IAA6007: Computer Architecture
Ch.5. Basic Computer Organization and Design

Wooil Kim
Dept. of Computer Science & Engineering
Incheon National University

2019 Fall

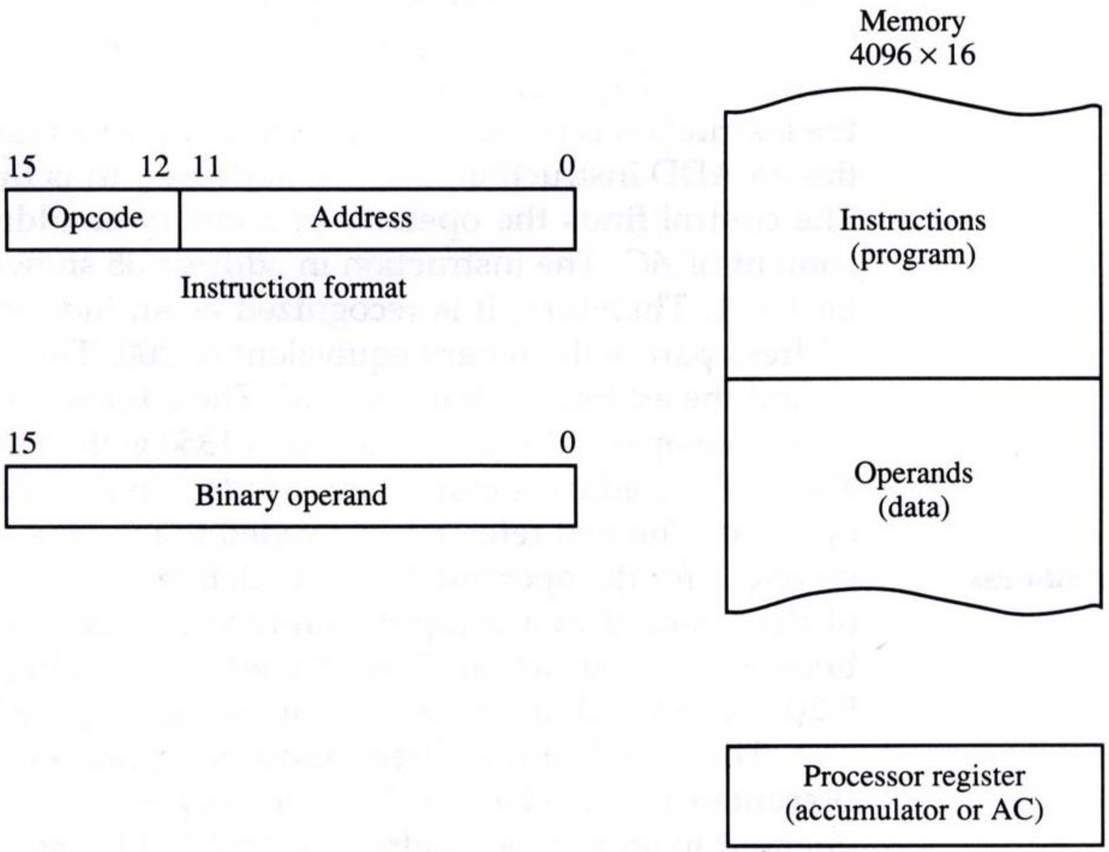
Outline

- 5.1 Instruction codes
- 5.2 Computer registers
- 5.3 Computer Instructions
- 5.4 Timing and control
- 5.5 Instruction cycle
- 5.6 Memory reference instructions
- 5.7 Input-output and interrupt
- 5.8 Complete computer description
- 5.9 Design of basic computer
- 5.10 Design of accumulator logic

5.1 Instruction codes

- A program – a set of instruction
- An instruction specifies a sequence microoperations

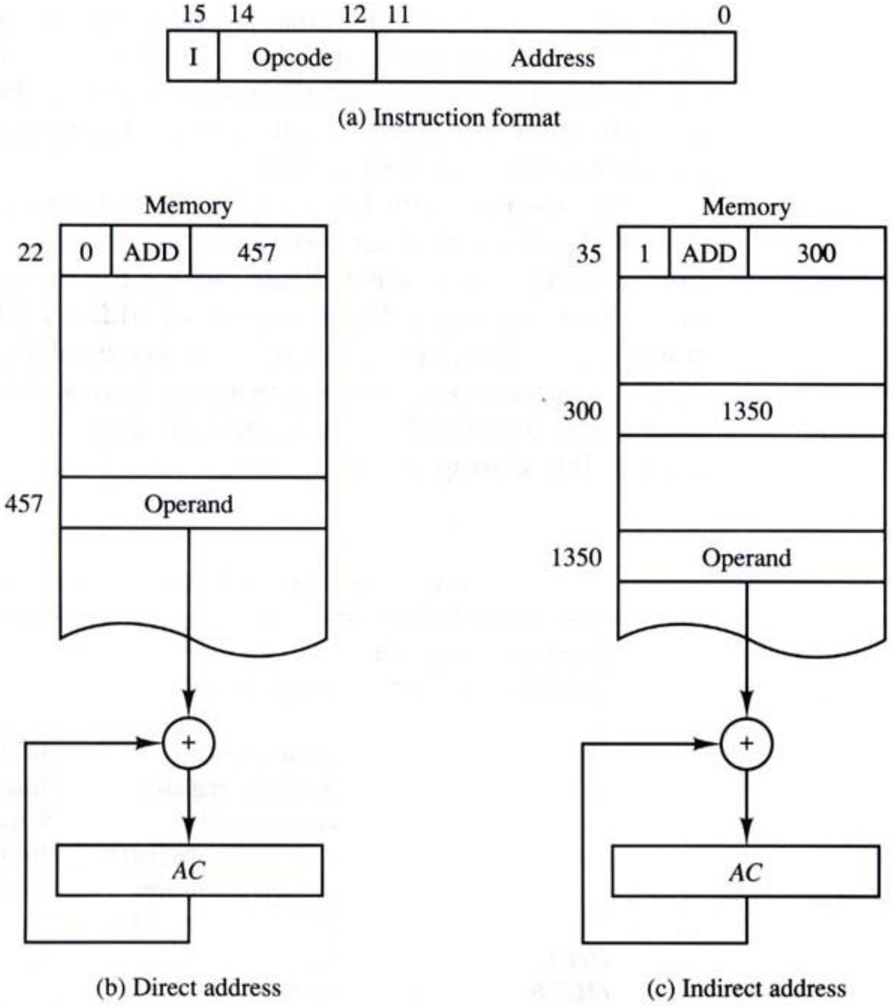
Figure 5-1 Stored program organization.



5.1 Instruction codes

- Accumulator (AC)
  - Computers that have a single processor register usually assign to it the name accumulator
- Effective address (유효주소)
  - When the second part of the instruction specifies
    - i) an operand (피연산자) – immediate
    - ii) the address of an operand – direct address
    - iii) the address of a memory word where the address of the operand can be found – indirect address
  - The indirect address instruction needs two references to memory to fetch an operand
  - Effective address – address of the operand

5.1 Instruction codes



(a) Instruction format

(b) Direct address

(c) Indirect address

Figure 5-2 Demonstration of direct and indirect address.

5.2 Computer registers

- A basic computer having a memory unit with a capacity of 4096 words & 16-bit word
- List of registers
  - Data Register (DR, 16), Address Register (AR, 12)
  - Accumulator (AC, 16), Instruction Register (IR, 16)
  - Program Counter (PC, 12), Temporary Register (TR, 16)
  - Input Register (INPR, 8), Output Register (OUTR, 8)

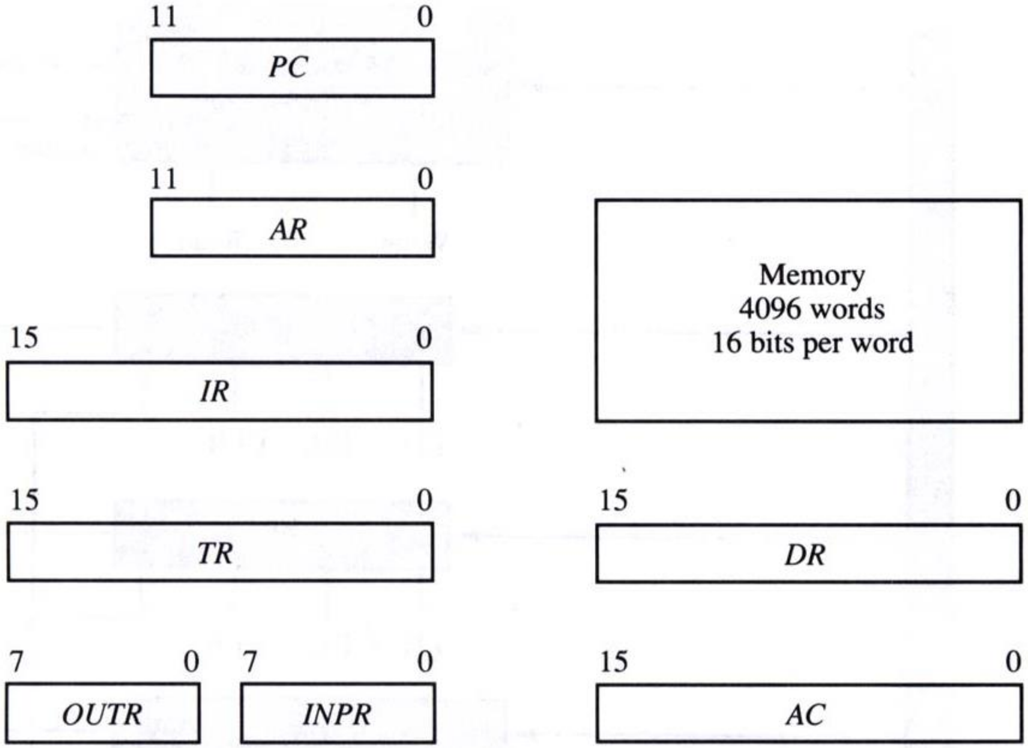


Figure 5-3 Basic computer registers and memory.

5.2 Computer registers

- Common bus system
  - Outputs of registers/memory are selected by selection inputs $S_2S_1S_0$
    - $S_2S_1S_0$ = 011: BUS <- DR
    - $S_2S_1S_0$ = 111: BUS <- M[AR]
  - When AR or PC is selected 4 most significant bits are set to 0's
  - When AR or PC receives information from bus only 12 least significant bits are transferred
  - For INPR and OUTR, 8 least significant bits are transferred

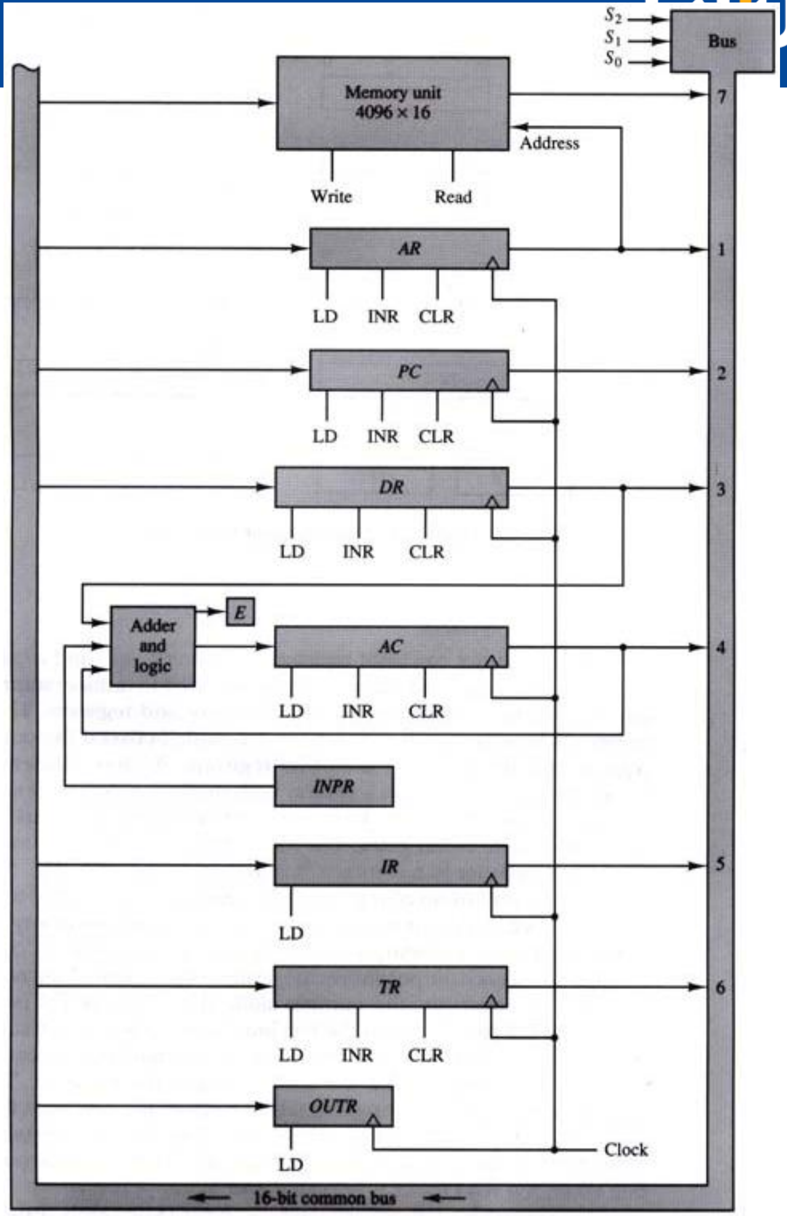
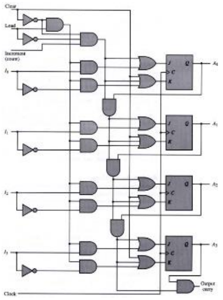


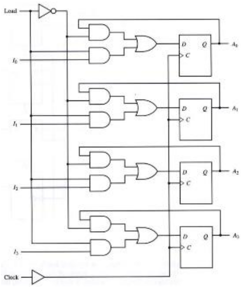
Figure 5-4 Basic computer registers connected to a common bus.

