**Laboratory No 2**

## The instruction set and programming of the microprocessor Intel® 8080 (КР580ИК80А)

### 1. The aim of the work

To get acquainted with the instruction set of the microprocessor Intel® 8080 (КР580ИК80А); to acquire skills in programming by an assembler.

### 2. General knowledge

### 2.1. Characteristics of the microprocessor

The microprocessor Intel® 8080 appeared on the market in 1974. It was the third microprocessor following Intel® 4004 (1971) and Intel® 8008 (1972).

The main parameters of the microprocessor Intel® 8080:

- clock frequency – 2 MHz;
- speed – 0.64 million operations per second;
- the number of transistors – 6000;
- resolution power of production technology – 6 μm;
- data bus width – 8 bits;
- address bus width – 16 bits;
- addressable memory – 64 KB.

The microprocessor efficiency compared to Intel® 8008 increased ten times, and the number of additional microcircuits maintaining the microprocessor work decreased from 20 to 6. The microprocessor was used in the street lightning control devices and the first personal computers *Altair*. Now due to its rather simple architecture and the instruction set the microprocessor is used for training purposes.

## 2.2. Unit of registers

The microprocessor has seven 8-bit registers for data storage. Register **A** called an accumulator is intended for information exchange with peripheral devices. When performing arithmetical, logical and displacement operations, one of operands is stored in a accumulator. The operation result is stored in it. Other six registers – **B, C, D, E, H** and **L** make up the so called general purpose register unit (GPR) and they can be used for storage of data and addresses. If 16-bit binary numbers have to be stored, these single 8-bit registers are joined into pairs **BC, DE, HL**. In the assembler language these pairs are identified as **B, D** and **H**.

Stack pointer **SP** (16-bit register) is intended for addressing the stack memory cells. A lower part **SL** and a higher part **SH** of the 8-bit register are accessible to the programmer separately.

The program counter **PC** (16-bit register) is intended for storing the instruction address. By selecting a routine instruction from the memory, the content of the counter is increased by a unity, i.e. another byte of the instruction is addressed (if there are no conditional or unconditional transitions in the program).

The flag register **F** (8 bits) is intended for the fixation of certain flags of the operation result. The flag is fixed in a corresponding bit writing 1 or 0:

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| **S** | **Z** | **0** | **AC** | **0** | **P** | **1** | **C** |

- bit **S** – sign flag: 1 – negative result, 0 – positive result;
- bit **Z** – zero flag: 1 – result equal to zero, 0 – result is not equal to zero;
- bit **AC** – auxiliary carry flag: 1 – when carry occurs from the third bit of binary number, 0 – there is no carry;
- bit **P** – parity flag: 1 – if there is even number of units in the result binary code, 0 – if there is odd number of units in the result binary code;

- bit **C** – carry flag: 1 – if the operation result does not fit in 8 bits (carry occurs from the highest bit or it was borrowed by performing the operation of subtraction).

These values of bits are incorporated in 16 and 32-bit flag registers of later generation Intel® architecture microprocessors as a lower byte of these registers.

### 2.3. Arithmetic logic unit

The 8-bit arithmetic logic unit (ALU) can perform arithmetical operations (addition and subtraction, with carry and without it), four logical and four displacement operations. By performing arithmetical and logical operations, one of operands is placed in the accumulator. The operation result remains in the accumulator. Only the cyclic displacement of the accumulator is performed.

The possibility of performing arithmetical operations with decimal numbers is also provided. For the decimal number storage the register bits are divided into two groups in fours and in each group one decimal number coded by 8421 code (BCD – Binary Coded Decimal) is stored.

ALU is directly related to the control unit consisting of the instruction register, where the first instruction byte is sent, and of the control signal formation device. In the latter there is the control memory where micro programs of various operations are stored. The control memory, thus the operation micro programs as well, are inaccessible to the user.

### 2.4. Data and instruction formats

Data are the processed binary information and results of its processing. Data are most frequently stored in the random access memory and in the process registers. The binary 8-bit number is called a byte. Addresses are of 16 bits, thus, like 16-bit data, they are stored in parts: a lower byte – **Data L** or **Adr L** and a higher byte – **Data H** or **Adr H**.

The instruction formats can be of one, two and three bytes. The values of bytes are presented in Table 1. In one byte instructions the operation code byte (OCB) is the only byte.

Table 1. Formats of instructions

| Format | 1 byte | 2 byte | 3 byte |
|---|---|---|---|
| Of one byte | OCB | – | – |
| Of two bytes | OCB | Data or port number | – |
| Of three bytes | OCB | Data L or Adr L | Data H or Adr H |

### 2.5. Operand addressing modes

To perform operation in the instruction, alongside with the operation type operands should also be indicated. The applied modes of the operand addressing will be discussed.

***Register addressing.*** Operands during this addressing are in general purpose registers, the addresses of which are indicated in the only OCB. As there are few registers, for their addressing three binary bits are sufficient, as shown in Table 2.

