**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 |  | *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 - addres 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

    sum\_num2 db 1,0,0,0,0,0,0,0,0,0,0,0,0,0

    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.