Non-profit
Joint-stock
company

ALMATY UNIVERSITY OF POWER ENGINEERING AND TELECOMMUNICATIONS

IT-engineering

DIGITAL CIRCUIT DESIGN
Part 2

Methodical guidelines for performance of Laboratory assignments
for students of specialty
5B070400 – Computer hardware and software

Almaty 2017

COMPILERS: O.T. Shanaev, G.D. Musapirova. A.N. Utegulov. Digital circuit design. Part 2. Methodical guidelines for performance laboratory assignments, designed for students of specialty 5B070400 – Computer hardware and software. – Almaty: AUPET, 2017. – 45 p.

These Methodical guidelines present the assignments for organization of laboratory classes for the second part of the "Digital circuit design" discipline, devoted to study of arrangement of microprocessor system operation.

Delivery of the proposed laboratory assignments is organized in the form of virtual experiments carried out by modeling the work of the corresponding program structures, completed on the basis of the application of various (in terms of size and method of addressing) instructions (related to the topic) and designed to organize the operation of a microprocessor system based on the Intel 8085 microprocessor.

Practical testing of the working capacity of the presented program structures is suggested to be carried out using the 8085 Simulator IDE, the functionality and tool set of which provide wide opportunities for it.

Pic. 6, tabl. 5, bibl. – 4 items.

Reviewer: senior teacher of «LCD department» G.S.Akhetova

Published according to the publication plan of the non-profit joint-stock company “Almaty University of Power Engineering and Telecommunications” for 2017.

Consolidated plan 2017 y., p.29

Oryngali Tolegenovich Shanaev
Gulzada.Dauletbekovna Musapirova
Aidos Nurzhanovich Utegulov

DIGITAL CIRCUIT DESIGN
Part 2

Methodical guidelines for performance of Laboratory assignments
for students of specialty
5B070400 – Computer hardware and software

Editor L.Ya. Korobeinikova
Standardization specialist N.K.Moldabekova

Signed in print
Circulation 30 ex.
Volume 2,8

Format 60x84/16
Typography paper № 1
Order Price 1400 tg

Copying Office
Non-profit joint-stock company
"Almaty University of Power Engineering and Telecommunications"
050013, Almaty, Baytursynov st, 126

Introduction

These methodical guidelines contain assignments that are devoted to the organization of laboratory classes for the second part of the discipline "Digital circuit design", devoted to the study to arrangement of microprocessor system operation, built on the basis of individual microprocessors. For the formation of the most well-rounded knowledge of the microprocessor system, it is necessary to know both the functions of its hardware blocks and to master the principles of drawing up programs of actions and interactions of the functional components of the system with the help of instructions from the instruction system of the microprocessor (software), which together provide the physical implementation of the solution to the stated problem.

Table 1 shows descriptions of Intel 8085 microprocessor instructions and their generalized characteristics.

Table 1 – Intel 8085 microprocessor instruction set

Mnemonic	Code	Flags	Byte	Tact	Cycle	Notations
1	2	3	4	5	6	7
Move instructions						
MOV r_1, r_2	01DDDSSS	–	1	4	1	$r_1 \leftarrow (r_2)$
MOV M, r	01110SSS	–	1	7	2	$[(HL)] \leftarrow (r)$
MOV r, M	01DDD110	–	1	7	2	$(r) \leftarrow [(HL)]$
MVI r, b_2	00DDD110	–	2	7	2	$r_1 \leftarrow b_2$
MVI M, b_2	36	–	2	10	3	$[(HL)] \leftarrow b_2$
LXI rp,b_3b_2	00PP0001	–	3	10	3	$rp \leftarrow b_3b_2$
LDA b_3b_2	3A	–	3	13	4	$A \leftarrow (b_3b_2)$
STA b_3b_2	32	–	3	13	4	$[b_3b_2] \leftarrow (A)$
LHLD b_3b_2	2A	–	3	16	5	$(HL) \leftarrow [b_3b_2]$
SHLD b_3b_2	22	–	3	16	5	$[b_3b_2] \leftarrow (HL)$
LDAX rp	00PP0010	–	1	7	2	$A \leftarrow [(r_p)]$
STAX rp	00PP1010	–	1	7	2	$[(rp)] \leftarrow (A)$
XCHG	EB	–	1	4	1	$(HL) \leftrightarrow (DE)$
Instructions for arithmetic and logical operations						
ADD r	10000SSS	+	1	4	1	$A \leftarrow (A) + (r)$
ADD M	86	+	1	7	2	$A \leftarrow (A) + [(HL)]$
ADI b_2	C6	+	2	7	2	$A \leftarrow (A) + b_2$
ADC r	10001SSS	+	1	4	1	$A \leftarrow (A) + (r) + (T_C)$
ADC M	8E	+	1	7	2	$A \leftarrow (A) + [(HL)] + (T_C)$
ACI b_2	CE	+	2	7	2	$A \leftarrow (A) + b_2 + (T_C)$
SUB r	10010SSS	+	1	4	1	$A \leftarrow (A) – (r)$
SUB M	96	+	1	7	2	$A \leftarrow (A) – [(HL)]$

Table 1 – Intel 8085 microprocessor instruction set (continued)

1	2	3	4	5	6	7
SUI b_2	D6	+	1	7	2	$A \leftarrow (A) - b_2$
SBB r	10011SSS	+	1	4	1	$A \leftarrow (A) - (r) - (T_C)$
SBB M	9E	+	1	7	2	$A \leftarrow (A) - [(HL)] - (T_C)$
SBI b_2	DE	+	2	7	2	$A \leftarrow (A) - b_2 - (T_C)$
INR r	00DDD100	(+)	1	4	1	$r \leftarrow (r) + 1$
INR M	34	(+)	1	10	3	$[(HL)] \leftarrow [(HL)] + 1$
DCR r	00DDD101	(+)	1	4	1	$r \leftarrow (r) - 1$
DCR M	35	(+)	1	10	3	$[(HL)] \leftarrow [(HL)] - 1$
INX rp	00PP0011	–	1	6	1	$rp \leftarrow (rp) + 1$
DCX rp	00PP1011	–	1	6	1	$rp \leftarrow (rp) - 1$
DAD rp	00PP1001	–	1	10	3	$[(HL)] \leftarrow [(HL)] + (r_p)$
DAA	27	+	1	4	1	$A \leftarrow (A)_{2\text{-}10}$
ANA r	10100SSS	+	1	4	1	$A \leftarrow (A) \wedge (r)$
ANA M	A6	+	1	4	1	$A \leftarrow (A) \wedge [(HL)]$
ANI b_2	E6	+	2	7	2	$A \leftarrow (A) \wedge b_2$
XRA r	10101SSS	+	1	4	1	$A \leftarrow (A) \oplus (r)$
XRA M	AE	+	1	7	2	$A \leftarrow (A) \oplus [(HL)]$
XRI b_2	EE	+	2	7	2	$A \leftarrow (A) \oplus b_2$
ORA r	10110SSS	+	1	7	2	$A \leftarrow (A) \vee (r)$
ORA M	B6	+	1	7	2	$A \leftarrow (A) \vee [(HL)]$
ORI b_2	F6	+	2	7	2	$A \leftarrow (A) \vee b_2$
CMP r	10111SSS	+	1	4	1	$(A) - (r)$
CMP M	BE	+	1	4	1	$(A) - [(HL)]$
CPI b_2	FE	+	2	7	2	$(A) - b_2$
CMA	2F	–	1	4	1	$A \leftarrow \overline{(A)}$
STC	37	C	1	4	1	$T_C \leftarrow 1$
CMC	3F	C	1	4	1	$T_C \leftarrow \overline{(T_C)}$
RLC	07	C	1	4	1	$A_{7\text{-}1} \leftarrow (A_{6\text{-}0})$, $A_0 \leftarrow (A_7)$, $T_C \leftarrow A_7$
RRC	0F	C	1	4	1	$A_{6\text{-}0} \leftarrow (A_{7\text{-}1})$, $A_7 \leftarrow (A_0)$, $T_C \leftarrow A_0$
RAL	17	C	1	4	1	$A_{7\text{-}1} \leftarrow (A_{6\text{-}0})$, $A_0 \leftarrow (T_C)$, $T_C \leftarrow A_7$
RAR	1F	C	1	4	1	$A_{6\text{-}0} \leftarrow (A_{7\text{-}1})$, $A_7 \leftarrow (T_C)$, $T_C \leftarrow A_0$
Control instructions						
JMP b_3b_2	C3	–	3	10	3	$PC \leftarrow b_3b_2$
J_{cond} b_3b_2	11CCCC01	–	3	10	3	Cond = 1: $PC \leftarrow b_3b_2$
CALL b_3b_2	CD	–	3	18	5	$SP \leftarrow (PC)$, $PC \leftarrow b_3b_2$

Table 1 –Intel 8085 microprocessor instruction set (end)

