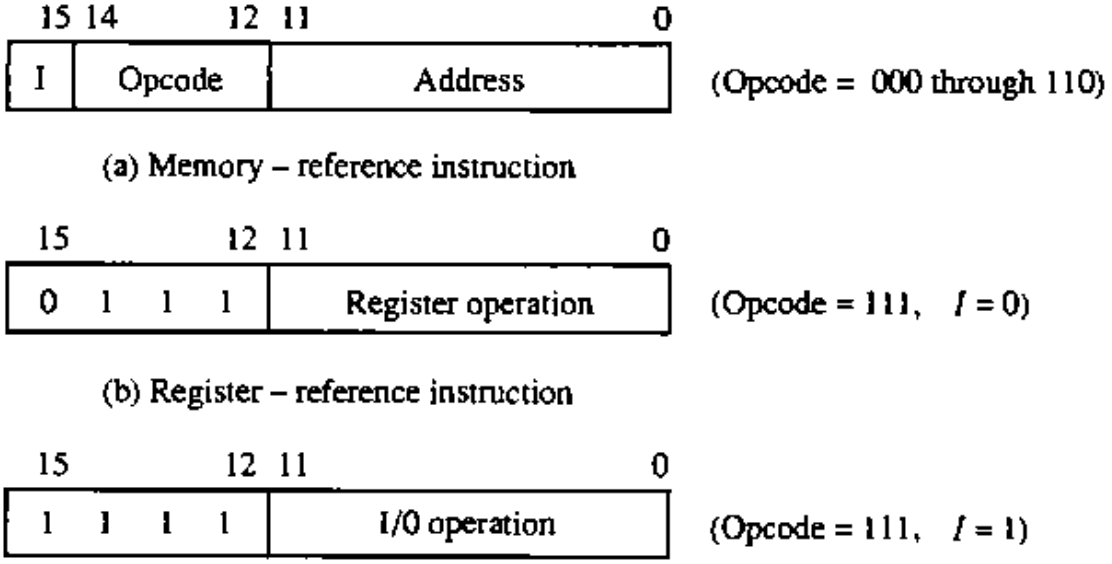


TABLE 5-2 Basic Computer Instructions

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

Instruction set Completeness

The instruction set is said to be complete if it has instructions to perform the following:

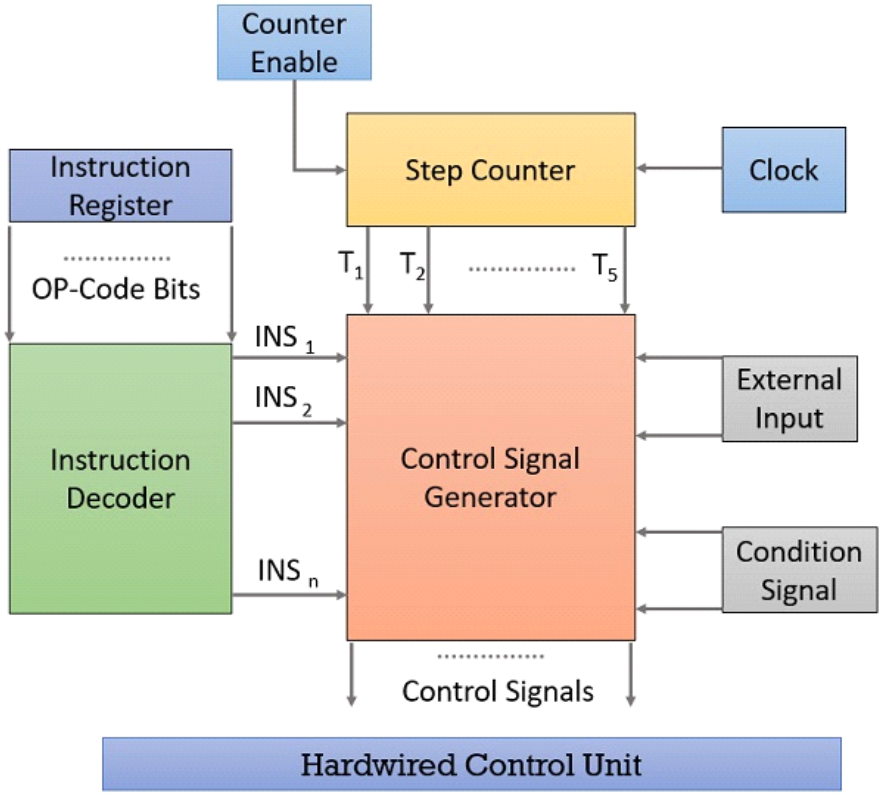
1. Arithmetic, Logical and Shift
2. Data transfer (MOV) between memory and processor registers
3. Program control instructions and status check instructions
4. Input and output instructions