5.2 Computer registers

- Common bus system
  - Registers with LD, INR, & CLR
    - Figure 2-11



  - Registers with LD
    - Figure 2-7



  - Register transfer using bus and operation performed in AL circuit can occur during the same clock cycle
    - $S_2S_1S_0 = 100$, LD(DR, AC) = 1
    - DR ← AC, AC ← DR

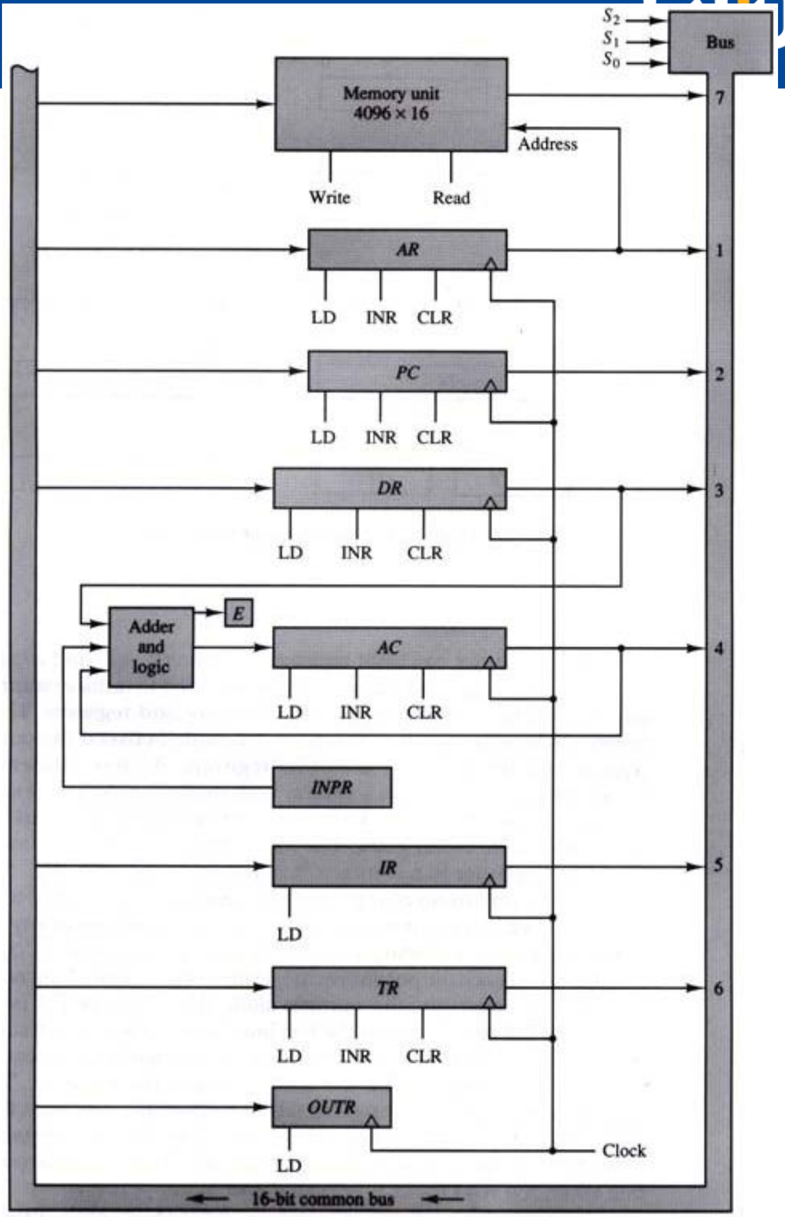
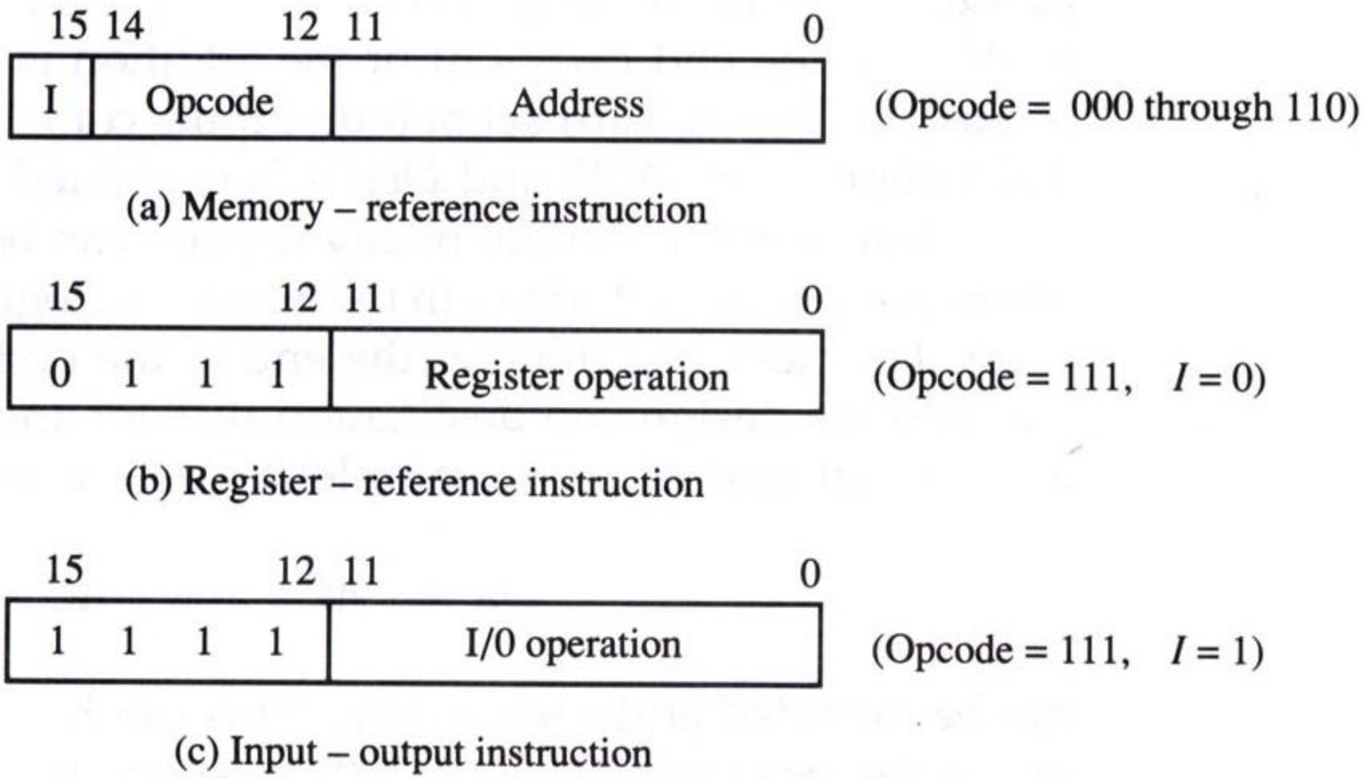


Figure 5-4 Basic computer registers connected to a common bus.

5.3 Computer Instructions

- Instruction format

Figure 5-5 Basic computer instruction formats.



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

5.3 Computer Instructions

- Instruction set

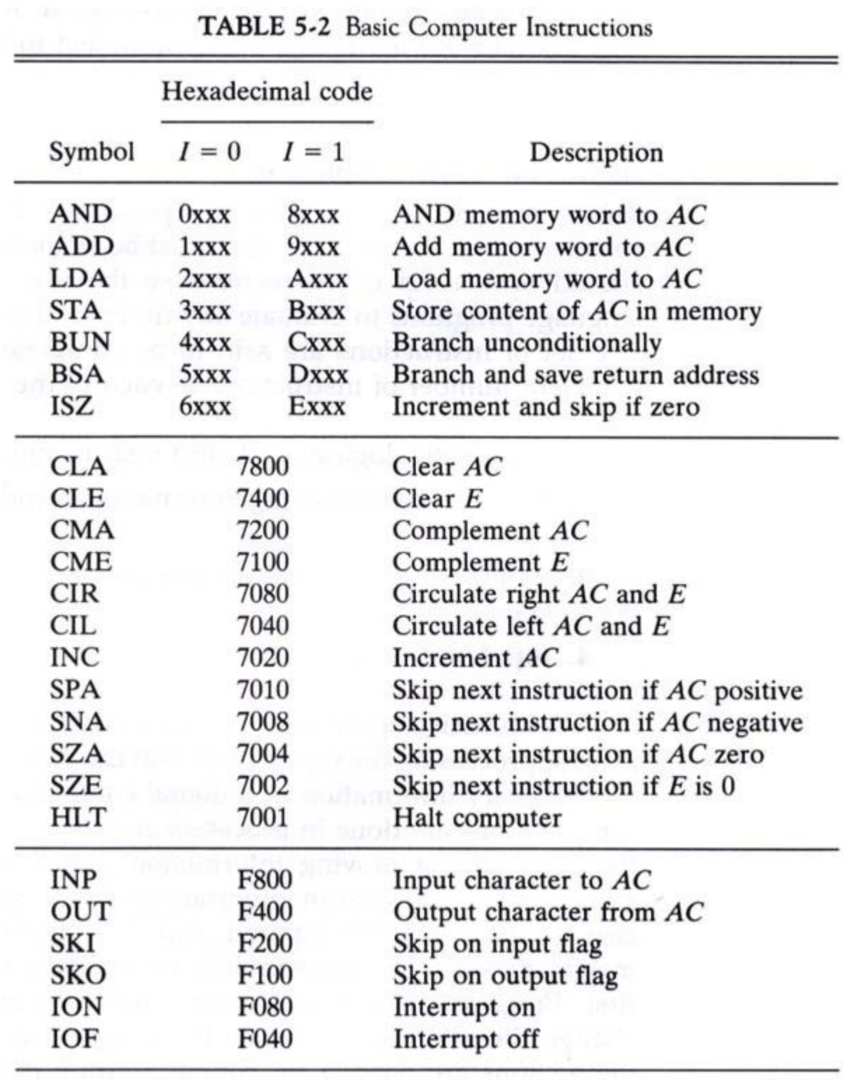


TABLE 5-2 Basic Computer Instructions

	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

5.4 Timing and control

- Control unit of basic computer

3x8 decoder

I4	I3	I2	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

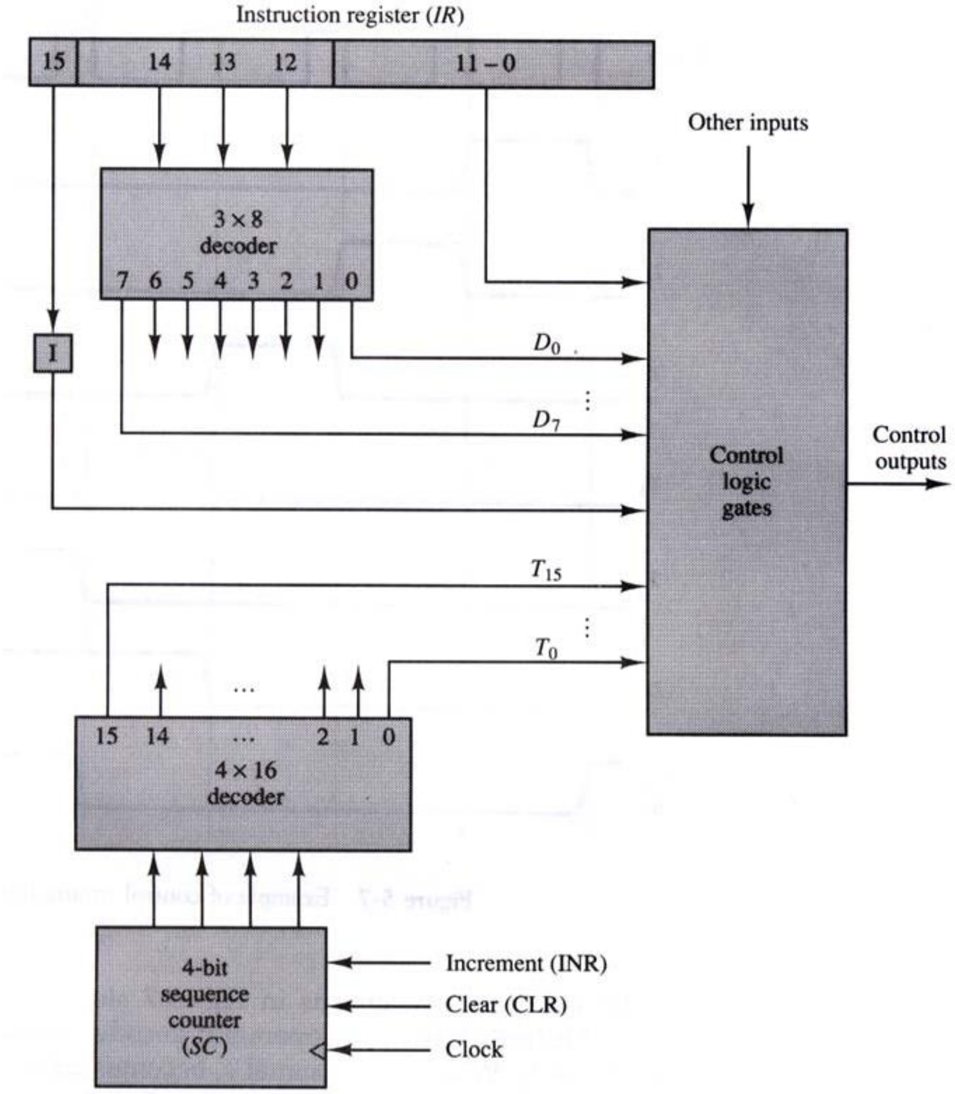


Figure 5-6 Control unit of basic computer.

