


**Федеральное государственное бюджетное образовательное
учреждение высшего образования "Белгородский государственный
технологический университет им. В.Г. Шухова"**

Кафедра программного обеспечения вычислительной техники и
автоматизированных систем.

Лабораторная работа № 3
Арифметические команды центрального процессора.
Вариант 13

Выполнил:
Студент группы КБ-211


_____ Коренев Д.Н.

Принял:

_____ Осипов О.В.

Цель работы: изучение арифметических команд центрального процессора для работы с целыми числами.

Задание

1. Написать программу для вычисления значения арифметического выражения согласно варианту задания. Все переменные, используемые в программе, требуется использовать как знаковые и расширять до размерности двойного слова. Результат должен быть записан в регистр EAX. Если результат содержит остаток от деления, оставить его в регистре EDX. Подобрать набор тестовых данных (не менее 3).

Вариант	Выражение	Размер входных параметров	Операция Размерность (2-е задание)
13	$ax + bx^2 + dx^3 - 14^3$	a, b, d – byte x – word	сложение 14 байт

2. Написать программу для сложения или вычитания целых беззнаковых чисел большой размерности (размерность и операция зависят от варианта задания). Младшие байты при этом хранить по младшему адресу. Подобрать наборы тестовых данных (не менее 3). Для выполнения этого задания изучить теоретический материал главы «Вычитание и сложение операндов большой размерности», начиная со страницы 176 учебника Юрова «Assembler».

```
.386
.model flat, stdcall
option casemap: none

include include\windows.inc
include include\kernel32.inc
include include\user32.inc
include include\msvcrt.inc

includelib user32.lib
includelib kernel32.lib
includelib msvcrt.lib

mem_dump32 MACRO mem_addr, rows
    LOCAL loop1, loop2
    LOCAL mem_dump_fmt, header_fmt, footer_fmt
    LOCAL r_edi, r_ebx, temp_mem_addr, stack_decr
    LOCAL holder_esi, holder_ebp, holder_ecx, holder_eax
    LOCAL reg_dump_fmt1, reg_dump_fmt2, reg_header
    .data
        reg_dump_fmt1 DB "| EAX=0x%08x EBX=0x%08x ECX=0x%08x EDX=0x%08x |",
            13, 10, 0
        reg_dump_fmt2 DB "| ESI=0x%08x EDI=0x%08x EBP=0x%08x ESP=0x%08x |",
```

```

        13, 10, 0
reg_header DB "+==--REGISTERS-", 51 DUP ("="), "+", 13, 10, 0
mem_dump_fmt DB "| %p", 9, " ", 16 DUP ("%02x "), "|", 13, 10, 0
header_fmt DB "+==--ADDR-", 7 DUP ("="),
        "-00-01-02-03-04-05-06-07-08-09-0A-0B-0C-0D-0E-0F-+", 13, 10, 0
footer_fmt DB "+", 41 DUP ("="), 45, 64, 75, 83, 69, 69, 78, 45, 61, 45,
        64, 75, 69, 82, 65, 83, 73, 46, 82, 85, 45, 61, 61, 43, 13, 10, 0
temp_mem_addr DD ?
r_edi DD ?
r_ebx DD ?
holder_esi DD ?
holder_ebp DD ?
holder_ecx DD ?
holder_eax DD ?
stack_decr DD 0
.code
; Сохраняем регистры
mov holder_esi, ESI
mov holder_ebp, EBP
mov holder_ecx, ECX
mov holder_eax, EAX

; Выводим регистры
invoke crt_printf, offset reg_header
invoke crt_printf, offset reg_dump_fmt1, EAX, EBX, ECX, EDX
invoke crt_printf, offset reg_dump_fmt2, ESI, EDI, EBP, ESP

invoke crt_printf, offset header_fmt
mov ESI, mem_addr
mov temp_mem_addr, mem_addr
mov r_edi, rows
mov r_ebx, mem_addr ; EBX - address for left column
mov EBP, 15 ; offset
loop2:
    mov ECX, 16
loop1:
    mov EAX, 0 ; EAX - байт данных
    mov AL, [ESI+EBP]
    push EAX
    inc stack_decr
    dec EBP
    dec ECX
    jnz loop1
    add EBP, 32
    push r_ebx
    push offset mem_dump_fmt
    inc stack_decr
    inc stack_decr
    call crt_printf
    inc stack_decr

```

```

add r_ebx, 16
dec r_edi
jnz loop2
invoke crt_printf, offset footer_fmt

; ЧИСТИМ СТЕК
mov EAX, stack_decr
imul EAX, 4
add ESP, EAX

; Восстанавливаем регистры
mov ESI, holder_esi
mov EBP, holder_ebp
mov ECX, holder_ecx
mov EAX, holder_eax
ENDM

```

print_equation **MACRO** a, b, d, x, res

LOCAL newline, hhd_fmt, hd_fmt, eq_fmt, pow_fmt, plus_fmt, mul_fmt, sub_fmt

.data

```

newline db 13, 10, 0
hhd_fmt db "%hhd", 0
hd_fmt db "%hd", 0
d_fmt db "%d", 0
eq_fmt db "=", 0
pow_fmt db "^", 0
plus_fmt db "+", 0
mul_fmt db "*", 0
sub_fmt db "-", 0

