**Ministry of Transport and Communications of Ukraine**

**State Administration of Communications of Ukraine**

**Odessa National Academy of Telecommunications after A.S. Popov**

Department of Computer Science and Microprocessors

**Computer Science and Microprocessors**

**Module №1**

**Units of computer facilities and microprocessor systems**

**for students**

# ARCHITECTURE AND SOFTWARE MODELS OF INTEL MICROPROCESSORS

**training area: telecommunications**

**for specialties: 7. 092402, 7. 092401, 7. 092404, 7. 092407**

Odessa 2009

УДК План НМВ 2009р.

**Writer:** Tkachenko A.A.

The basic features of architecture and software models of INTEL microprocessors from I8086 to I80486 are gave. The methodical guide contains 20 variants of home tasks to laboratory work.

IT IS APPROVED
by methodical council
of academy
Minutes № 6
from 10.02.2009

IT IS APPROVED
on department meeting
of Computer Science
and Microprocessors and
recommended for
printing
Minutes № 3
from 10.10.2008

## I. Foreword

Discipline general characteristic (quantity of credits ECTS - 6; modules - 4; substantial modules - 14; total hours - 216; including: lectures - 68 h.; laboratory works - 32 h.; practical trainings - 0 h.; independent work - 58 h.; individual work - 58 h.; semester 2.3, 2.4, 3.1, 3.2, a control kind: the complex task, the course work, the test.

## II. The purpose of discipline training

The purpose and subject matter of problems: knowledge formation concerning computer aids construction principles and microprocessor systems, creation and software debugging of them, ability to analyse, working out and operate such systems in telecommunications.

## III. The discipline content

Electronic computers nodes of: digital automatic machines, both their analysis and synthesis; memory devices, their classification and the organization; microprocessors (MP), construction principles and microprocessors and electronic computers functioning, universal microprocessors architecture, the organisation of memory and ways of addressing of operands in microprocessors.

Microprocessor systems (MPS): construction principles, ways of the organisation of data exchange in MPS, address space and its distribution in MPS, typical user MPS interface hardware and program means, the interruptions organisation in MPS; controllers in telecommunications, microcontrollers of conducting firms, control means and switching construction in systems of telecommunications at hardware and program levels; digital signals processors in telecommunications, conducting firms digital signals processors, telecommunications systems signals transformation modules construction on hardware and program levels; MPS productivity increase , multiprocessing system.

The MPS software: programming of MT INTEL firm, the raised word length MT programming of conducting firms; microcontrollers and processors digital signals programming.

**The module 1**: Units of computer facilities and microprocessor systems

Entrance requirements to module studying (knowledge and abilities from disciplines which provide studying of the given module).

| № | Content of knowledge | Code number |
|---|---|---|
| 1 | Number representations | KN.1 |
| 2 | Circuitry bases | KN.2 |
| | Content of abilities | |
| 1 | Designing of communication networks | AB.1 |
| 2 | Communication networks tumimg | AB.2 |

Structure of the test module 1

| The substantial module | Lecture (hours) | Study | | Self-instruction | Individual work |
|---|---|---|---|---|---|
| | | practical | laboratory | | |
| The module 1: The nodes of computer facilities and microprocessor systems (2 credits; 52 h.) | | | | | |
| 1. Computer aids | 4 | | 2 | 2 | 2 |
| 2. Microprocessors | 6 | | 2 | 4 | 4 |
| 3. Memory subsystems | 4 | | 2 | 4 | 4 |
| 4. Interfaces | 2 | | 2 | 4 | 4 |
| 1 module in total, h. | 16 | – | 8 | 14 | 14 |

Content of substantial modules (lecture hours - 16):

1.1 Computer aids (4 h.)
The content: Computer and microprocessor systems.; Data manipulation in computer systems.

1.2 Microprocessors (6 h.)
The content: Digital automaton. Digital automata synthesis.; Typical computer systems devices.; Microprocessors architecture. Software models of the 16- and 32-bit Intel microprocessors.

1.3 Memory subsystems (4 h.)
The content: Memory construction principles with the set organization.; Address space and its distribution in MPS. Memory segmentation. Operand addressing modes for the Intel microprocessors.