5.4 Timing and control

- The sequence counter SC can be incremented or cleared synchronously
- Relation between the clock
    - Ex) D_3T_4 : SC ← 0

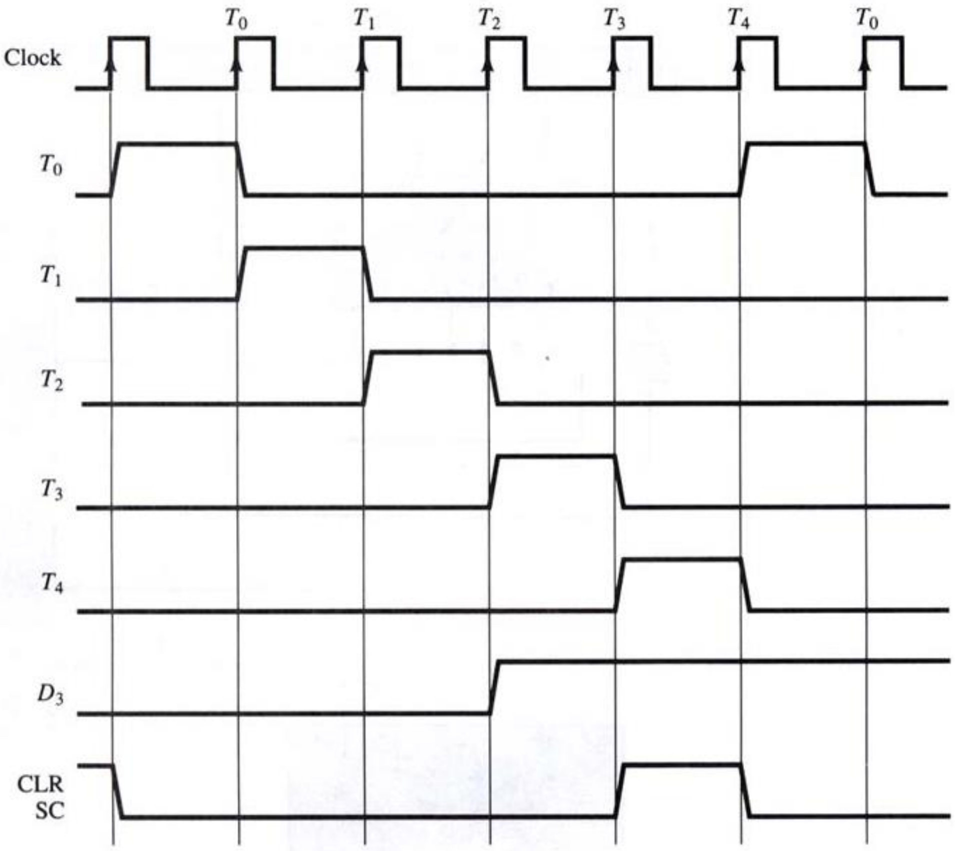
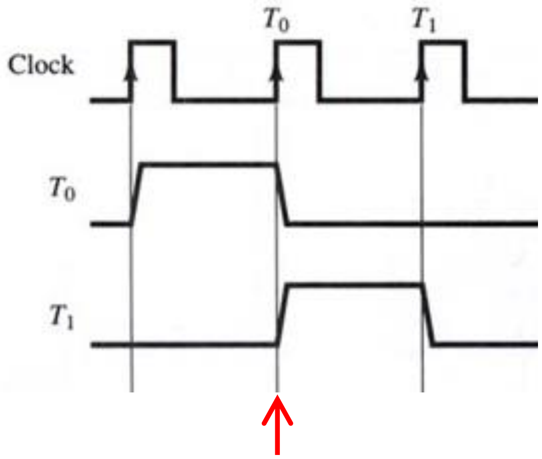


Figure 5-7 Example of control timing signals.

5.4 Timing and control

- Relation between the clock transition and timing signal
  - Ex) T_0 : AC ← PC
  - During the time T_0 is active, PC is loaded onto the bus and LD of AC is enabled
  - The actual transfer occurs at the end of the clock cycle when clock goes through a positive transition



actual transfer

5.5 Instruction cycle

- Instruction cycle consists of
  - 1. Fetch
  - 2. Decode
  - 3. Read effective address (if indirect address)
  - 4. Execute

- Fetch and decode
  - Initially, PC is loaded with the address of the first instruction and SC is cleared to 0
  - T_0 : AR ← PC
  - T_1 : IR ← M[AR], PC ← PC + 1
  - T_2 : D_0,…, D_7 ← Decode IR(12 - 14), AR ← IR(0 - 11), I ← IR(15)
  - It is assumed that memory cycle time is less than the clock time

5.5 Instruction cycle

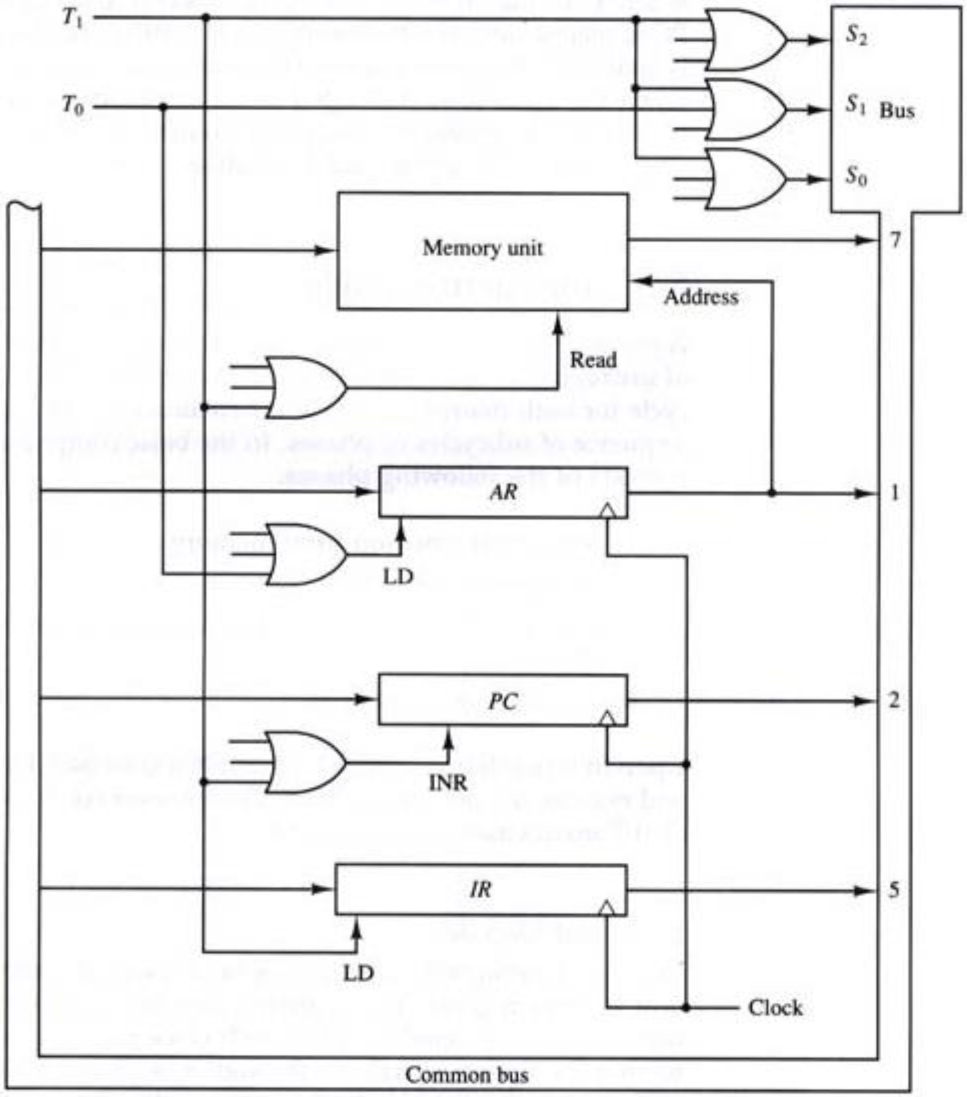


Figure 5-8 Register transfers for the fetch phase.

T_0 : AR ← PC

T_1 : IR ← M[AR], PC ← PC + 1

5.5 Instruction cycle

- Determine the type of instruction
  - $D_7'IT_3$: AR ← M[AR]
  - $D_7'I'T_3$: Nothing
  - $D_7I'T_3$: Execute a register reference instruction
  - D_7IT_3 : Execute an I/O instruction

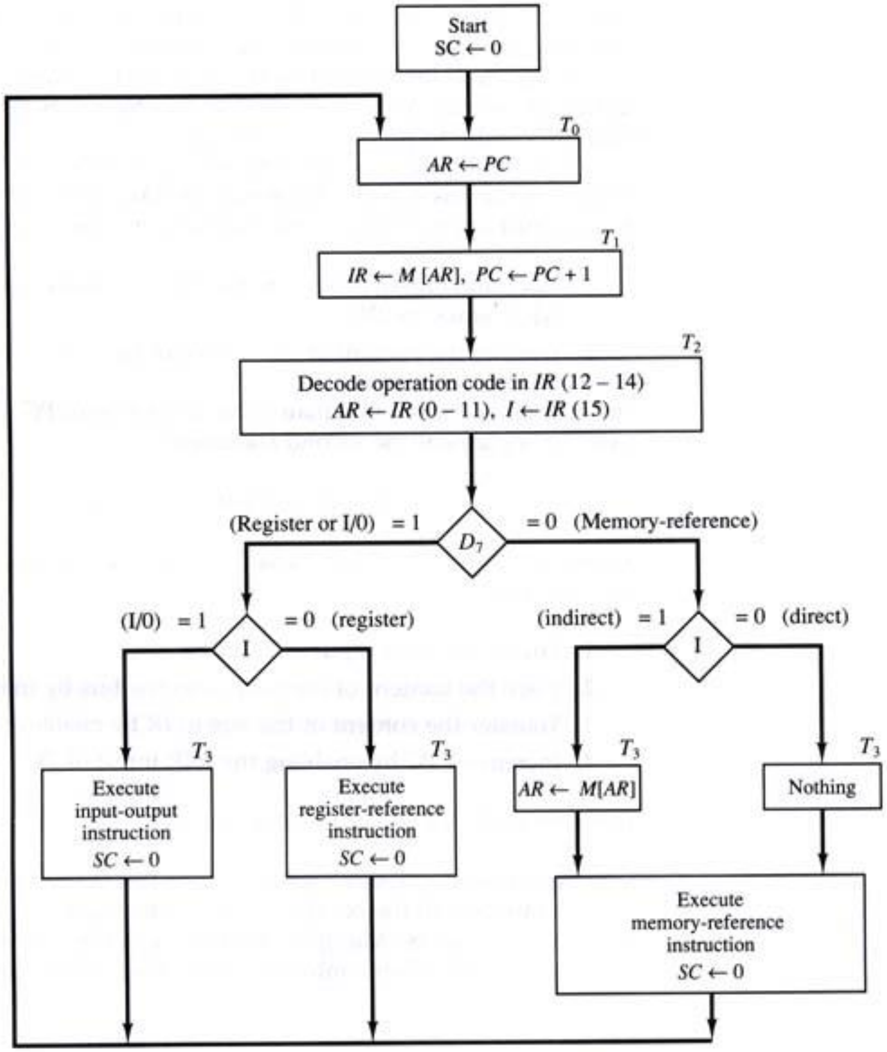
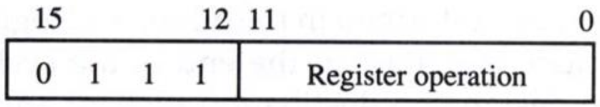

Start
SC ← 0
T_0
$AR \leftarrow PC$
T_1
$IR \leftarrow M[AR],\ PC \leftarrow PC + 1$
T_2
Decode operation code in IR (12 – 14)
$AR \leftarrow IR$ (0 – 11), $I \leftarrow IR$ (15)
(Register or I/0) = 1
D_7
= 0 (Memory-reference)
(I/0) = 1
I
= 0 (register)
(indirect) = 1
I
= 0 (direct)
T_3
Execute input-output instruction $SC \leftarrow 0$
T_3
Execute register-reference instruction $SC \leftarrow 0$
T_3
$AR \leftarrow M[AR]$
T_3
Nothing
Execute memory-reference instruction $SC \leftarrow 0$

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

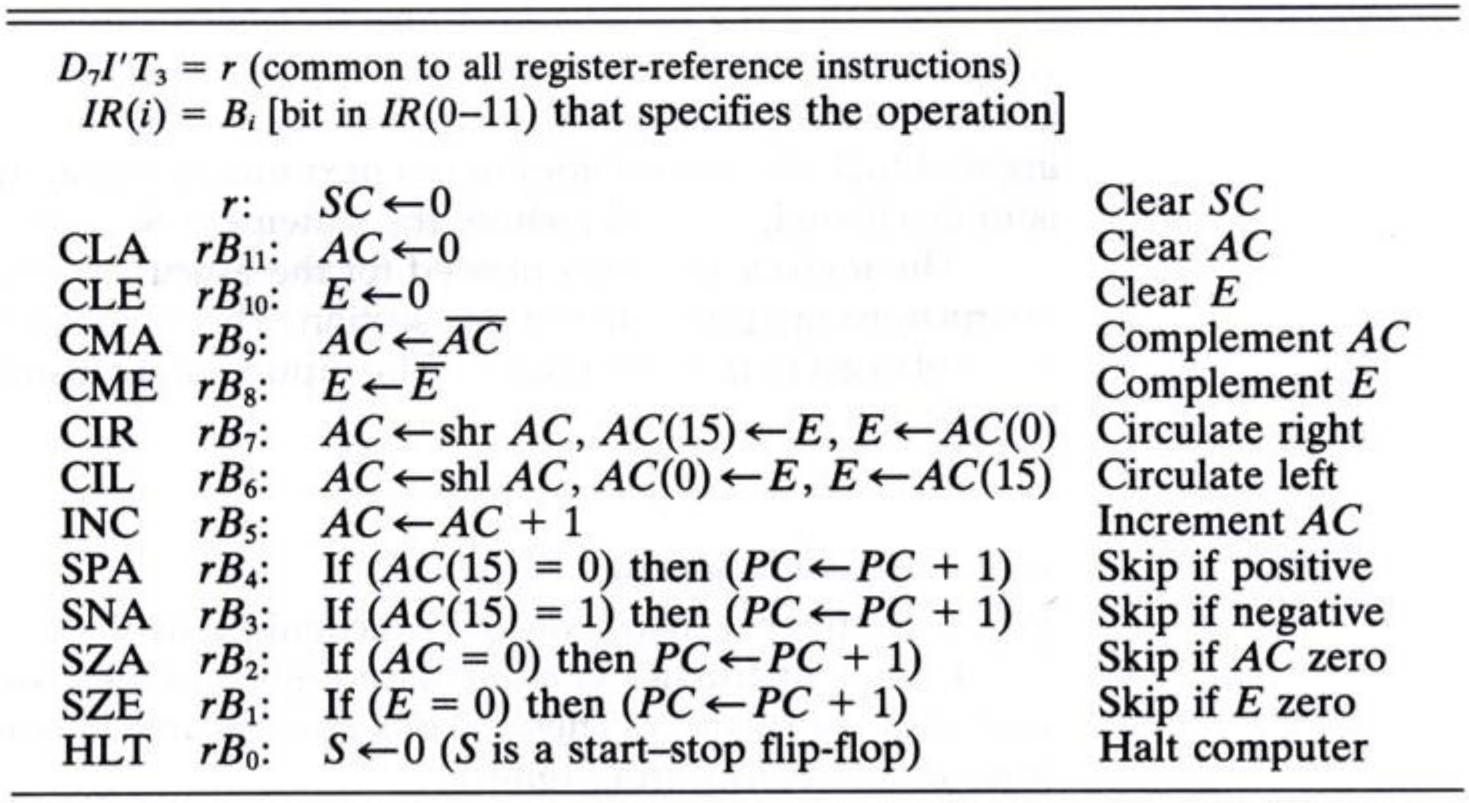
5.5 Instruction cycle

- Register reference instruction
  - B_i specifies the operation type (B_0~B_{11})
  - Ex) CLA: 0111 1000 0000 0000 = 7800, B_{11} = 1 (see Table 5-2)