1	2	3	4	5	6	7
POP rp	11PP0001	–	1	10	3	rp ← [(SP)], [(SP) + 1], SP ← (SP) + 2
C_{cond} b_3b_2	11CCC100	–	3	2/5	9/18	Cond = 1: SP ← (PC), PC ← b_3b_2
RET	C9	–	3	10	3	PC ← [(SP)]
R_{cond}	11CCC100	–	3	17/11	5/3	Cond = 1: PC ← [(SP)]
RST n	111nnn111	–	1	11	3	PC ← 8n
SPHL	E9	–	1	6	1	SP ← (HL)
Special instructions						
PUSH r_p	11PP0101	–	1	11	3	SP ← (SP) – 2; [(SP)], [(SP) + 1]← (r_p)
PUSH PSW	F5	–	1	11	3	SP ← (SP) – 2; [(SP)], [(SP) + 1]←(A),(F)
POP PSW	F1	–	1	10	3	F,A ← [(SP)],[(SP) + 1]; SP ← (SP) + 2
XTHL	E3	–	1	18	5	(HL) ↔ [(SP)], [(SP) + 1]
PCHL	F9	–	1	5	1	PC ← (HL)
IN port	DB	–	2	10	3	A ← (port)
OUT port	D3	–	2	10	3	port ← (A)
EI	FB	–	1	4	1	Interruption permission
BI	F3	–	1	4	1	Interruption prohibition
HLT	76	–	1	7	2	Stop
NOP	00	–	1	4	1	No operation
RIM	20	–	1	4	1	Reading mask of interruption
SIM	30	–	1	4	1	Writing mask of interruption

The first column of the table contains instruction mnemonics with registers denoted by r, register pairs by rp, memory cells with indirect addressing by M, third and second bytes of the instruction by b_3b_2, address of the computing unit by “port”. While accessing memory cells by indirect addressing, the addresses of these cells are taken from the register pair H (registers H and L) and, therefore, do not need to be specified in the instruction itself.

In the second column, the first byte of the bl instruction is given in a binary eight-bit representation, if it is necessary to specify the addresses of the operands in them or in two-digit hexadecimal notation for other cases. The numbers of the general purpose register addresses of the data source registers are expressed by the letter S (Data Source), data receiver registers by the letter D (Data Destination), register pairs - by the letter P (Pair). Substituting letter symbols by specific addresses given in table 2, we obtain codes for specific instruction options. The condition codes for the operation indicated in the instruction are indicated by the letter C (Condition), decoding of which is presented in table 3.

Table 2

Registers								Register pairs			
B	C	D	E	H	L	M	A	B	D	H	SP
000	001	010	011	100	101	110	111	00	01	10	11

Attributes are formed in the flags register, the format of which is presented in table 4.

In the code of the restart instruction RST, the three digits marked with the letter n are formed by the interrupt system or indicated by the programmer.

The comparison operation is performed by subtracting the operands and setting the result attributes (Z and S).

Table 3

CCC	Condition	Description of condition
000	NZ	Inequality to zero
001	Z	Equality to zero
010	NC	No carry
011	C	Carry
100	PO	Oddness
101	PE	Evenness
110	P	Plus
111	M	Minus

Table 4

Bits	7	6	5	4	3	2	1	0
Denotations	S	Z	0	AC	0	P	1	C

In the third column the dash means that the execution of the instruction is not accompanied by generation of flag-attributes, the plus sign indicates setting of all attributes, the plus sign in brackets means setting all of the attributes except for the presence or absence of the C carry, and the C symbol means that only the presence or absence of a carry attribute is being generated.

Delivery of the proposed laboratory assignments is organized in the form of virtual experiments carried out by modeling the work of the corresponding program structures, compiled on the basis of the application of various (in terms of size and method of addressing) instructions (related to the topic) and designed to organize the operation of a microprocessor system based on the Intel 8085 microprocessor.

Delivery of the laboratory assignments presented in the methodical guidelines is organized in the form of virtual experiments carried out by modeling the work of the relevant program structures, compiled on the basis of the application of various teams (on the topic) and intended for the implementation of a specific work of the microprocessor system based on the Intel 8085 microprocessor. Practical testing of the working capacity of the presented program

structures is suggested to be carried out using the 8085 Simulator IDE, the functionality and tool set of which provide ample opportunities for this.

In order to run the laboratory assignments on your computer, the 8085 Simulator IDE program must be installed. Before you start working with the simulator in C:\Program Files\8085 Simulator IDE it is recommended to create a folder under the name containing enough information about the student (group number, last name and first name) and then all the created files in it. This will make it easier to work with these files later.

All the works presented in the methodical guidelines are carried out in a uniform order, which is detailed in the assignment of the first lab. This order briefly looks as follows:

1) Prepare the simulator to work with the program implementing the task at hand and verify the reliability of the instruction codes written in memory.

2) Review the work of individual program blocks and analyze the results of each of them.

3) Investigate the operation of the program in a step mode and analyze the actions performed by the instructions in each of the program blocks and draw appropriate conclusions about the operation of each of program blocks.

The code for any instruction of the Intel 8085 microprocessor can be determined from Table 5, by sequentially reading the hrizontal and vertical values of its placement in the table.

Table 5

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	NOP	LXI B	STAX B	INX B	INR B	DCR B	MVI B	RLC		DAD B	LDAX B	DCX B	INR C	DCR C	MVI C	RRC	0
1		LXI D	STAX D	INX D	INR D	DCR D	MVI D	RAL		DAD D	LDAX D	DCX D	INR E	DCR E	MVI E	RAR	1
2	RIM	LXI H	SHLD	INX H	INR H	DCR H	MVI H	DAA		DAD H	LHLD	DCX H	INR L	DCR L	MVI L	CMA	2
3	SIM	LXI SP	STA	INX SP	INR M	DCR M	MVI M	STC		DAD SP	LDA	DCX SP	INR A	DCR A	MVI A	CMC	3
4	MOV B,B	MOV B,C	MOV B,D	MOV B,E	MOV B,H	MOV B,L	MOV B,M	MOV B,A	MOV C,B	MOV C,C	MOV C,D	MOV C,E	MOV C,H	MOV C,L	MOV C,M	MOV C,A	4
5	MOV D,B	MOV D,C	MOV D,D	MOV D,E	MOV D,H	MOV D,L	MOV D,M	MOV D,A	MOV E,B	MOV E,C	MOV E,D	MOV E,E	MOV E,H	MOV E,L	MOV E,M	MOV E,A	5
6	MOV H,B	MOV H,C	MOV H,D	MOV H,E	MOV H,H	MOV H,L	MOV H,M	MOV H,A	MOV L,B	MOV L,C	MOV L,D	MOV L,E	MOV L,H	MOV L,L	MOV L,M	MOV L,A	6
7	MOV M,B	MOV M,C	MOV M,D	MOV M,E	MOV M,H	MOV M,L	HLT	MOV M,A	MOV A,B	MOV A,C	MOV A,D	MOV A,E	MOV A,H	MOV A,L	MOV A,M	MOV A,A	7
8	ADD B	ADD C	ADD D	ADD E	ADD H	ADD L	ADD M	ADD A	ADC B	ADC C	ADC D	ADC E	ADC H	ADC L	ADC M	ADC A	8
9	SUB B	SUB C	SUB D	SUB E	SUB H	SUB L	SUB M	SUB A	SBB B	SBB C	SBB D	SBB E	SBB H	SBB L	SBB M	SBB A	9
A	ANA B	ANA C	ANA D	ANA E	ANA H	ANA L	ANA M	ANA A	XRA B	XRA C	XRA D	XRA E	XRA H	XRA L	XRA M	XRA A	A

Table 5 (continued)

B	ORA B	ORA C	ORA D	ORA E	ORA H	ORA L	ORA M	ORA A	CMP B	CMP C	CMP D	CMP E	CMP H	CMP L	CMP M	CMP A	B
C	RNZ	POP B	JNZ	JMP	CNZ	PUSH B	ADI	RST 0	RZ	RET	JZ		CZ	CALL	ACI	RST 1	C
D	RNC	POP D	JNC	OUT	CNC	PUSH D	SUI	RST 2	RC		JC	IN	CC		SBI	RST 3	D
E	RPO	POP H	JPO	XTHL	CPO	PUSH H	ANI	RST 4	RPE	PCHL	JPE	XCHG	CPE		XRI	RST 5	E
F	RP	POP PSW	JP	DI	CP	PUSH PSW	ORI	RST 6	RM	SPHL	JM	EI	CM		CPI	RST 7	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

1 Laboratory assignment № 1. Data transfers

Objectives:

- getting started with the structure of the microprocessor system (MPS) based on Intel 8085 microprocessor;
- mastering the work order with the program simulator 8085 Simulator IDE and learning its functionality;
- mastering the order of application of the peripheral device for data input / output;
- mastering the principles of organization of cyclic program structures;
- studying the actions performed by different (by size and method of addressing) instructions for transferring data from the Intel 8085 microprocessor.

1.1 Workplace equipment

Computer, 8085 Simulator IDE program.

