University of Ottawa, School of Electrical Engineering and Computer Science

Question 1 (10 points)

1. Which of the following choice(s) is/are correct?

(a) (2 pts) When a main program subroutine is called, return address is saved

- in a memory address, when the subroutine is called by a program ☐
- in Address Register (AR) when the subroutine is called by a program ☐
- in Control Address Register (CAR) when the subroutine is called by a microprogram in the control unit ☐
- in subroutine register (SBR) when the subroutine is called by a microprogram in the control unit ☐

(b) (2 pts) Interrupt is enabled by

- setting IEN=1 at the beginning of an interrupt subroutine (ISR) ☐
- setting IEN=1 before returning from an ISR ☐
- calling an ISR ☐
- fetching the first instruction of ISR ☐

2. Perform the following conversions.
   (a) (1 pt) $(204.75)_{10} = (\quad)_{16}$
   (b) (1 pt) $(126)_8 = (\quad)_{BCD}$

3. (2 pts) A digital computer represents its floating-point numbers using a signed 6-bit exponer and a signed normalized 10-bit mantissa. Negative exponent and mantissa values ar expressed in 2's complement. Show the binary values of the exponent and mantissa t represent $(21.1)_8$.

10-bit mantissa= _____

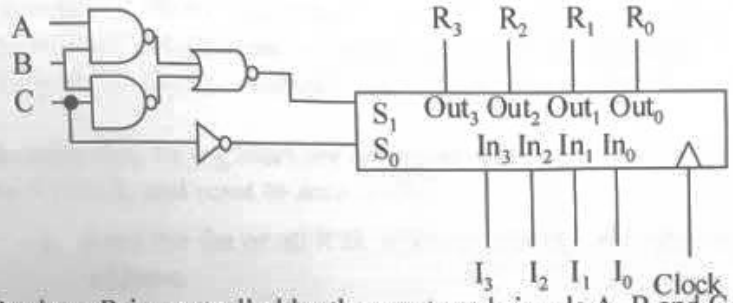
6-bit exponent= _____

4. (2 pts) Using the below table, translate the following machine codes which are stored in the memory of the 16-bit Basic Computer that was presented in the textbook (see Table 6 of the Annex and Figure 2) to their equivalent assembly language instructions.

Machine Code (in Hex)	Assembly Code
381D	
8456	
CBAC	
5777	

Question 2 (10 points)

The 4-bit register (R) employed in the following logic circuit has two control inputs (S_1, S_0). This register can perform three functions depending on the values of S_1 and S_0, as described in the following table.



S_1 S_0	Function
0 0	Load Parallel Inputs
0 1	Decrement
1 x	Circular shift left

Register R is controlled by three external signals A, B and C through the gates shown in the logic diagram. $I = (I_3I_2I_1I_0)$ is the register inputs and $R = (R_3R_2R_1R_0)$ is the output (content) of the register.

What will be the new value of R after the clock pulse?

a. If R = 0000, I = 0011, A = 0, B = 0, and C = 1.

R_{new} = ...

b. If R = 1001, I = 1100, A = 0, B = 0 and C = 0.

R_{new} = ...

c. If R = 0101, I = 1100, A = 1, B = 1 and C = 1.

R_{new} = ...

d. Explain why the register performs 3 functions instead of 4.

Question 3 (15 points)

Consider a computer similar to the one of Lab 4, and whose architecture is described in Figure 1 and Tables 1, 2, 3, 4, and 5. The instruction type is determined by the two most significant bits of the 8-bit register IR, as follows:
X_0 = IR(7)' IR(6)' denotes a memory-reference instruction (MRI) in direct addressing mode;
X_1 = IR(7)' IR(6) denotes a register-reference instruction (RRI); and
X_2 = IR(7) IR(6)' denotes a memory-reference instruction (MRI) in indirect addressing mode.

Assume that all registers are equipped with 3 control bits for loading the register (LD), increment by 1 (INC), and reset to zero (CLR).