Timing and Control

All registers are controlled by master clock generator

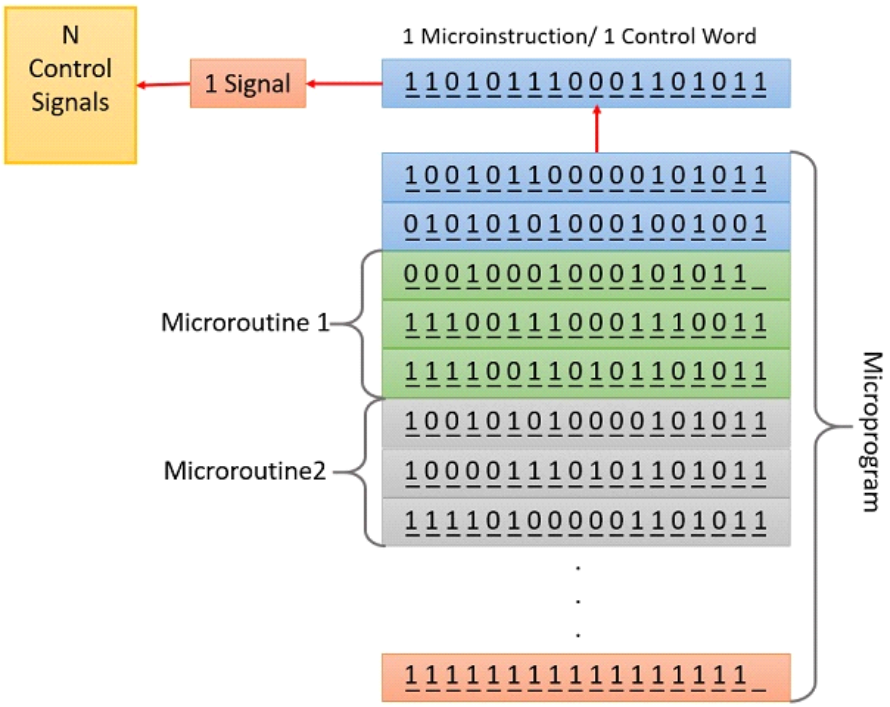
Two types of Control Organization

1. Hardwired Control
2. Microprogram Control

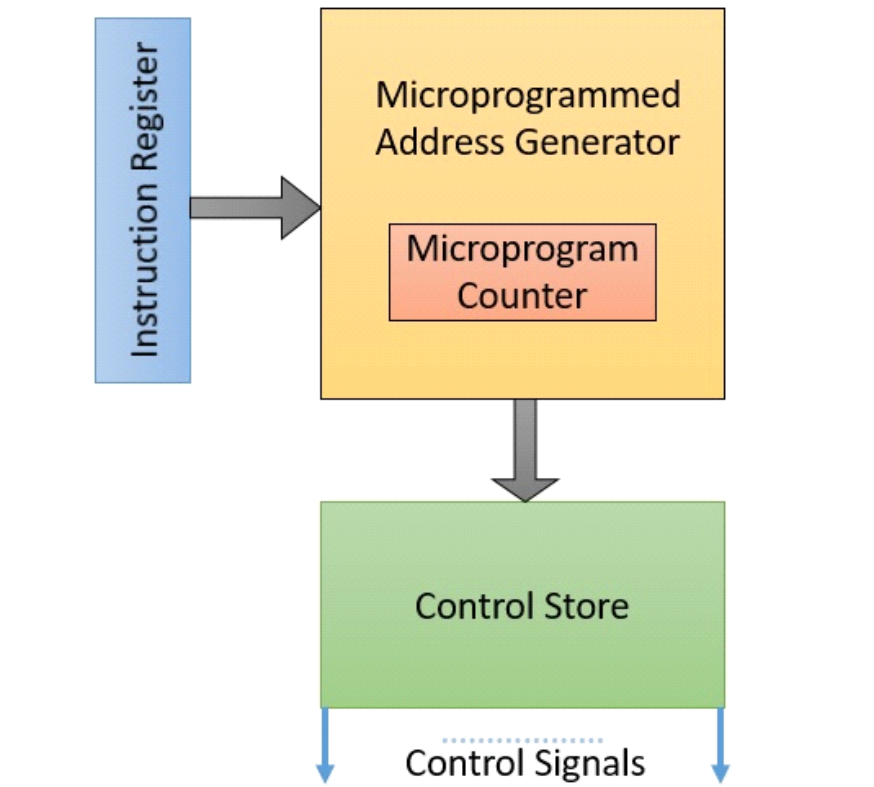


Microprogram Control

The organization of a **control word/ microinstruction, micro-routine** and **microprogram**.



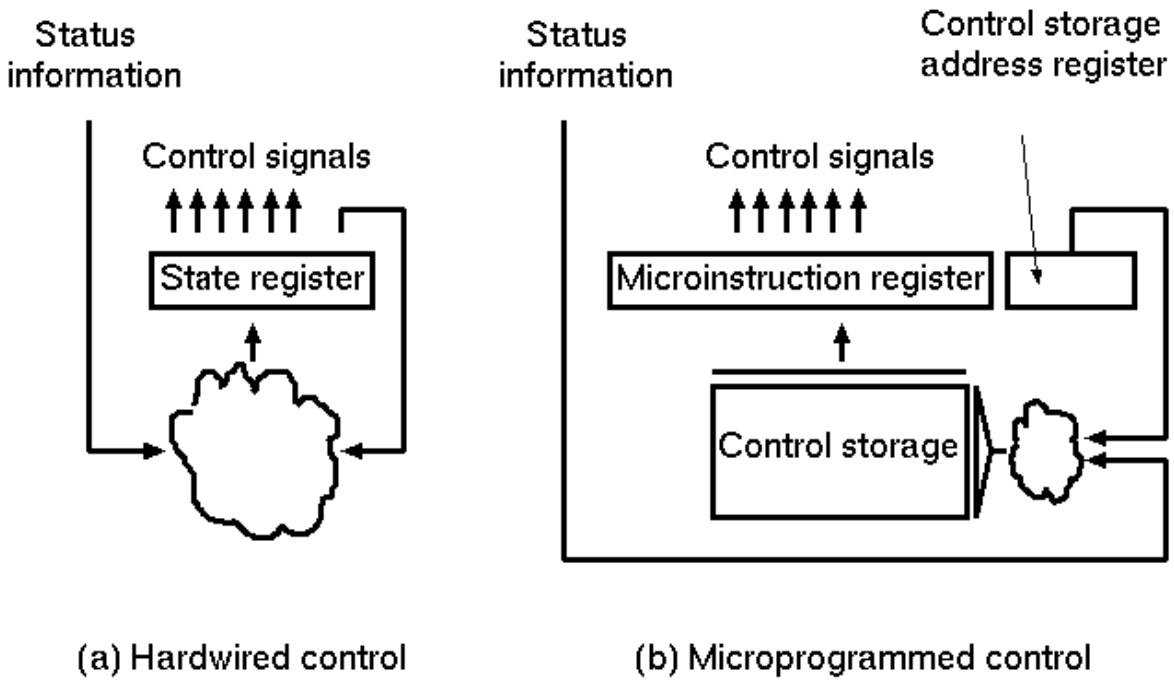
Organization of the **microprogram control unit**



Microprogrammed Control Unit Organization

Hardwired Control : Control logic is implemented with gates, flip-flops, decoders and other digital circuits
It can be optimized to produce a fast mode of operation.
Lacks flexibility.

Microprogrammed Control: Control information is stored in control memory.
The Control memory is programmed to initiate the required sequence of Microoperations.
More flexible.



(a) Hardwired control (b) Microprogrammed control

Comparison of Hardwired Versus Microprogram Control

Attributes	Hardwired Control	Microprogramming Control
Speed	Fast	Slow
Cost of Implementation	More	Cheaper
Flexibility	Difficult to modify	Flexible
Ability to handle complex instruction	Difficult	Easier
Decoding	Complex	Easy
Application	RISC	CISC
Instruction Set Size	Small	Large
Control Memory	Absent	Present

Control Unit

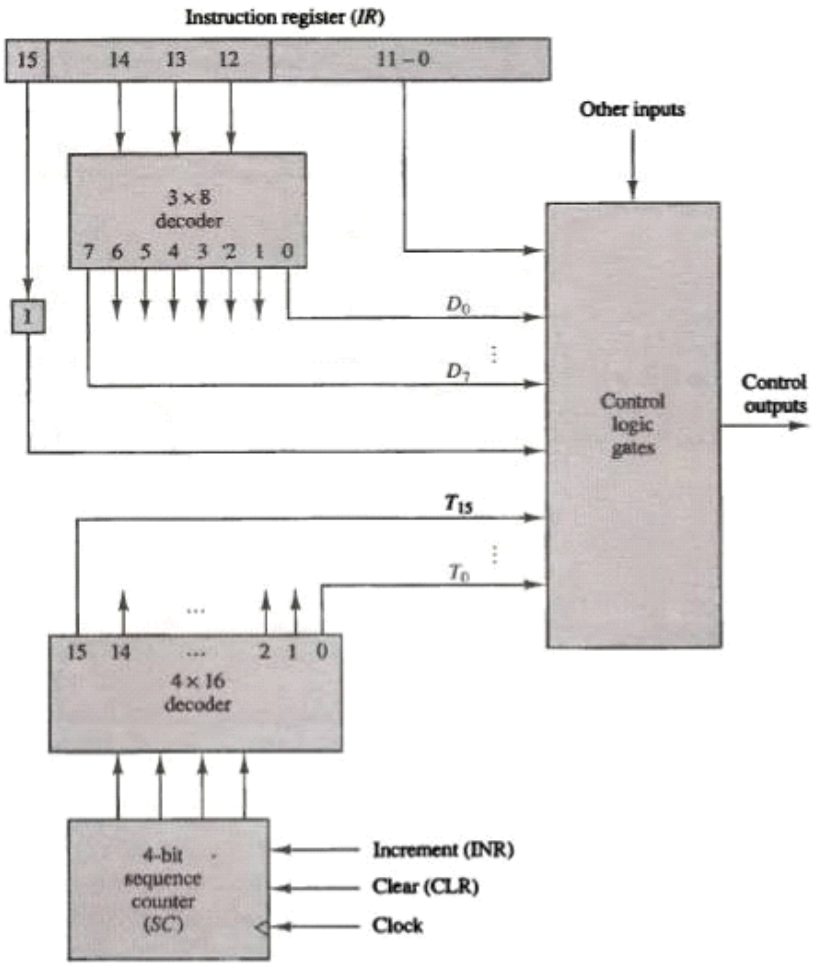


Figure 5-6 Control unit of basic computer.