```

.code

```

invoke crt_printf, offset hhd_fmt, a
invoke crt_printf, offset mul_fmt
invoke crt_printf, offset hd_fmt, x
invoke crt_printf, offset plus_fmt
invoke crt_printf, offset hhd_fmt, b
invoke crt_printf, offset mul_fmt
invoke crt_printf, offset hd_fmt, x
invoke crt_printf, offset pow_fmt
invoke crt_printf, offset hd_fmt, 2
invoke crt_printf, offset plus_fmt
invoke crt_printf, offset hhd_fmt, d
invoke crt_printf, offset mul_fmt
invoke crt_printf, offset hd_fmt, x
invoke crt_printf, offset pow_fmt
invoke crt_printf, offset hd_fmt, 3
invoke crt_printf, offset sub_fmt
invoke crt_printf, offset hd_fmt, 14
invoke crt_printf, offset pow_fmt
invoke crt_printf, offset hd_fmt, 3
invoke crt_printf, offset eq_fmt

```

```

        invoke crt_printf, offset d_fmt, res
        invoke crt_printf, offset newline
    ENDM

print_var MACRO var, var_size
    LOCAL loop1
    LOCAL stack_decr
    LOCAL holder_esi, holder_ebp, holder_ecx, holder_eax, holder_edi, holder_ebx
    LOCAL var_fmt
    .data
        var_fmt DB "0x", var_size - 1 DUP ("%02x_"), "%02x", 0
        holder_esi DD ?
        holder_ebp DD ?
        holder_ecx DD ?
        holder_eax DD ?
        holder_edi DD ?
        holder_ebx DD ?
        stack_decr DD 0
    .code
        mov holder_esi, ESI
        mov holder_ebp, EBP
        mov holder_ecx, ECX
        mov holder_eax, EAX
        mov holder_edi, EDI
        mov holder_ebx, EBX

        mov ESI, offset var
        mov EDI, 3
        mov EBX, offset var ; EBX - addres for left column
        mov EBP, 0 ; offset

        mov ECX, 0
    loop1:
        mov EAX, 0 ; EAX - data of byte
        mov AL, [ESI+EBP]
        push EAX
        inc stack_decr
        inc EBP
        inc ECX
        cmp ECX, var_size
        jne loop1
        push offset var_fmt
        inc stack_decr
        call crt_printf
        inc stack_decr
        add EBX, 16
        dec EDI

        ; Чистим стек
        mov EAX, stack_decr

```

```

    imul EAX, 4
    add ESP, EAX

; Восстанавливаем регистры
    mov ESI, holder_esi
    mov EBP, holder_ebp
    mov ECX, holder_ecx
    mov EAX, holder_eax
    mov EDI, holder_edi
    mov EBX, holder_ebx
ENDM

```

.data

```

a db 1
b db 2
d db 90
x dw 4
res dd 0

```

```

fmt_scanf_db db "%hhd", 0
fmt_scanf_dw db "%hd", 0

```

```

msg_in_a db "Input 'a': ", 0
msg_in_b db "Input 'b': ", 0
msg_in_d db "Input 'd': ", 0
msg_in_x db "Input 'x': ", 0

```

```

; ax+bx^2+dx^3-14^3=

```

```

sum_num1 db
0FFh,0FFh,0FFh,0FFh,0FFh,0FFh,0FFh,0FFh,0FFh,0FFh,0FFh,0FFh,0FFh,0FFh,0FFh
sum_num2 db 1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
sum_res db 14 DUP(0)

```

.code

start:

```

    invoke crt_printf, offset msg_in_a
    invoke crt_scanf, offset fmt_scanf_db, offset a

```

```

    invoke crt_printf, offset msg_in_b
    invoke crt_scanf, offset fmt_scanf_db, offset b

```

```

    invoke crt_printf, offset msg_in_d
    invoke crt_scanf, offset fmt_scanf_db, offset d

```

```

    invoke crt_printf, offset msg_in_x
    invoke crt_scanf, offset fmt_scanf_dw, offset x

```

```

movsx EAX, a      ; a
movsx ECX, x
imul EAX, ECX     ; ax

movsx EBX, b      ; b
imul EBX, ECX     ; bx
imul EBX, ECX     ; bx^2

movsx EDI, d      ; d
imul EDI, ECX     ; dx
imul EDI, ECX     ; dx^2
imul EDI, ECX     ; dx^3

mov EDX, 14       ; 14
imul EDX, 14      ; 14^2
imul EDX, 14      ; 14^3

add EAX, EBX      ; ax+bx^2
add EAX, EDI      ; ax+bx^2+dx^3
sub EAX, EDX      ; ax+bx^2+dx^3-14^3

mov res, EAX

mem_dump32 (offset a), 5
print_equation a, b, d, x, res


mov EAX, dword ptr sum_num1[0]
add EAX, dword ptr sum_num2[0]
mov dword ptr sum_res[0], EAX

mov EAX, dword ptr sum_num1[4]
adc EAX, dword ptr sum_num2[4]
mov dword ptr sum_res[4], EAX

mov EAX, dword ptr sum_num1[8]
adc EAX, dword ptr sum_num2[8]
mov dword ptr sum_res[8], EAX

movzx EAX, word ptr sum_num1[12]
movzx ECX, word ptr sum_num2[12]
adc EAX, ECX
mov dword ptr sum_res[12], EAX

mem_dump32 (offset sum_num1), 5
print_var sum_res, 15

```

```
push 0  
call ExitProcess ; Выход из программы  
  
end start
```

Вывод: в ходе лабораторной работы мы изучили арифметические команды центрального процессора для работы с целыми числами.