BASIC COMPUTER ORGANIZATION AND DESIGN

- **Instruction Codes**
- **Computer Registers**
- **Computer Instructions**
- **Timing and Control**
- **Instruction Cycle**
- **Register Reference Instructions**
- **Memory Reference Instructions**

INSTRUCTION CODES

- **Program:**

  A set of instructions that specify the *operations*, *operands*, and the *sequence* by which processing has to occur.

- **Instruction Code:**

  A group of bits that tell the computer to *perform a specific operation* (a sequence of micro-operation)

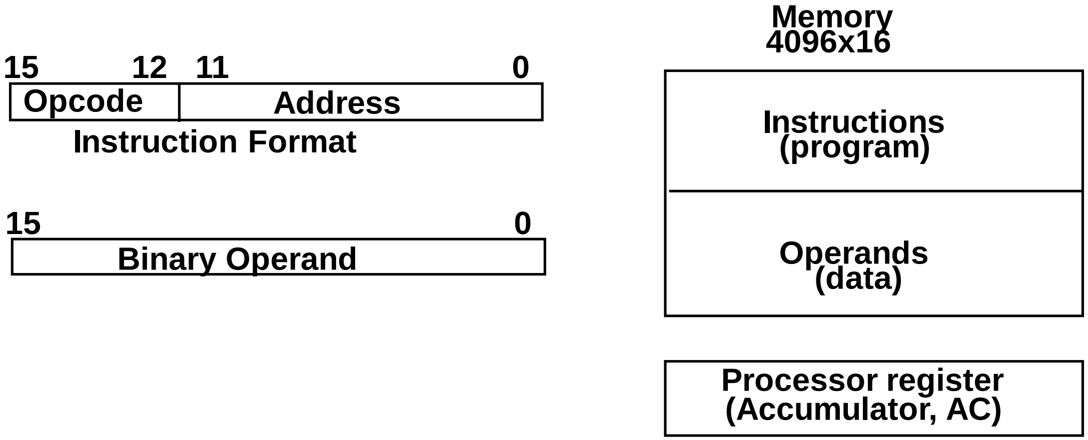
  --> *macro-operation*

  - usually divided into *operation code*, *operand address*, *addressing mode*, etc.

  - basic addressing modes

  Immediate*, *Direct*, *Indirect

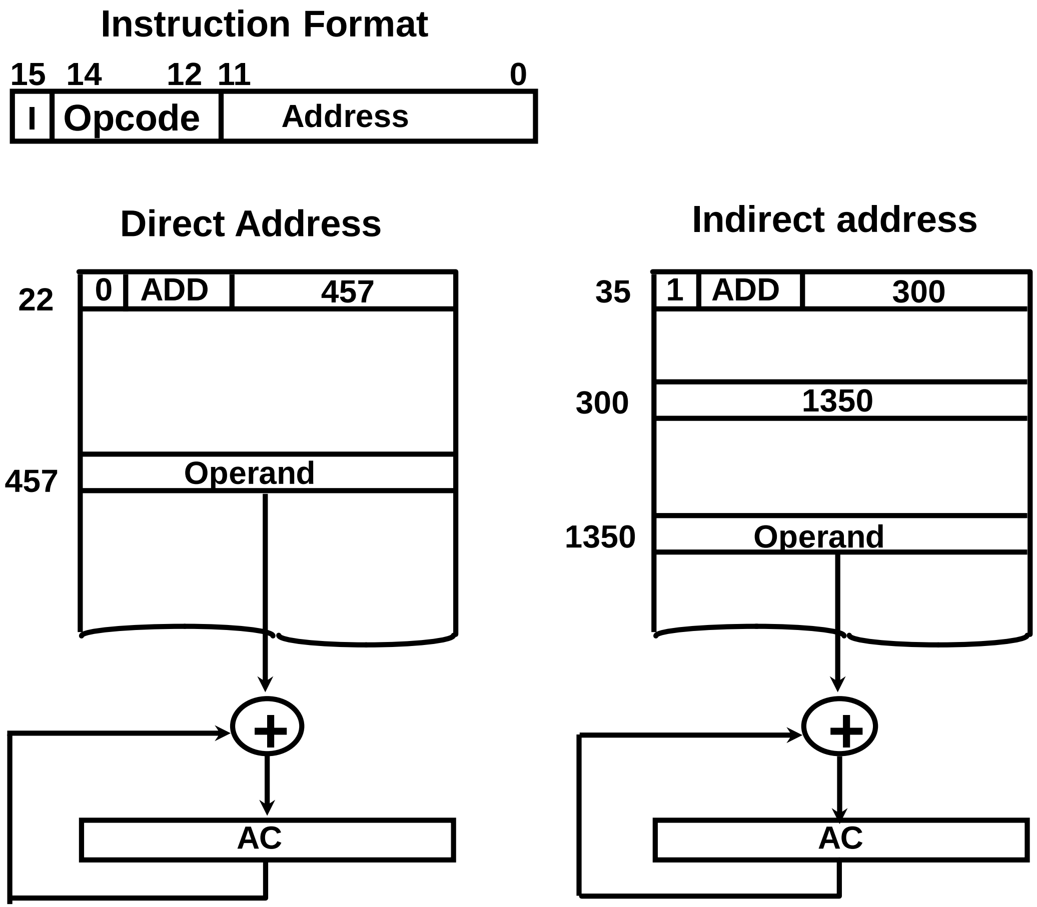
- **Simplest stored program organization**



Mano's Basic Computer

- Memory unit with 4096 l6-bit words
- Registers: *AR, PC, DR, AC, IR, TR, OUTR, INPR,* SC
- Flip-flops: I, S, *E, R, IEN, FGI, FGO*
- 3 x 8 op decoder and 4 x 16 timing decoder
- 16-bit common bus
- Control logic gates
- Adder and logic circuit connected to input of *AC*