1. Find the list of all RTL micro-operations in which the bus reads the contents of a memory address.

Derive the logic equation of the read memory control input, the control inputs of the affected registers and the bus select inputs required by the micro-operations found above.

Question 4 (15 points)

Four memory-reference instructions in the Basic Computer of the textbook's Chapter 5 are to be changed to the instructions specified in the following table, where "EA" denotes the operand's effective address.

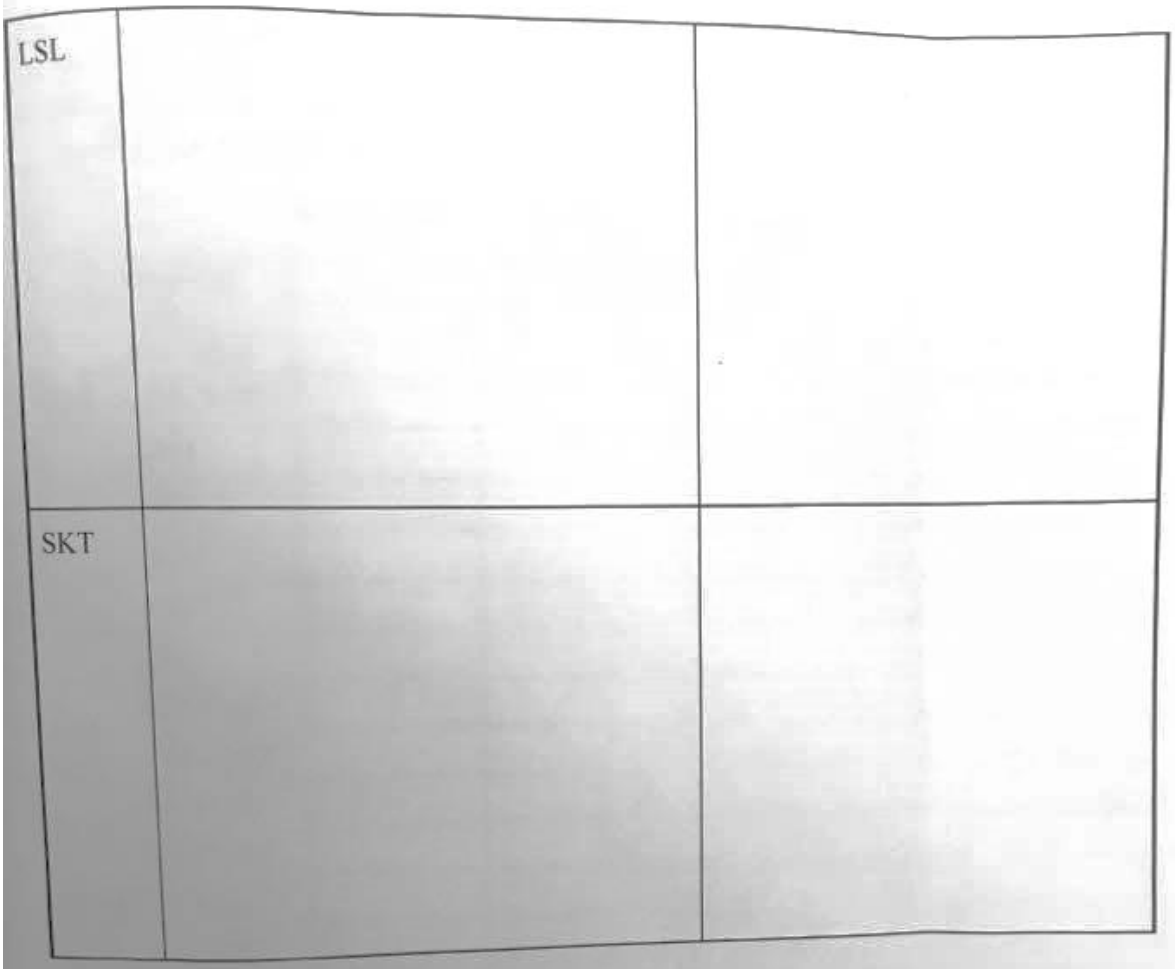
Symbol	Opcode	Symbolic designation	Description in words
NND	000	M[EA] ← (AC ∧ M[EA])'	Logic NAND of AC to memory
LSR	001	M[EA] ← shr(M[EA])	Logical shift right memory content
LSL	010	M[EA] ← shl(M[EA])	Logical shift left memory content
SKT	011	If (M[EA] mod 2 == 1) then PC ← PC + 1	Skip next 2 instructions if M[EA] is odd

