

CEG2136: Computer Architecture I / CEG2536: Architecture des Ordinateurs I  
FINAL EXAMINATION - SAMPLES

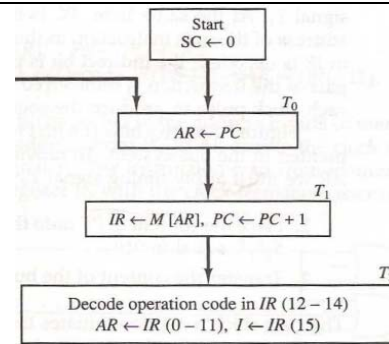
**A – SOLUTIONS ARE HIGHLIGHTED**

Q1.1 For 1 K memory locations you need:

- (a) 8 address lines
  - (b) 10 address lines**
  - (c) 12 address lines
  - (d) None of the above
- (A memory with a capacity of  $2^m$  words requires  $m$  address lines. So, for  $2^m = 2^{10}$ , we need  $m = 10$  address lines)

Q1.2 Which CPU register provides the address from which the next instruction opcode is to be fetched?

- (a) Instruction register IR
- (b) Accumulator AC
- (c) Program counter PC**
- (d) None of these



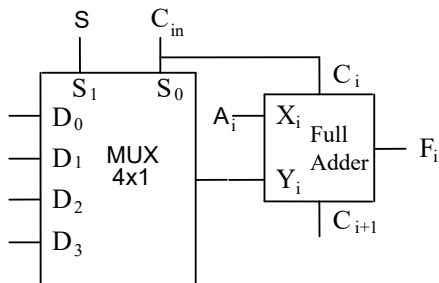
Q1.4 An arithmetic unit performs basic operations on two numbers A and B, under the control of two bits S and  $C_{in}$ , as shown in the following table:

S	$C_{in} = 0$	$C_{in} = 1$
0	$F = A + B$ (addition)	$F = A + 1$ (increment A)
1	$F = A - 1$ (decrement A)	$F = A + B' + 1$ (subtraction)

The following combinational circuit (multiplexor and full adder) is used to implement the functions described above for bit  $i$ . Select from the following combinations of multiplexor inputs which set of logic values implements correctly the above functions

S	$C_{in} = 0$	$C_{in} = 1$
0	$F = A_2 A_1 A_0 + B_2 B_1 B_0$	$F = A_2 A_1 A_0 + 0 0 0 + 1$
1	$F = A_2 A_1 A_0 + 1 1 1$	$F = A_2 A_1 A_0 + B'_2 B'_1 B'_0 + 1$

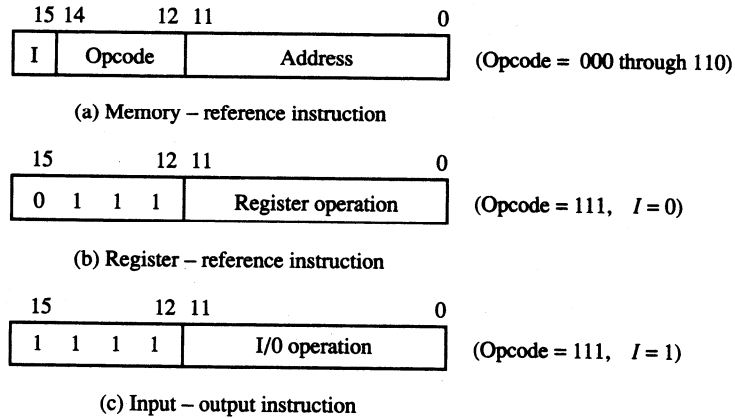
S	$C_{in}$	$X_i$	$Y_i$
00		$A_i$	$B_i$
01		$A_i$	0
10		$A_i$	1
11		$A_i$	$B'_i$



- (a)  $D_0=B'_i, D_1=0, D_2=1, D_3=B_i$
- (b)  $D_0=0, D_1=B_i, D_2=B'_i, D_3=1$
- (c)  $D_0=1, D_1=B'_i, D_2=B_i, D_3=0$
- (d)  $D_0=B_i, D_1=0, D_2=1, D_3=B'_i$**
- (e)  $D_0=1, D_1=B_i, D_2=B'_i, D_3=0$

Q2

The block diagram of the basic computer that was introduced in chapter 5 of Mano’s textbook is given in the annex, along with its instruction list. The instruction word is 16 bit long and has the following structure ...



Q2.1 At some point, the content of PC of the basic computer is 3AF (all numbers are in hexadecimal) and the content of AC is 2EC3, as shown in the following table. The content of memory is partially given below, as well:

MEMORY		BASIC COMPUTER REGISTERS		
Address	Memory content	Content before instruction execution	Content after instruction execution	
3AD	03B5	PC	3AF	3B0
3AE	ABBA	AC	2EC3	6A62
3AF	93AD	AR	0000	3B5
3B0	DEED	DR	0000	3B9F
3B1	7BEE	IR	0000	93AD
3B2	AD08	E	0	0
3B3	10BC	I	0	1
3B4	1CAA	SC		000
3B5	3B9F			
3B6	3BA0			

- a. What is the instruction that will be fetched and executed?  
**ADD 3AD I**
- b. Show the operands and the binary operation that will be performed in the AC when the instruction is executed.  
**2EC3 + 3B9F**
- c. Fill out the last column (“Content after instruction execution”) of the above table with 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, all shown at the end of the instruction cycle.

Q2.2 What is the result, in decimal, of the operation performed by the following assembly program?