1.2 The program for implementation of data transfers

```
;*****************************************************
                        Data Transfers
;*****************************************************
;                       Writing to Memory
;=============================================
;                       Single Data
                MVI     A,0Fh
                OUT     02h
                STA     00D0h
;=============================================
;                       Data Array-1
;-----------------------------------------------------------------------
                MVI     C,07h
                LXI     D,00D0h
```

```
            IN      01h         ;Initial Data
DT_1:       ADI     13h         ;Step
            OUT     02h
            INX     D
            STAX    D
            DCR     C
            JNZ     DT_1
;=============================================
;                   Data Array-2
;---------------------------------------------------------------------
            XCHG
            MVI     C,08h
DT_2:       ADI     09h         ;Step
            OUT     02h
            INX     H
            MOV     M,A
            DCR     C
            JNZ     DT_2
;*************************************************
;                   Data Access
;=============================================
;                   Conversion-1
;---------------------------------------------------------------------
            LXI     D,00EFh
            MVI     C,08h
CNV_1:      MOV     A,M
            OUT     03h
            RRC
            OUT     04h
            STAX    D
            DCX     H
            DCX     D
            DCR     C
            JNZ     CNV_1
;=============================================
;                   Conversion-2
;---------------------------------------------------------------------
            MVI     C,07h
CNV_2:      MOV     A,M
            OUT     03h
            RAR
            OUT     04h
            XCHG
            MOV     M,A
```

```
                DCX     H
                DCX     D
                XCHG
                DCR     C
                JNZ     CNV_2
;=============================================
;                       Single Access
;-----------------------------------------------------------------------
                LDA     00D0h
                OUT     03h
                CMA
                OUT     04h
                STA     00E0h
;=============================================
                HLT
;-----------------------------------------------------------------------
```

Program clarification.

This program is devoted to investigating the nature of actions when performing various (by size and method of addressing) instructions for data transfers.

In the first part of the program (Writing to Memory), composed in the form of several blocks, the data are transferred from the microprocessor to the memory: in the Single Data block - single forwarding by an instruction with direct addressing; in the blocks Data Array-1 and Data Array-2 - sending, the array of data (in the form values of members of the arithmetic progression) instructions with indirect addressing.

In blocks (Conversion-1, Conversion-2 and Single Access) of the second part of the program (Data Access) with the help of similar instructions used earlier, access to data from one part of memory is carried out, and after the corresponding transformation, they are transferred to another area of memory.

Entering the initial value from which the generation of values of the members of an arithmetic progression begins, and displaying the data values transferred between the microprocessor and the memory, are carried out using the peripheral device.

1.3 Working assignment

1.3.1 Prepare the simulator to work with the program for implementing data transfers:

1) Start the simulator program with the help of an  icon and select Tools / Assembler in the main simulator window (figure 1.1), while in the Assembler window - File / New.

2) In the Assembler window (figure 1.2) enter the instructions of the program presented above. In this case, the labels are written at the beginning of the line, and the instructions are shifted relatively to the beginning of the line (for example, with the tab key).

3) Save (File / Save As) the recorded program under the name Data Transfers in the previously prepared individual folder, resulting in a file with the extension asm.

4) Open the memory window by selecting Tools / Memory Editor in the main window (figure 1.3).

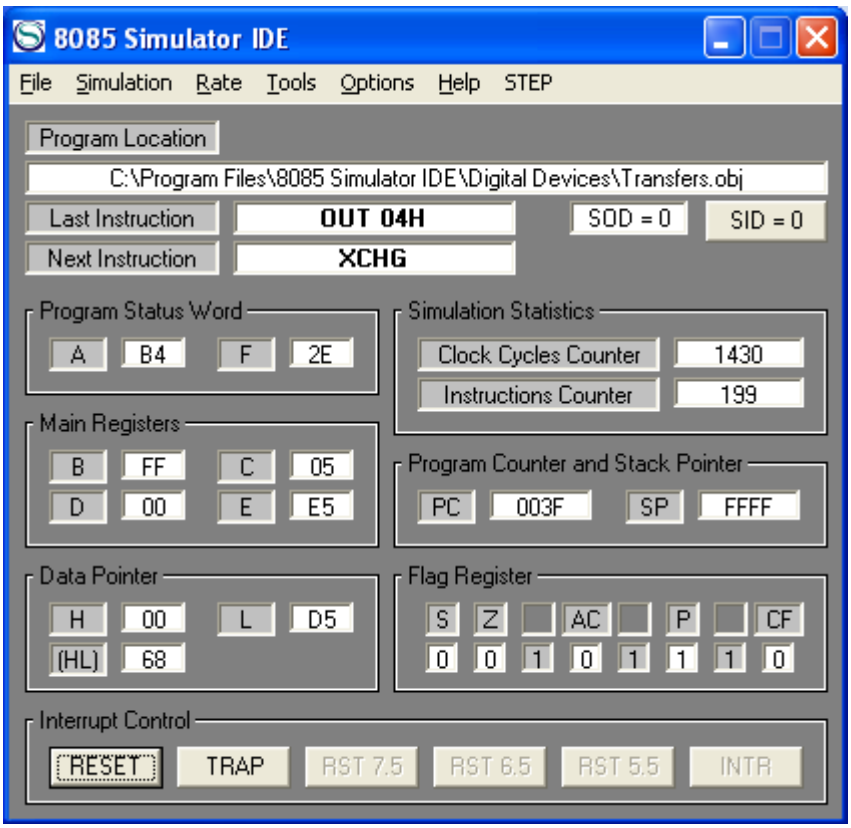

Figure 1.1

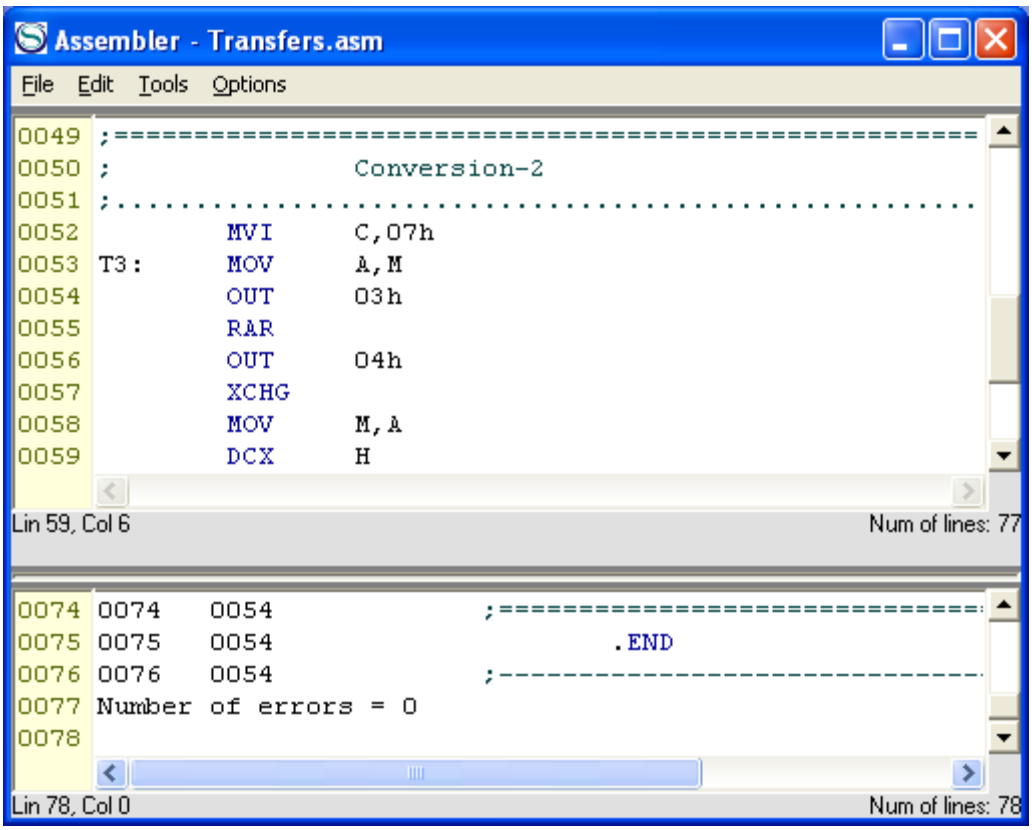

Figure 1.2

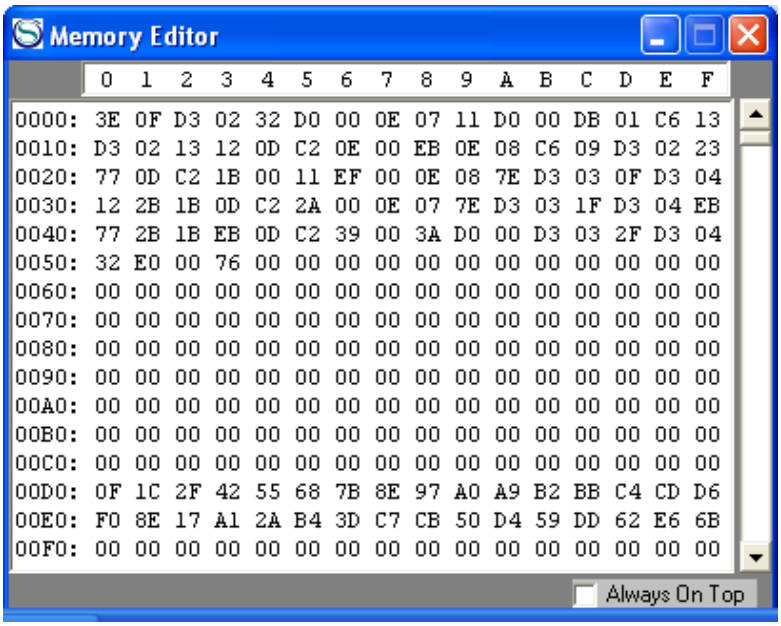

Figure 1.3

5) Assemble the program and load it into the memory of the simulator (select in the Assembler Tools / Assemble & Load window).