1.4 Interfaces (2 h.)
The content: Principles of the computer construction and its functioning. Interfaces.

Laboratory studies' themes of the module 1

| № | NUMBER, THE NAME OF LABORATORY WORKS | Hours |
|---|---|---|
| 1 | Arithmetic logic unit | 2 |
| 2 | Storage device | 2 |
| 3 | Architecture and software models of Intel microprocessors | 2 |
| 4 | Memory segmentation | 2 |
| | In total | 8 |

Initial knowledge and abilities from the module 1

| № | Content of knowledge | The code number |
|---|---|---|
| 1 | To know the construction principles of arithmetic logic and memory devices | KN.1 |
| 2 | To know architecture and operand addressing modes for the Intel microprocessors. | KN.2 |
| | Content of abilities | |
| 1 | To submit and treat entrance and initial numerical data for the further digital processing. To correlate logic changes and functions to digital signals which realize them. | AB.1 |
| 2 | To put and solve the problems connected with the analysis, working out and operation of microprocessor systems of different function, creation and to software debugging to them. | AB.2 |

**The module 2**: Programming of Intel microprocessors.

Entrance requirements to module studying (knowledge and abilities from disciplines which provide studying of the given module).

| № | Content of knowledge | The code number |
|---|---|---|
| 1 | Number representations | KN.1 |
| 2 | Circuitry bases | KN.2 |
| 3 | The general principles of programming | KN.3 |
| | Content of abilities | |
| 1 | Designing of communication networks | AB.1 |
| 2 | Communication networks tumimg | AB.2 |

Structure of the test module 2

| The substantial module | Lecture (hours) | Employment | | Self-instruction | Individual work |
|---|---|---|---|---|---|
| | | practical | laboratory | | |
| The module 2: Intel microprocessors programming (1 credit; 56 ч.) | | | | | |
| 1. Programming language Assembler-86 | 2 | | 2 | 5 | 5 |
| 2. The linear program organizations using Assembler-86 language | 2 | | 2 | 5 | 5 |
| 3. The branched and cyclic programs organization using Assembler-86 language | 14 | | 4 | 5 | 5 |
| 1 module in total, h. | 18 | – | 8 | 15 | 15 |

Content of substantial modules (lecture hours - 18):

2.1 Programming language Assembler-86 (2 h.)
The content: Low-level programming languages. Assembly programming language. Instruction and data formats. Operand addressing modes. Move instructions.

2.2 The linear program organisation using Assembly language (2 h.)
The content: Data conversion instruction in Assembly language. The linear programs.

2.3 The branched and cyclic programs organisation using Assembly language
The content: Conditional and unconditional jump instructions in Assembly language. The branched programs.; The organisation of cyclic programs.; Data exchange modes in MPS. The user interface software in typical MPS.; Organization of interrupts in MPS. Types of interrupts.; Productivity of microprocessors and estimation of it. Architecture of modern microprocessors.; Intel microprocessors using in telecommunication. Software support of telecommunications facilities nodes.

Laboratory studies' themes of the module 2

| № | NUMBER, THE NAME OF LABORATORY WORKS | Hours |
|---|---|---|
| 1 | The linear programs | 2 |
| 2 | The branched programs | 2 |
| 3 | The cyclic programs | 2 |
| 4 | Serial port RS-232-C | 2 |
| | In total | 8 |

Initial knowledge and abilities from the module 2

| № | Content of knowledge | The code number |
|---|---|---|
| 1 | To know the operand addressing modes, move instructions, conditional and unconditional jump instructions, organisation interrupts in MPS . | KN.1 |
| 2 | To know the software support of telecommunications facilities nodes. | KN.2 |
| | Content of abilities | |
| 1 | To put and solve the problems connected with the analysis, working out and operation of different functional MPS, creation and software debugging of them. | AB.1 |
| 2 | To analyze and develop separate telecommunications systems nodes which use computer means and microprocessors. To use typical digital blocks, nodes and elements for digital devices realization. | AB.2 |
| 3 | To put and solve the problems connected with a computer means choice, microprocessors behind their technical, operational and economic characteristics for systems of telecommunications. | AB.3 |

**The module 3**: Microprocessor systems (MPS) on universal microprocessors and its programming

Entrance requirements to module studying (knowledge and ability from disciplines which provide studying of the given module).

| № | Content of knowledge | The code number |
|---|---|---|
| 1 | The general MPS architecture | KN.1 |
| 2 | Microprocessors programming bases | KN.2 |
| | Content of abilities | |
| 1 | Communication networks designing | AB.1 |
| 2 | Communication networks tumimg | AB.2 |