ORG 100	Instruction effect
LDA OP	AC=FFA5
CMA	AC=005A
INC	AC=005B
ADD OP1	AC=0094
STA OP2	M(OP2)=0094
HLT	
OP1, 0039	
OP, FFA5	
OP2, 0	0094
END	

HEX94 = DEC(9 x 16 + 4) = 144 + 4 = 148

The machine code of this program is stored in a memory unit of 1 kilo-word of 16 bits implemented on an Altera FPGA; give the Quartus .mif file that describes this program

Assembly Program	Memory Address	Memory content
ORG 100		
LDA OP	100	2107
CMA	101	7200
INC	102	7020
ADD OP1	103	1106
STA OP2	104	3108
HLT	105	7001
OP1, 0039	106	0039
OP, FFA5	107	FFA5
OP2, 0	108	0000
END		

Quartus .mif file:

Addr	+0	+1	+2	+3	+4	+5	+6	+7
100	2107	7200	7020	1106	3108	7001	0039	FFA5
108	0000	0000	0000	0000	0000	0000	0000	0000
110	0000	0000	0000	0000	0000	0000	0000	0000

Q4.1 Examining the annexed block diagram of the Mano's basic computer, give the list of the blocks of the datapath and the list of those that form the computer's control unit.

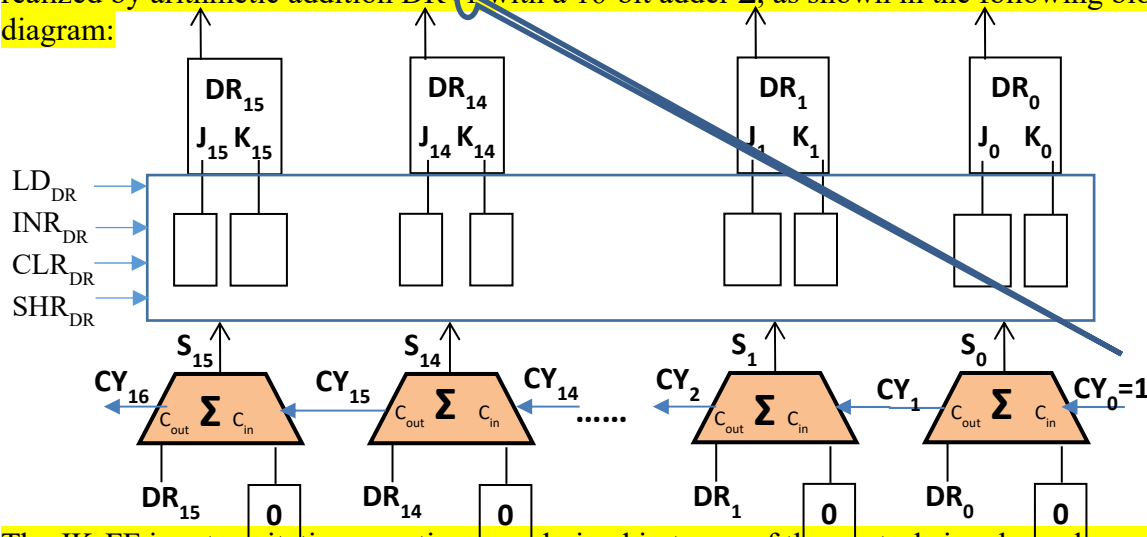
Q4.2 You have to expand the instruction list of your textbook basic computer with three more instructions (ASR, DIV, TAT) as shown in the following table; both circuits of the datapath and the control unit need to be extended. This basic computer is a simplified version of the one presented in the textbook as it does not have any provisions to deal with interrupts.

The datapath has to be redesigned to allow your DR register to support the following operations:

- $LD_{DR} = x T_4$ :  $DR \leftarrow (\text{bus})$  /Transfer value from bus to DR
- $INR_{DR} = y T_3$ :  $DR \leftarrow DR+1$  /Increment DR
- $CLR_{DR} = z T_3$ :  $DR \leftarrow 0$  /Clear DR
- $SHR_{DR} = v T_3 + w T_4$ :  $DR \leftarrow \text{shr } DR$  /Shift right DR

Design and draw the logic diagram of  $DR_i$  (bit  $i$  of register DR), using any type of combinational circuits and a JK flip-flop.

**Solution:** DR will be implemented as a *multi-function register* where incrementation  $INR_{DR}$  is realized by arithmetic addition  $DR+1$  with a 16-bit adder  $\Sigma$ , as shown in the following block diagram:



The JK-FF input excitation equations are derived in terms of the control signals as shown below:

$J^n$	$K^n$	$Q^{n+1}$	Function	Action:
0	0	$Q^n$	Hold	Do not include any term in J & K equations to keep the current state
0	1	0	Reset	To reset a JK-FF, add that term to the K equation, but not to J
1	0	1	Set	To set a JK-FF, add that term to the J equation, but not to K
1	1	$Q^n$	Toggle	To toggle a JK-FF, add that term to both J & K equations

To load value  $a$  into a JK-FF, AND  $a$  with the term and include in the J equation, while  $a'$  is AND-ed with the term and it is applied to K. In particular to load whatever next state, set  $a = Q^+$