6) According to table 5, check and verify the correctness of the instruction codes stored in memory.

7) In the simulator window open the Tools / Peripheral Devices window. In the opened Peripheral Devices window, click the Device 1 button and set the port

address as 01, and then set this port to IN. Similarly, by specifying the addresses of the following ports as 02, 03 and 04, set them to OUT (figure 1.4).

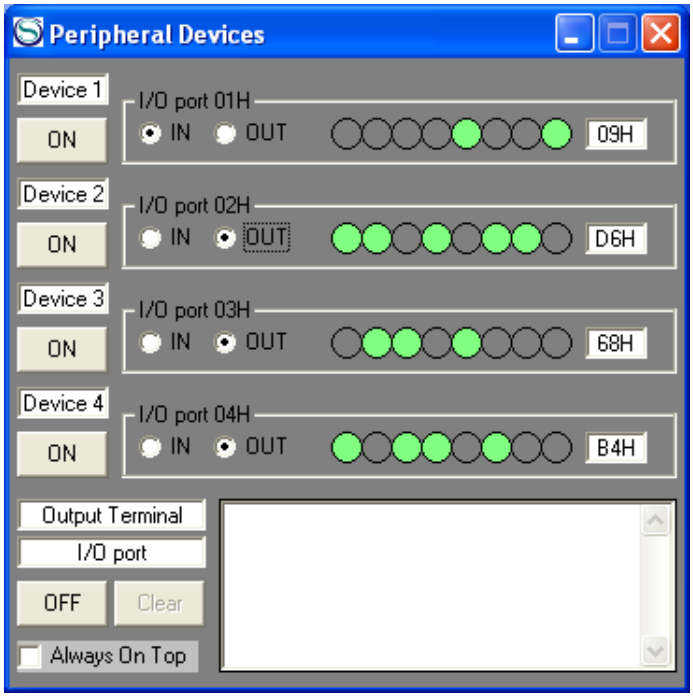

Figure 1.4

1.3.2 Observe the operation of individual program blocks:

1) In the main simulator window, select Tools / Breakpoints Manager and set the control points at the beginning of each block (figure 1.5).

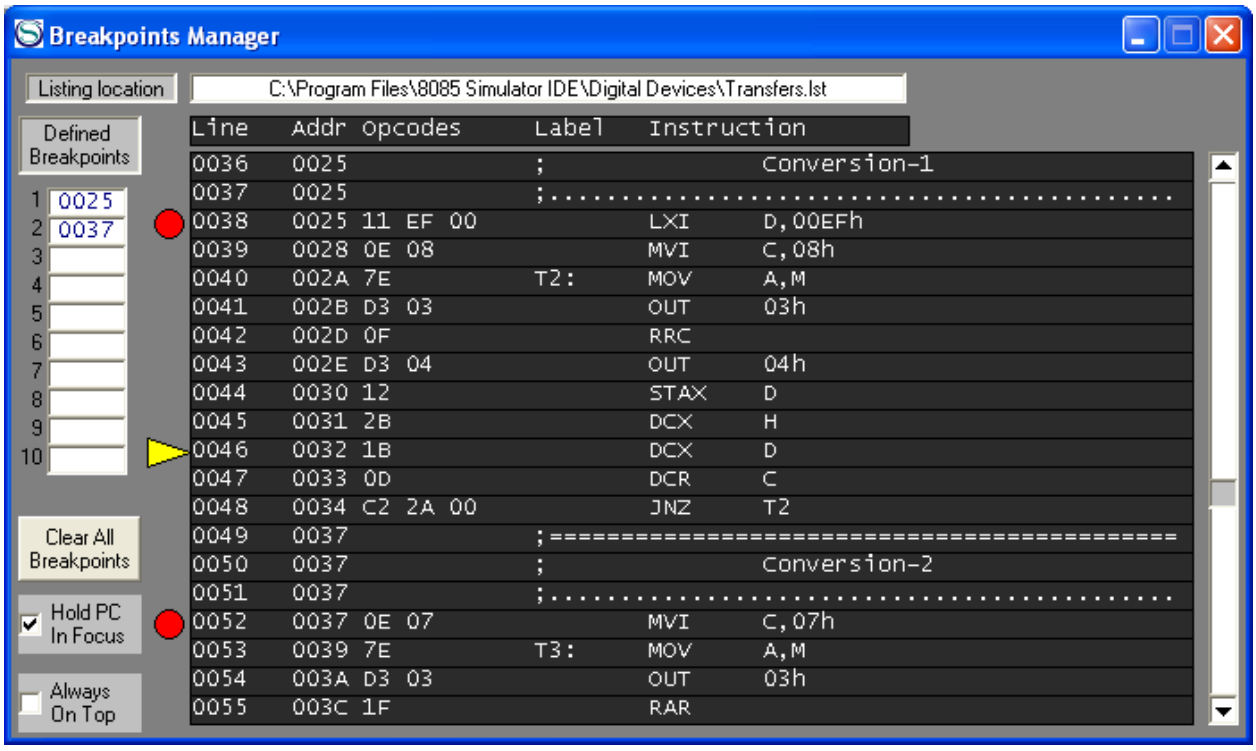

Figure 1.5

2) In the main simulator window, select the Rate / Normal simulation speed and run the simulation (Simulation / Start).

3) At the invitation of the peripheral device (figure 1.6), enter the initial value from which the values of the arithmetic progression begin to form.

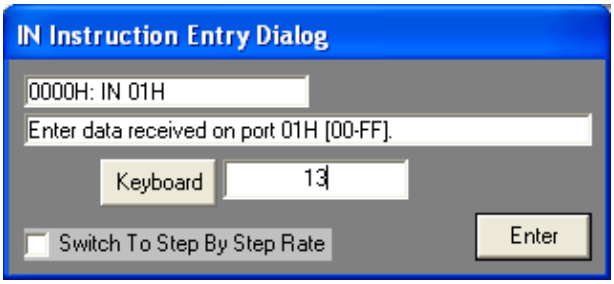

Figure 1.6

4) After each program stop at checkpoints, analyze the results of each of the program blocks, according to the data displayed in the corresponding memory cells.

5) After completion of the program, check the correctness of the results of each program block, namely the correspondence of the data written in the memory (in the blocks Data Array-1 and Data Array-1) to the values of the arithmetic progression members with the specified parameters.

1.3.3 Investigate the operation of the program and analyze the actions performed by the data transfer instructions in each of the program blocks:

1) Update the memory window, i.e. clear the memory area where the data received as a result of the previous program execution check was recorded. This can be done by re-loading the program into the memory of the simulator (by selecting File / Load Program in the main simulator window).

2) In the main simulator window, select the step by step simulation (Rate / Step By Step) and run the simulator (Simulation / Start). Note that at the top of the main window of the simulator appears button STEP, with which you can control the step-by-step mode of the simulator.

3) By sequentially pressing the STEP button and watching the changes in the contents of the corresponding registers displayed in the main simulator window and in the corresponding memory cells, analyze the actions of each instruction in separate blocks of the program. Write the specific observation results in the form of comments to the relevant instructions and draw conclusions about the operation of each of the program blocks.

1.4 Quiz questions

1. Explain the internal structure of the microprocessor and the purpose of its functional components.

2. Give examples of data transfer instructions with register addressing and explain their actions.

3. Give examples of data transfer instructions with direct addressing and explain their actions.

4. Give examples of data transfer instructions with indirect addressing and explain their actions.

5. Give examples of instructions for transferring data from an indirect address and explain their actions.

6. Explain the actions of the instructions LDA 00D0h and STA 00E0h.

7. Explain the actions of LDAX D and STAX B.

8. Explain the actions of the instructions LHLD 00D7h and SHLD 00E7h.

9. Explain the actions of the MOV M, C and MOV E, M instructions.

10. Explain the operation of the XCHG instruction.

11. Explain the actions of the instructions RAL, RAR, RLC and RRC.

12. Explain the order of data I / O using peripherals.

2 Laboratory assignment №2. Single-byte arithmetic

Objectives:

- studying the actions performed by single-byte arithmetic instructions with different addressing methods;
- mastering the principles of organization of cyclic program structures.

2.1 Workplace equipment

Computer, 8085 Simulator IDE.