(b) Register – reference instruction

TABLE 5-3 Execution of Register-Reference Instructions

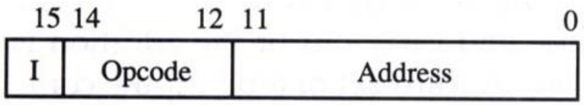


$D_7I'T_3 = r$ (common to all register-reference instructions)
$IR(i) = B_i$ [bit in $IR(0–11)$ that specifies the operation]

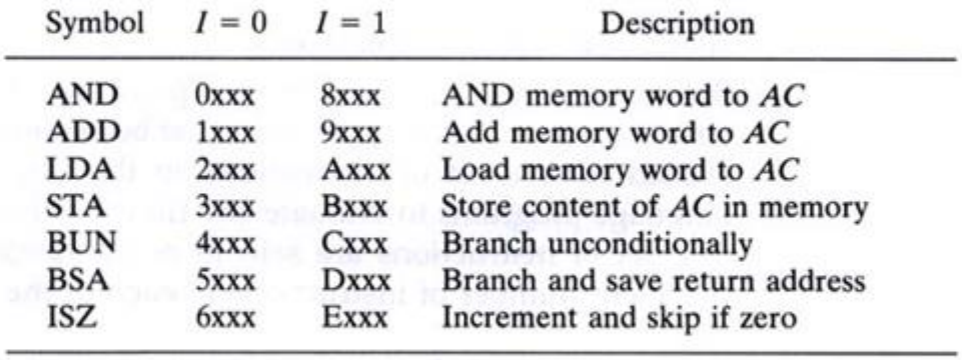
	r:	$SC \leftarrow 0$	Clear SC
CLA	rB_{11}:	$AC \leftarrow 0$	Clear AC
CLE	rB_{10}:	$E \leftarrow 0$	Clear E
CMA	rB_9:	$AC \leftarrow \overline{AC}$	Complement AC
CME	rB_8:	$E \leftarrow \overline{E}$	Complement E
CIR	rB_7:	$AC \leftarrow \text{shr } AC, AC(15) \leftarrow E, E \leftarrow AC(0)$	Circulate right
CIL	rB_6:	$AC \leftarrow \text{shl } AC, AC(0) \leftarrow E, E \leftarrow AC(15)$	Circulate left
INC	rB_5:	$AC \leftarrow AC + 1$	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

5.6 Memory reference instructions

- AND: and to AC
  - Direct: 0000, Indirect: 1000
  - D_0T_4 : DR ← M[AR]
  - D_0T_5 : AC ← AC ∧ DR, SC ← 0
- ADD: add to AC
  - Direct: 0001, Indirect: 1001
  - D_1T_4 : DR ← M[AR]
  - D_1T_5 : AC ← AC + DR, E ← C_{out}, SC ← 0
- LDA: load to AC
  - D_2T_4 : DR ← M[AR]
  - D_2T_5 : AC ← DR, SC ← 0
- STA: store AC
  - D_3T_4 : M[AR] ← AC, SC ← 0

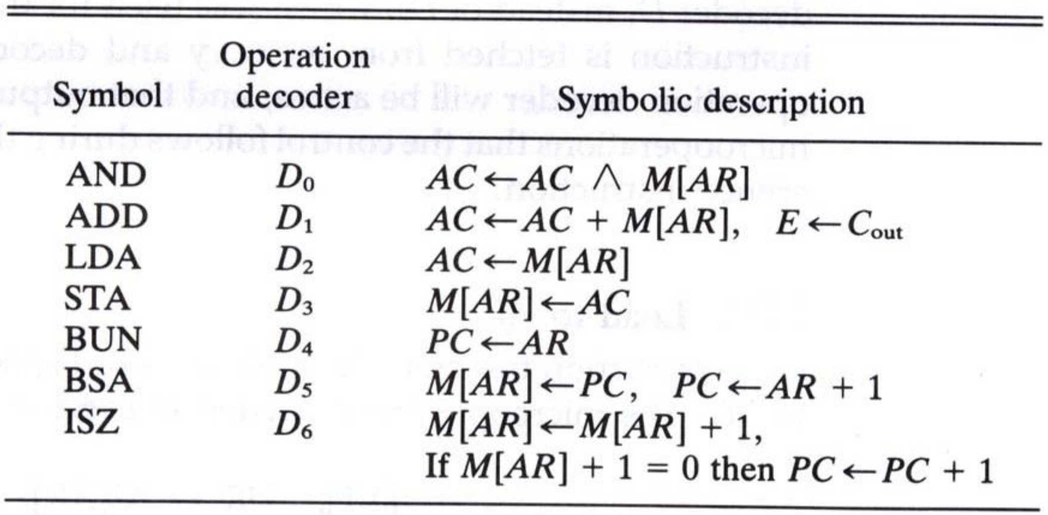


(a) Memory – reference instruction



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

TABLE 5-4 Memory-Reference Instructions

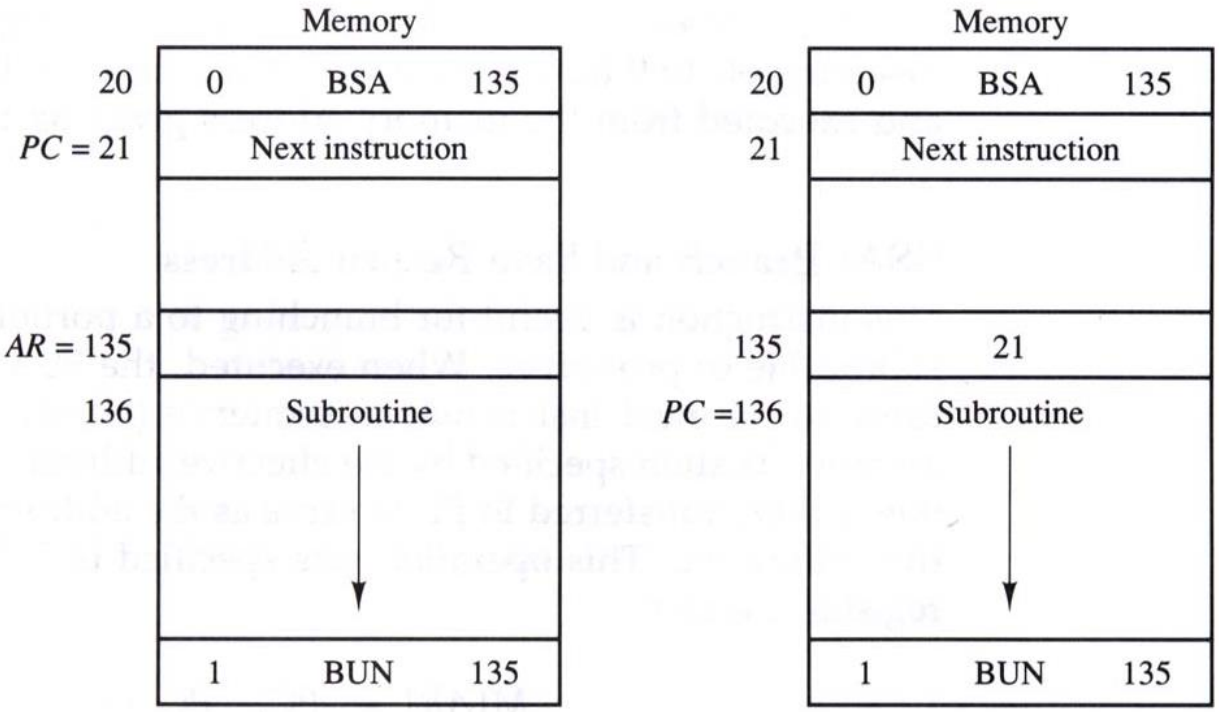


Symbol	Operation decoder	Symbolic description
AND	D_0	$AC \leftarrow AC \wedge M[AR]$
ADD	D_1	$AC \leftarrow AC + M[AR], \quad E \leftarrow C_{out}$
LDA	D_2	$AC \leftarrow M[AR]$
STA	D_3	$M[AR] \leftarrow AC$
BUN	D_4	$PC \leftarrow AR$
BSA	D_5	$M[AR] \leftarrow PC, \quad PC \leftarrow AR + 1$
ISZ	D_6	$M[AR] \leftarrow M[AR] + 1$, If $M[AR] + 1 = 0$ then $PC \leftarrow PC + 1$

5.6 Memory reference instructions

- BSA: branch and save return address
  - D_5T_4 : M[AR] ← PC, AR ← AR + 1
  - D_5T_5 : PC ← AR, SC ← 0

Figure 5-10 Example of BSA instruction execution.



M[135]←21, PC←135+1

(a) Memory, PC, and AR at time T_4

(b) Memory and PC after execution

5.6 Memory reference instructions

- BUN: branch unconditionally
  - D_4T_4 : PC ← AR, SC ← 0

- ISZ: increment and skip if zero
  - D_6T_4 : DR ← M[AR]
  - D_6T_5 : DR ← DR + 1
  - D_6T_6 : M[AR] ← DR, if (DR = 0) then (PC ← PC + 1), SC ← 0

5.6 Memory reference instructions

- Control flowchart

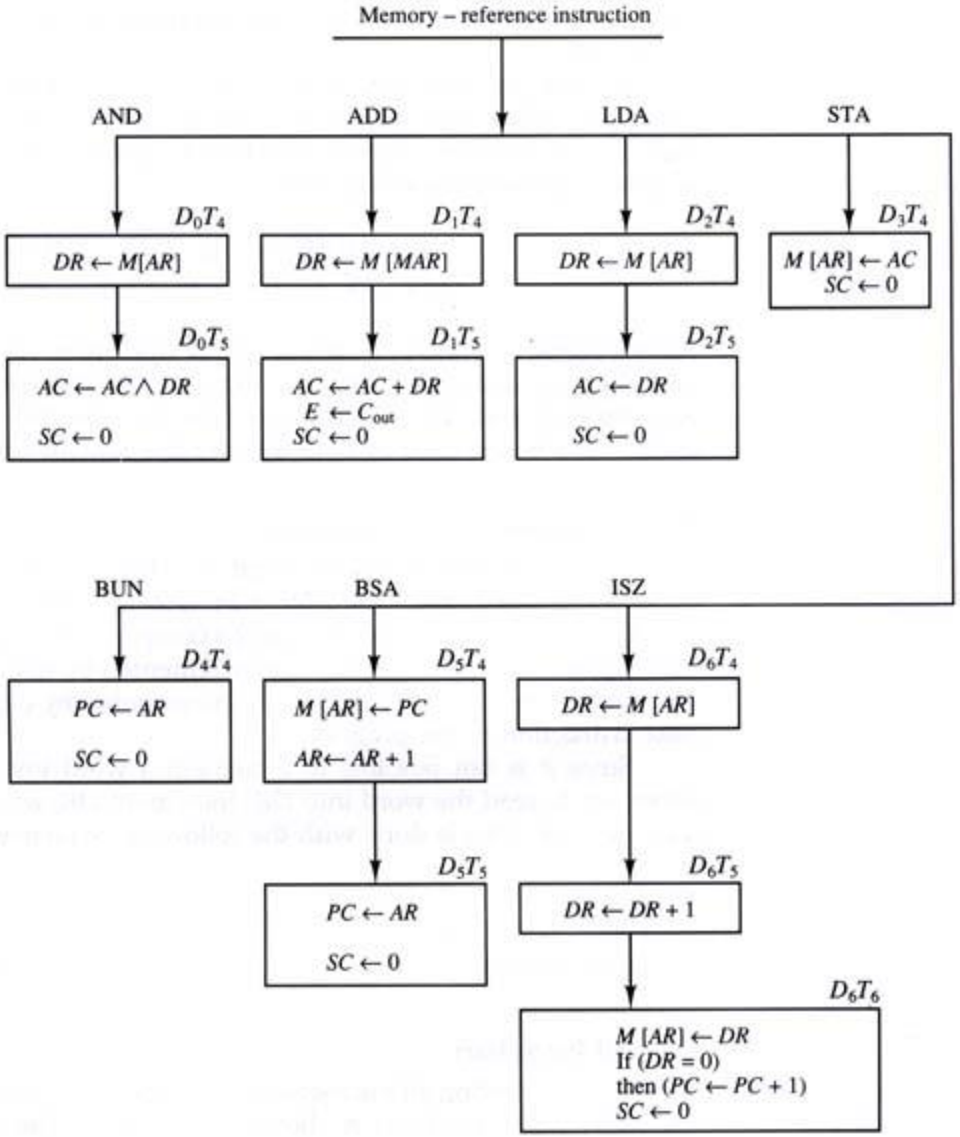
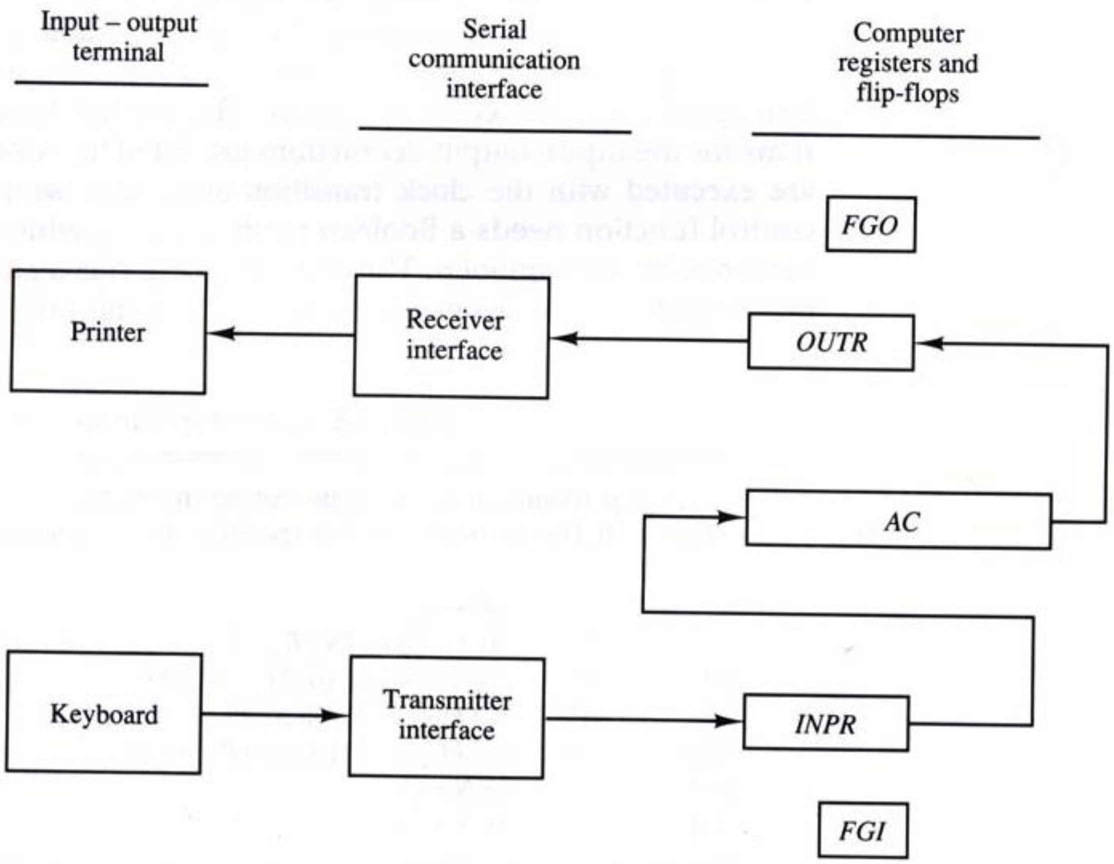


Figure 5-11 Flowchart for memory-reference instructions.