Clock Pulses

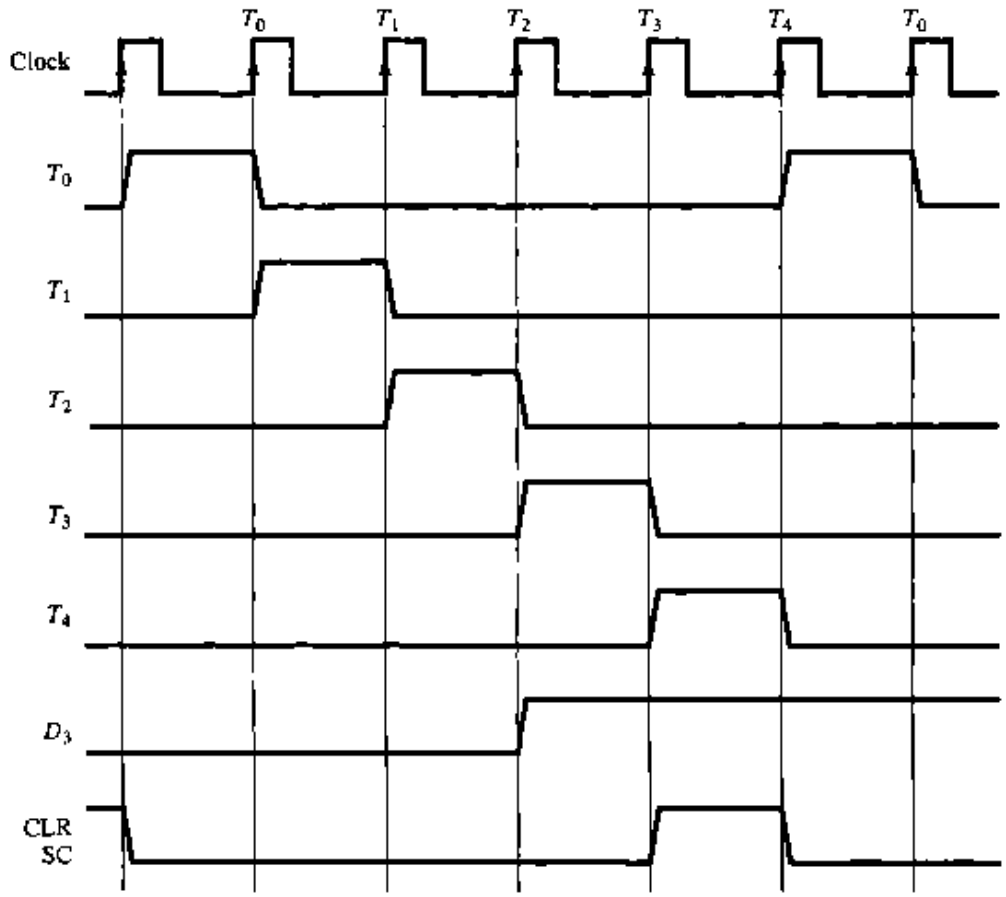


Figure 5-7 Example of control timing signals.

D_3T_4 : SC cleared to zero

Instruction Cycle

1. Fetch the instruction from memory
2. Decode the instruction
3. Read the effective address and fetch operand from memory
4. Execute the instruction

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

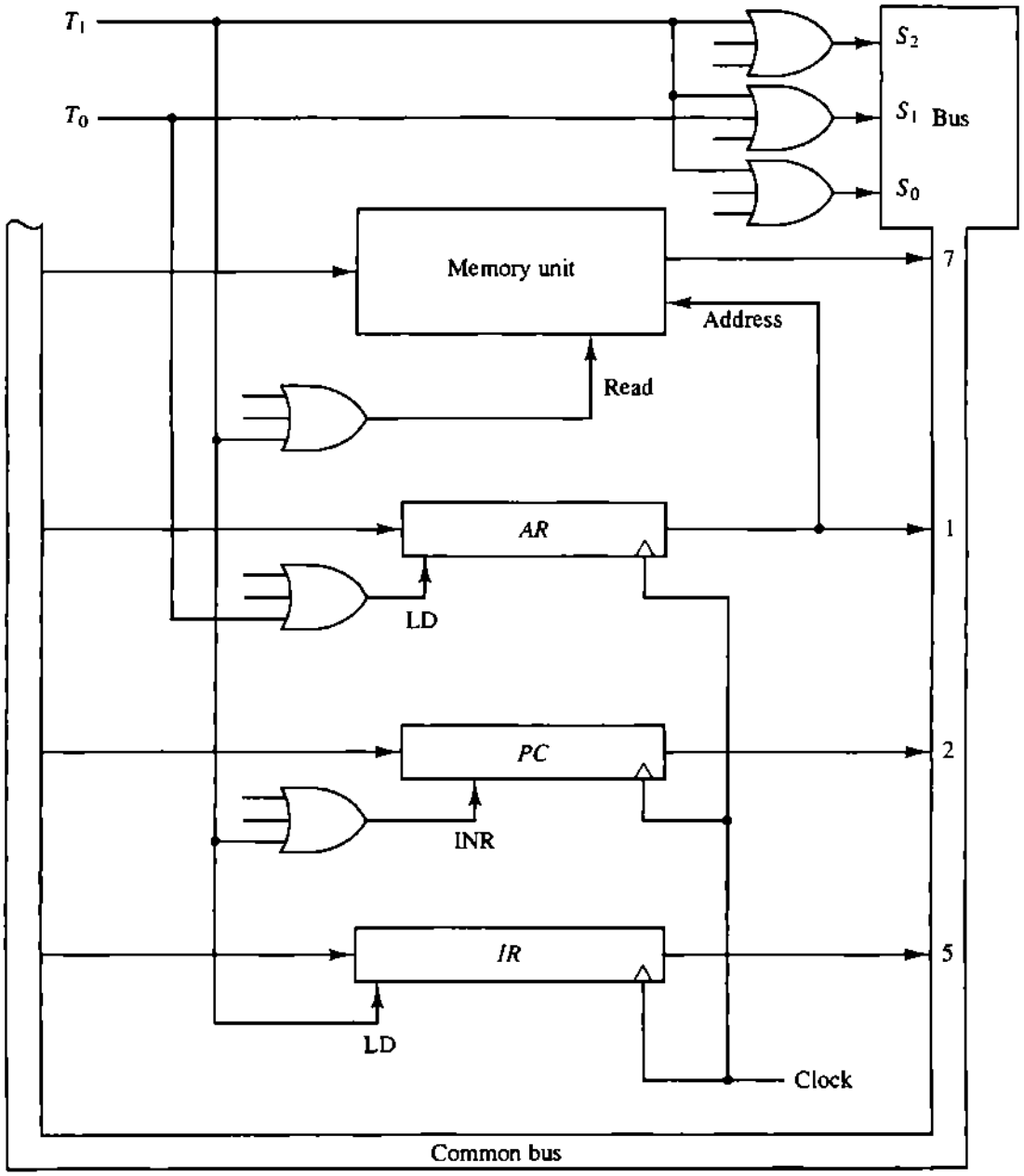


Figure 5-8 Register transfers for the fetch phase.