2.2 The program for implementation of single-byte arithmetic

```
;*****************************************************
;                       Single-Byte Arithmetic
;*****************************************************
;                       Entry Data (for Linear Series)
;-----------------------------------------------------------------------
            LXI     H,00D0h
            IN      01h         ;Data Quantity
            MOV     C,A
            MOV     M,A
            IN      01h         ;Step
            MOV     D,A
            IN      01h         ; Initial Data
;=========================================
;                       Data Array (Linear Series)
DT:         ADD     D
            OUT     02h
```

```
            INX     H
            MOV     M,A
            DCR     C
            JNZ     DT
;=============================================
;                   Cumulative Summation
;-----------------------------------------------------------------------
            LXI     H,00D0h
            MOV     C,M
            MVI     A,00h
            MOV     B,A
SUM:        INX     H
            ADD     M
            JNC     NC_M
            INR     B
;……...
NC_M:       OUT     03h
            MOV     E,A
            MOV     A,B         ;Temporary
            OUT     04h
            MOV     A,E         ;Recovery
;……...
            DCR     C
            JNZ     SUM
;=============================================
;                   Direct Addition
;-----------------------------------------------------------------------
            ADI     0E5h
            JNC     NC_D
            INR     B
;……...
NC_D:       OUT     03h
            MOV     E,A
            MOV     A,B
            OUT     04h
            MOV     A,E
;=============================================
;                   Result of Summation
;-----------------------------------------------------------------------
            LXI     H,00F0h
            MOV     M,A
            INX     H
            MOV     M,B
;*************************************************
```

```
;                       Series Subtraction
;-------------------------------------------------------------------------
            LXI     H,00F0h
            MOV     A,M
            INX     H
            MOV     B,M
;……...
            LXI     H,00D0h
            MOV     C,M
SUB:        INX     H
            SUB     M
            JNC     NB_M
            DCR     B
;……...
NB_M:       OUT     03h
            MOV     E,A
            MOV     A,B
            OUT     04h
            MOV     A,E
;……...
            DCR     C
            JNZ     SUB
;=========================================
;                       Direct Subtraction
;……...
            SUI     0E5h
            JNC     NB_D
            DCR     B
;……...
NB_D:       OUT     03h
            MOV     E,A
            MOV     A,B
            OUT     04h
            MOV     A,E
;=========================================
            HLT
;-------------------------------------------------------------------------
```

The program of clarification.

This program, composed in the form of several blocks, is devoted to the study of the nature of actions when performing various (by size and method of addressing) arithmetic instructions (by addition and subtraction) for single-byte data.

In the Entry Data block, the input parameters are entered (n is the number of

members, d is the step of the progression, and a0 is the initial value), through which data (in the form of arithmetic progression values) is generated in the Data Array and stored in memory.

In the Cumulative Summation block, the sum of the values of the arithmetic progression members stored in the memory in the previous block of the program is calculated.

In the Direct Summation block, the instruction for adding data is performed by an instruction with direct addressing.

The accumulation result is stored in memory in the Result of Summation block.

In the blocks of Series Subtraction and Direct Subtraction, the process of sequential subtraction of the data used for summing the data from the previously obtained value of the sum is implemented.

Entering the initial values for the formation of data (in the form of the values of the members of the arithmetic progression) is carried out using a peripheral device.

Any data changes made during program operation are displayed (after executing the corresponding instructions) in the channels of the peripheral device specified in the program.

2.3 Working assignment

2.3.1 Prepare the stimulator to work with the above program to implement single-byte arithmetic operations (Single-Byte Arithmetic) and verify the correctness of the stored instruction codes. Set port 01 to IN, and the rest to OUT**.**

2.3.2 Perform preliminary monitoring of the operation of individual blocks of the program using Breakpoints Manager and analyze the results of their operation, according to the data displayed in the respective memory cells (rate of simulation - Rate / Normal).

2.3.3 Investigate the operation of the program and examine the actions of the instructions in individual function blocks of the program in a step-by-step mode (Rate / Step By Step). Write the specific observation results in the form of comments to the relevant instructions and draw conclusions about the operation of each of the program blocks.

2.3.4 Upon completion of the program, verify the correctness of the results of each of the program blocks, namely:

- correspondence of the data written into the memory (in the Data Array block) to the values of the members of the arithmetic progression with the given parameters;

- equivalence of the result in the Cumulative Summation block to the analytically calculated value of the sum of the values of the terms of the arithmetic progression by the expression

$$S_n = \frac{2a_1+(n-1)d}{2} n,$$

where a_1 is the value of the first term;

n is the number of terms;

d is the step of the values of the arithmetic progression members;

- obtaining a zero result in the Series Subtraction block.

2.3.5 By running the program again and entering other data, check that the results obtained correspond to the results of the corresponding analytical calculation.

2.4 Quiz questions

1. Give examples of instructions for arithmetic addition and subtraction of data with register addressing and explain their actions.
2. Give examples of instructions for arithmetic addition and subtraction of data with direct addressing and explain their actions.
3. Give examples of instructions for arithmetic addition and subtraction of data with indirect addressing and explain their actions.
4. How do we obtain a two-byte result when adding single-byte data in the program under study?
5. How do we perform looping actions in the program under study?
6. Explain the actions of the ADD C and ADD M instructions.
7. Explain the actions of the SUB D and SUB M instructions.
8. Explain the actions of the SUI C5h and SUB M instructions.
9. Explain the actions of the ADC D and SBB E.
10. Explain the joint action of the DCR C and JNZ DT instructions used in the program.
11. Explain the combined effect of the ADD M and JNC NC_M instructions used in the program.
12. Explain the joint action of the SUB M and JNC NB_M instructions used in the program.

3 Laboratory assignment №3. Two-byte arithmetic

Objectives: to study the nature of the actions performed by single-byte arithmetic instructions with different addressing methods, when performing two-byte arithmetic operations.

3.1 Workplace equipment

Computer, 8085 Simulator IDE.

3.2 The program for implementing double-byte arithmetic

```
;*****************************************************
        ;                   Double-Byte Arithmetic
;*****************************************************
        ;                   Entry Data (for Linear Series)
;-----------------------------------------------------------------------
            LXI     H,00B0h         ;Initial Address
;.......................................................................
            IN      01h             ;Data Quantity
            STA     00A0h
            MOV     M,A
            IN      01h             ;Step (Low Byte)
            MOV     C,A
            IN      02h             ;Step (High Byte)
            MOV     B,A
            IN      01h             ;Initial Data (Low Byte)
            MOV     E,A
            IN      02h             ;Initial Data (High Byte)
            MOV     D,A
;=======================================
                    Data Array (Linear Series)
;-----------------------------------------------------------------------
DT:         MOV     A,E
            ADD     C
            OUT     03h
            INX     H
            MOV     M,A
            MOV     E,A
;.......................................................................
            MOV     A,D
            ADD     B
            OUT     04h
            INX     H
            MOV     M,A
            MOV     D,A
;.......................................................................
            LDA     00A0h
            DCR     A
            STA     00A0h
            JNZ     DT
;=======================================
                    Cumulative Summation
;-----------------------------------------------------------------------
            LXI     H,00B0h
            MVI     A,00h
```

```
            MOV     B,A
            LXI     D,0000h
            MOV     C,M
;.....................................................................................
SUM:        MOV     A,E
            INX     H
            ADD     M
            OUT     03h
            MOV     E,A
;.....................................................................................
            MOV     A,D
            INX     H
            ADC     M
            OUT     04h
            MOV     D,A
;.....................................................................................
            DCR     C
            JNZ     SUM
;=========================================
                    Direct Summation
;-------------------------------------------------------------------------
            MOV     A,E
            ADI     45h
            OUT     03h
            MOV     E,A
;.....................................................................................
            MOV     A,D
            ADI     23h
            OUT     04h
            MOV     D,A
;=========================================
                    Result of Summation
;-------------------------------------------------------------------------
            LXI     H,00F0h
            MOV     A,E
            OUT     03h
            MOV     M,A
;.....................................................................................
            INX     H
            MOV     A,D
            OUT     04h
            MOV     M,A
;=========================================
                    Series Subtraction
```

```
;-------------------------------------------------------------------
            LXI     H,00B0h
            MOV     C,M
;...................................................................................
SUB:        MOV     A,E
            INX     H
            SUB     M
            OUT     03h
            MOV     E,A
;...................................................................................
            MOV     A,D
            INX     H
            SBB     M
            OUT     04h
            MOV     D,A
;...................................................................................
            DCR     C
            JNZ     SUB
;=========================================
                    Direct Subtraction
;-------------------------------------------------------------------
            MOV     A,E
            SUI     45h
            OUT     03h
            MOV     E,A
;...................................................................................
            MOV     A,D
            SBI     23h
            OUT     04h
            MOV     D,A
;=========================================
            HLT
        ;-------------------------------------------------------------------------------
```

Program clarification.

This program, composed in the form of several blocks, is devoted to study of the nature of the actions when performing various (by the size and method of addressing) one-byte arithmetic instructions (addition and subtraction) during implementation of the corresponding operations on two-byte data.

In Entry Data block, the input parameters are entered (n is the number of members, d is the step of the progression, and a_0 is the initial value), through which data (in the form of double-byte arithmetic progression terms) is generated in the Data Array and stored in memory.

In the Cumulative Summation block, the sum of the double-byte values of the arithmetic progression terms stored in the memory in the previous block of the program is calculated.

In the Direct Summation block, the addition of double-byte data is performed by direct-addressing instructions.

The accumulation result is stored in memory in the Result of Summation block.

In blocks Series Subtraction and Direct Subtraction the process of consecutive subtraction of used for summation of two-byte data from the previously obtained value of the sum is realized.

Entering the initial values for the formation of data (in the form of the values of the members of the arithmetic progression) is carried out using a peripheral device.

Any data changes made during program operation are displayed (after executing the corresponding instructions) in the channels of the peripheral device specified in the program.