INDIRECT ADDRESS

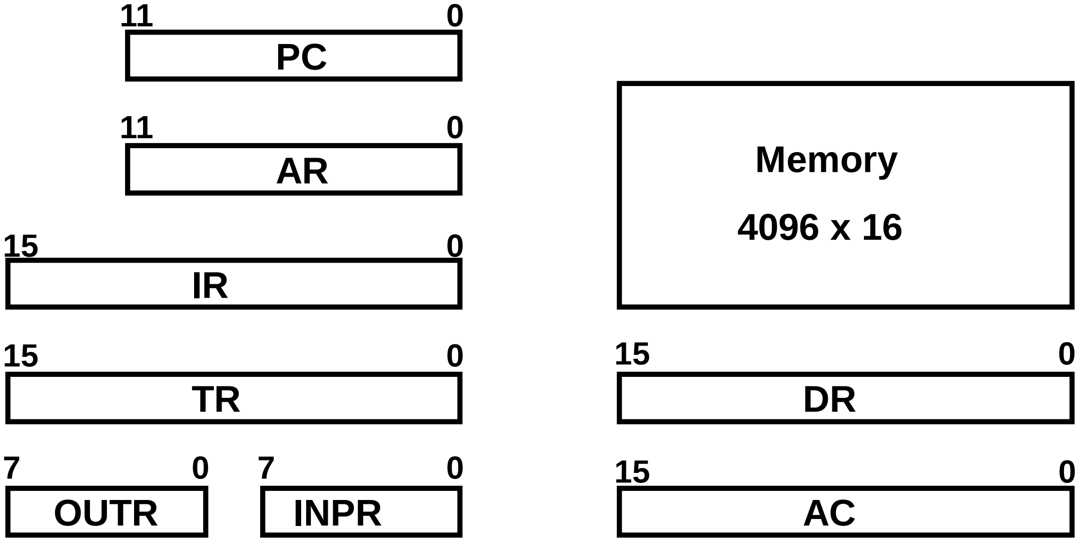


Effective Address(EA)

The address, that can be directly used without modification to access an operand for a computation-type instruction, or as the target address for a branch-type instruction

COMPUTER REGISTERS

Registers in the Basic Computer



List of BC Registers

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

Program Counter (PC)

- Holds memory address of next instruction to be executed.
- Next instruction is fetched after current instruction completes execution cycle.
- *PC* is incremented right after instruction is fetched from memory.
- *PC* value can be replaced by new address when executing a branch instruction

Register Control Inputs

- Load (LD)
- Increment (INR)
- Clear (CLR)

Common Bus

- Connects registers and memory
- Specific output selected by S2, S1 and S0.
- When register has length < 16 bits, high-order bus bits are set to 0.
- Register with LD enabled reads data from bus.
- When S2SlS0 = 111
  - Memory with Write enabled reads bus.
  - Memory with Read enabled puts data on bus.

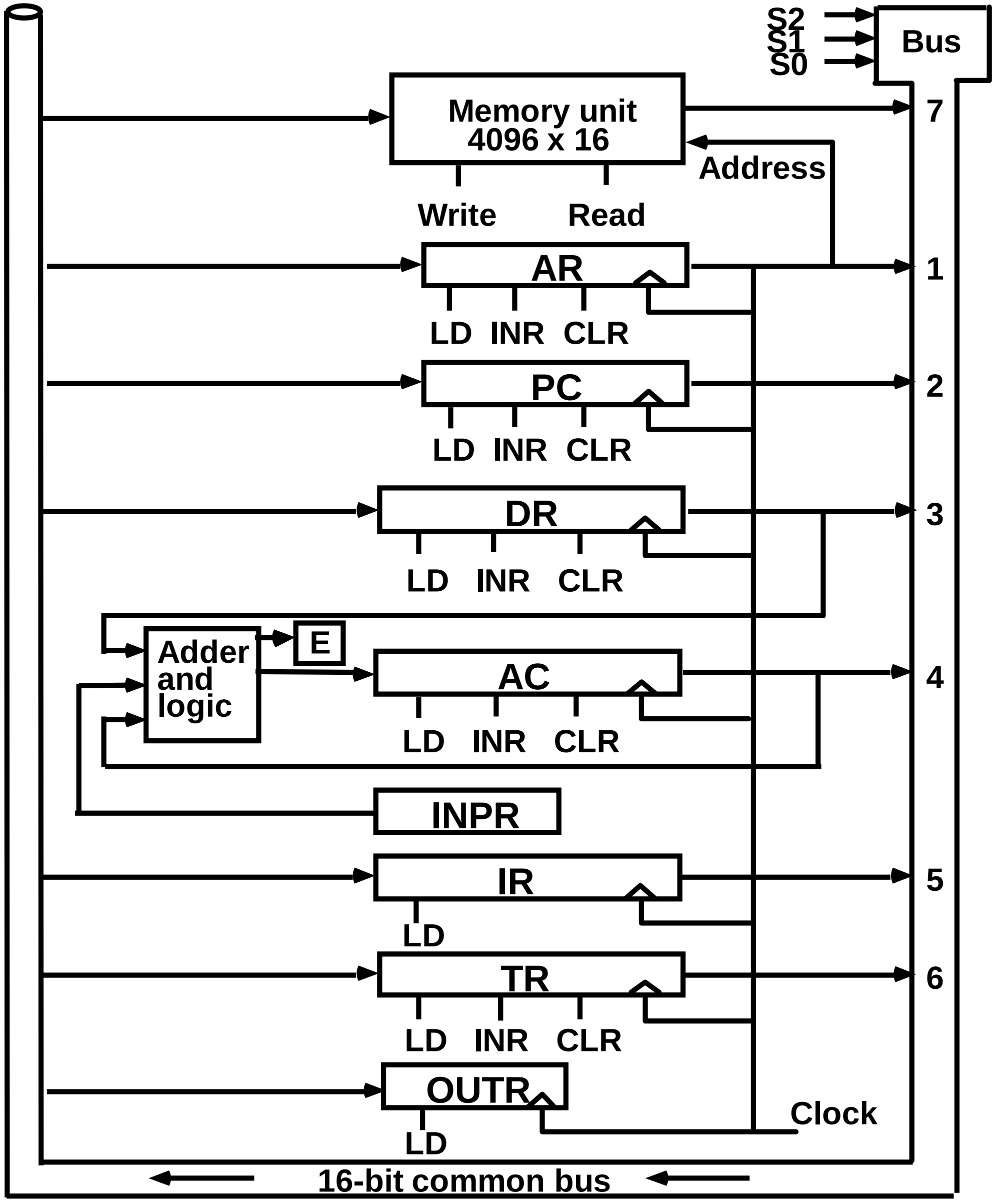
Address Register (AR)

- Always used to specify address within memory unit.
- Dedicated register eliminates need for separate address bus.
- Content of any register output connected to the bus can be written to memory.
- Any register input connected to bus can be target of memory read.
- As long as its LD is enabled.