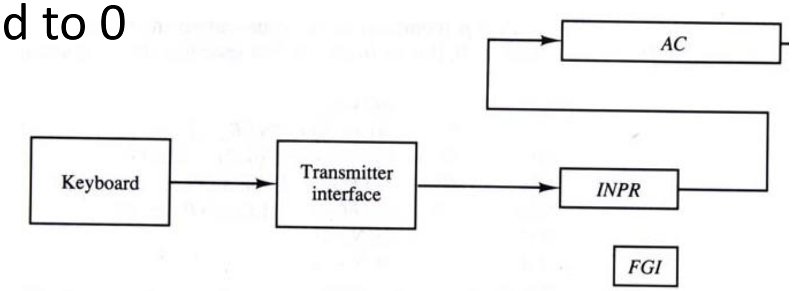
5.7 Input-output and interrupt

Figure 5-12 Input-output configuration.

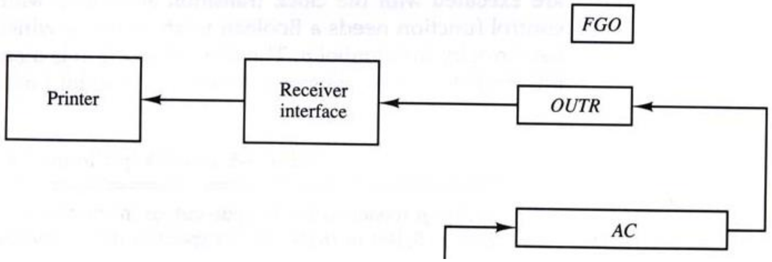


5.7 Input-output and interrupt

- Initially, FGI is set to 0
  - A new data from input device → INPR, FGI is set to 1
  - The data in INPR → AC, FGI is cleared to 0



- Initially, FGO is set to 1
  - A new data from AC → OUTR, FGO is cleared to 0
  - The data in OUTR → output device, FGO is set to 1



FGI/FGO = 1

New data ready for input to/output from computer (AC)

5.7 Input-output and interrupt

- Input-output instructions

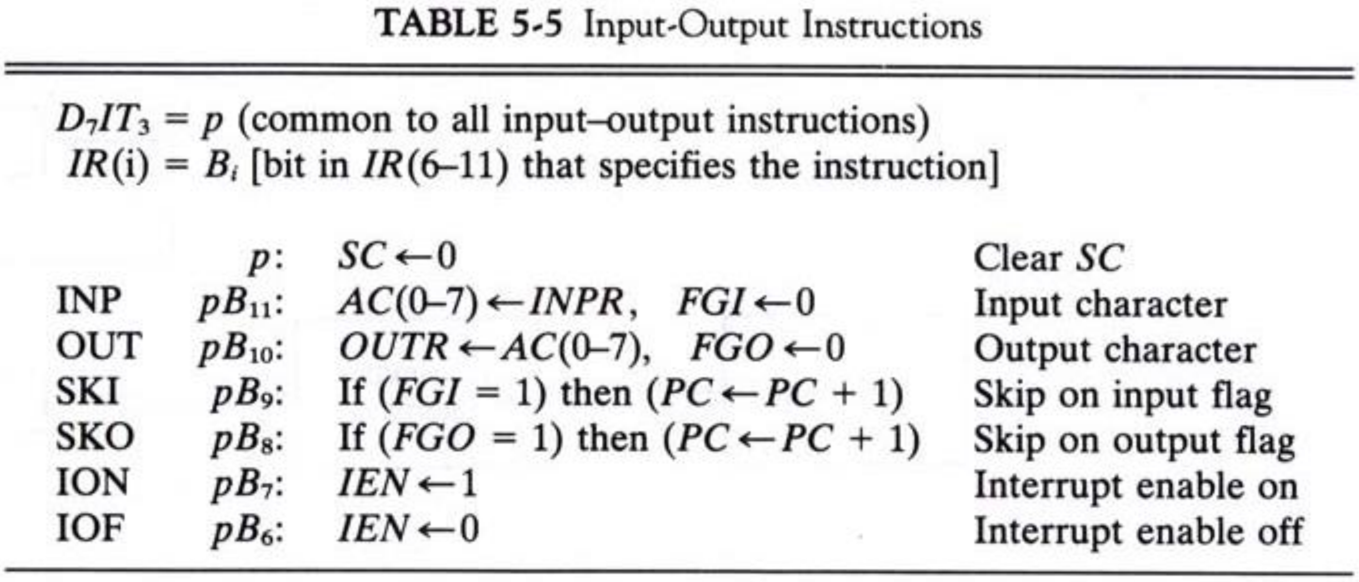


TABLE 5-5 Input-Output Instructions

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

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

5.7 Input-output and interrupt

- Program controlled transfer

```
        …
Loop: SKI
        BUN   Loop      ← waste a lot of time
        INP               in checking the flag
        …
```

- Program interrupt
  - Let the external device inform the computer when it is ready for transfer
  - Does not check the flags while the computer is running a program
  - Interrupt flip-flop R
    - When R = 0 the computer goes through an instruction cycle
    - If R = 1 it goes through an interrupt cycle
      - Return address is stored in a specific location

5.7 Input-output and interrupt

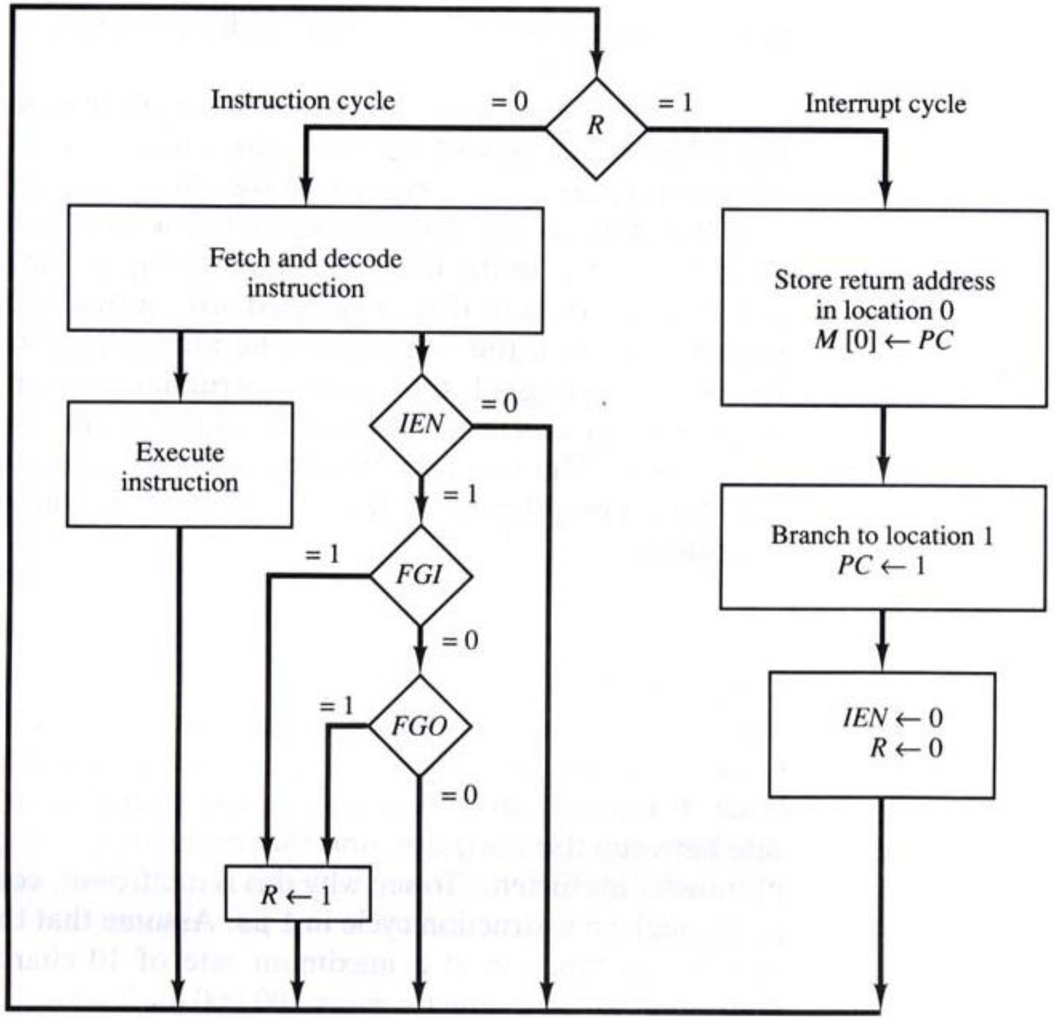


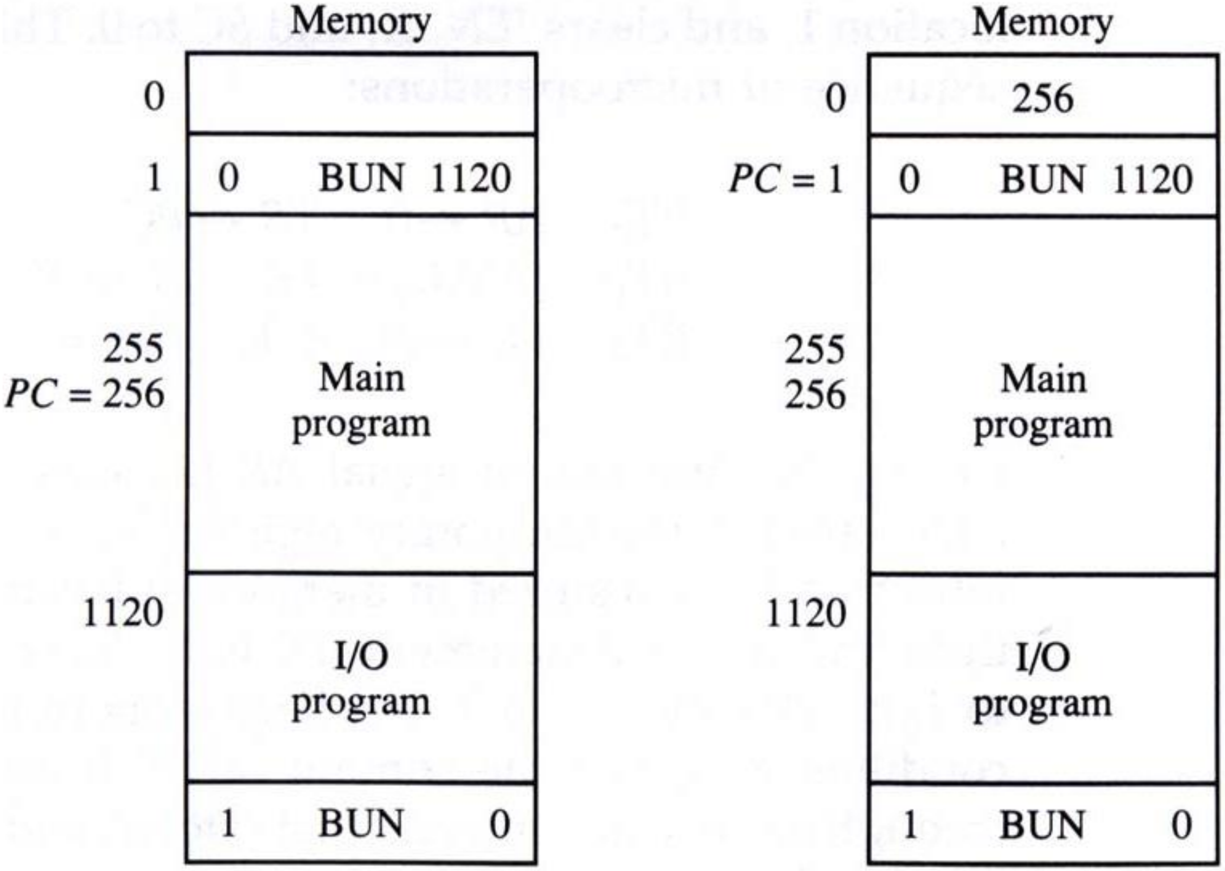
Figure 5-13 Flowchart for interrupt cycle.

5.7 Input-output and interrupt

- Interrupt cycle is a hardware implementation of a branch and save return address operation
- Interrupt cycle
  - (i) store return address in location 0, M[0] ← PC
  - (ii) branch to location 0, PC ← 1
  - (iii) IEN ← 0, R ← 0

5.7 Input-output and interrupt

Figure 5-14 Demonstration of the interrupt cycle.