Structure of the test module 3

| The substantial module | Lecture (hours) | Employment | | Self-instruction | Individual work |
|---|---|---|---|---|---|
| | | practical | laboratory | | |
| The module 3: Microprocessor systems (MPS) on universal microprocessors and its programming (2 credits; 52 ч.) | | | | | |
| 1. x-bit Motorola microprocessors | 4 | | 2 | 4 | 4 |
| 2. MPS construction on 32-bit Motorola MP | 6 | | 2 | 5 | 5 |
| 3. MPS software creation on 32-bit Motorola MP | 6 | | 4 | 5 | 5 |
| 1 module in total, h. | 16 | – | 8 | 14 | 14 |

Content of substantial modules (lecture hours - 16):

3.1 x-bit Motorola microprocessors (4 h.)
The content: Motorola MP MC68XXX. Software models of MP MC68000 and 68020.; Memory organisation and operand addressing modes in MP MC68XXX .

3.2 MPS construction on 32-bit Motorola MP (6 h.)
The content: Principles of MPS construction on MP MC68XXX.; Distribution of address space in MPS on MP MC68XXX. The organisation of a memory subsystem.; The organisation of a peripheral subsystem.

3.3 Creation of the software for MPS on 32-bit Motorola MP (6 ч.)
The content: Instruction set of MP MC68000. Examples of programs with different operand addressing modes in instructions.; Control transfer instruction in MP MC68XXX.; Construction of programs with structure "branching" and "cycle" in MP MC68XXX.

Laboratory studies' themes of the module 3

| № | NUMBER, THE NAME OF LABORATORY WORKS | Hours |
|---|---|---|
| 1 | Monitor instructions studying of Motorola MC 68xxx | 2 |
| 2 | Structure and operand addressing modes of typical instructions in Motorola MP 68xxx | 2 |
| 3 | Instruction set of Motorola MP 68xxx | 2 |
| 4 | Programming of Motorola MP 68xxx | 2 |
| | In total | 8 |

Initial knowledge and abilities from the module 3

| № | Content of knowledge | The code number |
|---|---|---|
| 1 | To know the structure and addressing modes in typical commands of Motorola MP 68xxx. | KN.1 |
| 2 | MPS construction principles. Memory subsystem organisation. Peripheral subsystem organisation. | KN.2 |
| | Content of abilities | |
| 1 | To put and solve the problems connected with the analysis, working out and microprocessor systems operation of different function, creation of the software debugging them. | AB.1 |
| 2 | To analyze and develop separate telecommunications systems nodes which use computer aids, microprocessors and microcontrollers. To use typical digital blocks, nodes and elements for digital devices realization. | AB.2 |

**The module 4**: Microprocessor systems on microcontrollers and DSP and its programming.

Entrance requirements to module studying (knowledge and ability from disciplines which provide studying of the given module).

| № | Content of knowledge | The code number |
|---|---|---|
| 1 | General MPS architecture | KN.1 |
| 2 | Microprocessors programming bases | KN.2 |
| | Content of abilities | |
| 1 | Communication networks designing | AB.1 |
| 2 | Communication networks tumimg | AB.2 |

Structure of the test module 4

| The substantial module | Lecture (hours) | Employment | | Self-instruction | Individual work |
|---|---|---|---|---|---|
| | | practical | laboratory | | |
| The module 4: Microprocessor systems on microcontrollers and DSP and its programming (1 credit; 56 h.) | | | | | |
| 1. Motorola microcontrollers | 4 | | 2 | 2 | 2 |
| 2. MPS construction on Motorola MC | 4 | | 2 | 3 | 3 |
| 3. Software creation for MPS on Motorola MC | 6 | | 2 | 5 | 5 |
| 4. Digital signals processors | 4 | | 2 | 5 | 5 |
| 1 module in total, h. | 18 | – | 8 | 15 | 15 |

Content of substantial modules (lecture hours - 16):

4.1 Motorola microcontrollers (4 h.)
The content: Motorola Microcontrollers (MC): HC05, HC08, HC11. Structure; MC intrinsics.

4.2 MPS construction on Motorola MC (4 h.)
The content: Adjustment of the MC intrinsics.; Typical MC programming examples.

4.3 Software creation for MPS on Motorola MC (6 h.)
The content: Motorola RISC-processors; General principles of the digital signals processing.; Construction principles of telecommunications nodes at program level.