$D_7' IT_3$: $AR \leftarrow M[AR]$
$D_7' I'T_3$: Nothing
$D_7 I'T_3$: Execute a register-reference instruction
$D_7 IT_3$: Execute an input–output instruction

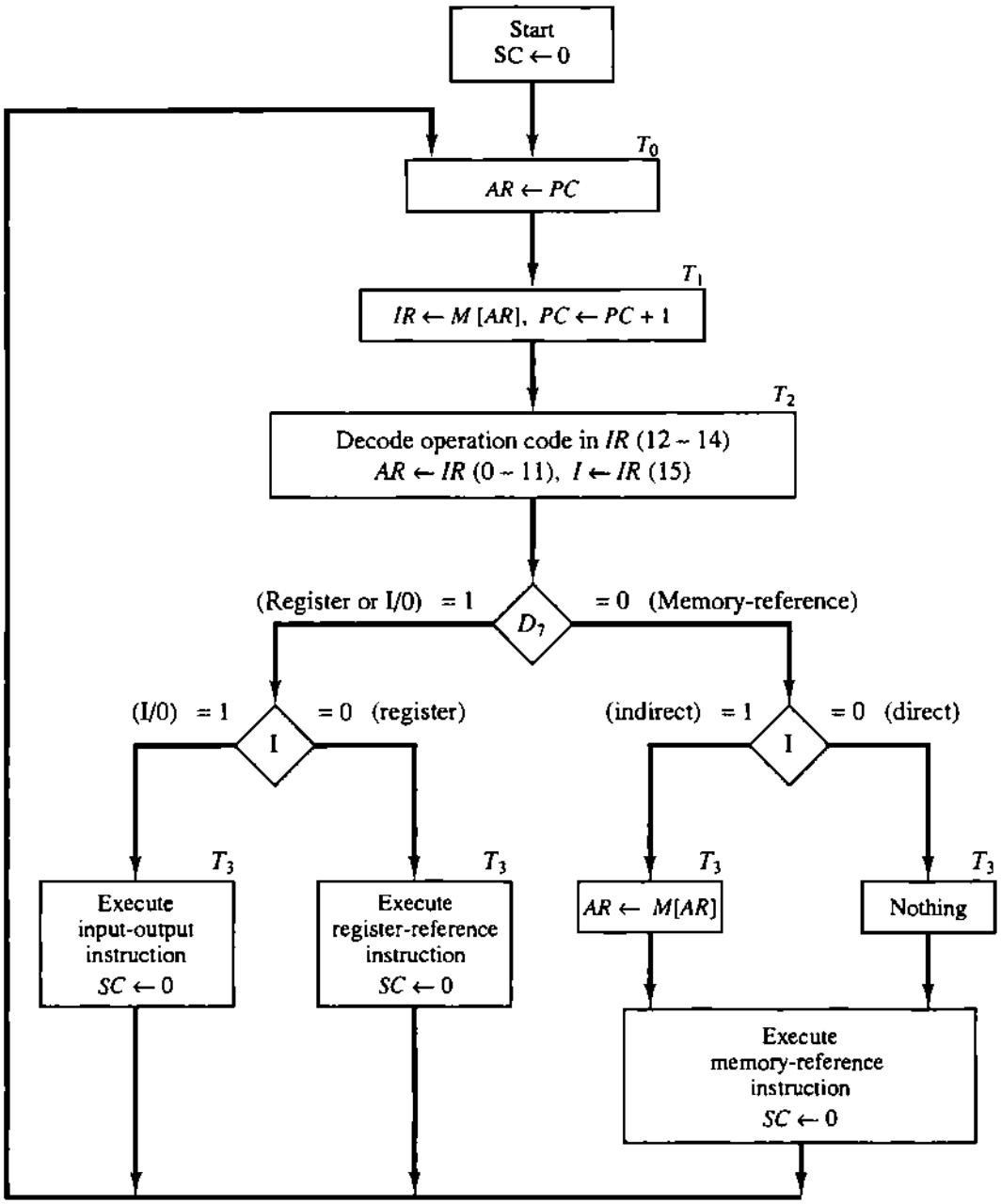


Figure 5-9 Flowchart for instruction cycle (initial configuration).

TABLE 5-3 Execution of Register-Reference Instructions

$D_7I'T_3 = r$ (common to all register-reference instructions)
$IR(i) = B_i$ [bit in $IR(0\text{–}11)$ 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

TABLE 5-4 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 $PC \leftarrow PC + 1$

AND to AC

D_0T_4: $DR \leftarrow M[AR]$
D_0T_5: $AC \leftarrow AC \wedge DR$, $SC \leftarrow 0$

ADD to AC

D_1T_4: $DR \leftarrow M[AR]$
D_1T_5: $AC \leftarrow AC + DR$, $E \leftarrow C_{out}$, $SC \leftarrow 0$

LDA : Load to AC

D_2T_4: $DR \leftarrow M[AR]$
D_2T_5: $AC \leftarrow DR, \quad SC \leftarrow 0$

STA: Store AC

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

BUN : Branch Unconditionally

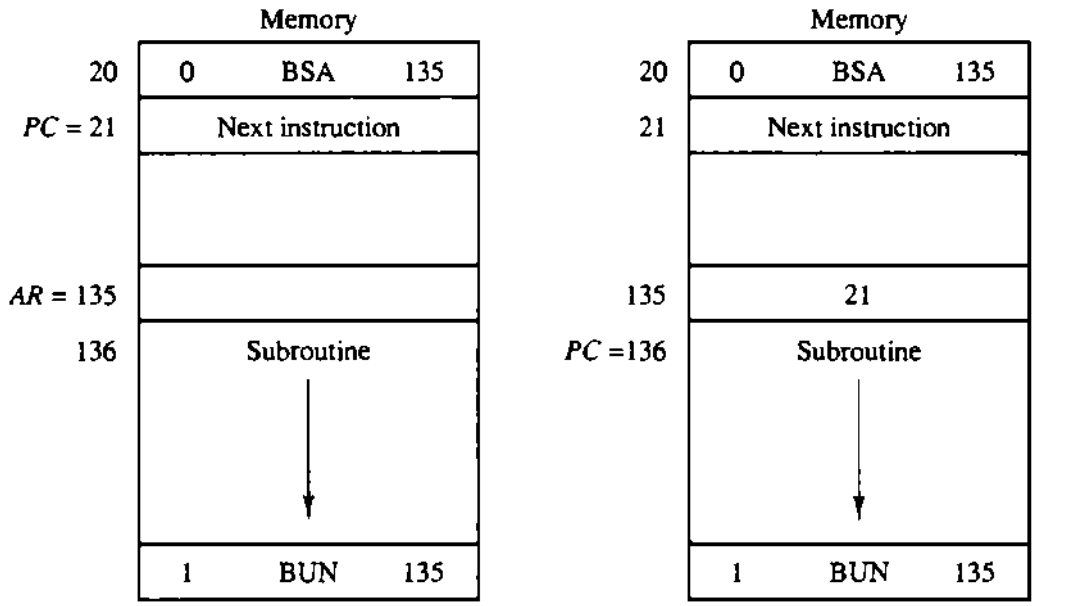
D_4T_4: $PC \leftarrow AR, \quad SC \leftarrow 0$

BSA : Branch and Save Return Address

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

Subroutine call

Figure 5-10 Example of BSA instruction execution.



(a) Memory, PC, and AR at time T_4 (b) Memory and PC after execution

D_5T_4: $M[AR] \leftarrow PC, \quad AR \leftarrow AR + 1$
D_5T_5: $PC \leftarrow AR, \quad SC \leftarrow 0$

ISZ: Increment and skip if zero

D_6T_4: $DR \leftarrow M[AR]$
D_6T_5: $DR \leftarrow DR + 1$
D_6T_6: $M[AR] \leftarrow DR$, if $(DR = 0)$ then $(PC \leftarrow PC + 1)$, $SC \leftarrow 0$