(a) Before interrupt

(b) After interrupt cycle

5.7 Input-output and interrupt

- Condition for setting R
  - $T_0'T_1'T_2'(IEN)(FGI + FGO) : R \leftarrow 1$
- This procedure can be done with the following sequence of micro-operations:
  - $RT_0 : AR \leftarrow 0, TR \leftarrow PC$
  - $RT_1 : M[AR] \leftarrow TR, PC \leftarrow 0$
  - $RT_2 : PC \leftarrow PC + 1, IEN \leftarrow 0, R \leftarrow 0, SC \leftarrow 0$
- Modified fetch phase: fetch and decode phase use the three control signals
  - $R'T_0$, $R'T_1$, $R'T_2$ instead of T_0, T_1, T_2

5.8 Complete computer description

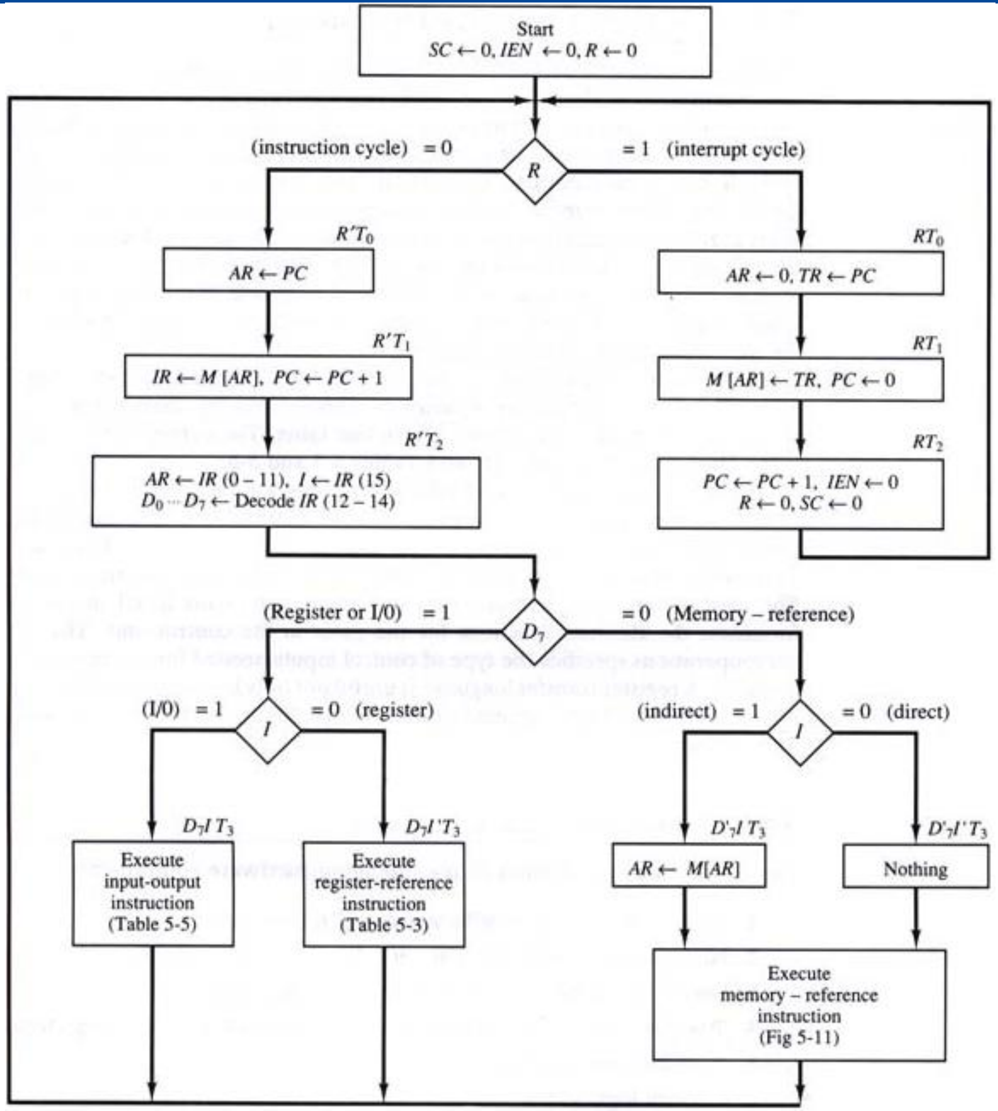


Figure 5-15 Flowchart for computer operation.

5.8 Complete computer description

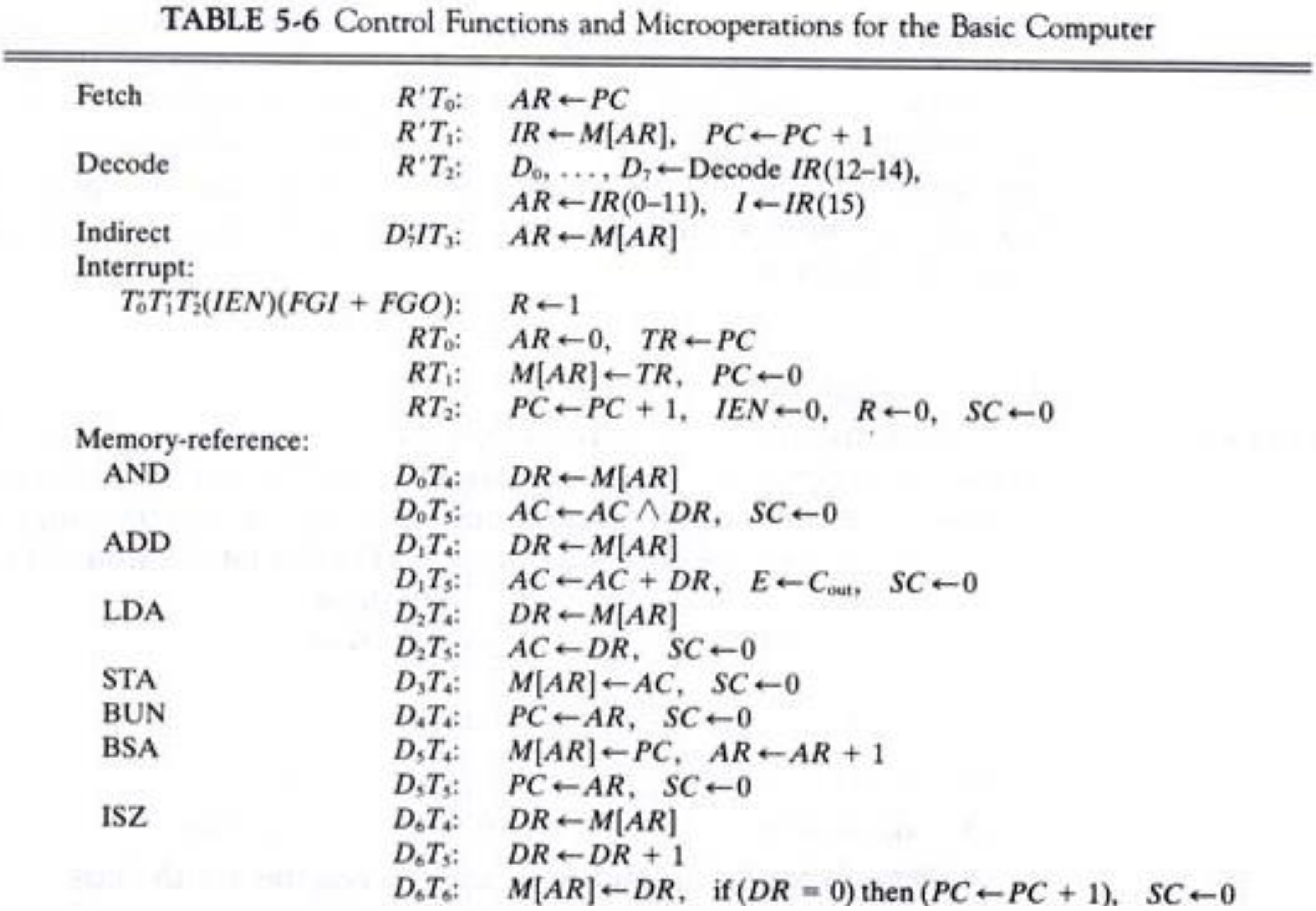
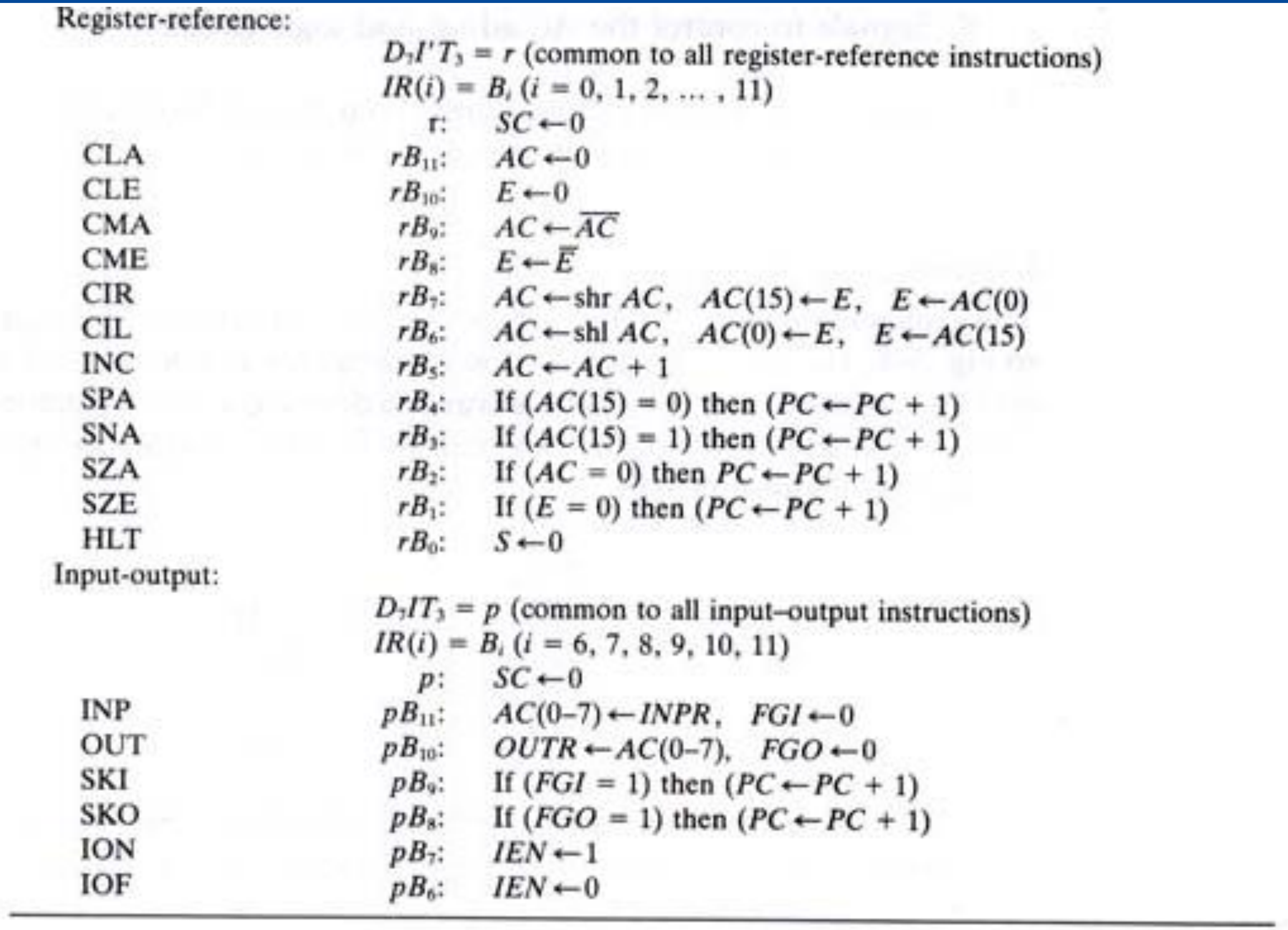


TABLE 5-6 Control Functions and Microoperations for the Basic Computer

Fetch	$R'T_0$:	$AR \leftarrow PC$
	$R'T_1$:	$IR \leftarrow M[AR],\quad PC \leftarrow PC + 1$
Decode	$R'T_2$:	$D_0, \ldots, D_7 \leftarrow$ Decode $IR(12\text{–}14),$ $AR \leftarrow IR(0\text{–}11),\quad I \leftarrow IR(15)$
Indirect	$D_7'IT_3$:	$AR \leftarrow M[AR]$
Interrupt:		
$T_0'T_1'T_2'(IEN)(FGI + FGO)$:		$R \leftarrow 1$
	RT_0:	$AR \leftarrow 0,\quad TR \leftarrow PC$
	RT_1:	$M[AR] \leftarrow TR,\quad PC \leftarrow 0$
	RT_2:	$PC \leftarrow PC + 1,\quad IEN \leftarrow 0,\quad R \leftarrow 0,\quad SC \leftarrow 0$
Memory-reference:		
AND	D_0T_4:	$DR \leftarrow M[AR]$
	D_0T_5:	$AC \leftarrow AC \wedge DR,\quad SC \leftarrow 0$
ADD	D_1T_4:	$DR \leftarrow M[AR]$
	D_1T_5:	$AC \leftarrow AC + DR,\quad E \leftarrow C_{out},\quad SC \leftarrow 0$
LDA	D_2T_4:	$DR \leftarrow M[AR]$
	D_2T_5:	$AC \leftarrow DR,\quad SC \leftarrow 0$
STA	D_3T_4:	$M[AR] \leftarrow AC,\quad SC \leftarrow 0$
BUN	D_4T_4:	$PC \leftarrow AR,\quad SC \leftarrow 0$
BSA	D_5T_4:	$M[AR] \leftarrow PC,\quad AR \leftarrow AR + 1$
	D_5T_5:	$PC \leftarrow AR,\quad SC \leftarrow 0$
ISZ	D_6T_4:	$DR \leftarrow M[AR]$
	D_6T_5:	$DR \leftarrow DR + 1$
	D_6T_6:	$M[AR] \leftarrow DR,\quad$ if $(DR = 0)$ then $(PC \leftarrow PC + 1),\quad SC \leftarrow 0$

5.8 Complete computer description



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:	$AC \leftarrow \text{shr } AC, \quad AC(15) \leftarrow E, \quad E \leftarrow AC(0)$
CIL	rB_6:	$AC \leftarrow \text{shl } AC, \quad AC(0) \leftarrow E, \quad E \leftarrow AC(15)$
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}:	$AC(0\text{–}7) \leftarrow INPR, \quad FGI \leftarrow 0$
OUT	pB_{10}:	$OUTR \leftarrow AC(0\text{–}7), \quad FGO \leftarrow 0$
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:	$IEN \leftarrow 0$

5.9 Design of basic computer

- The basic computer consists of the following components
  - A memory unit: 4096 (=2^{12}) words x 16 bit
  - Nine register: AR, PC, DR, AC, IR, TR, OUTR, INPR, SC
  - Seven flip-flop: I, S, E, R, IEN, FGI, FGO
  - Two decoders 3 x 8, 4 x 16
  - 16-bit common bus
  - Control logic gates
  - Adder and logic circuit

5.9 Design of basic computer

- Control functions for
  - Registers, Memory read/write
  - Flip-flops (set/clear/complement)
  - Common bus (S_0, S_1, S_2)
  - Adder and logic circuit

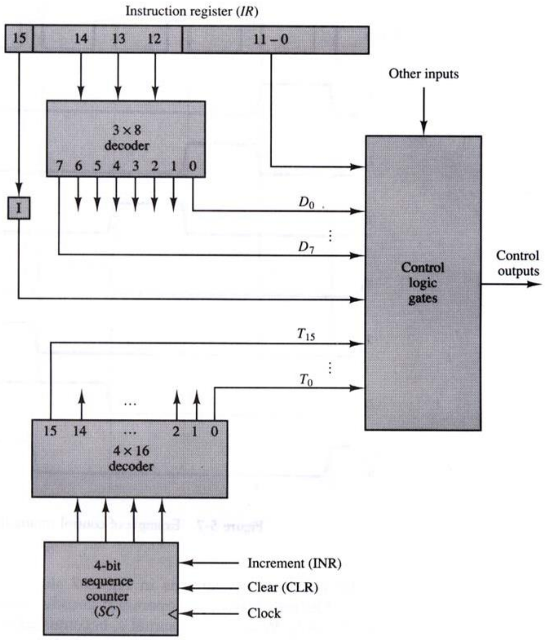


Figure 5-6 Control unit of basic computer.