4.4 Digital signals processors (4 h.)
The content: Architecture and construction principles of digital processors (DSP), its features and field of use.; Microprocessor systems in terms of the DSP, universal MP and MC.

Laboratory studies' themes of the module 4

| № | NUMBER, THE NAME OF LABORATORY WORKS | Hours |
|---|---|---|
| 1 | The organisation branched programs in Motorola MP 68xxx using Assembly language | 2 |
| 2 | The organisation cyclic programs in Motorola MP 68xxx using Assembly language | 2 |
| 3 | Instruction set of Motorola 68HC05microcontroller | 2 |
| 4 | Motorola 68HC05 microcontroller processor module and technological features structure studying. | 2 |
| | In total | 8 |

Initial knowledge and abilities from the module 4

| № | Content of knowledge | The code number |
|---|---|---|
| 1 | To know Instruction set of the Motorola 68HC05 microcontroller, processor module structure and technological features of the microcontroller. | KN.1 |
| | Content of abilities | |
| 1 | To put and solve the problems connected with a computer aids choice, microprocessors and microcontrollers behind its technical, operational and economic characteristics for telecommunications systems. | AB.1 |
| 2 | To create and adjust the software for digital signals processing devices in telecommunications systems using languages of specific microprocessors and microcontrollers. | AB.2 |
| 3 | To submit and treat entrance and initial numerical data for the further digital processing. To correlate logic changes and functions to digital signals which realize its. | AB.3 |

## 1. Study objective

The work purpose is acquaintance with architecture and software models of INTEL microprocessors: I8086, I8088, I80186, I80286, I80386, I80486, and also studying of receptions of loading software accessible registers of these microprocessors.

## 2. Key points

### 2.1. General characteristic of INTEL microprocessors

Development of INTEL microprocessors was passing by compatibility retention upwards at object and software levels. It means, that the software of older models supports with new models for which programs with using of its specificity are written.

The internal architecture of INTEL microprocessors since model I8086 has generally identical units:

1 ALU – the arithmetic logic unit which serves for performing arithmetic, logic and bit-shifting operand operations and indicates results flags.

2 General-purpose registers (GPR) or data registers which serve for data or subproduct holding:
- EAX – the register-accumulator can be used for arithmetic, logical, shift, rotate, or other similar operations.
- EBX – the base register. It is commonly used to hold indirect addresses. Can also be used in computation;
- EDX – the data register has two special purposes: it holds the overflow from certain arithmetic operations, and it holds I/O addresses when accessing data on the 80x86 I/O bus;
- ECX – the count register. It is often used to count off the number of iterations in a loop. Can also be used in computation;

3 Index registers and pointers:
- ESI – source index register is used for indexed, based indexed and register indirect addressing, as well as a source data address in string manipulation instructions;
- EDI – destination index register is used for indexed, based indexed and register indirect addressing, as well as a destination data address in string manipulation instructions;
- EBP – base pointer is a register pointing to data in stack segment. BP register is usually used for based, based indexed or register indirect addressing;
- ESP – stack pointer is a register pointing to program stack.