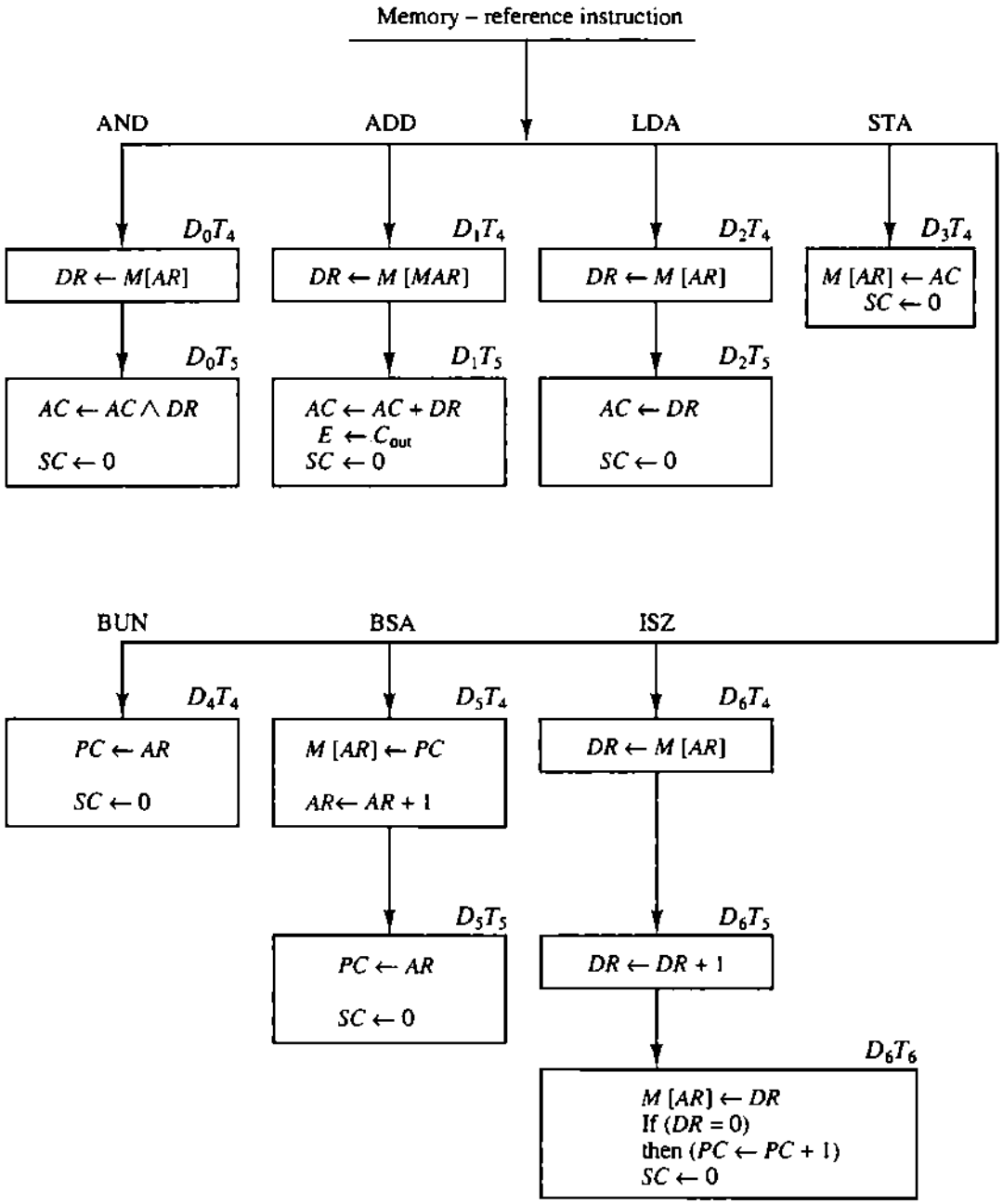


Figure 5-11 Flowchart for memory-reference instructions.

Input - Output and Interrupt

Figure 5-12 Input-output configuration.