**4.2.a** Applying these rules to our problem, the equations of the J and K inputs can be written directly:

$$J_{DRi} = xT_4 bus_i + yT_3 S_i + (vT_3 + wT_4) DR_{i+1} \quad \text{for } i = 0 \dots 14,$$

$$K_{DRi} = xT_4 bus'_i + yT_3 S'_i + (vT_3 + wT_4) DR'_{i+1} + zT_3$$

and, for **arithmetic** shift right, DR15 has to be copied to both most significant two bits, so,

$$J_{DR15} = xT_4 bus_{15} + yT_3 S_{15} + (vT_3 + wT_4) DR_{15}$$

$$K_{DR15} = xT_4 bus'_{15} + yT_3 S'_{15} + (vT_3 + wT_4) DR'_{15} + zT_3$$

**4.2.b** Alternatively, the initial table is re-written as shown below:

Function	Present state	Next state	JK FF inputs		JK-FF
RTL	$DR_i$	$DR_i^+$	$J_{DR}$	$K_{DR}$	Implemented as
$LD_{DR} = xT_4: DR \leftarrow (bus)$	0	$bus_i$	$bus_i$	$d(bus'_i)$	D-FF
	1	$bus_i$	$d(bus_i)$	$bus'_i$	
$INR_{DR} = yT_3: DR \leftarrow DR+1$ (implemented here with $DR \leftarrow S/\Sigma$ )	0	$S_i$	$S_i$	$d(S'_i)$	D-FF
	1	$S_i$	$d(S_i)$	$(S_i)'$	
$CLR_{DR} = zT_3: DR \leftarrow 0$	0	0	0	$d(1)$	JK Reset
	1	0	$d(0)$	1	
$SHR_{DR} = vT_3 + wT_4: DR \leftarrow shr DR$	0	$DR_{i+1}$	$DR_{i+1}$	$d(DR'_{i+1})$	D-FF
	1	$DR_{i+1}$	$d(DR_{i+1})$	$DR'_{i+1}$	

Don't care is marked in the above table with "d"

DR<sub>i</sub> variable can be eliminated from J and K equations if the inputs are made the same for both DR<sub>i</sub> = 0 or 1 as shown in the parentheses following the *d* 's in the table; the JK-FF's emulates D-FF.

$$J_{DR_i} = xT_4 bus_i + yT_3 S_i + (vT_3 + wT_4) DR_{i+1}$$

$$K_{DR_i} = xT_4 bus'_i + yT_3 S'_i + (vT_3 + wT_4) DR'_{i+1} + zT_3$$

for  $i = 0 \dots 14$  and

$$J_{DR_{15}} = x T_4 bus_{15} + y T_3 S_{15} + (v T_3 + w T_4) DR_{15}$$

$$K_{DR_{15}} = xT_4 bus'_{15} + y T_3 S'_{15} + (v T_3 + w T_4) DR'_{15} + zT_3$$

Since the second  $\Sigma$  operand is 0, the same can be realized by using half-adders instead of full-adders  $\Sigma$ :

$$S_j = DR_j \oplus CY_{j-1};$$

$$CY_j = DR_j \bullet CY_{j-1}$$

with  $j = 1 \dots 15$  and

$$S_0 = DR_0 \oplus 1; CY_0 = DR_0 \bullet 1$$

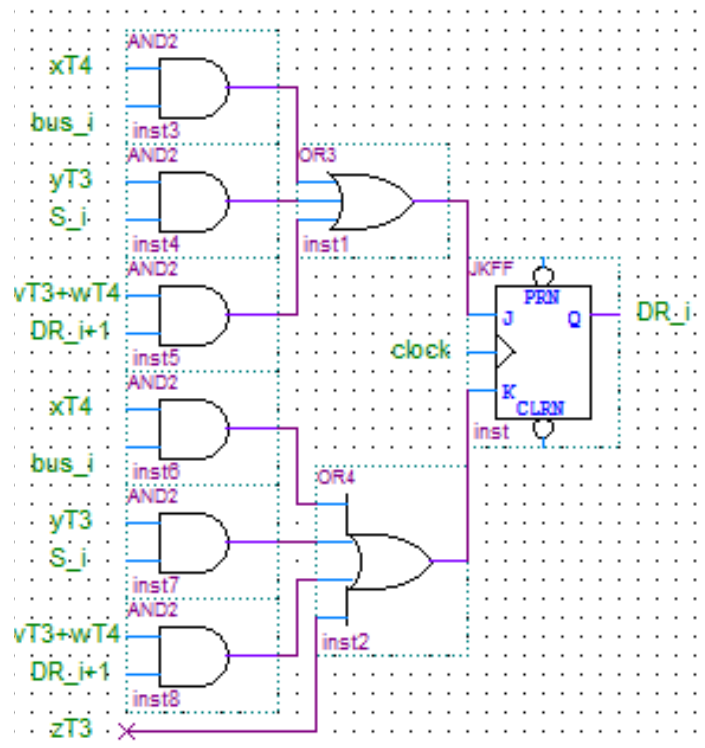
4.2.c Another alternate can employ a comprehensive table, as follows:

Inputs							Present state	Next state	J <sub>i</sub>	K <sub>i</sub>
xT <sub>4</sub>	yT <sub>3</sub>	zT <sub>3</sub>	vT <sub>3</sub> + wT <sub>4</sub>	bus <sub>i</sub>	S <sub>i</sub>	DR <sub>i+1</sub>	DR <sub>i</sub>	DR <sub>i</sub> '	J <sub>i</sub>	K <sub>i</sub>
LD <sub>DR</sub>	IN <sub>DR</sub>	CLR <sub>DR</sub>	SHR <sub>DR</sub>							
1	X	X	X	0	X	X	0	0	0	X
1	X	X	X	0	X	X	1	0	X	1
1	X	X	X	1	X	X	0	1	1	X
1	X	X	X	1	X	X	1	1	X	0
X	1	X	X	X	0	X	0	0	0	X
X	1	X	X	X	0	X	1	0	X	1
X	1	X	X	X	1	X	0	1	1	X
X	1	X	X	X	1	X	1	1	X	0
X	X	1	X	X	X	X	0	0	0	X
X	X	1	X	X	X	X	1	0	X	1
X	X	X	1	X	X	0	0	0	0	X
X	X	X	1	X	X	0	1	0	X	1
X	X	X	1	X	X	1	0	1	1	X
X	X	X	1	X	X	1	1	1	X	0

$$J_i = xT_4 bus_i + yT_3 S_i + (vT_3 + wT_4) DR_{i+1}$$

$$K_i = xT_4 bus'_i + yT_3 S'_i + zT_3 + (vT_3 + wT_4) DR'_{i+1}$$

$\Sigma$  with Half-adder:  
 $S_i = DR_i \oplus CY_{i-1}$   
 $CY_i = DR_i \bullet CY_{i-1}$