In the following table, write down the necessary micro-operations (in RTL) to perform the execution phase of each instruction. Note that the execution phase of such type of instructions starts at T_4. See figures 2 and 3 from Annex.

Note 1. The value in AC should not be changed by the execution of any instruction, unless the instruction specifies a change in its content. To this extent, you can use TR to store the temporary content of AC.
Note 2. Only msb and lsb of registers AC and E can be checked.

Symbol	RTL	Comments
NND		
LSR		

University of Ottawa, School of Electrical Engineering and Computer Science



Question 5 (20 points)

Use the table to the right to translate Program 1 in its equivalent machine code by specifying the machine code of each instruction/operand, and its address in the memory.

1. Write your answer in only the first two columns of the table (in hexadecimal).

	Address (hex)	Memory content before execution of program (hex)	AC content after first execution of instruction	AC content after final execution of instruction
ORG010				
LDA X				
STA TMP				
CMP LDA TMP I				
CMA				
INC				
ISZ X				
ADD X I				
SNA				
BUN WRX				
LDA TMP I				
STA RES				
BUN CHK				
WRX, LDA X I				
STA RES				
LDA X				
STA TMP				
CHK, ISZ CTR				
BUN CMP				
HLT				
ORG 200				Memory content after execution of the program
CTR, DEC -2	200	-2		
TMP, HEX 0	201	0		
RES, HEX 0	202	0		
X, HEX 0203	203	204		
HEX 0204	204	10		
HEX 0205	205	30		
HEX 0206	206	20		

University of Ottawa, School of Electrical Engineering and Computer Science

2. What is the content of the accumulator AC after the first and final execution of each instruction that affects it? Write down your answers in the third and fourth columns of the table. Note that you have to fill in only the rows with instructions that affect the accumulator.

3. After the execution of the program, what is the content of the part of the memory where the operands are stored? Write down your answers in the fourth column of the table. Note that you have to fill in only the rows corresponding to the memory locations where the operands are stored.

4. In just one sentence, what does the program do?

Question 6 (30 points)

All addresses in these questions are provided in hexadecimal format.

Write a program for the Basic Computer that counts the total number of 1's in an array of N numbers that are stored in memory starting with the address **F01**. This total number of 1's has to be stored by your program in the memory location at address **EFF**.

The main program invokes a subroutine (called ONE) which counts the number of 1's contained in a single memory location. For example, if the memory location contains the binary number 0000 0000 1001 1100, the value found by the subroutine will be 0000 0000 0000 0100, as the number of 1's in 0000 0000 1001 1100 is 4. This result (the number of 1's) is passed by the subroutine ONE to the calling program through the accumulator, and the address of the memory location being processed is passed to ONE through the accumulator AC, as well.

Your subroutine ONE should be stored in the memory of your Basic Computer beginning with address **A00**, while the main program starts at address **9C0**.

Before running the program, the operator loads N (the number of array's elements) at the memory location **F00** which is then followed by the elements of the array to be processed.

a) Write the subroutine ONE in assembly language using instructions of the Basic Computer given in the Table 6 from the Annex;
b) Write the main program that calls the subroutine ONE.

NOTE: You are allowed to use alternative ways to pass parameters to / from the subroutine, but full explanations should be provided.