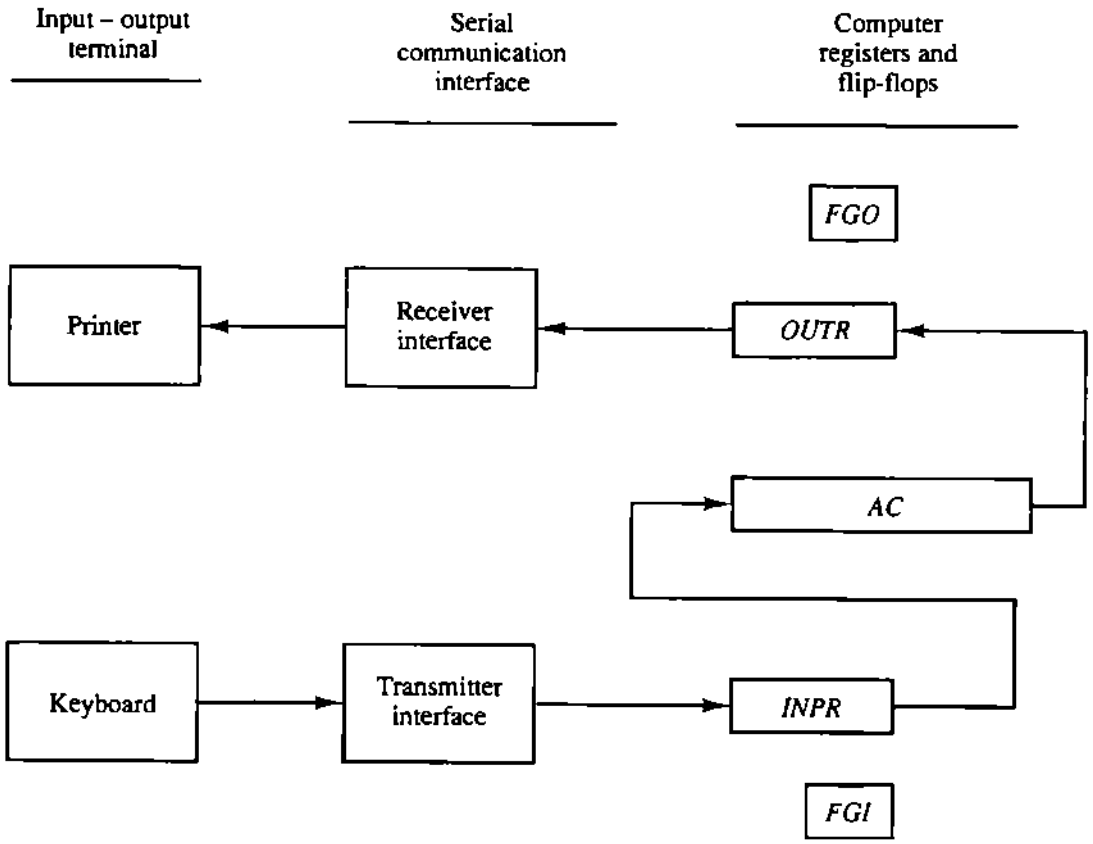


TABLE 5-5 Input-Output Instructions

$D_7IT_3 = p$ (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

Program Interrupt

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

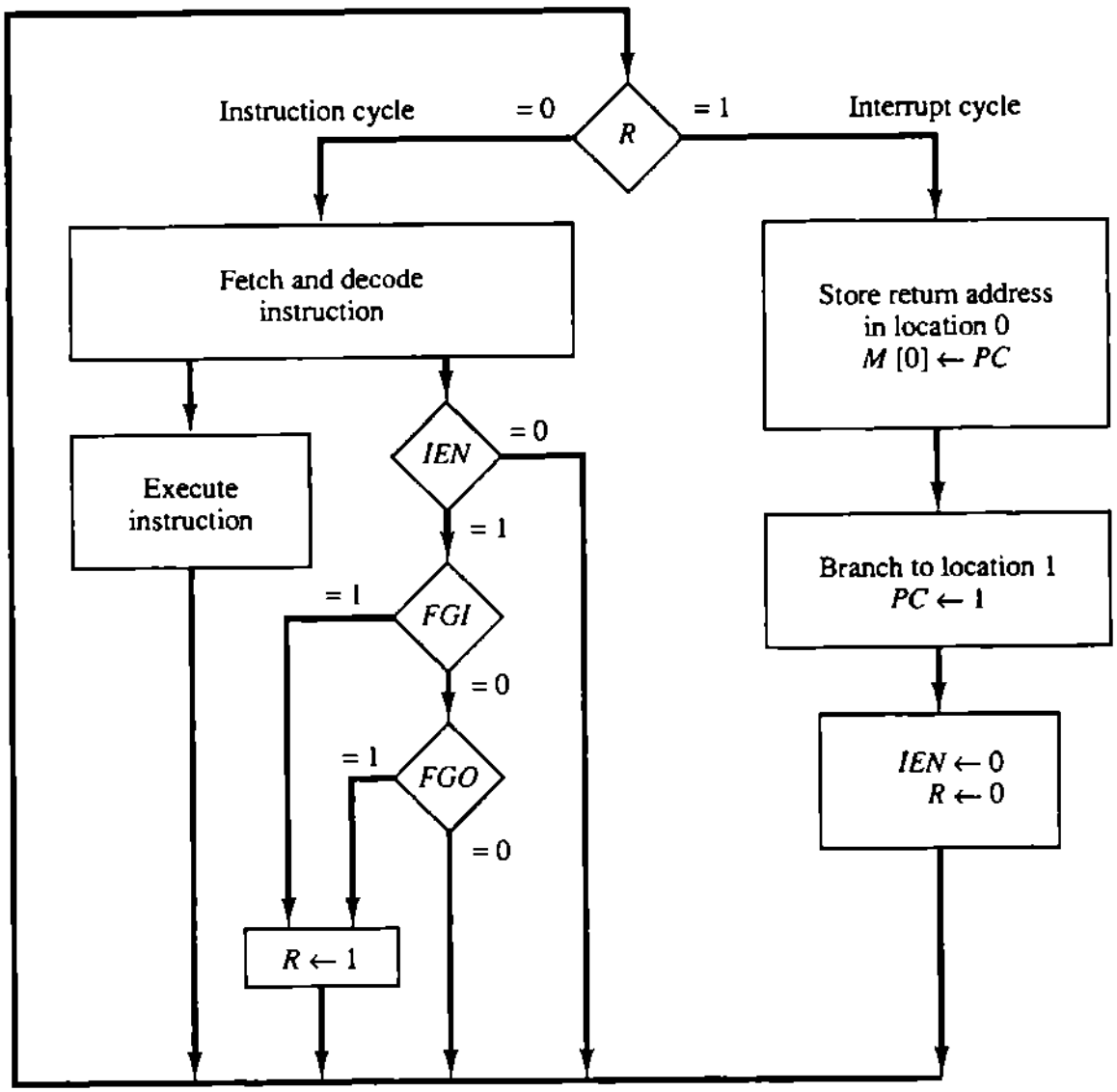


Figure 5-13 Flowchart for interrupt cycle.

Figure 5-14 Demonstration of the interrupt cycle.

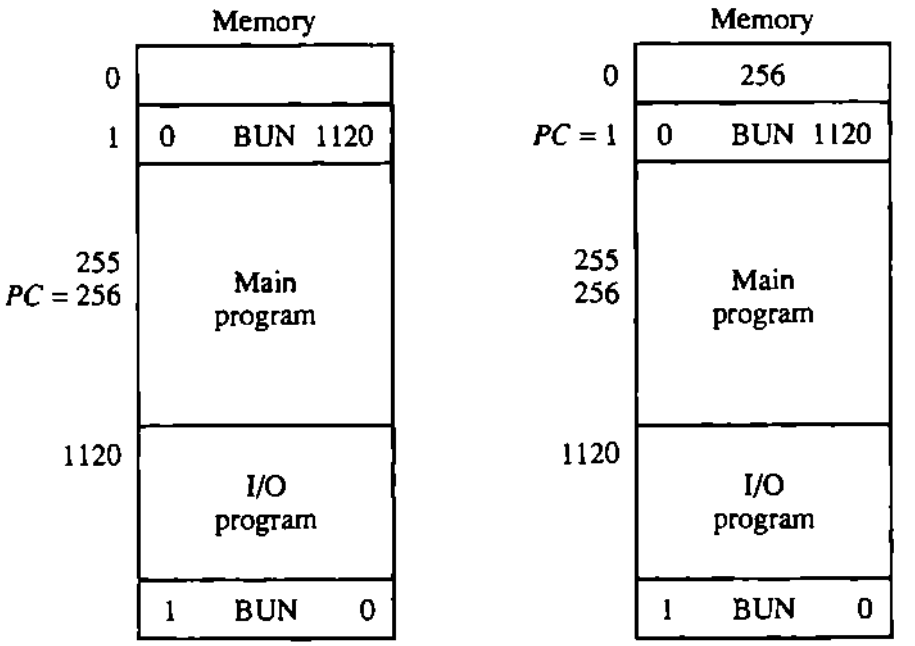
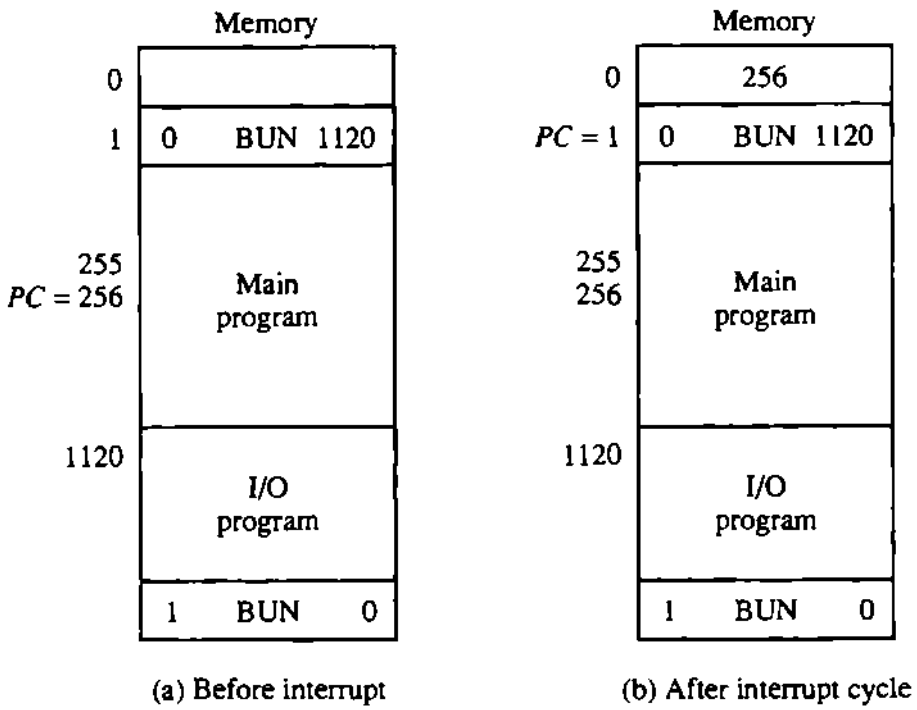


Figure 5-14 Demonstration of the interrupt cycle.