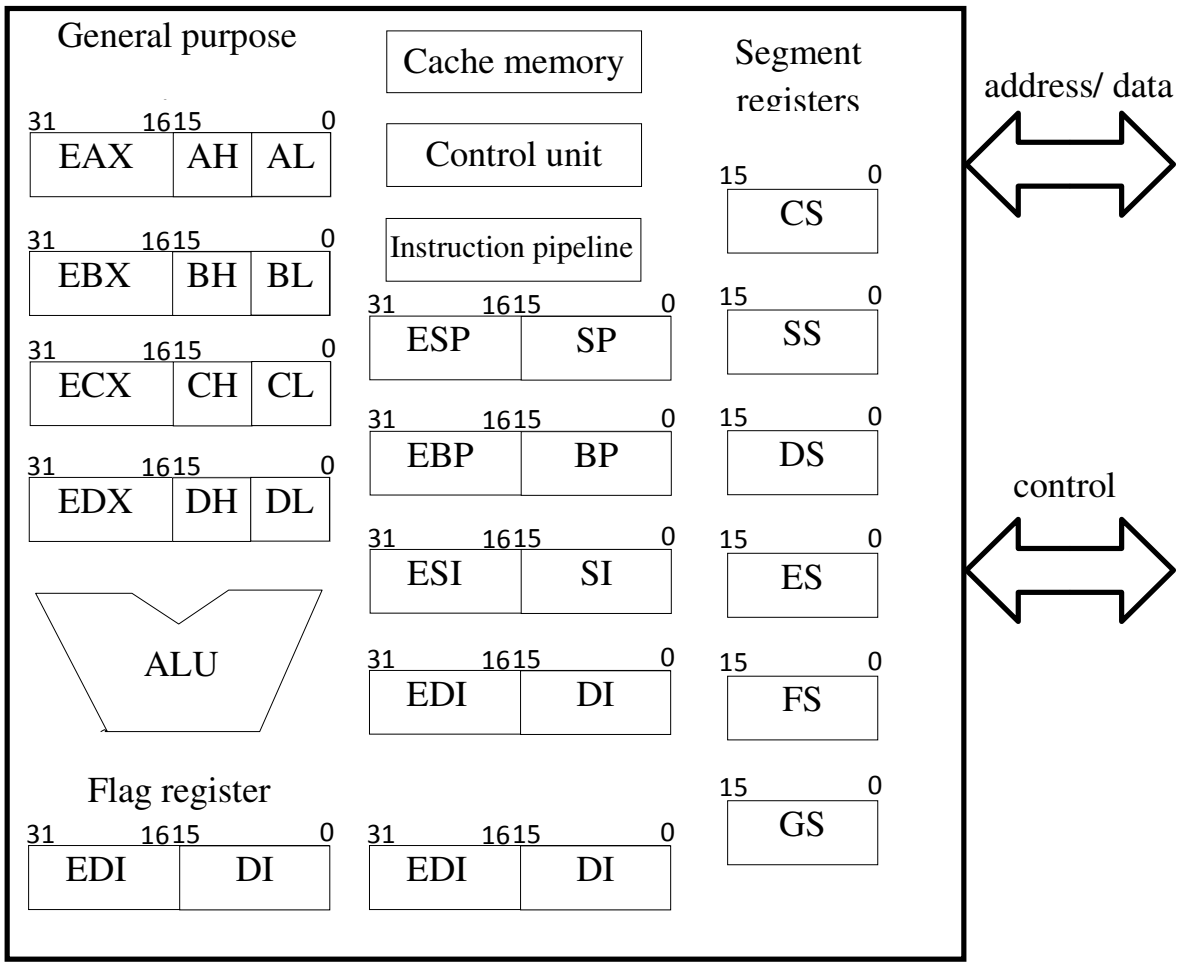


Figure 2.1 – Generic architecture of Intel microprocessors

4 Segment registers :

- SS – Stack segment is a 16-bit register containing address of 64KB segment with program stack. By default, the processor assumes that all data referenced by the stack pointer (SP) and base pointer (BP) registers is located in the stack segment;
- DS – data segment is a 16-bit register containing address of 64KB segment with program data. By default, the processor assumes that all data referenced by general registers (AX, BX, CX, DX) and index register (SI, DI) is located in the data segment;
- ES – extra segment is a 16-bit register containing address of 64KB segment, usually with program data. By default, the processor assumes that the DI register references the ES segment in string manipulation instructions.;
- CS – code segment is a 16-bit register containing address of 64 KB segment with processor instructions. The processor uses CS segment for all accesses to instructions referenced by instruction pointer (IP) register. CS register cannot be

changed directly. The CS register is automatically updated during far jump, far call and far return instructions.

5 Instruction pointer IP (EIP) is a 16(32)-bit register. IP register always works together with CS segment register and it points to the memory location that stores the next executable instruction. Branching is implemented by making changes to the instruction pointer. IP can be changed only indirectly (for example, a JUMP instruction will insert the operand into the instruction pointer).

6 Flag register EFLAGS can include result flags that record the results of certain kinds of testing, information about data that is moved, certain kinds of information about the results of comparisons or transformations, and information about some processor states.

7 Instruction pipeline sits in bus interface device. It consists of some contiguous bytes of memory, which are invoked from code segment independently of current instruction execution processing in ALU. Instruction pipeline allows increasing processor speed in some cases, specifically on linear sequence of instructions.

8 Cache (pronounced cash) memory is extremely fast memory that is built into a computer's central processing unit (CPU) (on-chip cache, I80486), or located next to it on a separate chip (external cache, I80386). The CPU uses cache memory to store instructions that are repeatedly required to run programs, improving overall system speed. Cache memory can consist of two parts: data memory and instruction memory.