The 8-bit Computer of Lab 4 ANNEX

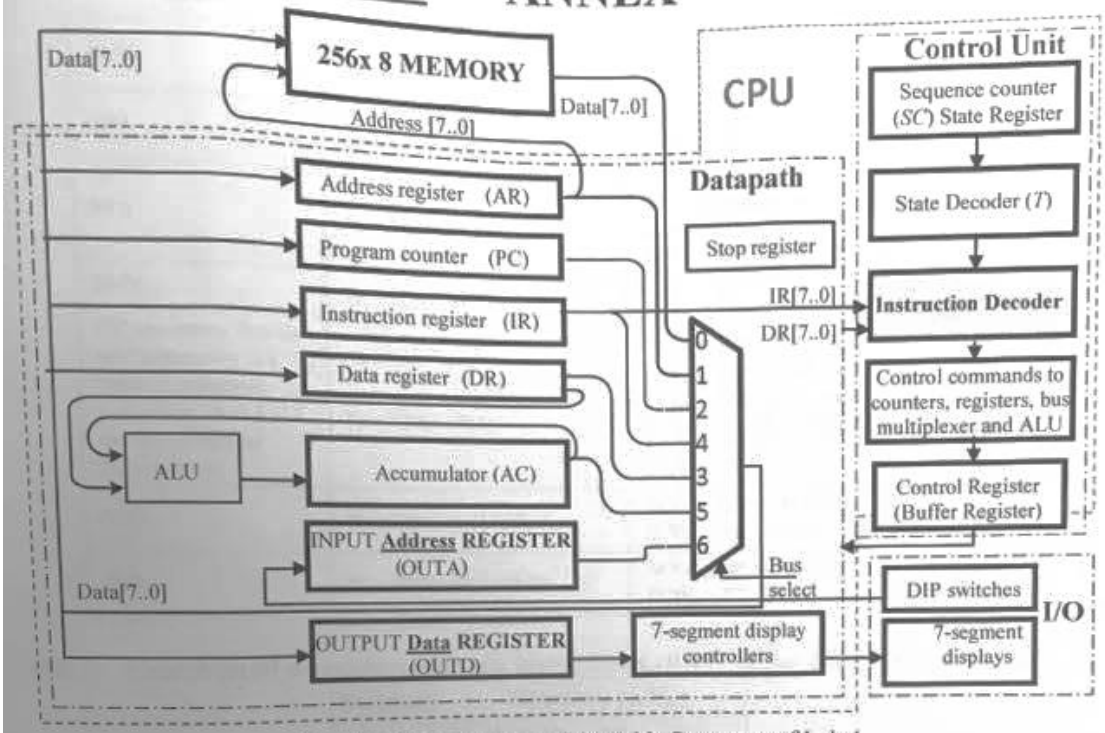


Fig. 1a. Block diagram of the 8-bit Computer of Lab 4

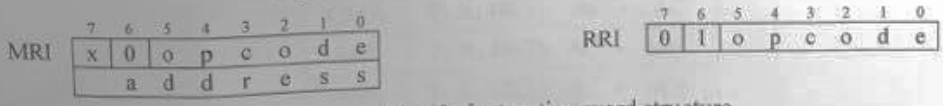


Fig. 1b. Instruction word structure

Table 1: Starting micro-operations in the instruction cycle of the 8-bit mini computer