Interrupt Cycle

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

Modified Fetch Phase

Fetch and decode phase will happen when the control function is $R'T_0$, $R'T_1$, $R'T_2$

Interrupt Cycle

RT_0: $AR \leftarrow 0$, $TR \leftarrow PC$
RT_1: $M[AR] \leftarrow TR$, $PC \leftarrow 0$
RT_2: $PC \leftarrow PC + 1$, $IEN \leftarrow 0$, $R \leftarrow 0$, $SC \leftarrow 0$

Design of Basic Computer

The basic computer consists of the following hardware components:

1. A memory unit with 4096 words of 16 bits each
2. Nine registers: *AR*, *PC*, *DR*, *AC*, *IR*, *TR*, *OUTR*, *INPR*, and *SC*
3. Seven flip-flops: *I*, *S*, *E*, *R*, *IEN*, *FGI*, and *FGO*
4. Two decoders: a 3 × 8 operation decoder and a 4 × 16 timing decoder
5. A 16-bit common bus
6. Control logic gates
7. Adder and logic circuit connected to the input of *AC*

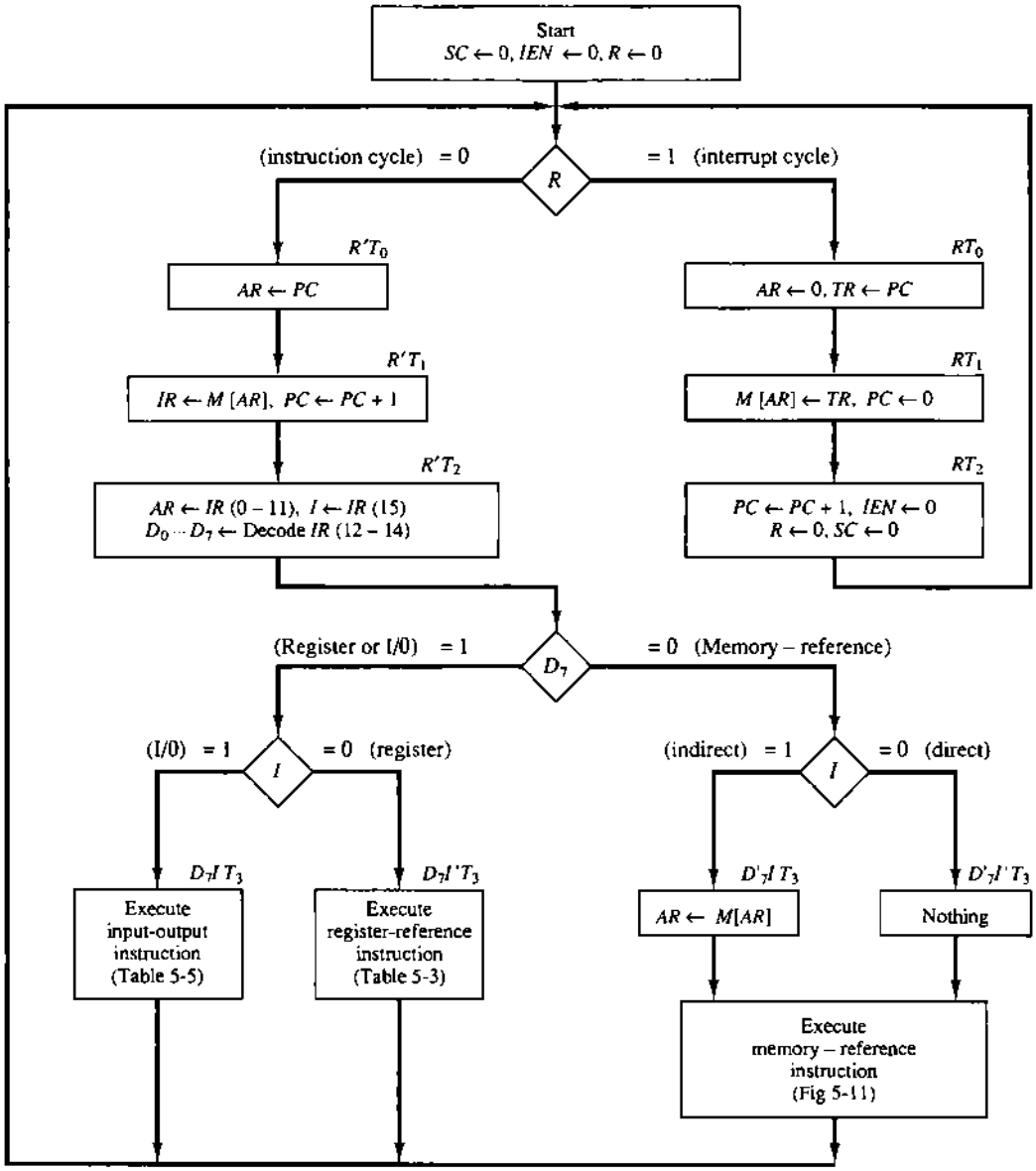


Figure 5-15 Flowchart for computer operation.

TABLE 5-6 Control Functions and Microoperations for the Basic Computer

Fetch	$R'T_0$:	$AR \leftarrow PC$
	$R'T_1$:	$IR \leftarrow M[AR]$, $PC \leftarrow PC + 1$
Decode	$R'T_2$:	$D_0, \ldots, D_7 \leftarrow$ Decode $IR(12\text{–}14)$, $AR \leftarrow IR(0\text{–}11)$, $I \leftarrow IR(15)$
Indirect	$D_7'IT_3$:	$AR \leftarrow M[AR]$
Interrupt:		
	$T_0'T_1'T_2'(IEN)(FGI + FGO)$:	$R \leftarrow 1$
	RT_0:	$AR \leftarrow 0$, $TR \leftarrow PC$
	RT_1:	$M[AR] \leftarrow TR$, $PC \leftarrow 0$
	RT_2:	$PC \leftarrow PC + 1$, $IEN \leftarrow 0$, $R \leftarrow 0$, $SC \leftarrow 0$
Memory-reference:		
AND	D_0T_4:	$DR \leftarrow M[AR]$
	D_0T_5:	$AC \leftarrow AC \wedge DR$, $SC \leftarrow 0$
ADD	D_1T_4:	$DR \leftarrow M[AR]$
	D_1T_5:	$AC \leftarrow AC + DR$, $E \leftarrow C_{out}$, $SC \leftarrow 0$
LDA	D_2T_4:	$DR \leftarrow M[AR]$
	D_2T_5:	$AC \leftarrow DR$, $SC \leftarrow 0$
STA	D_3T_4:	$M[AR] \leftarrow AC$, $SC \leftarrow 0$
BUN	D_4T_4:	$PC \leftarrow AR$, $SC \leftarrow 0$
BSA	D_5T_4:	$M[AR] \leftarrow PC$, $AR \leftarrow AR + 1$
	D_5T_5:	$PC \leftarrow AR$, $SC \leftarrow 0$
ISZ	D_6T_4:	$DR \leftarrow M[AR]$
	D_6T_5:	$DR \leftarrow DR + 1$
	D_6T_6:	$M[AR] \leftarrow DR$, if $(DR = 0)$ then $(PC \leftarrow PC + 1)$, $SC \leftarrow 0$
Register-reference:		
	$D_7I'T_3 = r$ (common to all register-reference instructions)	
	$IR(i) = B_i$ $(i = 0, 1, 2, \ldots, 11)$	
	r:	$SC \leftarrow 0$
CLA	rB_{11}:	$AC \leftarrow 0$
CLE	rB_{10}:	$E \leftarrow 0$
CMA	rB_9:	$AC \leftarrow \overline{AC}$
CME	rB_8:	$E \leftarrow \overline{E}$
CIR	rB_7:	$AC \leftarrow$ shr AC, $AC(15) \leftarrow E$, $E \leftarrow AC(0)$
CIL	rB_6:	$AC \leftarrow$ shl AC, $AC(0) \leftarrow E$, $E \leftarrow AC(15)$
INC	rB_5:	$AC \leftarrow AC + 1$
SPA	rB_4:	If $(AC(15) = 0)$ then $(PC \leftarrow PC + 1)$
SNA	rB_3:	If $(AC(15) = 1)$ then $(PC \leftarrow PC + 1)$
SZA	rB_2:	If $(AC = 0)$ then $PC \leftarrow PC + 1$
SZE	rB_1:	If $(E = 0)$ then $(PC \leftarrow PC + 1)$
HLT	rB_0:	$S \leftarrow 0$
Input-output:		
	$D_7IT_3 = p$ (common to all input–output instructions)	
	$IR(i) = B_i$ $(i = 6, 7, 8, 9, 10, 11)$	
	p:	$SC \leftarrow 0$
INP	pB_{11}:	$AC(0\text{–}7) \leftarrow INPR$, $FGI \leftarrow 0$
OUT	pB_{10}:	$OUTR \leftarrow AC(0\text{–}7)$, $FGO \leftarrow 0$
SKI	pB_9:	If $(FGI = 1)$ then $(PC \leftarrow PC + 1)$
SKO	pB_8:	If $(FGO = 1)$ then $(PC \leftarrow PC + 1)$
ION	pB_7:	$IEN \leftarrow 1$
IOF	pB_6:	$IEN \leftarrow 0$