## 2.2. Architecture features of certain INTEL processors

### 2.2.1. I8086 microprocessor

All internal registers of MP are 16-bit and contain two bytes. Data registers have the name AX, BX, CX, DX. Unlike all other groups of registers these registers can be divided into two one-byte:

- AX – AH and AL;
- BX – BH and BL;
- CX – CH and DL;
- DX – DH and DL,

to which it is possible to address program-independently.

Index registers, pointers and segment registers are 16-bit and have the name without letter E:

BP, SI, DI, SP, CS, SS, DS, ES, IP

ALU serves for performing of arithmetic and logic operations over 16- or 8-bit figures.

FLAGS register contains 16 bits, but ALU sets only 9 flags:

- CF (0) – carry flag is set if a carry out of the most significant bit of an operand occurs (addition). Also commonly set if a borrow occurs in a subtract;
- PF (2) – parity flag indicates an even number of 1 bits in low byte;
- AF (4) – auxiliary carry flag is set if a carry out of the most significant bit of a BCD operand occurs (binary coded decimal addition). Also commonly set if a borrow occurs in a BCD subtract;
- ZF (6) – zero flag indicates a zero result or an equal comparison;
- SF (7) – sign flag indicates a negative result or comparison;
- TF (8) – trap flag controls debug interrupt generation after instructions;
- IF (9) – interrupt flag controls whether interrupts are enabled;
- DF (10) – direction flag determines the direction of string operations (set for autoincrement, cleared for autodecrement). Works with registers SI and DI;
- OF (11) – overflow flag indicates a signed arithmetic overflow occurred.

Microprocessor I8088 is identical I8086 MP for programmer point of view.

### 2.2.2. I80186 microprocessor

I80186 MP – the 16-bit processor similar to processors I8086/8088 by the architecture and is compatible with them on object code level. I80188 microprocessor differs from I80186 by external data bus capacity which has eight bits.

### 2.2.3. I80286 microprocessor

I80286 MP is 16-bit processor and can work in 2 modes: real and protected. In the real mode this processor is completely similar to I8086, but all instructions are carried out faster at the expense of clock frequency increase and instructions execution modification. The protected mode is not considered here.

### 2.2.4. I80386 and I80486 microprocessors

I80386 and I80486 microprocessors are constructed by 32-bit architecture. It concerns to ALU, all data registers, pointers and index registers, the flag register and the instruction pointer. Segment registers are 16-bit. It is possible to access to the 16-bit registers which are a part of the 32-bit. 32-bit registers have prefix E in the name: EAX, EBX, ECX, EDX, EBP, ESI, EDI, ESP, EIP. But it is impossible to address only to the high part of 32-bit registers. I80386 and I80486 microprocessors have two new segment registers FS and GS which together with register ES form group of three additional data segments. EFLAGS register is also 32-bit, but additional flags are used in the protected mode and are not considered here. In I80386 and I80486 MP all data registers can be used for indirect addressing, for example, for indirect addressing with scaling that simplifies access to array elements, which consist of words and long words.

### 2.2.5. Software models of Intel microprocessors

Software models of MP are software accessible nodes, videlicet: all general purpose registers, pointers, segment registers, flag register (file register) and also ALU.

## 2.3. Registers loading

Initialization of general purpose registers EAX, EBX, ECX, EDX and their parts, and also pointers and index registers ESP, EBP, ESI, EDI is possible by using of direct addressing. Loading of segment registers SS, DS, ES, FS, GS can be carried out through general purpose registers, and in I80486 MP only through the accumulator AX. The code segment register CS is software inaccessible.

Examples of registers loading:

MOV EAX, 12345678H; Load of 12345678H into 32-bit EAX

MOV AX, 1234H; Load of 1234H into 16-bit accumulator AX

MOV AH, 34H; Load of 34H into high eight bits of the accumulator

MOV AL, 12H; Load of 12H into low eight bits of the accumulator

MOV BP, 0400H; Load of 0400H into base pointer

MOV SP, 0200H; Load of 0200H into stack pointer

MOV SS, AX; Load of initial segment address 3412H from AX into segment register SS

MOV DS, AX; Load of initial segment address 3412H from AX into segment register DS

MOV ES, AX; Load of initial segment address 3412H from AX into segment register ES