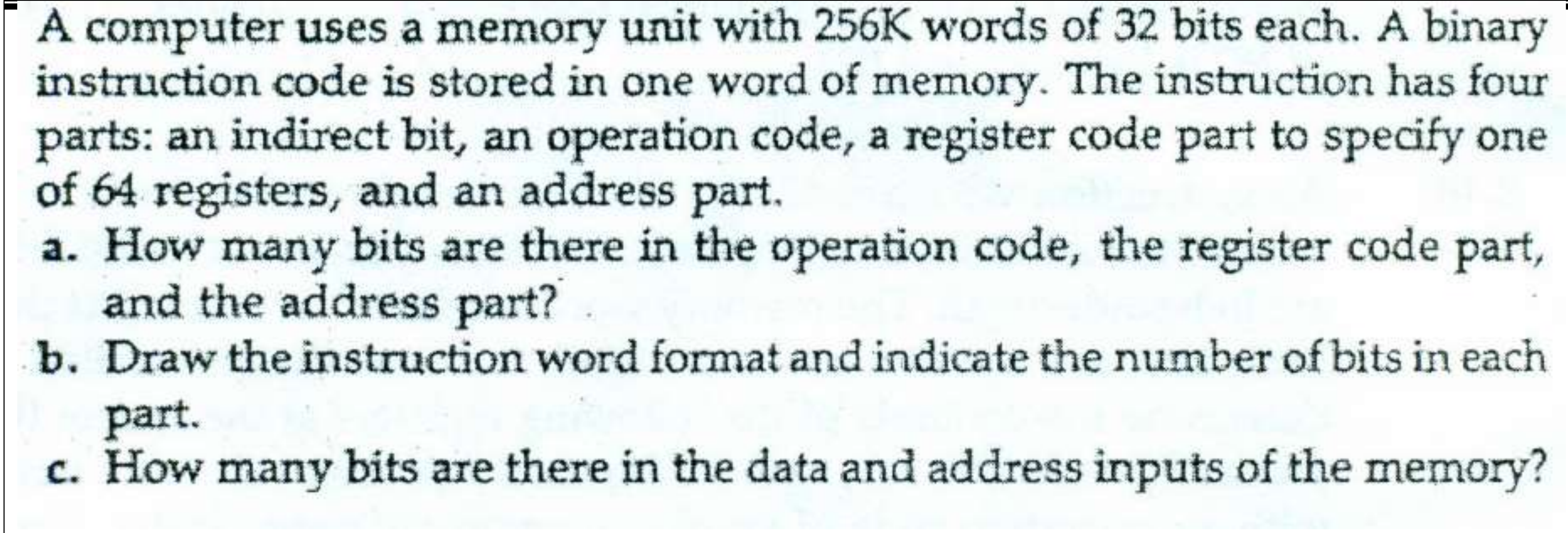
Accumulator (AC)

- Input comes from adder and logic circuit
- Adder and logic circuit
- Input
- 16-bit output of AC
- 16-bit data register *(DR)*
- 8-bit input register *(INPR)*
- Output
- 16-bit input of *AC*
- *E* flip-flop (extended AC bit. aka overflow)
- *DR* and *AC* input used for arithmetic and logic microoperations

COMMON BUS SYSTEM



Question



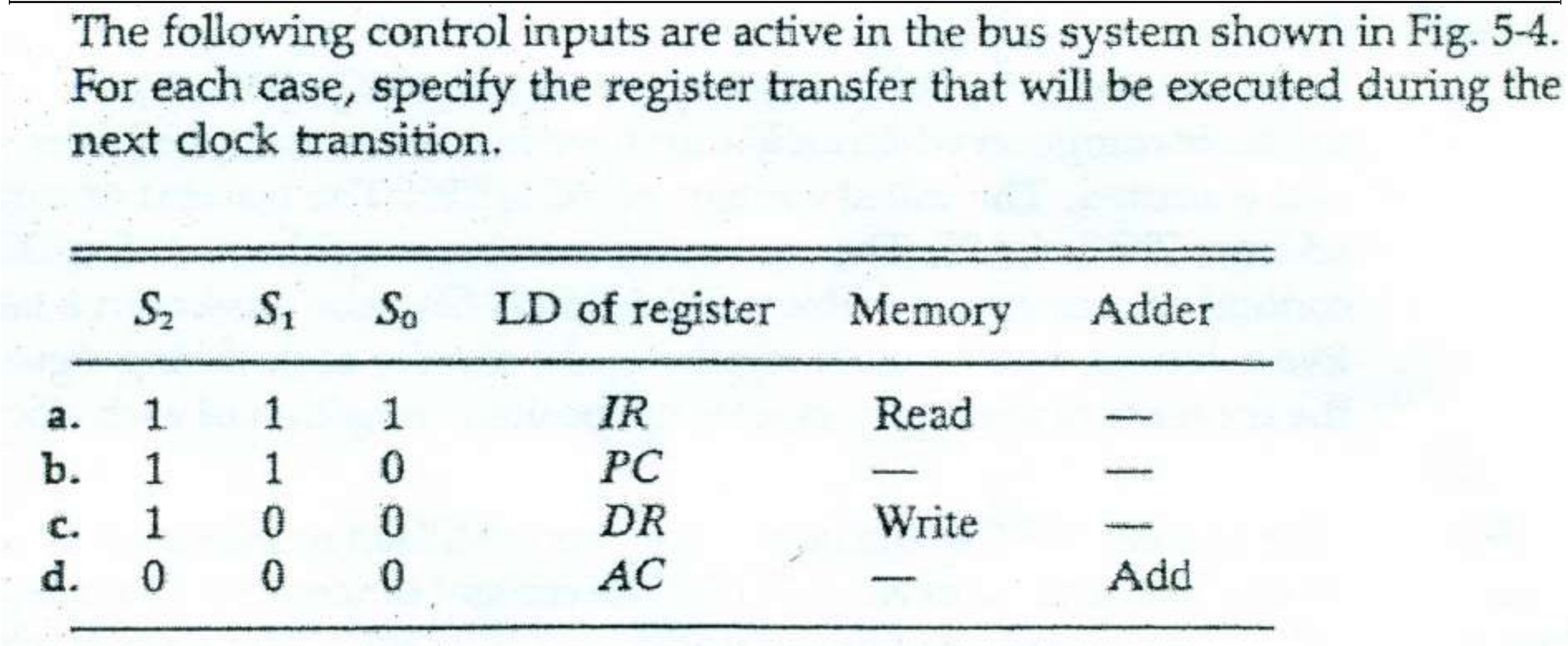
A computer uses a memory unit with 256K words of 32 bits each. A binary instruction code is stored in one word of memory. The instruction has four parts: an indirect bit, an operation code, a register code part to specify one of 64 registers, and an address part.

a. How many bits are there in the operation code, the register code part, and the address part?

b. Draw the instruction word format and indicate the number of bits in each part.

c. How many bits are there in the data and address inputs of the memory?