Control logic Gates

The outputs of the control logic circuit are:

1. Signals to control the inputs of the nine registers
2. Signals to control the read and write inputs of memory
3. Signals to set, clear, or complement the flip-flops
4. Signals for S_2, S_1, and S_0 to select a register for the bus
5. Signals to control the AC adder and logic circuit

Control of Registers and Memory

All statements that change the content of AR

$R'T_0$: $AR \leftarrow PC$
$R'T_2$: $AR \leftarrow IR(0\text{–}11)$
$D_7'IT_3$: $AR \leftarrow M[AR]$
RT_0: $AR \leftarrow 0$
D_5T_4: $AR \leftarrow AR + 1$

1-3 - Transfer of Information from a register or memory to AR

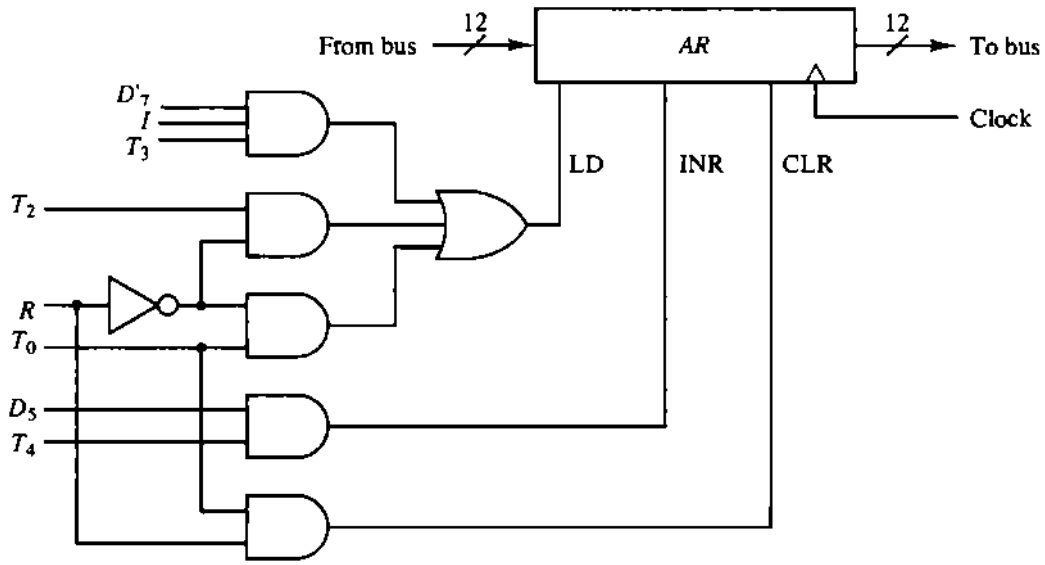
These control functions can be combined into 3 Boolean expression as follows:

$$\text{LD}(AR) = R'T_0 + R'T_2 + D_7'IT_3$$
$$\text{CLR}(AR) = RT_0$$
$$\text{INR}(AR) = D_5T_4$$

Read operation that reads M[AR] can be specified as

$$\text{Read} = R'T_1 + D_7'IT_3 + (D_0 + D_1 + D_2 + D_6)T_4$$

Figure 5-16 Control gates associated with AR.



Control of Single Flip-Flop

$$pB_7: \quad IEN \leftarrow 1$$
$$pB_6: \quad IEN \leftarrow 0$$

where $p = D_7IT_3$ and B_7 and B_6 are bits 7 and 6 of IR, respectively. Moreover, at the end of the interrupt cycle IEN is cleared to 0.

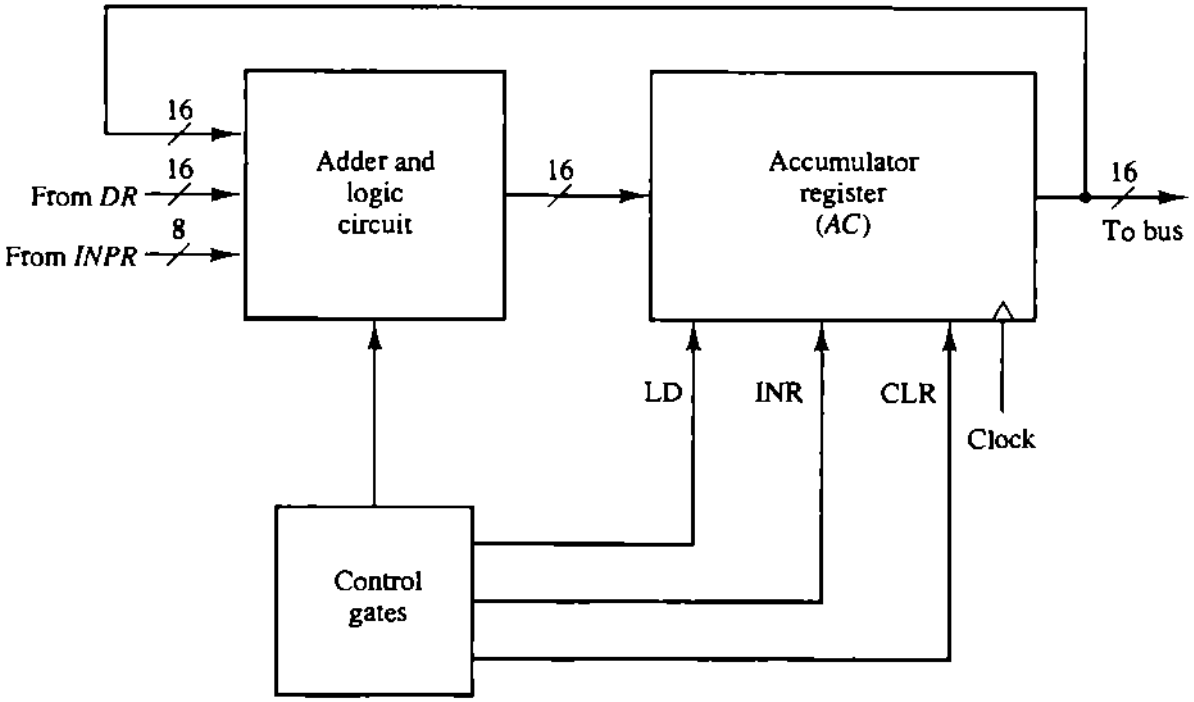
$$RT_2: \quad IEN \leftarrow 0$$

Design of Accumulator Logic

All statements that change the content of Accumulator

D_0T_5:	$AC \leftarrow AC \wedge DR$	AND with DR
D_1T_5:	$AC \leftarrow AC + DR$	Add with DR
D_2T_5:	$AC \leftarrow DR$	Transfer from DR
pB_{11}:	$AC(0\text{–}7) \leftarrow INPR$	Transfer from $INPR$
rB_9:	$AC \leftarrow \overline{AC}$	Complement
rB_7:	$AC \leftarrow \text{shr } AC, \quad AC(15) \leftarrow E$	Shift right
rB_6:	$AC \leftarrow \text{shl } AC, \quad AC(0) \leftarrow E$	Shift left
rB_{11}:	$AC \leftarrow 0$	Clear
rB_5:	$AC \leftarrow AC + 1$	Increment

Figure 5-19 Circuits associated with AC.



Control of AC Register

Figure 5-20 Gate structure for controlling the LD, INR, and CLR of AC.