Table 2. Addresses of registers

| Register | Address | Register | Address |
|---|---|---|---|
| **B** | 000 | **H** | 100 |
| **C** | 001 | **L** | 101 |
| **D** | 010 | **M** | 110 |
| **E** | 011 | **A** | 111 |

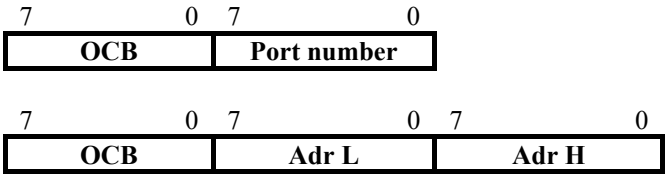
where **M** is the memory cell, the address of which is stored in the register pair **HL**.

For example, the format of instruction **MOV Reg1, Reg2** is the following:

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **0** | **1** | **Reg1** | | | **Reg2** | | |

By entering specific register addresses instead of **Reg1** and **Reg2**, we obtain a binary code of the instruction.

***Direct addressing.*** Instructions directly addressing the operands can be of two or three bytes. Their formats are:




where **Adr L** and **Adr H** are lower and higher bytes of the address.

***Register indirect addressing.*** This mode of addressing differs from the register mode in which not the operand itself is stored in registers, but the memory address of this operand. Since the address is of 16 bytes, a pair of registers is needed for its storage. For example, the instruction **MOV D, M** means that the data from the memory cell, which address is indicated by the register pair **HL** (**M** in this case means a register pair **HL**).will be sent to register **D**.

***Immediate addressing.*** These are data of one or two bytes connected to OCB. The instruction format is of two or three bytes and is similar to the format of directly addressable instructions. The difference is that in the case of the immediate addressing after OCB not data bytes, but the address bytes follow (Table 1).

## 3. Task

1. To switch on a computer and get acquainted with the software intended for this work.
2. To switch on the training microprocessor kit (TMK) and prepare for work.
3. To program and execute 3 short programs with freely selected instructions for each mode of the operand addressing. In executing instructions use a directive:

**“СТ” <Program initial address> ”␣” <Interrupt address> “ВП”**.

***Note:*** *The interrupt address is taken larger by unity than the program address of the last byte.*

To write the experimental results in Table 3.

Table 3. Techniques of the operand addressing

| Address | Instruction code | | Initial data | Results | Notes |
|---|---|---|---|---|---|
| | Mnemonic | Hexadecimal | | | |
| | | | | | |

4. To select instructions of one, two and three bytes and execute them in a step mode (by computer cycles). Write the results in Table 4.

Table 4. Results of instruction execution in computer cycles

| Step number | Address (Hex) | Data (Hex) | State (Bin) | Comment |
|---|---|---|---|---|
| Instruction mnemonic code | | | | |
| | | | | |

***Note:*** *Computer cycles only of selected instructions should be written in the table according to the principle: one instruction – one cycle of the instruction selection.*

5. To create and execute the program which would sum up two operands from different memory cells and would write the result into the third memory cell. For the operand addressing to select different modes of operand addressing. When writing the program and its execution results refer to Table 3.

6. To create and execute the program which would compare two operands, would subtract a smaller operand from the larger one and to write difference in the stack at the selected address. Then writing the program and its execution results refer to Table 3.

## 4. Contents of the report

1. The aim of the work.

2. Examples of executing instructions with different operand addressing modes (Table 3).

3. Examples of executing step instructions (Table 4).

4. Programs of tasks 5 and 6, and results of their execution (Table 3).

5. Conclusions.

## 5. Test questions

1. What are the main characteristics of the microprocessor Intel® 8080?

2. What are general purpose registers of the microprocessor Intel® 8080?

3. What are bit values of the microprocessor Intel® 8080 flag register?

4. What is the assignment of the program counter?

5. What is the stack and what is its assignment?

6. What operations can the arithmetic logic unit of the microprocessor Intel® 8080 perform?

7. Enumerate and explain the instruction formats of the microprocessor Intel® 8080.

8. What operand addressing is called register addressing?

9. What operand addressing is called register indirect addressing?

10. What operand addressing is called direct addressing?

11. What operand addressing is called immediate addressing?

## References

1. Uffenbeck J. Microcomputers and Microprocessors: The 8080, 8085, and Z-80 Programming, Interfacing and Troubleshooting. USA, New Jersey: Pearson Prentice Hall; 3rd Edition. June 18, 1999. 729 p.

2. Coffron J. Microprocessor Programming, Troubleshooting and Interfacing: The Z80, 8080 and 8085. USA, New Jersey: Pearson Prentice Hall; 1st Edition. October 28, 1997. 528 p.

3. Gražulevičius A. Mikroprocesoriai. Laboratorinių darbų užduotys ir metodikos nurodymai. Vilnius: Technika, 2000. 60 p.