Instant	Description	Notation (RTL)
T_0	Load the program counter PC in AR, and increment PC	T_0 : AR ← PC T_0 S' : PC ← PC + 1
T_1	Read instruction from memory and put it in the instruction register IR	T_1 : IR ← M [AR]
T_2	This cycle is not used to allow for the new value of the instruction to propagate in the controller	(nothing)
T_3	If X_1 => it is a *register - reference instruction* (RRI) that will be executed now	T_3X_1 : execute an RRI instruction (Table 4) T_3X_1 : SC ← 0
	If X_0 or X_2 (MRI), load the address of the second instruction word to AR, i.e., memory address of the operand and place it in AR, and increment PC. Recall that $(X_0 + X_2) = IR(6)'$	$T_3IR(6)'$: AR ← PC $T_3IR(6)'$ S': PC ← PC + 1
T_4	Read the second instruction word (memory address) into AR	$T_4IR(6)'$: AR ← M [AR]
T_5	If *direct addressing*, don't do anything, as the operand address is already in AR since T_4	T_5X_0 : (nothing)
	If *indirect addressing*, read the *operand address* from memory location pointed to by AR	T_5X_2 : AR ← M [AR]
starting from T_6	Execute the memory-reference instructions (MRI) described in Table 2	(see Table 2)

Table 2: Execution of a memory-reference instructions (MRI) in the 8-bit mini computer

Symbol		Notation (RTL)
ADD	$Y_0 = IR(6)' IR(1) IR(0)'$	$T_6 Y_0$: DR ← M [AR] $T_7 Y_0$: AC ← AC + DR , SC ← 0
LDA	$Y_1 = IR(6)' IR(2)$	$T_6 Y_1$: DR ← M [AR] $T_7 Y_1$: AC ← DR , SC ← 0
STA	$Y_2 = IR(6)' IR(3)$	T_6 : (cycle not used to allow for the address bus to stabilize) $T_7 Y_2$: M [AR] ← AC , SC ← 0
BUN	$Y_3 = IR(6)' IR(4)$	$T_6 Y_3$: PC ← AR , SC ← 0
ISZ (assuming that the next instruction is a memory-reference instruction, stored at 2 memory locations further down)	$Y_4 = IR(6)' IR(5)$	$T_6 Y_4$: DR ← M [AR] $T_7 Y_4$: DR ← DR + 1 $T_8 Y_4$: M [AR] ← DR $T_9 Y_4$: if (DR = 0)S' then (PC ← PC + 1) $T_{10} Y_4$: if (DR = 0)S' then (PC ← PC + 1) $T_{10} Y_4$: SC ← 0
AND	$Y_5 = IR(6)' IR(1)' IR(0)$	$T_6 Y_5$: DR ← M [AR] $T_7 Y_5$: AC ← AC ∧ DR , SC ← 0
SUB	$Y_6 = IR(6)' IR(1) IR(0)$	$T_6 Y_6$: DR ← M [AR] $T_7 Y_6$: AC ← AC − DR , SC ← 0

Table 3: Execution of a register-reference instructions (RRI) of the 8-bit mini computer

Symbol	Notation (RTL)
CLA	$T_3 X_1 IR(0)$: AC ← 0
CMA	$T_3 X_1 IR(1)$: AC ← AC'
ASL	$T_3 X_1 IR(2)$: AC ← ashl
ASR	$T_3 X_1 IR(3)$: AC ← ashr
INC	$T_3 X_1 IR(4)$: AC ←AC+1
HLT	$T_3 X_1 IR(5)$: S ← 1

Table 4:
Function table of the 8-bit ALU

S2	S1	S0	Operation
0	0	0	F = AC + DR
0	0	1	F = AC − DR
0	1	0	F = ashl AC
0	1	1	F = ashr AC
1	0	0	F = AC ∧ DR
1	0	1	F = AC ∨ DR
1	1	0	F = DR (transfer)
1	1	1	F = AC'