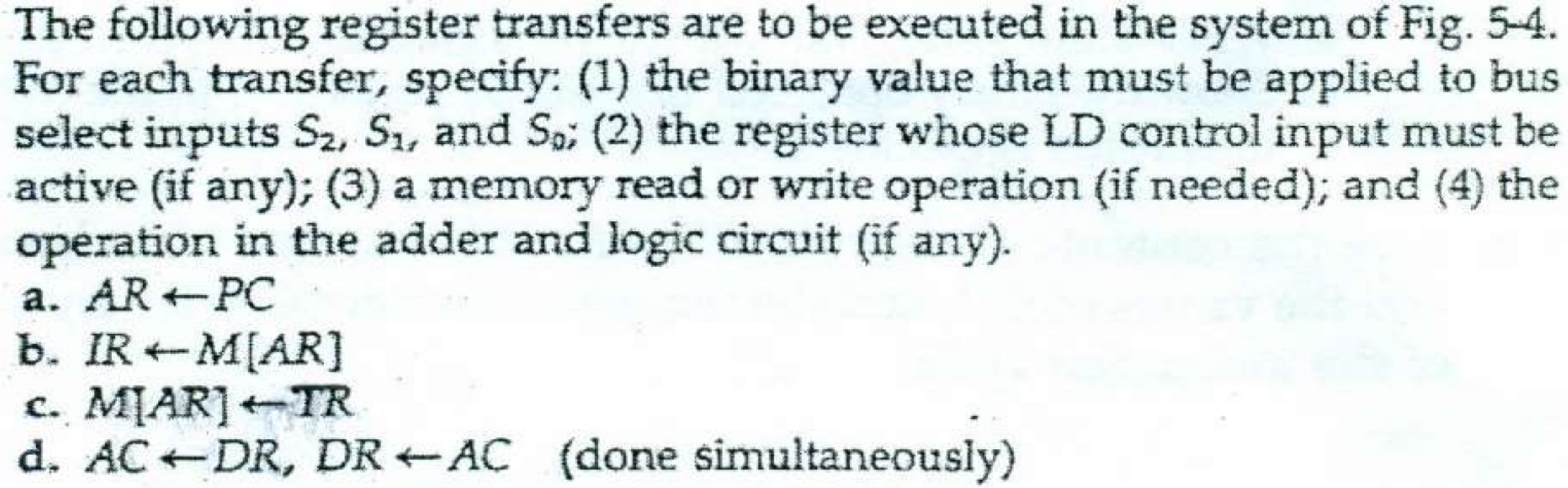
Question



The following control inputs are active in the bus system shown in Fig. 5-4. For each case, specify the register transfer that will be executed during the next clock transition.

	S_2	S_1	S_0	LD of register	Memory	Adder
a.	1	1	1	*IR*	Read	—
b.	1	1	0	*PC*	—	—
c.	1	0	0	*DR*	Write	—
d.	0	0	0	*AC*	—	Add

Question



The following register transfers are to be executed in the system of Fig. 5-4. For each transfer, specify: (1) the binary value that must be applied to bus select inputs S_2, S_1, and S_0; (2) the register whose LD control input must be active (if any); (3) a memory read or write operation (if needed); and (4) the operation in the adder and logic circuit (if any).

a. $AR \leftarrow PC$

b. $IR \leftarrow M[AR]$

c. $M[AR] \leftarrow TR$

d. $AC \leftarrow DR$, $DR \leftarrow AC$ (done simultaneously)

COMPUTER(BC) INSTRUCTIONS

Basic Computer Instruction code format

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

15	14 12	11 0
I	Opcode	Address

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

15 12	11 0
0 1 1 1	Register operation

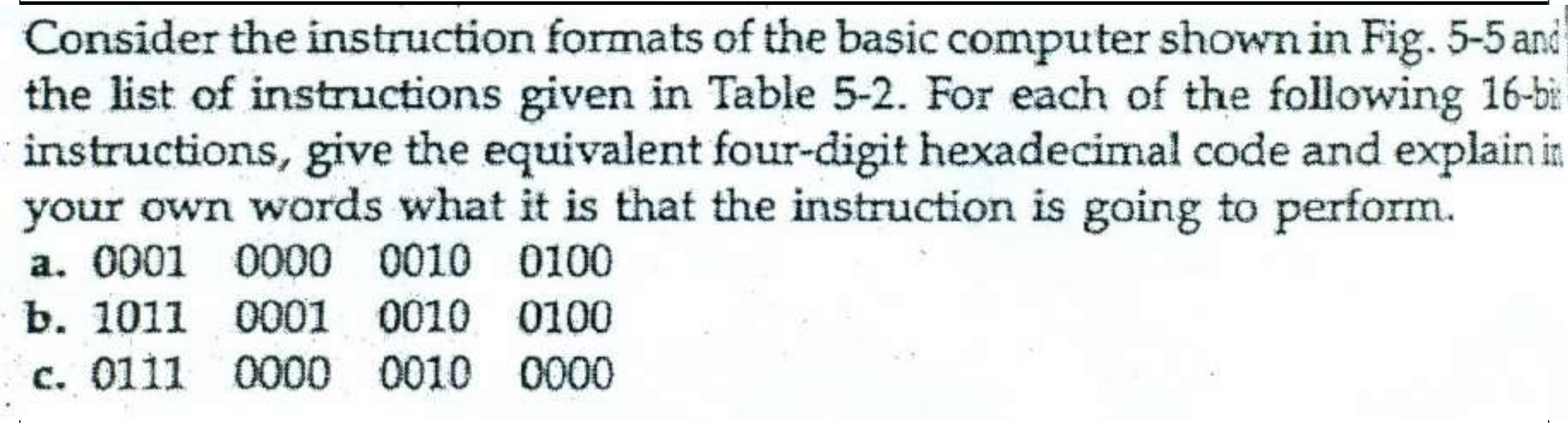
Input-Output Instructions (OP-code =111, I = 1)

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

BASIC COMPUTER INSTRUCTIONS

	Hex 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 AC from memory
STA	3xxx	Bxxx	Store content of AC into 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 instr. if AC is positive
SNA	7008		Skip next instr. if AC is negative
SZA	7004		Skip next instr. if AC is zero
SZE	7002		Skip next instr. if E is zero
HLT	7001		Halt computer
INP	F800		Input character to AC
OUT	F400		Output character from AC

Question



Consider the instruction formats of the basic computer shown in Fig. 5-5 and the list of instructions given in Table 5-2. For each of the following 16-bit instructions, give the equivalent four-digit hexadecimal code and explain in your own words what it is that the instruction is going to perform.

a. 0001 0000 0010 0100

b. 1011 0001 0010 0100

c. 0111 0000 0010 0000

INSTRUCTION SET COMPLETENESS

A computer should have a set of instructions so that the user can construct machine language programs to evaluate any function that is known to be computable.