3.3 Working assignment

3.3.1 Prepare the simulator to work with the above program to implement double-byte arithmetic operations (Double-Byte Arithmetic) and verify the correctness of the stored instruction codes. Set the ports 01 and 02 to IN, and the rest to OUT.

3.3.2 Perform preliminary monitoring of the operation of individual program blocks using Breakpoints Manager and analyze the results of their work, according to the data displayed in the corresponding memory cells (rate of simulation - Rate / Normal).

3.3.3 Investigate the operation of the program and examine the actions of the instructions in individual function blocks of the program in a step-by-step mode (Rate / Step By Step). Specific observation results are written in the form of comments to the relevant teams and draw conclusions about the work of each of the program blocks.

3.3.4 Upon completion of the program, verify the correctness of the results of each of the program blocks, namely:

- correspondence of the data written into the memory (in the Data Array block) to the values of the members of the arithmetic progression with the given parameters;

3.3.5 After restarting the program and entering other data, check that the data received is consistent with the results of the corresponding analytical calculation.

3.4 Quiz questions

1. Give examples of instructions with register addressing, used to implement the operation of double-byte arithmetic data addition and explain their actions.

2. Give examples of instructions with direct addressing used to implement the operation of double-byte arithmetic data addition and explain their actions.

3. Give examples of instructions with register addressing used to implement the operation of double-byte arithmetic subtraction of data and explain their actions.

4. Give examples of instructions with direct addressing used to implement the operation of double-byte arithmetic subtraction of data and explain their actions.

5. How is the execution of operations for adding double-byte data implemented in the program under study?

6. How is the execution of operations for subtracting double-byte data implemented in the program under investigation?

7. Explain the nature of the instructions for adding double-byte data with direct addressing.

8. Explain the actions of the ADD M and ADC M instructions.

9. Explain the actions of the SUB M and SBB M.

10. Explain the actions of the SUI 3Dh and SBI 5Ah instructions.

11. How is the formation of double-byte data values intended for the execution of subsequent arithmetic operations implemented in the program under investigation?

12. How are operations of direct addition and subtraction of two-byte data carried out in the program under investigation?

4 Laboratory assignment №.4. Organization of work with the stack

Objectives:

- mastering the principles of organizing data transfers between the stack and the registers of the microprocessor;
- studying the nature of the actions performed by the two-byte arithmetic summation instructions.

4.1 Workplace equipment

Computer, 8085 Simulator IDE.

4.2 The program for organizing work with the stack

```
;*****************************************************
;                          Stack
;*****************************************************
;                          Entry Data (for Linear Series)
```

```
;-------------------------------------------------------------------
            IN      01h             ;Data Quantity
            MOV     B,A
            IN      01h             ;Step (Low Byte)
            MOV     E,A
            IN      02h             ;Step (High Byte)
            MOV     D,A
            IN      01h             ;Initial Data (Low Byte)
            MOV     L,A
            IN      02h             ; Initial Data (High Byte)
            MOV     H,A
;===================================================
;                   Data Array (Linear Series)
;-------------------------------------------------------------------
            LXI     SP,00B0h        ;Stack Pointer
;.....................................................................
            MOV     C,B
DT:         DAD     D
;.....................................................................
            MOV     A,L
            OUT     03h
            MOV     A,H
            OUT     04h
;.....................................................................
            PUSH    H
            DCR     C
            JNZ     DT
;===================================================
;                   Cumulative Summation
;-------------------------------------------------------------------
            MOV     C,B
            LXI     H,0000h     ;Doublet-Byte Accumulator
SUM:        POP     D
            DAD     D
;.....................................................................
            MOV     A,L
            OUT     03h
            MOV     A,H
            OUT     04h
;.....................................................................
            DCR     C
            JNZ     SUM
;===================================================
;                   Direct Addition
```

```
;-------------------------------------------------------------------
        LXI     D,2C1Dh
        DAD     D
;...................................................................................
        MOV     A,L
        OUT     03h
        MOV     A,H
        OUT     04h
;===========================================
        HLT
;-------------------------------------------------------------------
```

Program clarification.

This program is devoted to studying the principles of organizing data transfers between the stack and the registers of the microprocessor and implementing the operations of adding double-byte data using the DAD instruction.

In the Entry Data block, the input parameters are entered (n is the number of members, d is the step of the progression, and a0 is the initial value), through which data (in the form of double-byte arithmetic progression terms) is generated in the Data Array and stored in memory.

In the Cumulative Summation block, the sum of the double-byte values of the arithmetic progression terms stored in the memory in the previous block of the program is calculated.

In the Direct Summation block, the instruction for adding double-byte data is executed by direct-addressing instructions.

The initial values for data generation (in the form of byte values of double-byte arithmetic progression terms) are entered using a peripheral device.

Any data changes made during program operation are displayed (after executing the corresponding instructions) in the channels of the peripheral device specified in the program.

4.3 Working assignment

4.3.1 Prepare the simulator to work with the above program for implementing two-byte arithmetic operations (Stack) and verify the correctness of the stored instruction codes. Set the ports 01 and 02 to IN, and the rest to OUT.

4.3.2 Perform preliminary monitoring of the operation of individual program blocks using Breakpoints Manager and analyze the results of their operation, according to the data displayed in the corresponding memory cells (rate of simulation is Rate / Normal).

4.3.3 Examine the operation of the program and examine the actions of the instructions in individual function blocks of the program in step-by-step mode (Rate / Step By Step). Write the specific observation results in the form of

comments to the relevant instructions and draw conclusions about the operation of each of the program blocks.

4.3.4 Upon completion of the program, verify the correctness of the results of each of the program blocks, namely:

- correspondence of the two-byte data written in the memory (in the Data Array block) to the values of the arithmetic progression members with the specified parameters;
- reliability of the obtained summation result (in the register pair HL) by calculating the sum of the terms of the arithmetic progression by the formula:

$$S_n = \frac{2a_1 + (n-1)d}{2} n$$

and directly adding the value 2C1Dh.

4.4 Quiz questions

1. Explain the actions of the PUSH H and PUSH D instructions.
2. How do the contents of the SP register change when executing the PUSH H and PUSH D instructions?
3. Explain the actions of the POP H and POP B instructions.
4. How do the contents of the SP register change when the POP H and POP B instructions are executed?
5. How is the formation of double-byte data values intended for performing subsequent addition operations implemented in the program under investigation?
6. Explain the actions of the DAD B and DAD D.
7. Which single-byte instructions replace the execution of the DAD D instruction?
8. Which single-byte instructions replace the execution of the DAD H instruction?
9. Explain the results of executing LXI instructions D, 2C1Dh and DAD D.
10. How is it possible to write table data into memory?
11. How is it possible read the table data from memory?
12. How could I subtract double-byte data using DAD instructions?

5 Laboratory assignment №.5. Implementation of the multiplication instruction

Objective: mastering the principles of compiling complex analytical programs.

5.1 Workplace equipment

Computer, 8085 Simulator IDE.

5.2 Program for implementing the multiplication operation

```
;*****************************************************
;                    Multiplication
;*****************************************************
;                    Preparation
;-----------------------------------------------------------------
            IN      01h         ;Multiplicand
            MOV     E,A
            MVI     D,00h
            IN      02h         ;Multiplier
            LXI     H,0000h     ;Result
            MVI     C,08h       ;Digit Counter
;=================================================
;                    Implementation
;-----------------------------------------------------------------
SH:         DAD     H           ;Double-Byte Shift
            RLC
            JNC     NC
            DAD     D           ;Double-Byte Addition
NC:         DCR     C
            JNZ     SH
;……..
            MOV     A,L
            OUT     03h
            MOV     A,H
            OUT     04h
;=================================================
            HLT
;-----------------------------------------------------------------
```

Program Clarification.

The proposed program for multiplying single-byte binary numbers is composed according to the matrix multiplication algorithm:

1) In view of the fact that the result of multiplying eight-digit numbers will be a 16-digit number, and in connection with the fact that partial additions are also performed on hexadecimal numbers, the register pair H, which is cleared at the beginning of the program execution, was adopted as a 16-bit accumulator.

2) The first number (multiplicand) is entered into the register E of the register pair D, and the register D is zeroed, as a result of which the multiplier becomes a 16-digit number.

3) To ensure the possibility of a shift, the second number (multiplier) is placed in register A (eight-bit accumulator).

4) To provide control over the number of shift operations, the number of factor multipliers (eight) is placed in register C.

5) Before each partial multiplication, the result accumulated before (in register pair H) is shifted to the left (by doubling).

6) The next value of the factor bit is determined by shifting the contents of register A (multiplier) to the left and, if it is equal to zero, partial multiplication is complete (that is, previous shift of the register pair H), and if it is equal to one, contents of the register pair D are added to the contents of the register pair H.

7) After each partial multiplication, contents of the shift counter are reduced by one and if it is equal to zero, multiplication program completes its work.

Input of single-byte data, over which it is necessary to perform the multiplication operation, and also the output of the multiplication result are carried out using a peripheral input / output device.

5.3 Working assignment

5.3.1 Prepare the simulator to work with the above program to implement the multiplication operation (Multiplication) and check the correctness of the stored instruction codes. Set the ports 01 and 02 to IN, and the rest to OUT.

5.3.2 Perform preliminary monitoring of the operation of individual program blocks (Rate / Normal):

1) At the invitation of the peripheral device, enter data values for multiplication.