Q5

Q5.1 RTL notation is used in the following table to describe the FETCH cycle of a memory reference instruction. Give short explanations of the respective micro-operations in the last column

State	RTL Micro-operations	Explanations
$T_0$	$AR \leftarrow PC$	Address of the instruction to be executed is loaded into the address register
$T_1$	$IR \leftarrow M[AR]$ $PC \leftarrow PC + 1$	The instruction is loaded in the Instruction Register Address of next instruction is prepared
$T_2$	$D_0, D_1, \dots, D_7 \leftarrow DecodeIR(14-12)$ $AR \leftarrow IR(11-0)$ $I \leftarrow IR(15)$	Opcode is decoded Address of operand (if direct addressing) or pointer to the address (if indirect addressing) goes to AR Direct/indirect bit goes to bit I
$I \cdot D_7 \cdot T_3$	$AR \leftarrow M[AR]$	

Q5.2a Use RTL notation to describe the EXECUTION cycle of each of the three newly added instructions; give the corresponding control signals in terms of the instruction code and the sequence counter. Use as many lines for your table as you need; if needed more, please use the back of the page.

<b>ASR F020</b>	<b>Arithmetic Right Shift (DR &lt;- DR/2)</b>
<b>DIV F010</b>	<b>Divide by 4 (DR &lt;- DR/4)</b>
<b>TAT F008</b>	<b>Swap AC with DR (DR &lt;-&gt; AC)</b>

Instr.	Condition	RTL Micro-operation	Control signals						
			Bus Select $S_2, S_1, S_0$	SHR <sub>DR</sub>	LD <sub>DR</sub>	LD <sub>AC</sub>	INR <sub>SC</sub>	CLR <sub>SC</sub>	ALU <sub>DR</sub>
ASR	$D_7 I T_3 IR_5$	$DR \leftarrow shr DR$ $SC \leftarrow 0$	000	1	0	0	0	1	0
TAT	$D_7 I T_3 IR_3$	$AC \leftarrow DR,$ $DR \leftarrow AC$ $SC \leftarrow 0$	100	0	1	1	0	1	1
DIV	$D_7 I T_3 IR_4$	$DR \leftarrow shr DR$ $SC \leftarrow SC + 1$	000	1	0	0	1	0	0
	$D_7 I T_4 IR_4$	$DR \leftarrow shr DR$ $SC \leftarrow 0$	000	1	0	0	0	1	0

Q5.2b Derive the equations of the control signals from the above table:

$$SHR_{DR} = D_7 I T_3 IR_5 + D_7 I T_3 IR_4 + D_7 I T_4 IR_4$$

$$LD_{DR} = D_7 I T_3 IR_3 = LD_{AC} = ALU_{DR}$$

$$INR_{SC} = D_7 I T_3 IR_4$$

$$CLR_{SC} = D_7 I T_3 IR_5 + D_7 I T_3 IR_3 + D_7 I T_4 IR_4$$

$$S_2 = D_7 I T_3 IR_3$$

$$S_1 = 0$$

$$S_0 = 0$$

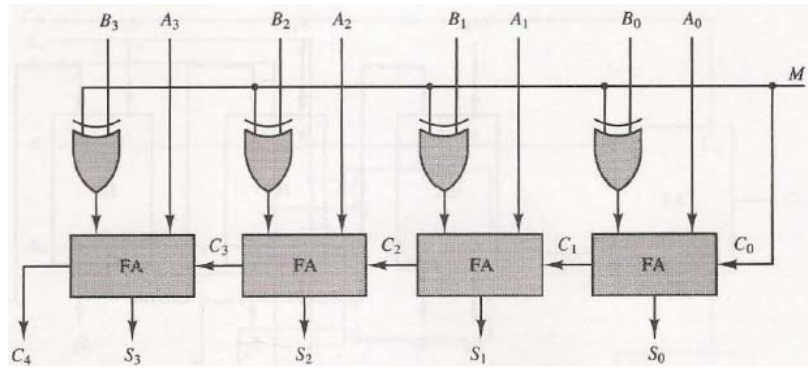
**NOTE:** The term  $D_7 I T_3 IR_x$  is obtained from the instruction in the instruction register IR as follows:

<b>ASR</b>	F	0	2	0
	IR	1111	0000	0010 0000
Decoder	I	$D_7$		$IR_5$
	<b>TAT</b>	F	0	1
IR	1111	0000	0001	0000
	Decoder	I	$D_7$	$IR_4$
<b>DIV</b>	F	0	0	8
	IR	1111	0000	0000 1000
Decoder	I	$D_7$		$IR_3$

## B

### Question 1

Consider the following logic diagram:



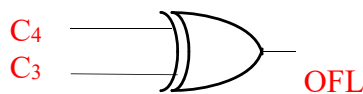
- a) What is the  $C_4, S_3, S_2, S_1, S_0$  value if:
- (1) Register A = 1101, Register B = 0110, and M=0
  - (2) Register A = 1101, Register B = 0110, and M=1

Write your calculations and results in the last two columns of the following table

- b) Given that registers A, B and S contain signed numbers in **2's complement representation**, convert the binary numbers  $A_3 A_2 A_1 A_0, B_3 B_2 B_1 B_0$  and  $S_3 S_2 S_1 S_0$  to decimal, and write your results and calculations in the following table:

$A_3 A_2 A_1 A_0$ (binary/ decimal)	$B_3 B_2 B_1 B_0$ (binary/ decimal)	M	$C_4$	a) $S_3 S_2 S_1 S_0$ (binary)	b) $S_3 S_2 S_1 S_0$ (decimal)	c) Are the results provided by the logic circuit correct?, i.e., are they in accord with the results you obtain if you do the same operations yourself
$A = 1101 = - A $ $ A  = -1101 =$ 2's compl(1101) $ A  = 0011 = 3$ $A = -3$	0110 = +6	0	1	Carry: $c_4 c_3 c_2 c_1 c_0$ 11000 1101 +0110 =(1)0011	$S = A + B$ $S = -3 + 6$ = 0011 = +3	<b>CORRECT!</b>
1101	0110 = +6	1	1	Carry: 10011 1101 +1001 (1)0111	$S = A + B$ $S = -3 - 6$ = 0111 = +7 !?	$A - B = -3 - 6 = -9$ => <b>Overflow</b> , since the result should be in the domain [-8, +7)

- d) Expand your adder with a circuit that can signalize overflow





### Question 4

Two memory-reference instructions in the Basic Computer are to be changed to the instructions specified in the following table.

Symbol	Opcode	Symbolic designation	Description in words
ADM	001	$M[EA] \leftarrow AC + M[EA]$	Add AC to memory
BNA	011	If $(AC < 0)$ then $(PC \leftarrow EA)$	Branch if AC is negative

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** that 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 content of AC temporary.

### Question 5

Column (3) of the following table shows the content of a memory segment of the Basic Computer that stores a machine language program and its operands.

- Convert from binary to hexadecimal both the address and the memory contents (columns (1) and (3)) and write your results in columns (2) and (4), respectively.
- Use Table 6 from Annex to convert the machine language program (stored in the first six memory locations) to symbolic operation codes (assembly language). Use Table 6 from Annex to convert

Symbol	RTL
ADM	$D_1T_4: TR \leftarrow AC$ $D_1T_5: DR \leftarrow M[EA]$ $D_1T_6: AC \leftarrow AC + DR$ $D_1T_7: M[EA] \leftarrow AC$ $D_1T_8: DR \leftarrow TR$ $D_1T_9: AC \leftarrow DR, SC \leftarrow 0$
BNA	$D_3T_4: \text{If } (AC(15) - 1) \text{ then } (PC \leftarrow AR), SC \leftarrow 0$

the machine language program (stored in the first six memory locations) to symbolic operation codes (assembly language). Write the assembly language instructions in the corresponding cells of column (5) of the above table. Use hexadecimal representation for the address field of the instructions.

- Fill out the last column (6) of the table with the content of the accumulator after the execution of each instruction.

Memory Address		Memory Content		Assembly /operands in HEX	AC after execution of instr.
Binary (1)	Hex (2)	Binary (3)	Hex (4)	(5)	(6)
1000 0001 0111	817	0111 1000 0000 0000	7800	CLA	0000
1000 0001 1000	818	0010 1000 0001 1101	281D	LDA 81D	0121
1000 0001 1001	819	0001 1000 0001 1110	181E	ADD 81E	0130
1000 0001 1010	81A	1000 1000 0001 1111	881F	AND 81F I	0100
1000 0001 1011	81B	0011 1000 0001 1101	381D	STA 81D	0100
1000 0001 1100	81C	0111 0000 0000 0001	7001	HLT	0100
1000 0001 1101	81D	0000 0001 0010 0001	0121		
1000 0001 1110	81E	0000 0000 0000 1111	000F		
1000 0001 1111	81F	0000 1000 0010 0000	0820		
1000 0010 0000	820	0000 0001 1100 0100	01C4		

The instruction list

Symbol	Hex code	Symbol	Hex code
AND	0 or 8	CIR	7080
ADD	1 or 9	CIL	7040
LDA	2 or A	INC	7020
STA	3 or B	SPA	7010
BUN	4 or C	SNA	7008
BSA	5 or D	SZA	7004
ISZ	6 or B	SZE	7002
CLA	7800	HLT	7001
CLE	7400	INP	F800
CMA	7200	OUT	F400
CME	7100	SKI	F200

## Question 6

a) Write a subroutine (MUL) in assembly language that multiplies two positive numbers by a repeated addition method. For example, to multiply 6 x 4, the subroutine evaluates the product by adding the multiplicand (6) four times (multiplier), i.e., 6+6+6+6.

Write your program using instructions of the Basic Computer given in the Table 6 from the Annex. Your subroutine should be stored in the memory of your Basic Computer beginning with address A00.

Before invoking your subroutine MUL, the calling program places:

- the multiplicand (6 in the above example) in memory location at address 9FF;
- the multiplier (4 in the above example) in the accumulator.

The result (the product of the two numbers) is passed by subroutine to the calling program through the accumulator.

**NOTE:** You are allowed to use different ways to pass parameters, but full explanations should be provided.

### Solution:

ORG HEX 9FF
MLP, 0000
MUL, 0000
CMA
INC /initialize CTR to
STA CTR /"negative multiplier"
CLA
LOP, ADD MLP
ISZ CTR
BUN LOP
BUN MUL I
CTR, 0000

b) This part is independent of part (a).

The coordinates of two 2-dimensional vectors ( $a_x$ ,  $a_y$ ,  $b_x$ ,  $b_y$ ) are stored at consecutive addresses starting at F01. Write an assembly language program that calculates the scalar (dot) product of two 2-dimensional vectors, assuming that all their coordinates are positive numbers:

$$p = a_x b_x + a_y b_y$$

Your program is stored in memory beginning with location at address 100, and it has to place the product "p" at address F00.