Table 5:
Function table of the 8-bit bus

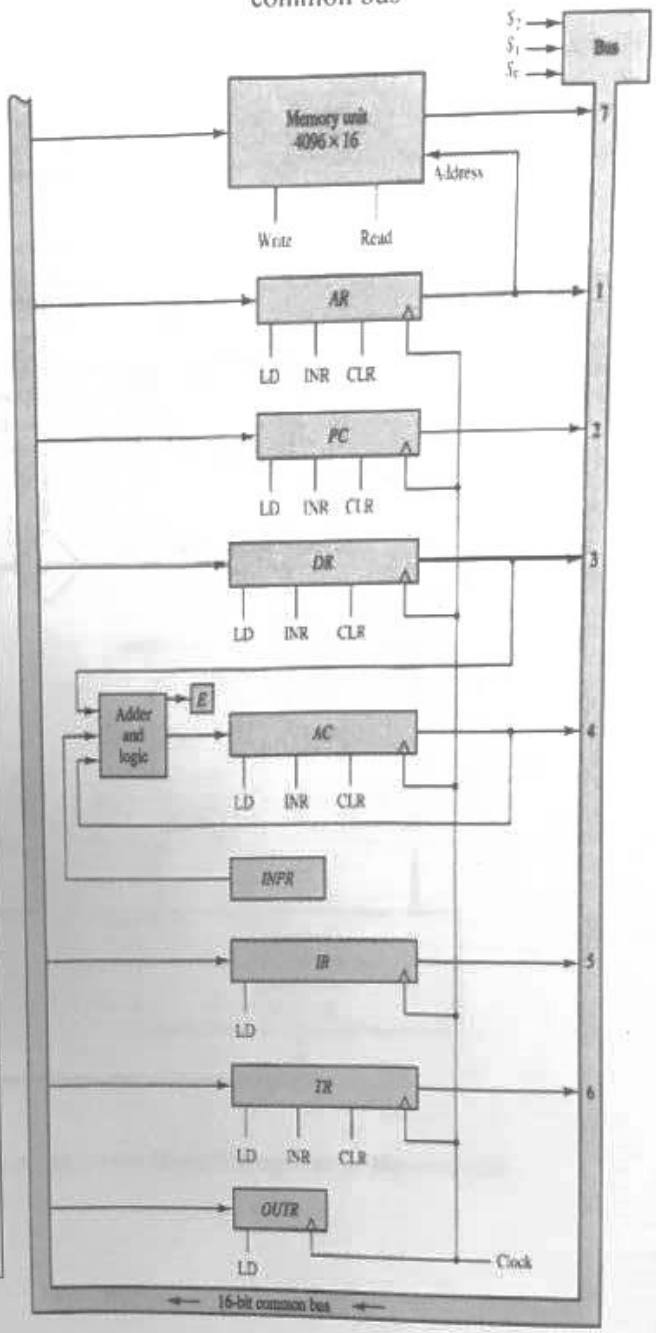
S2	S1	S0	Register placed on the bus
0	0	0	Memory
0	0	1	AR
0	1	0	PC
0	1	1	DR
1	0	0	IR
1	0	1	AC
1	1	0	OUTA
1	1	1	None

The 16-bit Basic Computer of the textbook

Table 6: Instructions list

Symbol	Hex code	Description
AND	0 or 8	AND *M* to *AC*
ADD	1 or 9	Add *M* to *AC*, carry to *E*
LDA	2 or A	Load *AC* from *M*
STA	3 or B	Store *AC* in *M*
BUN	4 or C	Branch unconditionally to *m*
BSA	5 or D	Save return address in *m* and branch to *m* + 1
ISZ	6 or E	Increment *M* and skip if zero
CLA	7800	Clear *AC*
CLE	7400	Clear *E*
CMA	7200	Complement *AC*
CME	7100	Complement *E*
CIR	7080	Circulate right *E* & *AC*
CIL	7040	Circulate left *E* and *AC*
INC	7020	Increment *AC*,
SPA	7010	Skip if *AC* is positive
SNA	7008	Skip if *AC* is negative
SZA	7004	Skip if *AC* is zero
SZE	7002	Skip if *E* is zero
HLT	7001	Halt computer
INP	F800	Input information and clear flag
OUT	F400	Output information and clear flag
SKI	F200	Skip if input flag is on
SKO	F100	Skip if output flag is on
ION	F080	Turn interrupt on
IOF	F040	Turn interrupt off