2) After completion of the program, verify the validity of the results obtained (displayed in ports 03 and 04 of the peripheral device) by comparison with the results of the manual calculation.

5.3.3 Investigate the operation of the program and examine the actions of the instructions in individual function blocks of the program in a step-by-step mode (Rate / Step By Step). Write the specific observation results in the form of comments to the relevant instructions and draw conclusions about the operation of each of the program blocks.

5.3.4 Re-run the program and enter other data, check the correspondence of the results with the results of the corresponding manual calculation.

5.4 Quiz questions

1. Explain the multiplication algorithm adopted as the basis for its software implementation.

2. Explain the order of the steps in the preparatory phase of the multiplication program.

3. How is the determination of the next bit of the multiplier implemented in the program under investigation?

4. Suggest another way to determine the next bit of the multiplier.

5. Explain the actions of the DAD H and DAD D.

6. Which single-byte instructions replace the execution of the DAD H instruction?

7. Which single-byte instructions replace the execution of the DAD D instruction?

8. Explain the actions of the RLC and RAL instructions.

9. Explain the actions of the RRC and RAR instructions.

10. Explain the order of data I / O using the peripheral device.

11. How is the cycle of the actions carried out in the program under study?

12. Propose an algorithm for implementing the operation of dividing single-byte data.

6 Laboratory assignment №. 6. Working with subprograms

Objective: mastering the principles of organization of work with subprograms.

6.1 Workplace equipment

Computer, 8085 Simulator IDE.

6.2 The program for organization of work with subprograms

```
;*****************************************************
;                    Composite Program
;*****************************************************
;                     SUM=a1*a2+a3*a4+sum(ai,n,d)
;-----------------------------------------------------------------
        LXI     SP,00FFh
;=================================================
;                     Mul1=a1*a2
;-----------------------------------------------------------------
        IN      01h             ;Multiplicand
        MOV     E,A
        IN      02h             ;Multiplier
        CALL    MUL
        PUSH    H
        CALL    PER
;=================================================
;                     Mul2=a3*a4
;-----------------------------------------------------------------
        IN      01h             ;Multiplicand
        MOV     E,A
        IN      02h             ;Multiplier
        CALL    MUL
```

```
            PUSH    H
            CALL    PER
;=============================================
;                   Sum0(ai,n,d)
;---------------------------------------------------------------------
            IN      01h             ;Data Quantity - n
            MOV     C,A
            IN      02h             ;Step - d
            MOV     D,A
            IN      01h             ; Initial Operand – a0
            MOV     E,A
            CALL    SUM
            PUSH    H
            CALL    PER
;=============================================
;                   Sum=Mul1+Mul2+Sum0
;---------------------------------------------------------------------
            POP     H
            POP     D
            DAD     D
            POP     D
            DAD     D
            CALL    PER
;=============================================
            HLT
;*************************************************
MUL:                ;Multiplication (subprogram)
;---------------------------------------------------------------------
            MVI     D,00h
            LXI     H,0000h
            MVI     C,08h
ML:         DAD     H
            RLC
            JNC     NC
            DAD     D
NC:         DCR     C
            JNZ     ML
            RET
;=============================================
SUM:                ; Cumulative Summation (subprogram)
;---------------------------------------------------------------------
            LXI     H,0000h
SM:         MOV     A,E
            ADD     D
```

```
                MOV     E,A
                MOV     A,L
                ADD     E
                JNC     NCS
                INR     H
NCS:            MOV     L,A
                DCR     C
                JNZ     SM
                RET
;=============================================
PER:                    ; Displaying
;-------------------------------------------------------------------------
                MOV     A,L
                OUT     03H
                MOV     A,H
                OUT     04H
                RET
;-------------------------------------------------------------------------
```

The program clarification.

At the beginning of the main program, the memory area for the stack is allocated by assigning a specific value (address) to the SP stack pointer.

In each of the Mul1 and Mul2 blocks, data is entered for multiplication and the multiplication subprogram (MUL), followed by the display subprogram (PER), is called.

In Sum0 block, the data for summation is entered and the summation subprogram (SUM) is called, and then the display subprogram (PER) is called.

In the block Sum = ... of the main program, the final result is determined, i.e. the sum of the partial results obtained in the previous steps and stored in the stack part of the memory is calculated.

The program uses three subprograms:

1) Multiplication - multiplies two numbers entered before the call of the subprogram.

2) Cumulative Summation - determines the sum of the values of the members of the arithmetic progression, the parameters of which are set before calling this sub-program.

3) Displaying - displays the results received in the subprograms and stores them on the stack.

6.3 Working assignment

6.3.1 Prepare the simulator for working with the aforementioned program for organizing the work with subprograms (Composite Program) and verify the correctness of the instruction codes stored in memory. Set the ports 01 and 02 to IN, and the rest to OUT.

6.3.2 Perform preliminary monitoring of the operation of individual blocks and program subprograms (Rate / Normal) by entering the values of the corresponding data at the invitation of the peripheral device. After the program completes, check the reliability of the results obtained in the subprograms (displayed in memory and on ports 03 and 04 of the peripheral device) by comparing them with the results of the corresponding manual calculation.

6.3.3 Investigate the operation of the program and examine the actions of the instructions in individual function blocks and subprograms of the program in a step-by-step mode (Rate / Step By Step). Write the specific observation results in the form of comments to the appropriate instruction and draw conclusions about the work of each of the blocks and subprograms of the program.

6.3.4 Re-run the program and enter other data, check the correspondence of the results with the results of the corresponding manual calculation.

6.4 Quiz questions

1. Explain the principles of organizing the work with subprograms.
2. Explain the actions that are performed while the subprogram is called.
3. How do you allocate a specific memory area for the stack?
4. What function does the stack perform when working with subprograms?
5. How do the contents of the SP register change when a subprogram is called?
6. Explain the actions performed when returning from the subprogram.
7. How do the contents of the SP register change when returns from a sub-program?
8. What instructions could replace the subprogram call?
9. What instructions could replace the return from the sub-program
10. Explain the actions of the PUSH H and PUSH D instructions.
11. Explain the actions of the POP H and POP D instructions.
12. How is the work with tabular data organized using the stack?

7 Laboratory assignment №7. Binary-to-decimal conversion

Objective: mastering the principles of implementing the binary-to-decimal conversion.

7.1 Workplace equipment

Computer, 8085 Simulator IDE.

7.2 The program for implementing the binary-to-decimal conversion

```
;**************************************************
```

```
;                       BinDec
;***************************************************
;                       Setup Cleaning
;---------------------------------------------------------------------
            LXI     H,00D0h
            MVI     C,2Fh
            MVI     A,00h
CLR         MOV     M,A
            INX     H
            DCR     C
            JNZ     CLR
;=====================================================
;                       Multiplication
;---------------------------------------------------------------------
            IN      01h         ;Multiplicand
;.....................................................................................
            LXI     H,00D0h
            MOV     M,A
            INX     H
            MVI     M,00h
;.....................................................................................
            MOV     E,A
            MVI     D,00h
            IN      02h         ;Multiplier
;.....................................................................................
            LXI     H,00E0h
            MOV     M,A
            INX     H
            MVI     M,00h
;.....................................................................................
            LXI     H,0000h
            MVI     C,08h
;=============================================
SH:         DAD     H
            RLC
            JNC     NC
            DAD     D
NC:         DCR     C
            JNZ     SH
;-------------------------------------------------------------------------
            MOV     A,L
            OUT     03h
            MOV     A,H
            OUT     04h
```

```
;.....................................................................................
            XCHG
            LXI     H,00F0h
            MOV     M,E
            INX     H
            MOV     M,D
;=============================================
;                   Preparation to BinDec
;=============================================
            LXI     H,00D0h
            MOV     E,M
            INX     H
            MOV     D,M
            LXI     H,00D8h
            CALL    BD
;-------------------------------------------------------------------------------
            LXI     H,00E0h
            MOV     E,M
            INX     H
            MOV     D,M
            LXI     H,00E8h
            CALL    BD
;-------------------------------------------------------------------------------
            LXI     H,00F0h
            MOV     E,M
            INX     H
            MOV     D,M
            LXI     H,00F8h
            CALL    BD
;-------------------------------------------------------------------------------
            HLT
;*****************************************************
BD:                 ;BinDec
;-------------------------------------------------------------------------------
            MVI     M,0FFh
;-----------------------------------------
Dig4:       DCX     H
D4:         MOV     A,E
            SUI     10h         ;Dec(10000)=Hex(2710)
            MOV     C,A
            MOV     A,D
            SBI     27h
            MOV     B,A
            JC      Dig3
```

```
;-------------------------------------------------------------------------------
                INR     M
                MOV     E,C
                MOV     D,B
                JMP     D4
;-------------------------------------------------------------------------------
Dig3:           DCX     H
D3:             MOV     A,E
                SUI     0E8h        ;Dec(1000)=Hex(03E8)
                MOV     C,A
                MOV     A,D
                SBI     03h
                MOV     B,A
                JC      Dig2
;-------------------------------------------------------------------------------
                INR     M
                MOV     E,C
                MOV     D,B
                JMP     D3
;-------------------------------------------------------------------------------
Dig2:           DCX     H
D2:             MOV     A,E
                SUI     64h         ;Dec(100)=Hex(0064)
                MOV     C,A
                MOV     A,D
                SBI     00h
                MOV     B,A
                JC      Dig1
;-------------------------------------------------------------------------------
                INR     M
                MOV     E,C
                MOV     D,B
                JMP     D2
;-------------------------------------------------------------------------------
Dig1 :          DCX     H
D1:             MOV     A,E
                SUI     0Ah         ;Dec(10)=Hex(0A)
```

Program Clarification.

