**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**



ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ

**Лабораторная работа №3**

по дисциплине: Архитектура вычислительных систем

тема: «Арифметические команды центрального процессора»

Выполнил: ст. группы ПВ-233

Мовчан Антон Юрьевич

Проверили:

ст. пр. Осипов Олег Васильевич

Белгород 2025 г.

**Лабораторная работа №3  
Арифметические команды центрального процессора  
Вариант 8**



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

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). Каждая строка исходного кода программы обязательно должна быть прокомментирована. Программы без подробных комментариев не принимаются!
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».

1. Исходный код:

.686

.model flat, stdcall

option casemap: none

include kernel32.inc

include msvcrt.inc

includelib kernel32.lib

includelib msvcrt.lib

.DATA

x db 10

y db 20

z dw 30

str\_fmt db "x = %d, y = %d, z = %hd, R = %d", 0

.CODE

START:

movsx EAX, x *; Расширение байта x до двойного слова в EAX*

add EAX, 10 *; EAX = EAX + 10*

mov EBX, EAX *; EBX = EAX*

movsx EAX, y *; Расширение байта y до двойного слова в EAX*

sub EAX, 5 *; EAX = EAX - 5*

mul EBX *; EAX = EAX \* EBX*

mov EBX, EAX *; EBX = EAX*

movsx EAX, z *; Расширение слова z до двойного слова в EAX*

mov ECX, 3 *; ECX = 3*

cdq *; Расширение до двойного слова*

idiv ECX *; EAX = EDX:EAX / ECX(3)*

mov EDX, EAX *; EDX = EAX*

movsx EAX, z *; Расширение слова z до двойного слова в EAX*

sub EAX, EDX *; EAX = EAX - EDX*

mul EBX *; EAX = EAX \* EBX*

sub EAX, 7\*7\*7\*7 *; EAX = EAX - 2401*

*; Вызов функции printf("x = %d, y = %d, z = %hd, R = %d", (int)x, (int)y, z, EAX)*

push EAX *; Поместим в стек итоговый результат. Выводим его как 4-байтовое со спецификатором %d*

push dword ptr z *; Для числа z типа short используем спецификатор %hd*

movsx EAX, x *; EAX = (int)x. Число x расширим до 4-байтового и используем для него спецификатор %d*

push EAX

movsx EAX, y *; EAX = (int)y. Число y расширим до 4-байтового и используем для него спецификатор %d*

push EAX

push offset str\_fmt

call crt\_printf

add ESP, 5\*4 *; Очистка стека от аргументов*

push 0

call ExitProcess *; Выход из программы*

END START

|  |  |  |  |
| --- | --- | --- | --- |
| x | y | z | Результат |
| 10 | 20 | 30 | 3599 |
| -10 | -20 | -30 | -2401 |
| -100 | 10 | -40 | 9749 |

Результат выполнения программы:







2. Исходный код:

.686

.model flat, stdcall

option casemap: none

include kernel32.inc

include msvcrt.inc

includelib kernel32.lib

includelib msvcrt.lib

.DATA

a db 7Fh, 0A3h, 0C1h, 0B9h, 0E0h, 8Dh, 45h, 0F2h, 0A1h, 0B3h, 7Ch, 9Dh, 58h, 0E4h, 0A6h *; 15 байт*

b db 12h, 0D9h, 0A7h, 0F5h, 0C4h, 0E3h, 8Bh, 01h, 0FAh, 76h, 0D2h, 0B9h, 0C5h, 8Eh, 30h *; 15 байт*

r db 15 dup(?) *; Для результата резервируется 15 байт*

.CODE

START:

mov EAX, dword ptr a[0]

sub EAX, dword ptr b[0]

mov dword ptr r[0], EAX

mov EAX, dword ptr a[4]

sbb EAX, dword ptr b[4]

mov dword ptr r[4], EAX

mov EAX, dword ptr a[8]

sbb EAX, dword ptr b[8]

mov dword ptr r[8], EAX

mov AX, word ptr a[12]

sbb AX, word ptr b[12]

mov word ptr r[12], AX

mov AL, byte ptr a[14]

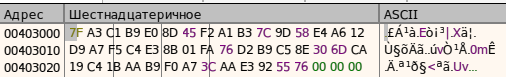
sbb AL, byte ptr b[14]

mov byte ptr r[14], AL

push 0

call ExitProcess

END START



Вывод: в ходе выполнения л.р. я изучил арифметические команды центрального процессора для работы с целыми числами.