Instruction Types

Functional Instructions

- Arithmetic, logic, and shift instructions
- ADD, CMA, INC, CIR, CIL, AND, CLA

Transfer Instructions

- Data transfers between the main memory and the processor registers
- LDA, STA

Control Instructions

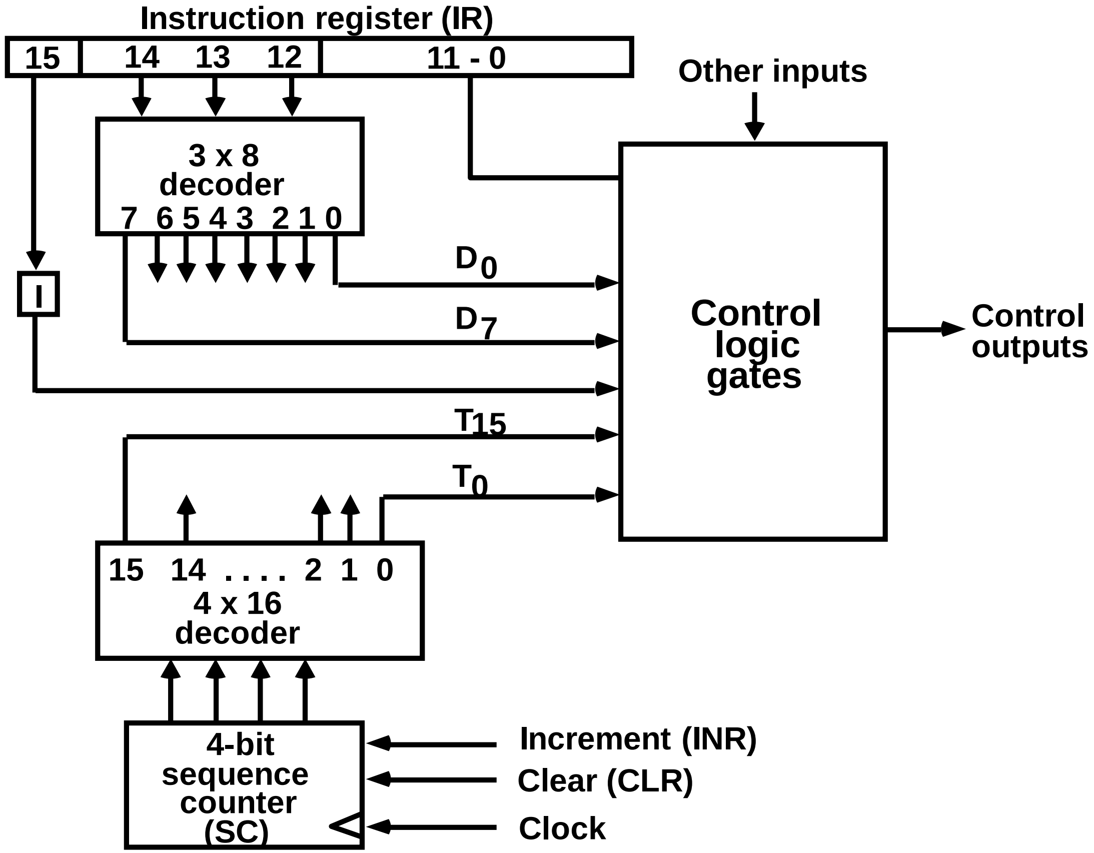
- Program sequencing and control
- BUN, BSA, ISZ

Input/Output Instructions

- Input and output
- INP, OUT

TIMING AND CONTROL

Control unit of basic computer



Control unit implementation

Hardwired Implementation
Microprogrammed Implementation

INSTRUCTION CYCLE

- Fetch instruction from memory

- Decode the instruction

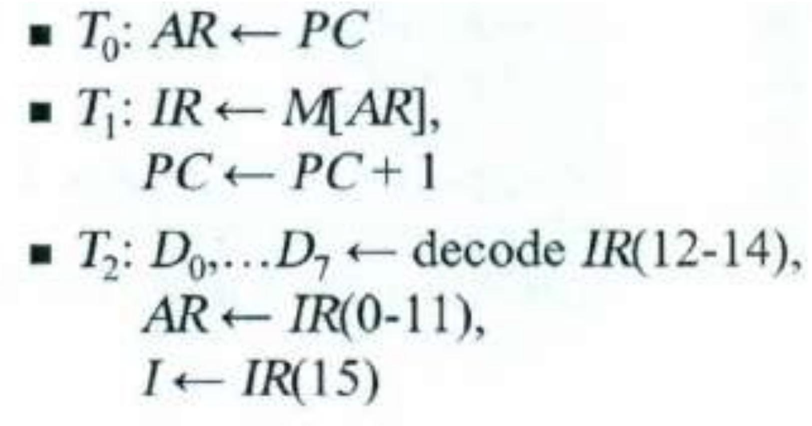
- Read effective address from memory if indirect address