To calculate partial products ( $a_x b_x$  and  $a_y b_y$ ), you should call subroutine MUL. Parameters have to be passed as follows:

- memory location 9FF is used to pass a factor, say  $a_x$ ;
- the other factor (i.e.,  $b_x$ ) is passed to the subroutine through the accumulator;
- the product  $a_x b_x$  is passed back to your program by subroutine MUL through the accumulator.

### Solution:

ORG HEX 100
CLA /to initialise P = 0
STA P
LDA AX /prepare $a_x$ in MLP
STA MLP
LDA BX /prepare $b_x$ in AC
BSA MUL / call MUL to get $a_x b_x$
ADD P / $P \leftarrow P + a_x b_x = 0 + a_x b_x$
STA P
LDA AY /prepare $a_y$ in MLP
STA MLP
LDA BY /prepare $b_y$ in AC
BSA MUL / call MUL to get $a_y b_y$
ADD P / $P \leftarrow P + a_y b_y = a_x b_x + a_y b_y$
STA P
HLT
ORG HEX F00
P
AX
AY
BX
BY

# ANNEX

## The 8-bit Computer of Lab 4

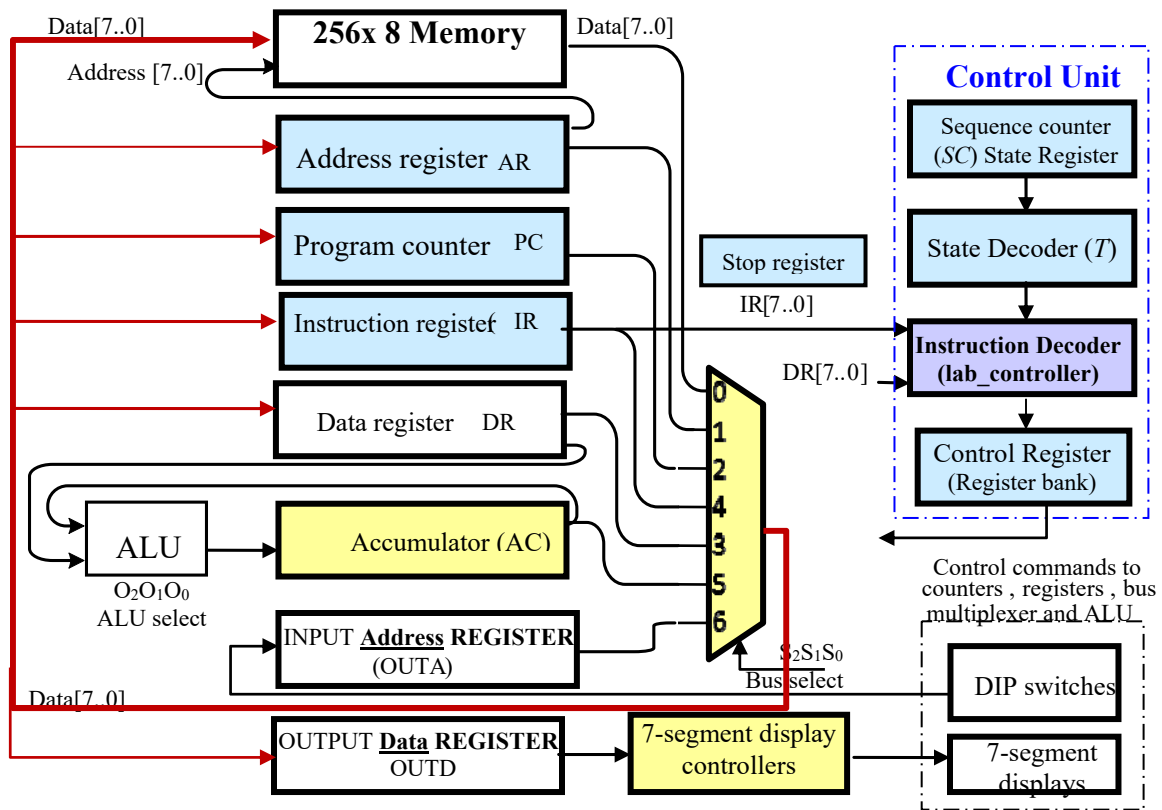


Fig. 1. Block diagram of the 8-bit Computer of Lab 4

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

Instant	Description	Notation (RTL)
T <sub>0</sub>	Load the program counter PC in AR, and increment PC	T <sub>0</sub> : AR ← PC T <sub>0</sub> S' : PC ← PC + 1
T <sub>1</sub>	Read instruction from memory and put it in the instruction register IR	T <sub>1</sub> : IR ← M [AR]
T <sub>2</sub>	This cycle is not used to allow for the new value of the instruction to propagate in the controller	(nothing)
T <sub>3</sub>	If X <sub>1</sub> => it is a <i>register - reference instruction</i> that will be executed now (RRI)	T <sub>3</sub> X <sub>1</sub> : execute an RRI instruction (Table 4)
T <sub>3</sub>	If X <sub>0</sub> or X <sub>2</sub> , read the memory address of the operand and place it in AR, and increment PC. Recall that (X <sub>0</sub> + X <sub>2</sub> ) = IR(6)'	T <sub>3</sub> IR(6)' : AR ← PC T <sub>3</sub> IR(6)' S' : PC ←
T <sub>4</sub>	Read memory address into AR	T <sub>4</sub> IR(6)' : AR ← M
T <sub>5</sub>	If <i>indirect addressing</i> , read the <i>operand address</i> from memory location pointed to by AR	T <sub>5</sub> X <sub>2</sub> : AR ← M [AR]
T <sub>5</sub>	If <i>direct addressing</i> , don't do anything, as the operand address is already in AR since T <sub>6</sub>	T <sub>5</sub> X <sub>0</sub> : (nothing)
starting from T <sub>6</sub>	Execute the memory-reference instructions (MRI) described in Table 2	(see Table 2)