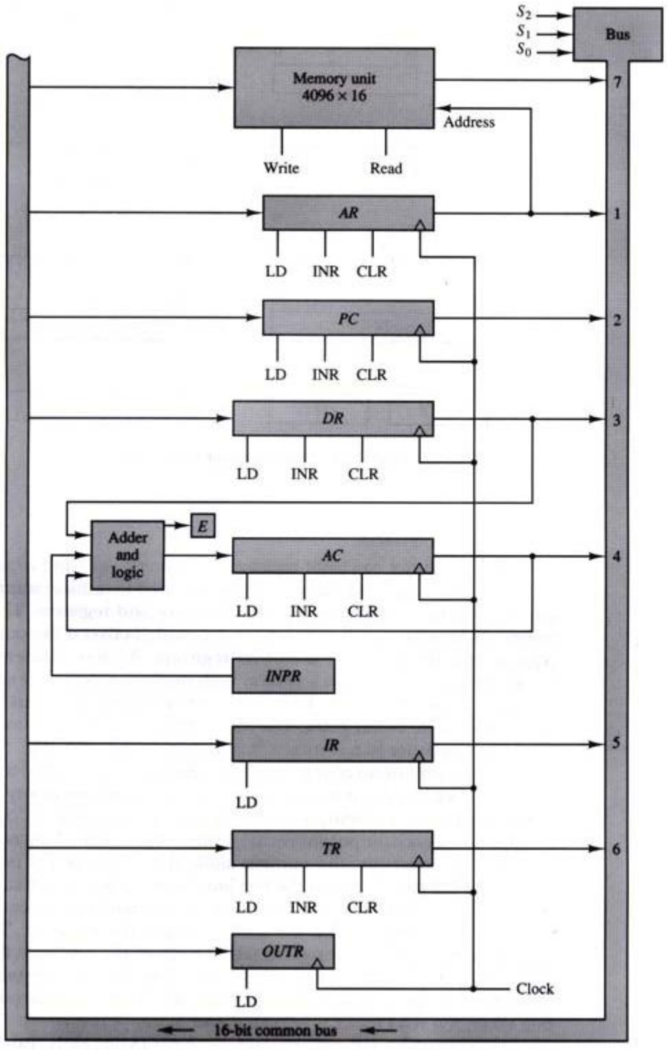


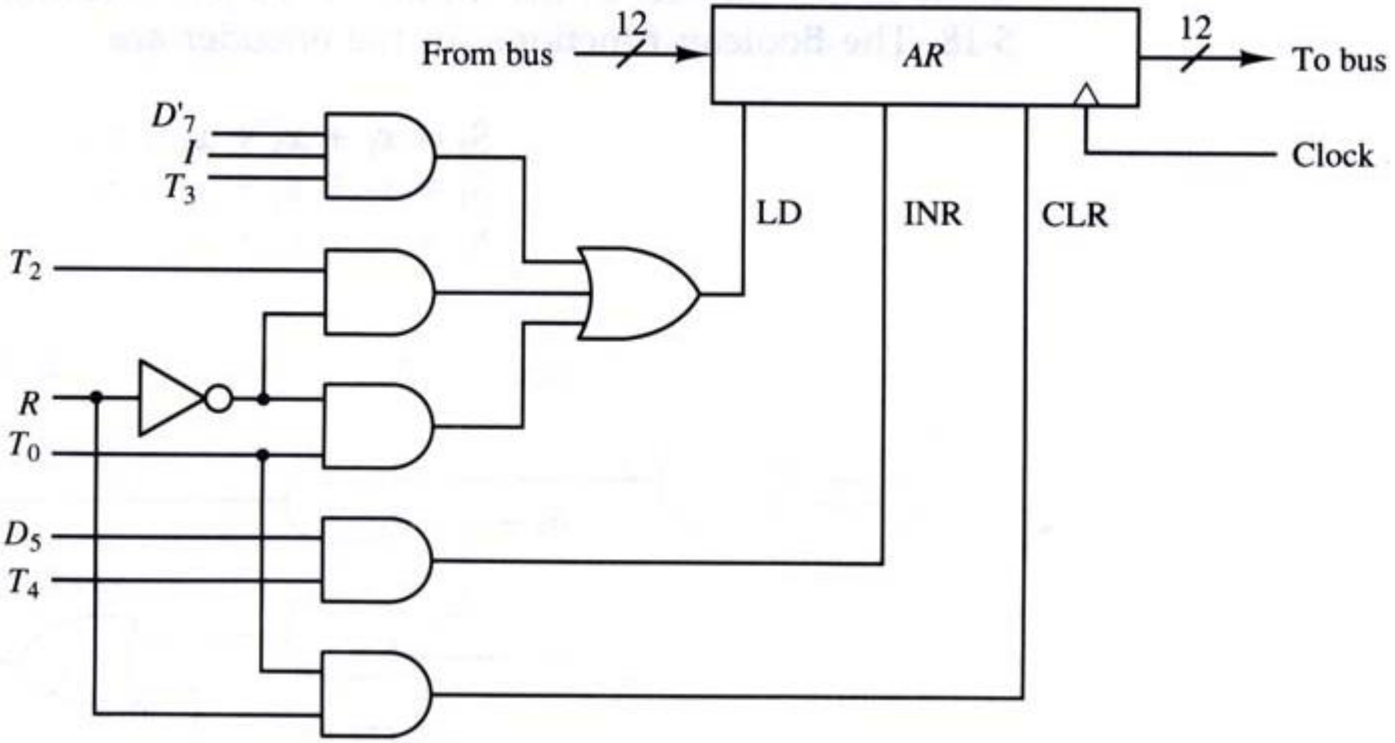
Figure 5-4 Basic computer registers connected to a common bus.

5.9 Design of basic computer

- Control of registers; ex) AR register
  - Statements that change the content of AR
    - $R'T_0$: AR ← PC
    - $R'T_2$: AR ← IR(0 - 11)
    - $D_7'IT_3$: AR ← M[AR]
    - RT_0 : AR ← 0
    - D_5T_4 : AR ← AR + 1
  - Control functions
    - $LD(AR) = R'T_0 + R'T_2 + D_7'IT_3$
    - $CLR(AR) = RT_0$
    - $INR(AR) = D_5T_4$

5.9 Design of basic computer

Figure 5-16 Control gates associated with *AR*.



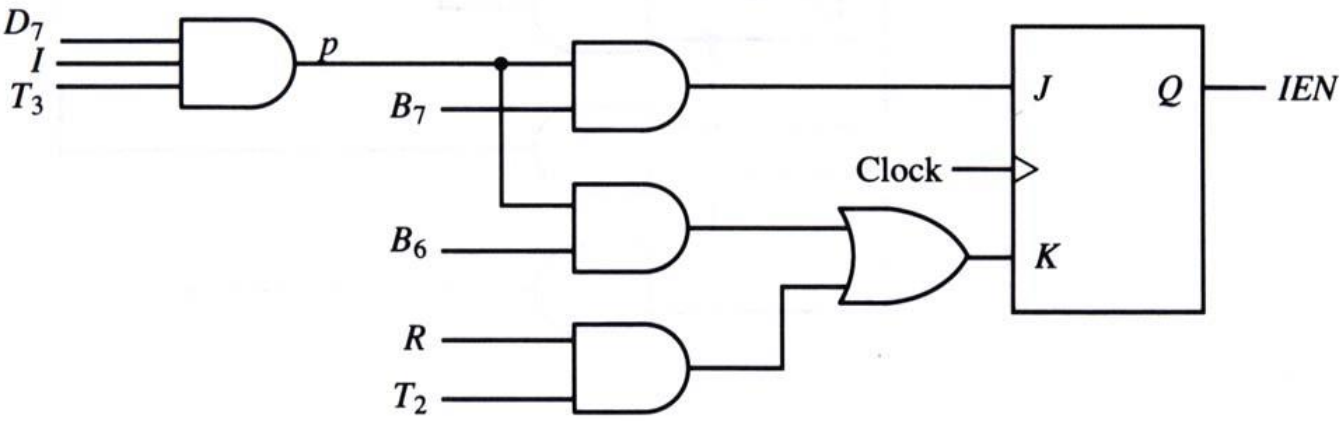
5.9 Design of basic computer

- Control function for reading input of memory
  - Find the statements including “← M[AR]”
  - Read = $R'T1 + D_7'IT_3 + (D_0 + D_1 + D_2 + D_6)T_4$

5.9 Design of basic computer

- Control of single flip-flops
  - Control of IEN
    - pB_7 : IEN ← 1
    - pB_6 : IEN ← 0
    - RT_2 : IEN ← 0

Figure 5-17 Control inputs for *IEN*.



5.9 Design of basic computer

- Control of common bus

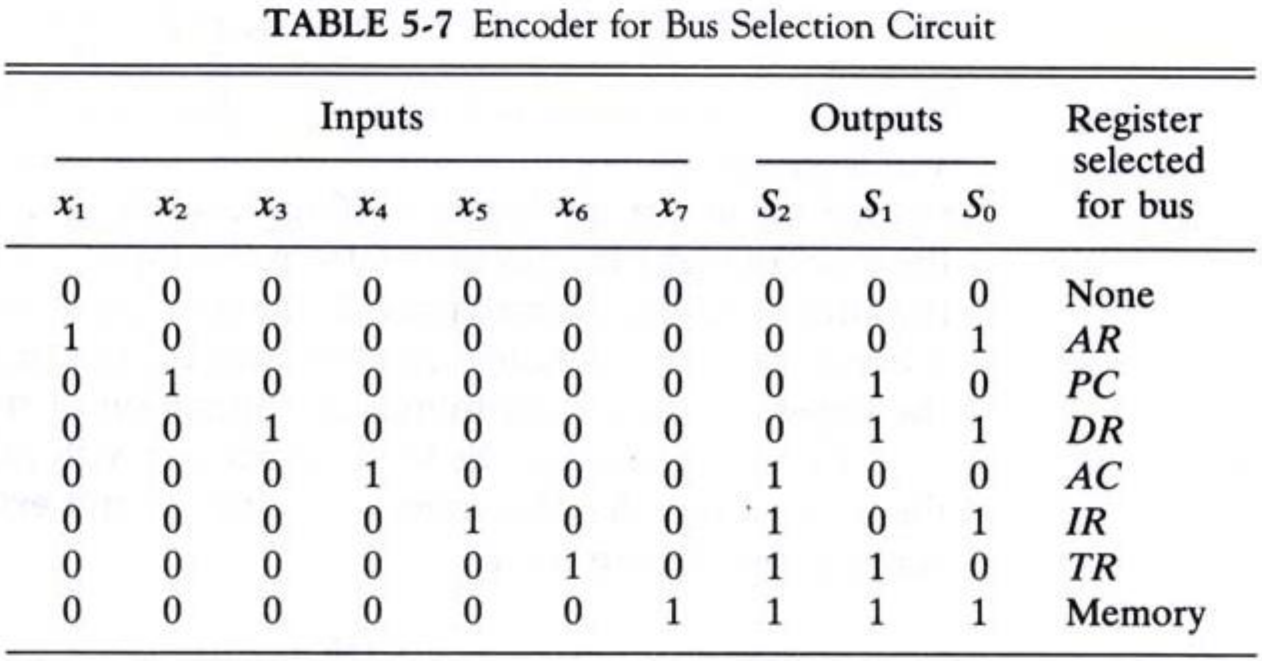


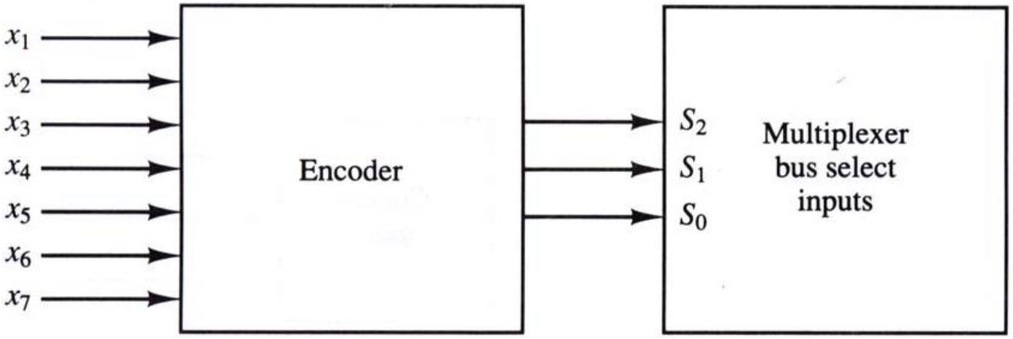
TABLE 5-7 Encoder for Bus Selection Circuit

Inputs							Outputs			Register selected for bus
x_1	x_2	x_3	x_4	x_5	x_6	x_7	S_2	S_1	S_0	
0	0	0	0	0	0	0	0	0	0	None
1	0	0	0	0	0	0	0	0	1	*AR*
0	1	0	0	0	0	0	0	1	0	*PC*
0	0	1	0	0	0	0	0	1	1	*DR*
0	0	0	1	0	0	0	1	0	0	*AC*
0	0	0	0	1	0	0	1	0	1	*IR*
0	0	0	0	0	1	0	1	1	0	*TR*
0	0	0	0	0	0	1	1	1	1	Memory

5.9 Design of basic computer

- Boolean function for the encoder
  - S0 = $x_1 + x_3 + x_5 + x_7$
  - S1 = $x_2 + x_3 + x_6 + x_7$
  - S2 = $x_4 + x_5 + x_6 + x_7$

Figure 5-18 Encoder for bus selection inputs.