- Execute the instruction

Common bus

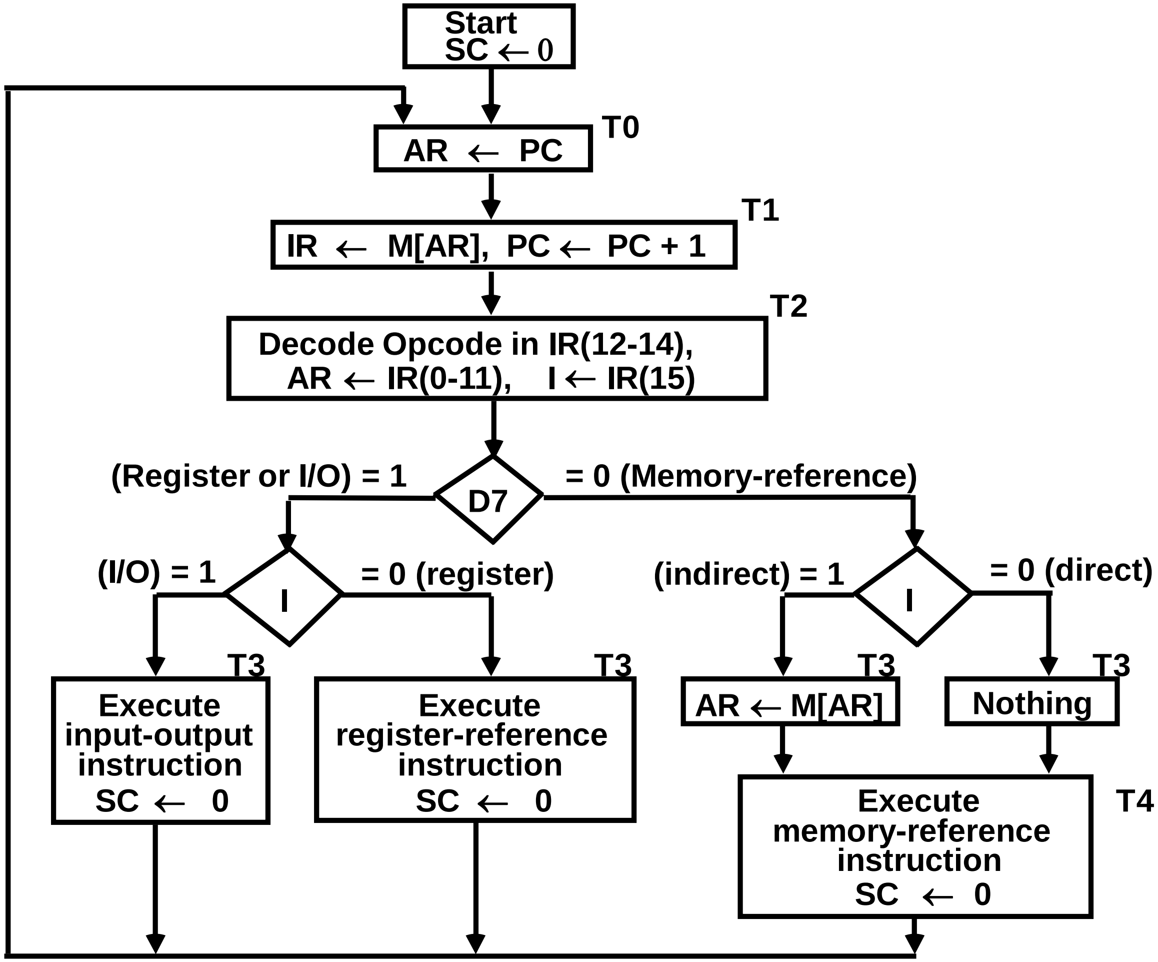
Fetch And Decode

- se cleared to 0, generating timing signal *To*
- After each clock pulse, se is incremented
- Fetch and decode microoperations



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

DETERMINE THE TYPE OF INSTRUCTION


Start
SC ← 0
T0
AR ← PC
T1
IR ← M[AR], PC ← PC + 1
T2
Decode Opcode in IR(12-14),
AR ← IR(0-11), I ← IR(15)
(Register or I/O) = 1
D7
= 0 (Memory-reference)
(I/O) = 1
I
= 0 (register)
(indirect) = 1
I
= 0 (direct)
T3
Execute input-output instruction SC ← 0
T3
Execute register-reference instruction SC ← 0
T3
AR ← M[AR]
T3
Nothing
Execute memory-reference instruction SC ← 0
T4

$D'_7 I T_3$: AR ← M[AR]

$D'_7 I' T_3$: Nothing

$D_7 I' T_3$: Execute a register-reference instr.

$D_7 I T_3$: Execute an input-output instr.

REGISTER REFERENCE INSTRUCTIONS

Register Reference Instructions are identified when

- $D_7 = 1$, $I = 0$
- Register Ref. Instr. is specified in $b_0 \sim b_{11}$ of IR
- Execution starts with timing signal T_3

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

	r:	SC ← 0
CLA	rB_{11}:	AC ← 0
CLE	rB_{10}:	E ← 0
CMA	rB_9:	AC ← AC’
CME	rB_8:	E ← E’
CIR	rB_7:	AC ← shr AC, AC(15) ← E, E ← AC(0)
CIL	rB_6:	AC ← shl AC, AC(0) ← E, E ← AC(15)
INC	rB_5:	AC ← AC + 1
SPA	rB_4:	if (AC(15) = 0) then (PC ← PC+1)
SNA	rB_3:	if (AC(15) = 1) then (PC ← PC+1)
SZA	rB_2:	if (AC = 0) then (PC ← PC+1)
SZE	rB_1:	if (E = 0) then (PC ← PC+1)
HLT	rB_0:	S ← 0 (S is a start-stop flip-flop)

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 $PC \leftarrow PC+1$

- The effective address of the instruction is in AR and was placed there during timing signal T_2 when I = 0, or during timing signal T3 when I = 1
- Memory cycle is assumed to be short enough to complete in a CPU cycle
- The execution of MR Instruction starts with T_4

AND to AC

D_0T_4: $DR \leftarrow M[AR]$ Read operand

D_0T_5: $AC \leftarrow AC \wedge DR, SC \leftarrow 0$ AND with AC

ADD to AC

D_1T_4: $DR \leftarrow M[AR]$ Read operand

D_1T_5: $AC \leftarrow AC + DR, E \leftarrow C_{out}, SC \leftarrow 0$ Add to AC and store carry in E

MEMORY REFERENCE INSTRUCTIONS

LDA: Load to AC

D_2T_4: $DR \leftarrow M[AR]$

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

STA: Store AC

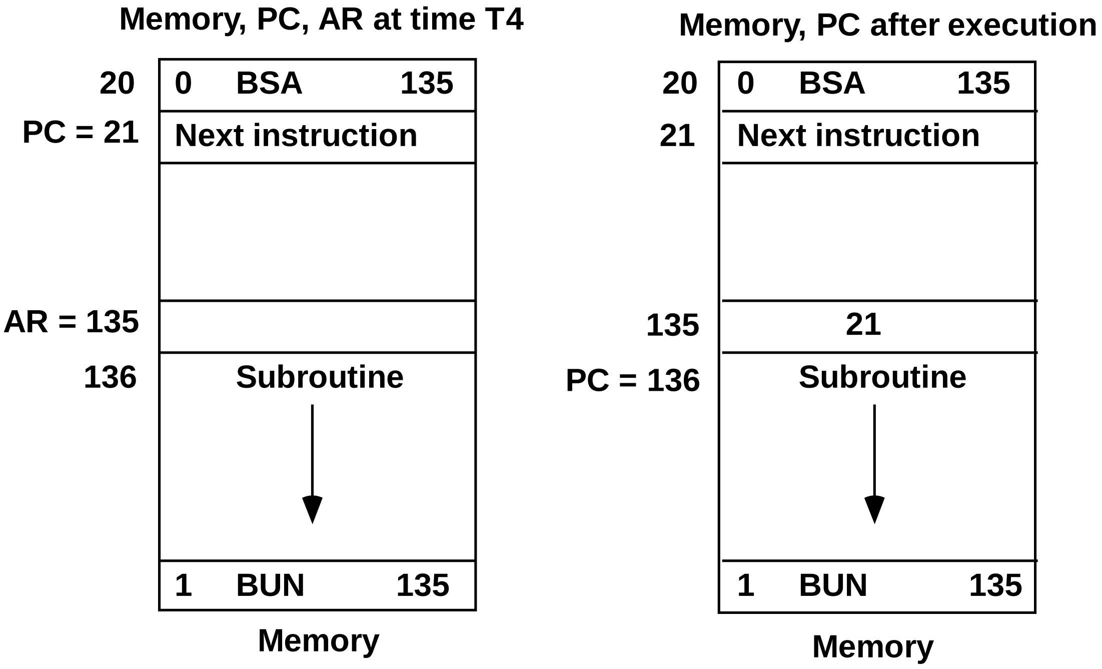
D_3T_4: $M[AR] \leftarrow AC, SC \leftarrow 0$