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

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

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

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

Table 4: Function table of the 8-bit ALU

$O_2 O_1 O_0$	ALU operation
000	$F = AC + DR$ / addition
001	$F = AC - DR$ / subtraction
010	$F = \text{ashl } AC$ / arithmetic shift left
011	$F = \text{ashr } AC$ / arithmetic shift right
100	$F = AC \wedge DR$ / AND
101	$F = AC \vee DR$ / OR
110	$F = AC'$ / 1's complement
111	$F = DR$ / transfer DR to 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 / L'ordinateur de base 16 bits du manuel

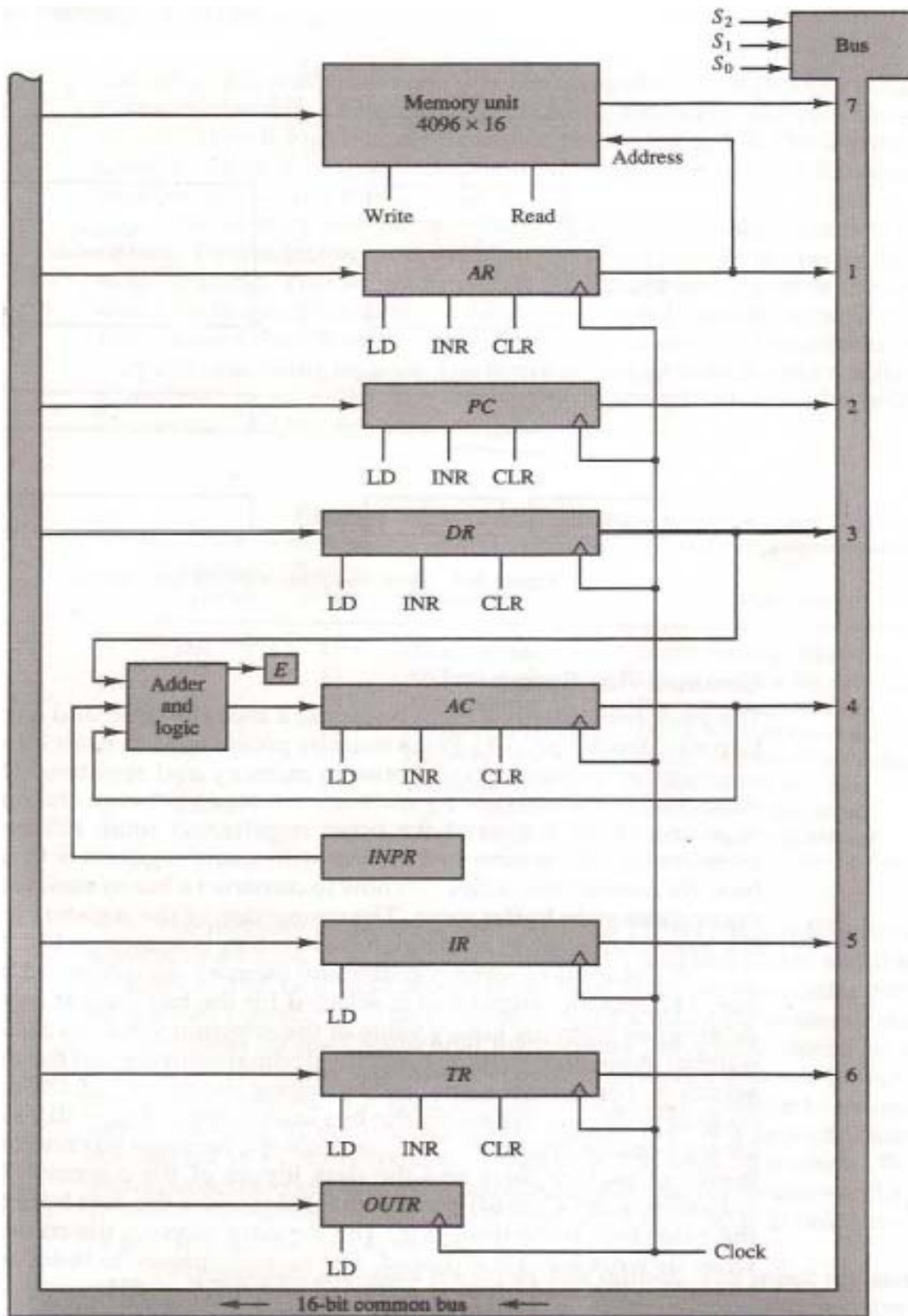


Fig. 2. Basic Computer registers connected to a common bus

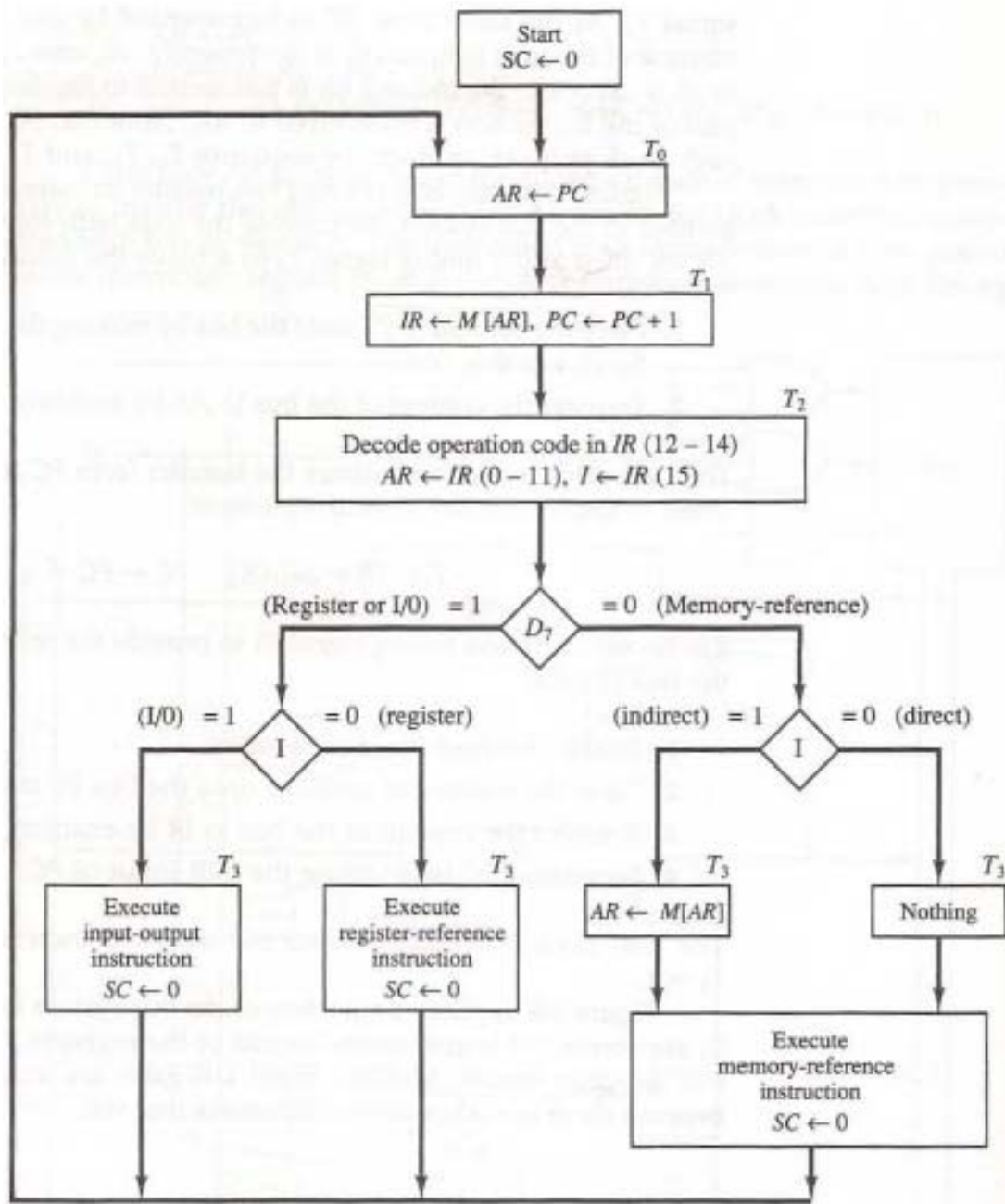


Fig. 3. Principle flowchart for instruction cycle of the 16-bit **Basic Computer** of the textbook

Table 6.

## Basic Computer 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 $E, AC$
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

Table 7. RTLs of the Basic Computer

<b>Fetch</b>	$R'T_0:$	
	$R'T_1:$	
<b>Decode</b>	$R'T_2:$	
<b>Indirect</b>	$D_7IT_3:$	$AR \leftarrow M[AR]$
<b>Interrupt:</b>	$T_0'T_1'T_2'IEN(FGI+FGO):$	$R \leftarrow 1$
	$RT_0:$	$AR \leftarrow 0, TR \leftarrow PC$
