Practical lesson No14-15

Subject. Working with Assembly Commands

Purpose of the lesson

Master basic x86 assembly commands for data manipulation, arithmetic, and flow control. Learn how to write, build, and debug programs using TASM in DOSBox. Develop skills in analyzing machine code and processor flags.

Objectives

- Study the groups of commands: movement (MOV), arithmetic (ADD, SUB, CMP), control (JMP, JZ).
- Write simple programs for calculation and conditional logic.
- Complete hands-on tasks to debug and optimize your code.
- Learn how to work with flags (ZF, CF) for conditional transitions.

Materials Needed

- A computer with DOSBox and TASM installed (from Lesson No13).
- Text editor (EDIT in DOSBox or Notepad++ for .asm files).
- Notepad for register and flag schemas.

Theoretical part

Main groups of assembly commands

Assembly commands are divided into categories: data movement (MOV - copies a value), arithmetic (ADD - addition, SUB - subtraction, CMP - comparison with setting flags), logical (AND, OR) and control (JMP - unconditional transition, JZ - transition if ZF=1). Flags are bits in the FLAGS register: ZF (zero), CF (carry), SF (sign). After the CMP, the flags define the result (equal, greater, less than).

Example explanation: The MOV AX command, 5 is like "put 5 in the AX box" (register). ADD AX, 3 - "add 3 to AX" (AX=8). CMP AX, 10—Compare AX to 10, set ZF=0 (not equal), CF=0 (no borrowing). JZ label — "if ZF=1, go to label" (conditional if).

Example table (basic commands and flags):

Group	Team	Description	Example	Influence on flags
Moving	MOV	Copying data	MOV AX, BX	No
Arithmetic	ADD	Addition	ADD AX, 5	ZF, CF, SF
Comparison	CMP	Comparison (does not change)	CMP AX, BX	ZF, CF, SF
Transition	JMP	Unconditional transition	JMP loop	No
Condition	JZ	Transition if ZF=1	CMP AX,0; JZ end	Uses ZF

Example diagram (registers and flags after ADD; described as block diagram): Horizontal line "AX Register" (16 bits: 0008h after ADD 5+3). On the left is "Input1: 0005h", on the right is "Input2: 0003h", down arrow to "ALU: +", output to AX. Below is "FLAGS Register" (bar: bit 0=CF=0, bit 6=ZF=0, bit 7=SF=0). Arrows from ALU to flags: "No overflow → CF=0", "Not zero → ZF=0".

Working with Loops and Conditions

Loops: LOOP (decreases CX, transition if CX≠0). Conditions: After CMP – JG (more), JL (less). In TASM: Use labels (label:) for transitions.

Example explanation: For the sum of 1+2+3: MOV CX,3 (counter); MOV AX,0 (amount); loop: ADD AX,CX; DEC CX; JNZ loop — AX=6 after 3 iterations (CX=0).

Practical part

Demo 1: Arithmetic Operations

Description: A program for adding and comparing numbers.

Steps (in DOSBox, EDIT arith.asm):

1. . MODEL SMALL; . STACK 100h; . DATA num1 DW 10; num2 DW 5; . CODE MAIN: MOV AX,num1; ADD AX,num2; CMP AX,20; MOV AH,4Ch; INT 21h; END MAIN.
2. Assemble: TASM arith.asm; TLINK arith.obj /t; arith (run - no output, but check in DEBUG: D AX - 0015h).
3. IN DEBUG: L arith.exe; R AX (check 0015h); T (step: after ADD AX=15).

Expected result: After ADD AX=15, CMP sets ZF=0 (15<20).

Analysis: Why doesn't the CMP change the AX? (Flags only; for conditions).

Demo 2: Conditional Transition and Loop

Description: A program with a loop to output numbers 1-3.

Steps (EDIT loop.asm):

1. . MODEL SMALL; . STACK 100h; . DATA; . CODE MAIN: MOV CX,3; MOV BX,1; loop_start: MOV AX,BX; ; (здесь вывод, упрощенно) INC BX; DEC CX; JNZ loop_start; MOV AH,4Ch; INT 21h; END MAIN.
2. Collect and run: TASM loop.asm; TLINK loop.obj /t; loop.
3. In TD (if available): TD loop.exe, BP loop_start, G – Step, observe CX (3→2→1→0).

Expected result: The cycle is executed 3 times, BX=4 after INC.

Analysis: JNZ uses ZF from DEC (CX≠0 → ZF=0, junction).

Independent Assignments for Students

1. **Task 1:** Write sub.asm: MOV AX,15; SUB AX,7 (AX=8); CMP AX,10; JL less (tag: MOV BX,1); JMP end; less: MOV BX,0; end: . Build, in DEBUG: T after SUB (AX=0008h), after CMP (flags: SF=0, ZF=0).

   Example table (flags after CMP 8<10):

   Flag Significance Why

   CF 1 Loan (8<10)

   ZF 0 Not equal to

   SF 0 Positive

2. **Task 2:** Create cond.asm: MOV AX,5; CMP AX,10; JG greater; MOV DX,0; JMP end; greater: MOV DX,1; end: MOV AH,4Ch; INT 21h. Build, test (DX=0, transition did not work).
3. **Task 3:** Extend loop.asm: Add an output (use INT 21h,09h with a string). Build, run, and you'll see "123" (simulate).

Control: The teacher checks the .asm files and the DEBUG/TD output.

Final questions and assignments for students

Questions for self-examination (10 questions, answer in writing)

1. What does the MOV command do and does it affect the flags?
2. Describe the difference between ADD and CMP.
3. What is the ZF flag and when is it installed?
4. How does JMP compare to JZ?
5. Why do I need CX in a LOOP?
6. What happens after DEC CX in JNZ?
7. Give an example of a SUB with CF=1.
8. How is CMP used for if?
9. Why does the RET terminate the procedure?
10. What is a tag in assembler?

Homework (5 tasks)

1. Describe (1 page) the effect of arithmetic commands on flags (ADD/SUB examples).
2. Write a program fact.asm (factorial 3=6 with a loop), build and debug.
3. Create sort.asm (comparing two numbers, printing the larger one with INT 21h).
4. Compare x86 and ARM commands (Table 3 Commands, 1 page).
5. The gcd.asm program (GCF of two numbers with the CMP/JG cycle), test on 12 and 18.