In the proposed program, a binary-to-decimal subroutine has been added to the previously studied multiplication program, with the help of which a corresponding conversion of both the entered single-byte multipliers and a two-byte result is performed.

The values of the corresponding digits in the decimal representation of the number are determined by dividing the converted number by the weight values of the corresponding digits of the decimal number system. In view of the fact that there is no division instruction in the instruction set of the Intel 8085 microprocessor, the division operation is performed programmatically using the subtraction instructions.

The hexadecimal values of the converted numbers and the results of the corresponding conversion are output to the allocated memory locations. In order to provide visibility of the observation of the conversion results, the results are displayed in the vicinity of the cells with the FFh content.

7.3 Working Assignment

7.3.1 Prepare the simulator to work with the above program for implementing the binary-to-decimal conversion (BinDec) and verify the correctness of the stored instruction codes. Set the ports 01 and 02 to IN, and the rest to OUT.

7.3.2 Perform preliminary monitoring of the operation of individual blocks of the program (simulation speed - Rate / Normal):

– at the invitation of the peripheral device, enter data values for multiplication;

– after completion of the program, verify the validity of the obtained multiplication results and the binary-to-decimal conversion of the corresponding data.

7.3.3 Investigate the operation of the program and examine the actions of the instructions in individual function blocks of the program in a step-by-step mode (Rate / Step By Step). Specific observation results should be written in the form of comments to the appropriate teams and draw appropriate conclusions about the work of each of the program blocks.

7.3.4 Re-run the program and enter other data, verify the validity of the obtained multiplication results and the corresponding binary-decimal conversion.

7.4 Quiz questions

1. Explain the binary-to-decimal conversion algorithm adopted in the program under investigation.
2. Explain the principles of implementing the binary-to-decimal conversion algorithm in the program.
3. How is the division operation practically carried out in the program under study?
4. Suggest the ways to reduce the size of the program.
5. Suggest the other algorithms for the binary-decimal conversion.
6. Explain the actions of SUI instructions 10h and SBI 27h.
7. Explain the actions of the SUI 0E8h and SBI 03h instructions.

8. Explain the actions of SUI 64h and SBI 00h.

9. How does the program determine the value of the fifth digit of the decimal representation of a given number?

10. How does the program determine the value of the fourth digit of the decimal representation of a given number?

11. How does the program determine the value of the third digit of the decimal representation of a given number?

12. How does the program determine the values of the second and first digits of the decimal representation of a given number?

8 Laboratory assignment №8. Working with interrupts

Objective: mastering the principles of organization of work with interrupts.

8.1 Workplace equipment

Computer, 8085 Simulator IDE.

8.2 The program for managing interrupts

```
;*****************************************************
;                       Interrupts
;*****************************************************
            JMP         0050h       ;Jump to Main Routine
;=============================================
            .ORG        002Ch       ;Interrupt Routine 5.5
;---------------------------------------------------------------------
            JMP         Rst55
;=============================================
            .ORG        003Ch       ;Interrupt Routine 7.5
;---------------------------------------------------------------------
            JMP         Rst75
;*****************************************************
            .ORG        0050H       ;Main Rountine
;---------------------------------------------------------------------
            LXI         SP,00FFh
            MVI         A,00h
            MOV         B,A
            OUT         01h
            OUT         02h
;---------------------------------------------------------------------
MR0:        MVI         A,00h
MR:         INR         A
            OUT         01h
```

```
                JNZ         MR
                INR         B
                MOV         A,B
                OUT         02h
;.....................................................................................
                RIM                     ;Read Interrupt Mask
;.....................................................................................
                MVI         A,0BH       ;Interrupt Mask
                SIM                     ;Set Interrupt Mask
                EI                      ;Enable Interrupts
;.....................................................................................
                RIM
;.....................................................................................
                JNZ         MR0
                DI                      ;Disable Interrupts
;.....................................................................................
                RIM
;.....................................................................................
                JMP         MR0
                HLT
;*************************************************
;=================================================
Rst55:                      ;Interrupt Routine 5.5
;-------------------------------------------------------------------------
                PUSH        PSW
                PUSH        B
;-------------------------------------------------------------------------
                MVI         E,00h
                MVI         B,08h
                MVI         A,00h
                OUT         04h
Int5_0:         MOV         C,B
                MVI         A,80h
Con5_0:         RLC
                MOV         H,A
                ORA         E
                MOV         L,A
                OUT         04h
                MOV         A,H
                DCR         C
                JNZ         Con5_0
                DCR         B
                MOV         E,L
                JNZ         Int5_0
```

```
;-------------------------------------------------------------------------
            MVI     B,08h
Int5_1:     MOV     C,B
            MVI     A,07Fh
Con5_1:     RLC
            MOV     H,A
            ANA     E
            MOV     L,A
            OUT     04h
            MOV     A,H
            DCR     C
            JNZ     Con5_1
            DCR     B
            MOV     E,L
            JNZ     Int5_1
;……...
            POP     B
            POP     PSW
;……...
            RET                 ;Return from Interrupt
;=========================================
Rst75:              ;Interrupt Routine 7.5
;-------------------------------------------------------------------------
            PUSH    PSW
            PUSH    B
;……...
            MVI     E,00h
            MVI     B,08h
            MVI     A,00h
            OUT     03h
Int7_0:     MOV     C,B
            MVI     A,01h
Con7_0:     RRC
            MOV     H,A
            ORA     E
            MOV     L,A
            OUT     03h
            MOV     A,H
            DCR     C
            JNZ     Con7_0
            DCR     B
            MOV     E,L
            JNZ     Int7_0
;-------------------------------------------------------------------------
```

```
            MVI     B,08h
Int7_1:     MOV     C,B
            MVI     A,0FEh
Con7_1:     RRC
            MOV     H,A
            ANA     E
            MOV     L,A
            OUT     03h
            MOV     A,H
            DCR     C
            JNZ     Con7_1
            DCR     B
            MOV     E,L
            JNZ     Int7_1
;……...
            POP     B
            POP     PSW
;……...
            RET                 ;Return from Interrupt
;-------------------------------------------------------------------------
```

Program Clarification.

In the main program (location starting with cell 0050h of memory), two-byte numbers are sequentially formed and displayed in ports 1 and 2 of the peripheral device.

In the interrupt routines, dynamic images (displayed in ports 3 and 4 of the peripheral device) are created using logical instructions.

8.3 Working assignment

8.3.1 Prepare the simulator to work with the above program for managing interrupts (Interrupts) and verify the correctness of the stored instruction codes. Set all ports of the peripheral device to OUT.

8.3.2 Perform preliminary monitoring of the program (simulation speed - Rate / Normal):

1) Observing the display of sequentially generated numbers and the state of the RST 7.5 button, note that the interrupt permission is given (by activating the RST button 7.5) after the first transfer to the second byte.

2) After generating the interrupt signal (by pressing the button RST 7.5), observe the operation of the corresponding interrupt program.

3) Note that after returning to the main program, the sequential generation of numbers continues from the value that was generated at the time of the interrupt.

8.3.3 Investigate the operation of the program and examine the actions of the instructions in individual function blocks of the program in a step-by-step mode

(Rate / Step By Step). Specific observation results should be written in the form of comments to the appropriate teams and draw appropriate conclusions about the work of each program blocks.

8.3.4 Modify the program to work with the interrupt RST 5.5 and test its operation.

8.3.5 Make changes to the program to work with both the RST 5.5 interrupt and the RST 7.5 interrupt and test its operation.

8.4 Control questions

1. List the output pins of the Intel 8085 microprocessor related to interrupts and characterize the corresponding interrupt signals.
2. Explain how to work with interrupt routines.
3. Where is the TRAP interrupt service program located?
4. Where is the RST 7.5 interrupt service program located?
5. Where is the RST 6.5 interrupt service program located?
6. Where is the RST 5.5 interrupt service program located?
7. How is the RST 7.5 request masked?
8. How is the RST 6.5 request masked?
9. How is the RST 5.5 request masked?
10. What hardware actions are performed when an interrupt request signal arrives?
11. How is it possible to save and restore the contents of the accumulator and the microprocessor's flags register when switching to and returning from the interrupt routine?
12. How is it possible to store and restore the contents of the microprocessor's general-purpose registers when the interrupt subroutine is called and returned from it?

List of references

1 Shanaev O.T. Digital Systems. – Almaty: AUPET, 2013.

2 Ugryumov E. P. Digital Circuit Design. – SPb.: BHV- Petersburg, 2010.

3 Novikov Y.V., Skorobogatov P.K. Microprocessor technology fundamentals. – M.: BINOM, 2009.

4 Volovich G.I. Analog and analog-digital circuit design. – M.: Dodeka, 2005.

Contents