BUN: Branch Unconditionally

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

BSA: Branch and Save Return Address

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



Memory, PC, AR at time T4

20 | 0 BSA 135
PC = 21 | Next instruction
AR = 135 |
136 | Subroutine
1 BUN 135

Memory

Memory, PC after execution

20 | 0 BSA 135
21 | Next instruction
135 | 21
PC = 136 | Subroutine
1 BUN 135

Memory

MEMORY REFERENCE INSTRUCTIONS

BSA:

D_5T_4: M[AR] ← PC, AR ← AR + 1

D_5T_5: PC ← AR, SC ← 0

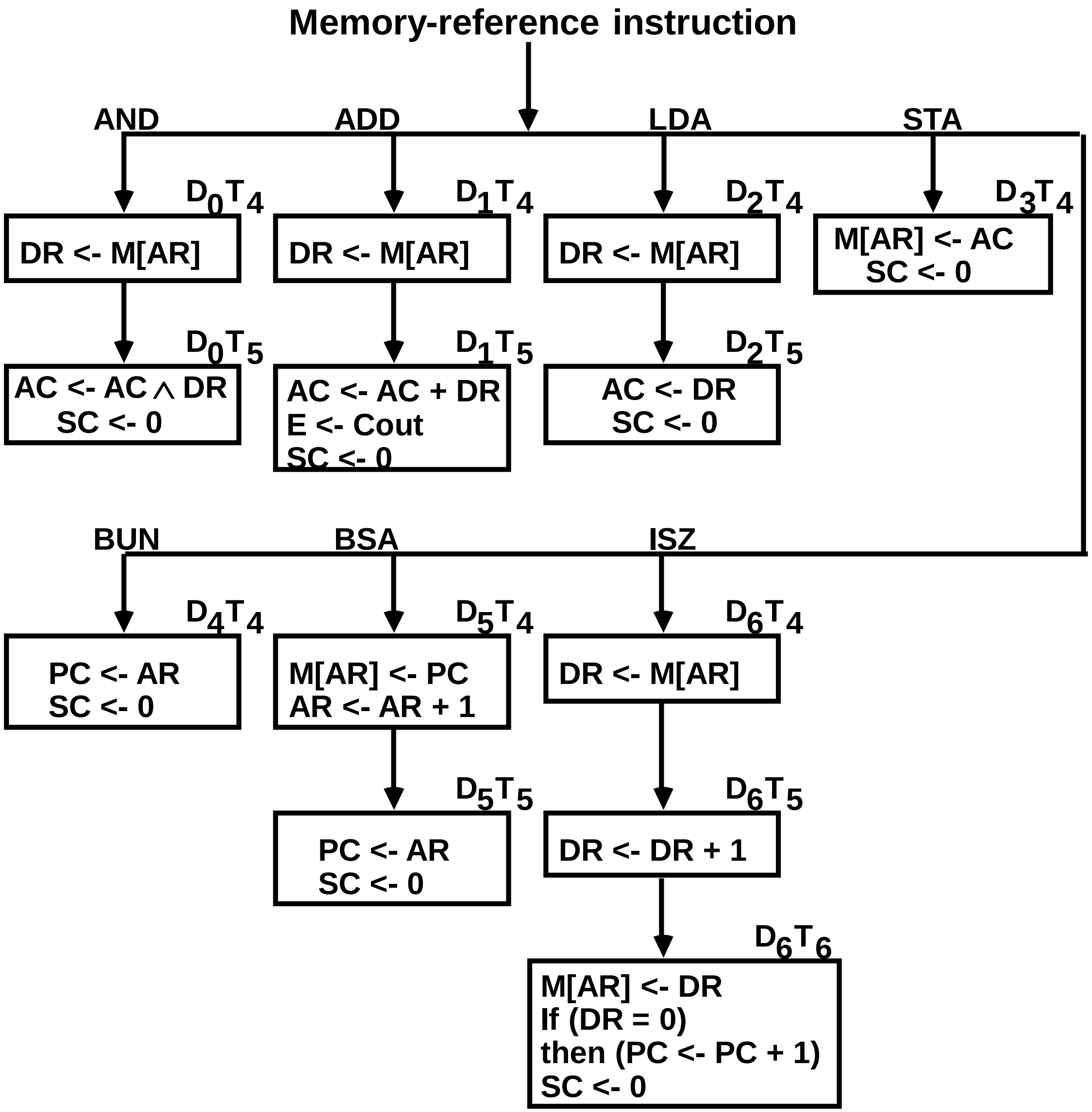
ISZ: Increment and Skip-if-Zero

D_6T_4: DR ← M[AR]

D_6T_5: DR ← DR + 1

D_6T_4: M[AR] ← DR, if (DR = 0) then (PC ← PC + 1), SC ← 0

FLOWCHART FOR MEMORY REFERENCE INSTRUCTIONS



Memory-reference instruction

AND

D_0T_4

DR <- M[AR]

D_0T_5

AC <- AC ∧ DR
SC <- 0

ADD

D_1T_4

DR <- M[AR]

D_1T_5

AC <- AC + DR
E <- Cout
SC <- 0

LDA

D_2T_4

DR <- M[AR]

D_2T_5

AC <- DR
SC <- 0

STA

D_3T_4

M[AR] <- AC
SC <- 0

BUN

D_4T_4

PC <- AR
SC <- 0

BSA

D_5T_4

M[AR] <- PC
AR <- AR + 1

D_5T_5

PC <- AR
SC <- 0

ISZ

D_6T_4

DR <- M[AR]