Fig. 2. **Basic Computer** registers connected to a common bus



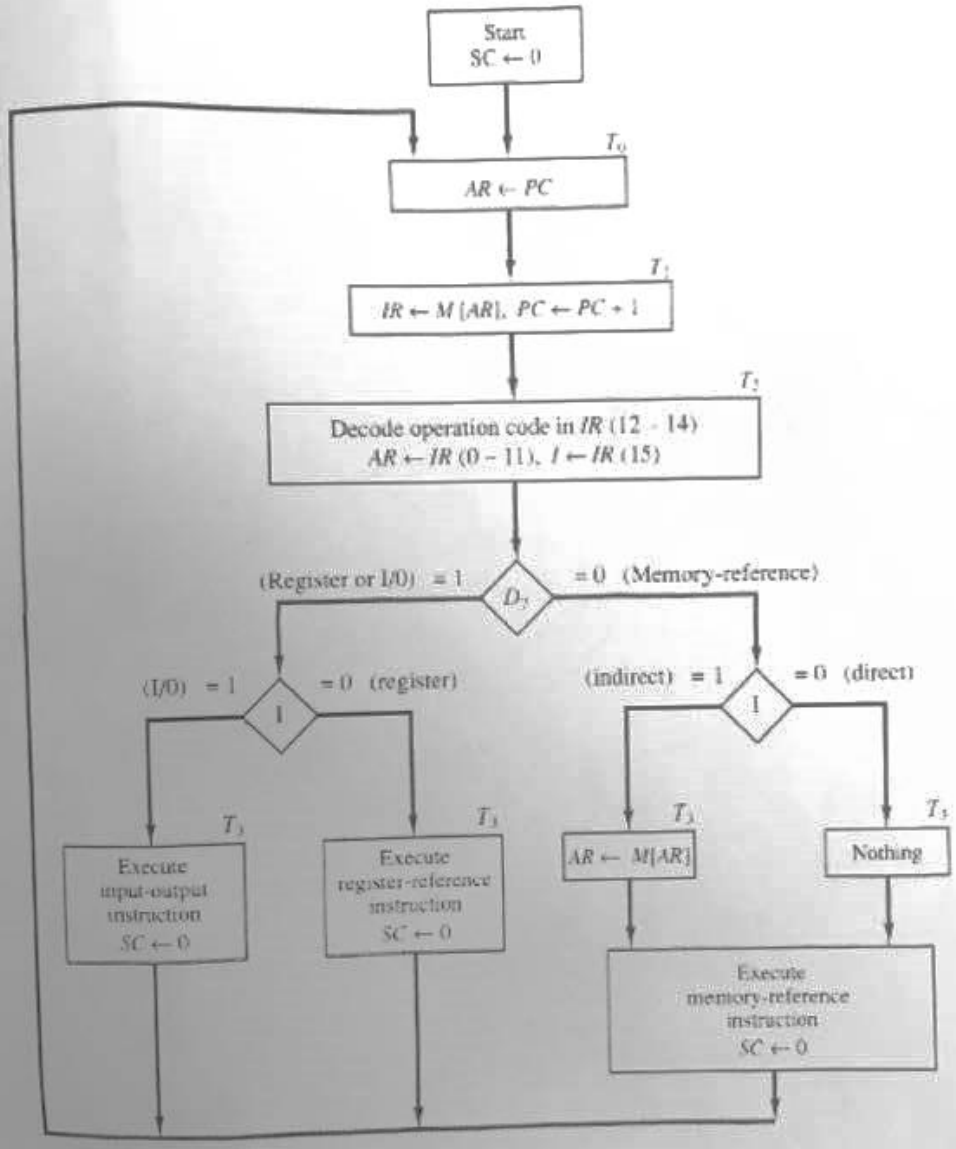


Fig. 3. Flowchart for instruction cycle of the 16-bit **Basic Computer** of the textbook