MOV SI, 400H; Load of 400H into index register SI

PUSH BX; Load of BX into top of a stack

POP CX; Load of top of a stack into CX

MOV AX, CS; Storing of a code segment in AX

POP BX; Load of top of a stack into CX

MOV SS, BX; Load of BX into SS

MOV AX, 0000H; Nulling of AX

PUSH DS; Stack initialization by the initial address

PUSH AX; Stack initialization by the zero address

## 3. Control questions

3.1 What kind of architecture features do allow to Intel microprocessors to be compatible?

3.2 Which groups of registers do Intel microprocessors contain?

3.3 How general purpose registers of MP are used?

3.4 What kind of appointment do pointers and index registers have?

3.5 What kind of appointment do segment registers have?

3.6 Which loading features do segment registers have?

3.7 Which nodes of Intel MP does the software model contain?

## 4. Homework

4.1 Make an example of 16-bit MP architecture.

4.2 Write the program using Assembly language for the 16-bit processor according to the algorithm given in the verbally-descriptive form. Variants parity is defined by penultimate figure of the student's card, and number – last.

4.3 Prepare a table of the program performance report in a step-by-step mode like on example:

Table 4.1 – Step-by-step program performance

| № | IP | Instruction mnemonic | AX | BX | CX | DX | SP | BP | SI | DI | CS | DS | SS | ES |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | |

4.4 Give answers to the control questions.

## Even variants

### № 1

1. Load the register AX by datum 1234H.
2. Store contents of AX register into a stack.
3. Load the register CX by quantity of cycles 3H.
4. Store contents of count register into the index register DI.
5. Zero the registers AX, BX, CX.
6. Restore contents of the count register.

### № 2

1. Load the count register by a decimal constant 145D.
2. Load the register AX by datum 1234H.
3. Store contents of AH register into the register DL.
4. Store contents of AL register into the register DH.
5. Zero the register AX.
6. Restore contents of the register AX.

### № 3

1. Load the segment register DS by initial address 4000H.
2. Load the register BX by a decimal constant -5D.
3. Load the register DX by a binary constant 1101B.
4. Swap the contents of the registers BX and DX.
5. Load the registers DX and AX by a dividend 12345678H.
6. Load the register BX by a divisor FAB0H.

### № 4

1. Load the segment register SS by address 2000H.
2. Load the register AX from the segment register DS.
3. Load the segment register ES from the count register.
4. Load the stack by contents of segment register ES.
5. Zero the count register.
6. Restore the contents of count register.

### № 5

1. Load the register AX by number 1010B.
2. Move the contents of AX register into DX register.
3. Swap the high and low bytes in the register DX.
4. Swap the contents of the registers AX and DX.
5. Load the register BX by number 2000H.
6. Load a segment of a stack from register BX.

**№ 6**

1. Combine the stack and additional segments.
2. Load the register AL by a constant 11110000B.
3. Load the index register DI by datum C50FH.
4. Load the index register SI by datum F12AH.
5. Swap the contents of index registers.
6. Zero the register BH.

**№ 7**

1. Load the register AL by number -5H**.**
2. Load the register BH by number ABH.
3. Swap the contents of the registers AL and BH.
4. Store only the high byte of datum in the AX.
5. Load the stack segment by address 17ABH.
6. Load the base pointer BP by datum 34ABH.

**№ 8**

1. Load the register BX by datum ABCDH**.**
2. Load the stack by contents of register BX.
3. Load the register AX by datum 1234H.
4. Store the contents of AX in the stack.
5. Swap the contents of AX and BX.
6. Load the additional segment register by address 3000H.

**№ 9**

1. Combine the stack segment and the data segment.
2. Load the registers AX, BX, DX by data 1234H, 5678H, 9ABCH accordingly.
3. Swap the contents of the registers AX and DX.
4. Load the CX by number 1000D.
5. Load the code segment register into the stack.
6. Store the contents of CX in the stack.

**№ 0**

1. Load the data segment register by number 3000H.
2. Load the additional segment register by number 4000H.
3. Swap the contents of these registers.
4. Load the register BH by number ABH.
5. Load the register BL by number 12H.
6. Swap the contents of registers BH and BL.

## Odd variants

### № 1

1. Load the register AH by number 2000H.
2. Load the segment registers DS and SS by the initial address of a segment from AX.
3. Zero the registers AX, BX, CX, DX by the different ways.
4. Load the stack pointer SP by number 50H.
5. Load the register BX by number 4261H.
6. Swap the contents of registers BH and CL.