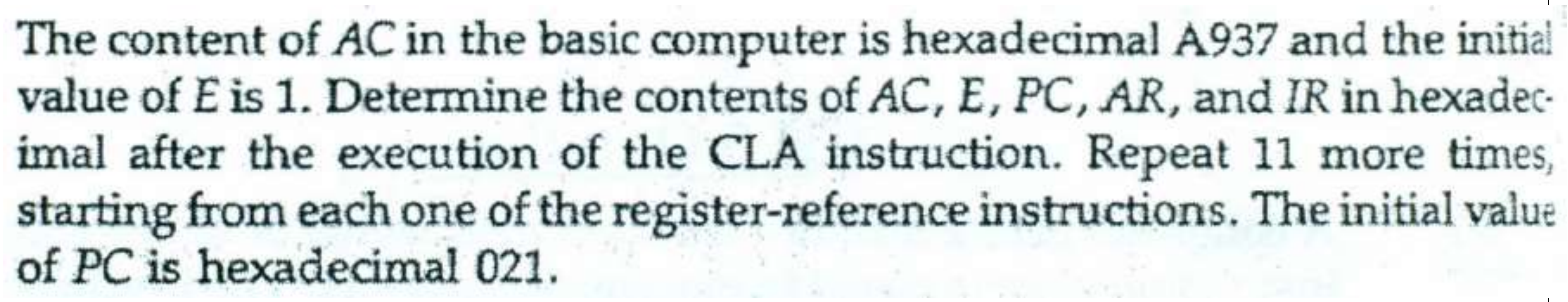
D_6T_5

DR <- DR + 1

D_6T_6

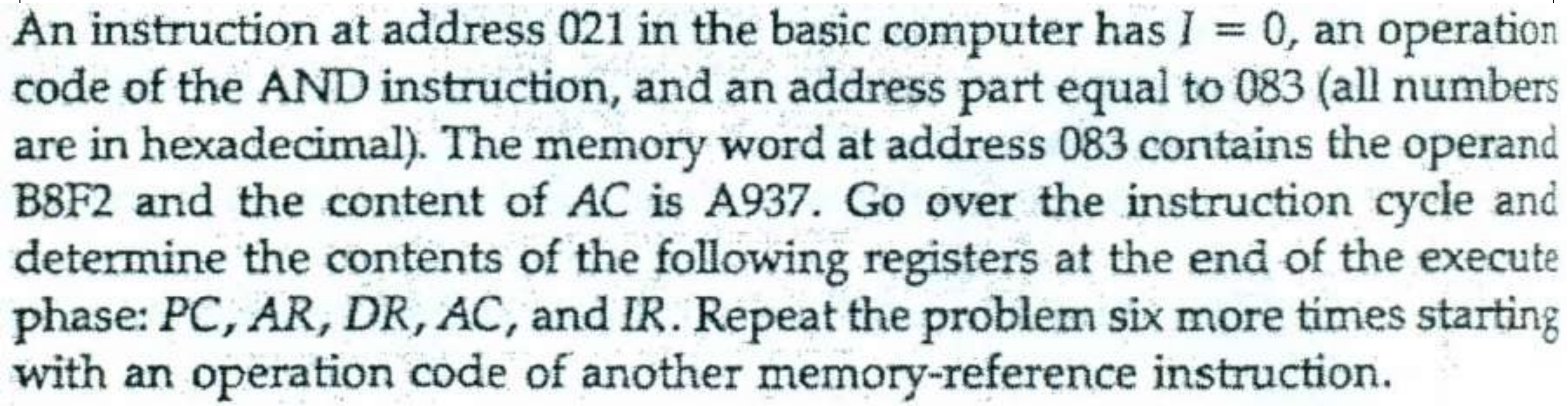
M[AR] <- DR
If (DR = 0)
then (PC <- PC + 1)
SC <- 0

Question



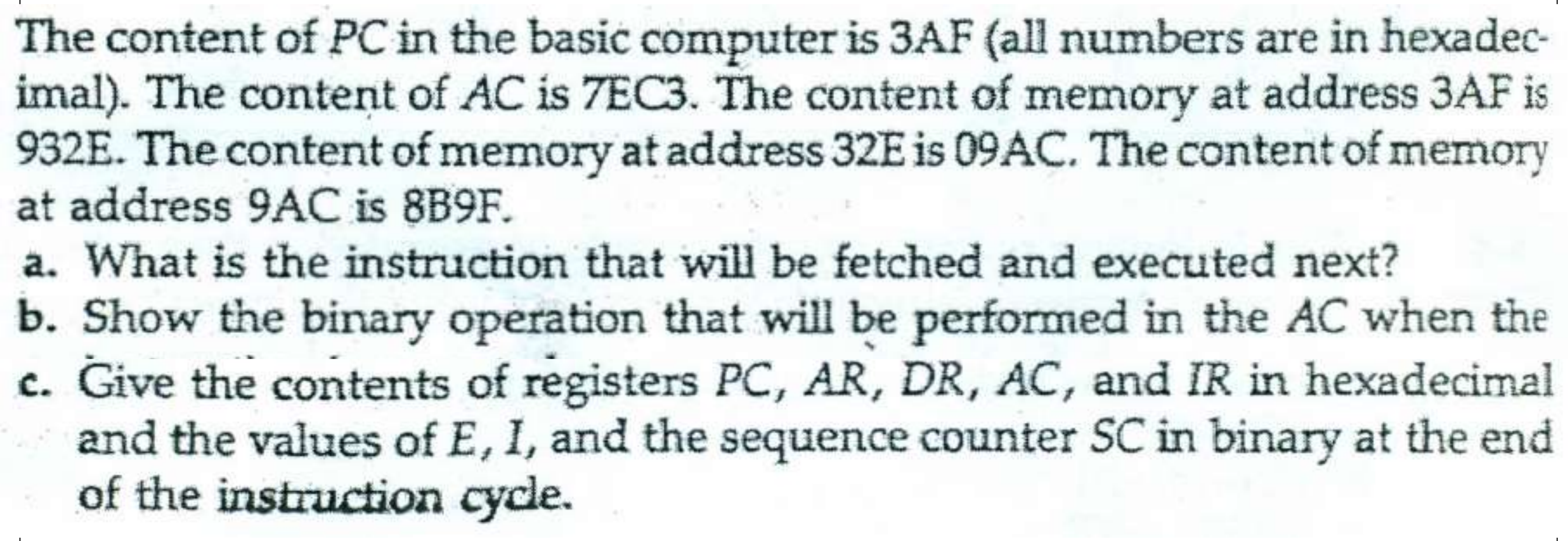
The content of *AC* in the basic computer is hexadecimal A937 and the initial value of *E* is 1. Determine the contents of *AC*, *E*, *PC*, *AR*, and *IR* in hexadecimal after the execution of the CLA instruction. Repeat 11 more times, starting from each one of the register-reference instructions. The initial value of *PC* is hexadecimal 021.

Question



An instruction at address 021 in the basic computer has $I = 0$, an operation code of the AND instruction, and an address part equal to 083 (all numbers are in hexadecimal). The memory word at address 083 contains the operand B8F2 and the content of *AC* is A937. Go over the instruction cycle and determine the contents of the following registers at the end of the execute phase: *PC*, *AR*, *DR*, *AC*, and *IR*. Repeat the problem six more times starting with an operation code of another memory-reference instruction.

Question



The content of *PC* in the basic computer is 3AF (all numbers are in hexadecimal). The content of *AC* is 7EC3. The content of memory at address 3AF is 932E. The content of memory at address 32E is 09AC. The content of memory at address 9AC is 8B9F.

a. What is the instruction that will be fetched and executed next?

b. Show the binary operation that will be performed in the *AC* when the

c. Give the contents of registers *PC*, *AR*, *DR*, *AC*, and *IR* in hexadecimal and the values of *E*, *I*, and the sequence counter *SC* in binary at the end of the instruction cycle.