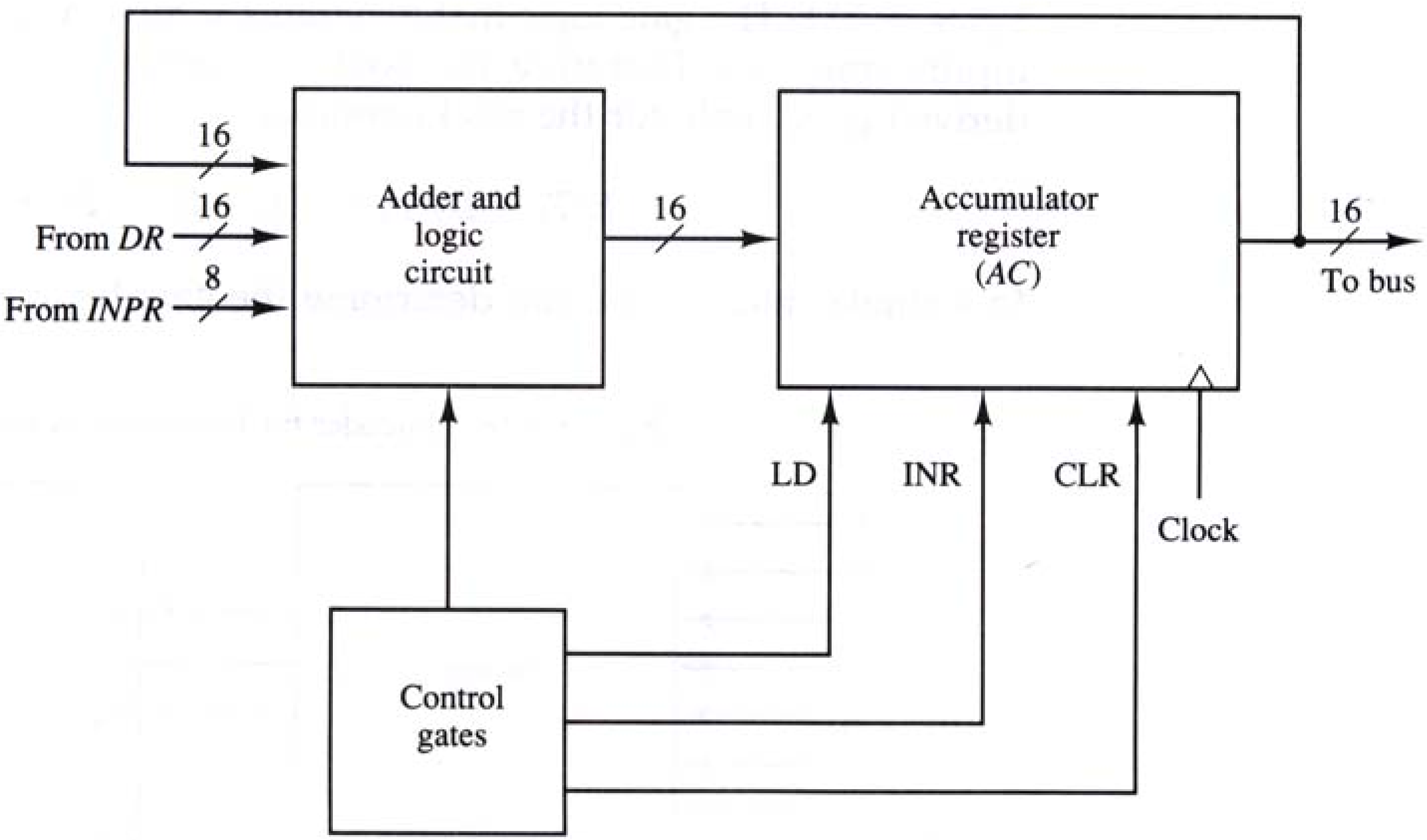
5.9 Design of basic computer

- Ex) to find the logic that makes x_1 = 1, find all the register transfer statements that have AR as a source
  - D_4T_4 : PC ← AR
  - D_5T_5 : PC ← AR
  - $x_1 = D_4T_4 + D_5T_5$
- Select memory as a source = read memory
  - $x_7 = R'T1 + D_7'IT_3 + (D_0 + D_1 + D_2 + D_6)T_4$

5.10 Design of accumulator logic

- The statements that change the content of AC

Figure 5-19 Circuits associated with AC.



5.10 Design of accumulator logic

- The statements that change the content of AC

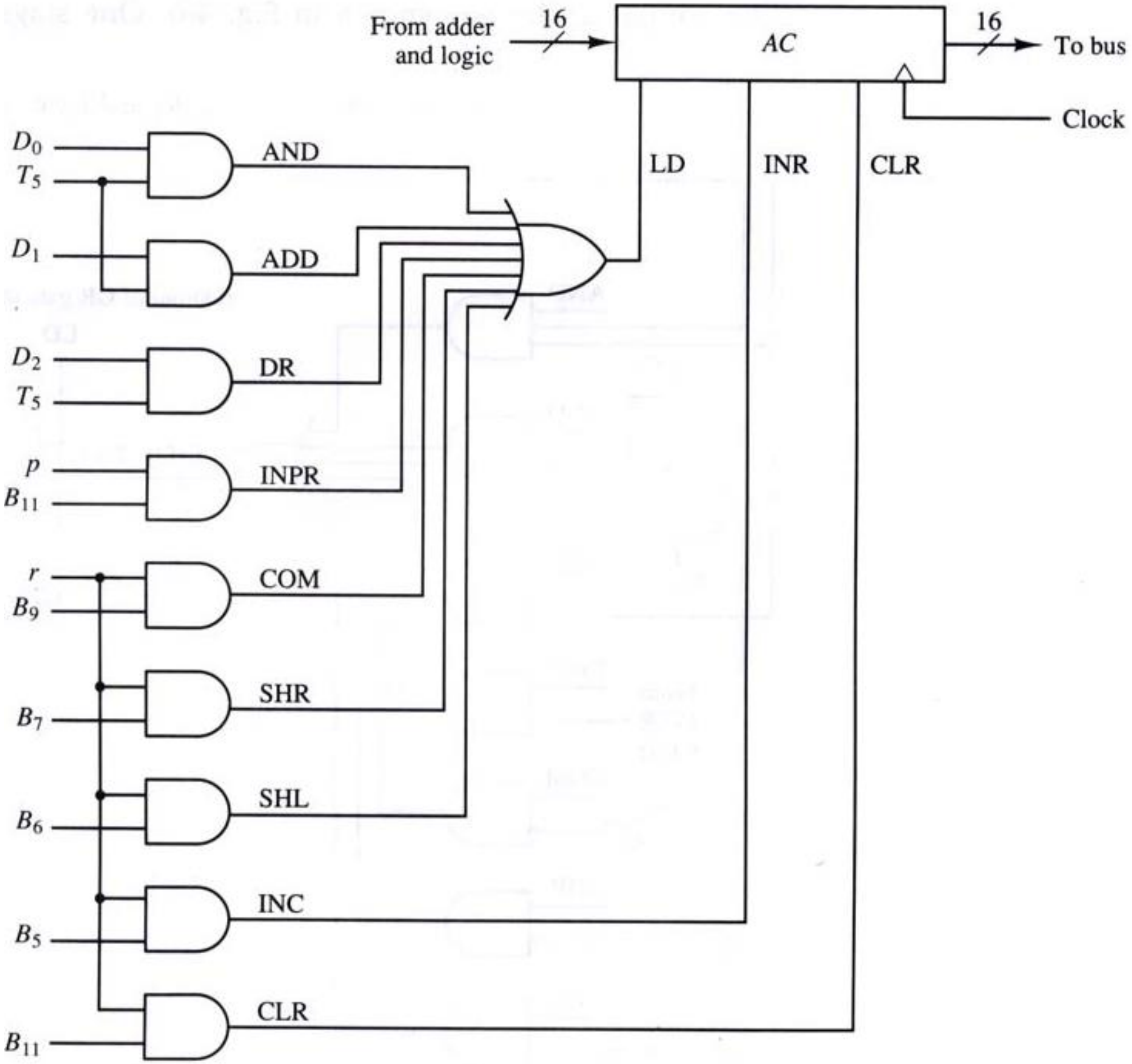
$$
\begin{aligned}
&D_0T_5 : AC \leftarrow AC \wedge DR \\
&D_1T_5 : AC \leftarrow AC + DR \\
&D_2T_5 : AC \leftarrow DR \\
&pB_{11} : AC(0-7) \leftarrow INPR \\
&rB_9 : AC \leftarrow (AC)' \\
&rB_7 : AC \leftarrow \text{shr } AC,\ AC(15) \leftarrow E \\
&rB_6 : AC \leftarrow \text{shl } AC,\ AC(0) \leftarrow E \\
&rB_{11} : AC \leftarrow 0 \\
&rB_5 : Ac \leftarrow AC + 1
\end{aligned}
$$

5.10 Design of accumulator logic

- Control of AC register

D_0T_5 : AC ← AC ∧ DR
D_1T_5 : AC ← AC + DR
D_2T_5 : AC ← DR
pB_{11} : AC(0 – 7) ← INPR
rB_9 : AC ← (AC)'
rB_7 : AC ← shr AC, AC(15) ← E
rB_6 : AC ← shl AC, AC(0) ← E
rB_{11} : AC ← 0
rB_5 : Ac ← AC + 1

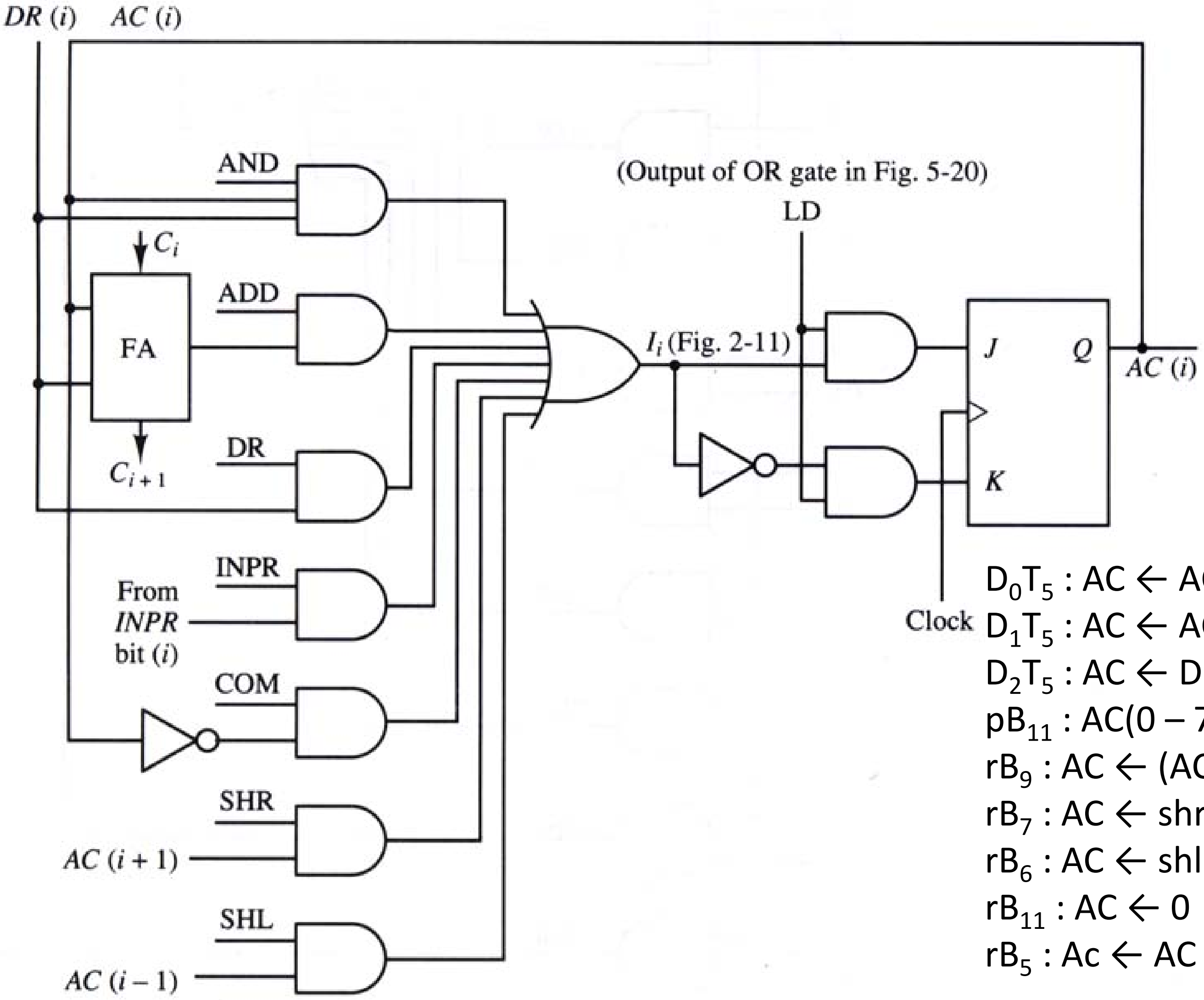
Figure 5-20 Gate structure for controlling the LD, INR, and CLR of AC.



5.10 Design of accumulator logic

- Adder and logic circuit

Figure 5-21 One stage of adder and logic circuit.



D_0T_5 : AC ← AC ∧ DR

D_1T_5 : AC ← AC + DR

D_2T_5 : AC ← DR

pB_{11} : AC(0 – 7) ← INPR

rB_9 : AC ← (AC)’

rB_7 : AC ← shr AC, AC(15) ← E

rB_6 : AC ← shl AC, AC(0) ← E

rB_{11} : AC ← 0

rB_5 : Ac ← AC + 1

Problems

- 5-2, 5-3, 5-4, 5-6, 5-9, 5-10, 5-12,
- 5-13, 5-18, 5-19, 5-21, 5-22, 5-23