	$RT_1:$	$M[AR] \leftarrow TR, PC \leftarrow 0$
	$RT_2:$	$PC \leftarrow PC + 1, IEN \leftarrow 0, R \leftarrow 0, SC \leftarrow 0$
<b>Memory - reference instructions</b>		
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	$D_6T_4:$	$DR \leftarrow M[AR]$
	$D_6T_5:$	$DR \leftarrow DR + 1$
	$D_6T_6:$	$M[AR] \leftarrow DR, SC \leftarrow 0$
	$D_6T_6DR':$	$PC \leftarrow PC + 1$
<b>Register - reference instructions</b>		
	$D_7IT_3 = r$	(common to all RRI)
	$IR(i) = B_i$	( $i = 0, 1, 2, \dots, 11$ )
	$r:$	$SC \leftarrow 0$
CLA	$rB_{11}:$	$AC \leftarrow 0$
CLE	$rB_{10}:$	$E \leftarrow 0$
CMA	$rB_9:$	$AC \leftarrow AC'$
CME	$rB_8:$	$E \leftarrow E'$
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	$rB_4AC'(15):$	$PC \leftarrow PC + 1$
SNA	$rB_3AC(15):$	$PC \leftarrow PC + 1$
SZA	$rB_2AC':$	$PC \leftarrow PC + 1$
SZE	$rB_1E':$	$PC \leftarrow PC + 1$
HLT	$rB_0:$	$S \leftarrow 0$
<b>Input - output instructions</b>		
	$D_7IT_3 = p$	(common to all IOI)
	$IR(i) = B_i$	( $i = 6, 7, 8, 9, 10, 11$ )
	$p:$	$SC \leftarrow 0$
INP	$pB_{11}:$	$AC(0-7) \leftarrow INPR, FGI \leftarrow 0$
OUT	$pB_{10}:$	$OUTR \leftarrow AC(0-7), FGO \leftarrow 0$
SKI	$pB_9FGI:$	$PC \leftarrow PC + 1$
SKO	$pB_8FGO:$	$PC \leftarrow PC + 1$
ION	$pB_7:$	$IEN \leftarrow 1$
IOF	$pB_6:$	$IEN \leftarrow 0$