### № 2

1. Combine the additional and stack segments.
2. Load AH from the stack segment register.
3. Load BX from the data segment register.
4. Swap the contents of DS and SS registers.
5. Load the contents of DS and SS registers into a stack.
6. Swap the contents of DS and SS registers.

### № 3

1. Load into the register AH number 3000H.
2. Combine the stack, additional and data segments.
3. Load into the register CX the counter of cycles.
4. Store its value in the stack.
5. Zero the counter.
6. Restore the counter.

### № 4

1. Load the segment register DS by number C100H.
2. Load the stack pointer by number 0000H. Load the base pointer BP by number 5FFAH.
3. Store the contents of the base pointer in the stack.
4. Zero the base pointer.
5. Restore the contents of the base pointer.
6. Move contents of the base pointer into the counter.

### № 5

1. Move the contents of CS register into the register AX**.**
2. Combine all segments, except the code segment.
3. Zero all general purpose registers.
4. Load the register DX by a constant -112D.
5. Store the contents of DX register in the stack.
6. Invoke the contents of DX register from the stack into the register AX.

**№ 6**

1. Load the register AL by number 10001101B.
2. Load the register AH by number -122D.
3. Load the base pointer from DX register.
4. Load the index register DI by number AB34H.
5. Load the index register SI from AX register.
6. Swap the contents of index registers.

**№ 7**

1. Load the register AL by number CDH.
2. Load the register AH by number ABH.
3. Store the contents of AX register in BX register.
4. Zero the register AX.
5. Restore the register AH.
6. Store the contents of AX register in the stack.

**№ 8**

1. Load the register SI by number A7B8H.
2. Load the register DI by number B8A7H.
3. Store the contents of register SI in the stack.
4. Store the contents of register DI in the stack.
5. Swap the contents of SI and DI registers with stack use.
6. Swap the contents of SI and DI registers again.

**№ 9**

1. Load the counter by number ABH.
2. Load the register BX from the count register.
3. Store the contents of register BX in the stack.
4. Zero the contents of register DX.
5. Copy the contents of register BX from a stack into DX register.
6. Swap the contents of CX and DX registers.

**№ 0**

1. Load the DX and AX registers by a dividend 87654321H.
2. Load the register DX from the count register.
3. Load the segment register SS by address 4000H.
4. Load the contents of register SS into the stack.
5. Copy the contents of register SS into the register DS.
6. Swap the contents of SS and DS registers.

## 5. Laboratory task

5.1 Invoke DEBUG program. The processor in this program is used as I80286 in a real mode that is works with 16-bit registers.

5.2 Display the contents of all MP registers by means of the instruction:

R <ENTER>

5.3 Set the contents of IP register and code segment register CS:

(CS) =7000H, (IP) =100H.

5.4 Enter the program, written in a homework, starting from (CS):(IP) address by means of the instruction

A(CS):(IP) <ENTER>

After each pressing a key <ENTER> on the screen there will be a current address of a code segment on which it is possible to enter the Assembly command. Input of an instruction is carried out by a key <ENTER>. By the errors detection in instructions it is necessary to write correctly instructions to the same addresses.

5.5 Assemble the entered program, for what to enter the instruction

u (IP start)... (IP end) <ENTER>

After that the program in the form of the object module will be located in memory to those addresses on which instructions were entered.

5.6 Look over the area of memory occupied with the program. Find machine codes of the first instruction.

5.7 Execute the program in a step-by-step mode. For this purpose it is necessary to enter the instruction:

T = (CS): (IP start) 1 <ENTER>

owing to which the first command will be executed. On the screen there will be contents of all registers and the instruction that will be carried out next. Analyse the result of the first instruction's performance. For performance of following instruction it is necessary to enter the directive:

T <ENTER>

5.8 Fill in the report of laboratory work performance in the form of table for all program instructions.

5.9 Analyse the report of the program performance in a step-by-step mode.

5.10 Use next directive to escape from Debug program

Q <ENTER>

## 6. Contents of the protocol

It is necessary to record the title and purpose of the laboratory work, to prepare homework according your variant, to represent Intel microprocessors architecture and give answers to the control questions.

## 7. The literature list

7.1 Майко Г.В. Ассемблер для IBM PC. – М.: Бизнес - Информ, Сирин, 1997.

7.2 Нортон П. Персональный компьютер фирмы IBM и операционная система MS DOS. – М.: Радио и связь, 1992.