**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по лабораторной работе №3**

**по дисциплине «Организация ЭВМ и систем»**

Тема: Изучение организации ветвлений в программах на языке ассемблера

|  |  |  |
| --- | --- | --- |
| Студент гр. 3388 |  | Шубин П.А. |
| Преподаватель |  | Фирсов М.А. |

Санкт-Петербург

2024

**Цель работы.**

Изучение принципов разработки программ на языке ассемблера с использованием условных переходов, арифметических операций и работы с числами со знаком.

**Основные теоретические положения.**

Инструкция CMP (от слова compare - сравнить) позволяет сравнить значения и установить флаги. Благодаря чему мы можем использовать результат сравнения для выполнения условного перехода.

**Команды условного перехода:**

JE - переход при равенстве операндов

JNE - переход при неравенстве операндов

JG/JNLE - переход если первый операнд больше

JL/JNGE - переход если первый операнд меньше

JGE - переход если первый операнд больше или равен

JLE - переход если первый операнд меньше или равен

Команда NEG выполняет изменение знака числа путем вычисления дополнительного кода. Механизм работы: инвертирование всех битов операнда и добавление единицы к результату.

Команда SHL осуществляет логический сдвиг влево. Биты операнда сдвигаются влево на указанное число позиций. Освободившиеся справа позиции заполняются нулями. Один сдвиг влево умножает число на 2, двойной сдвиг - на 4, тройной - на 8.

**Задание**

Разработать на языке Ассемблер iX86 программу, которая по заданным целым значениям a,b,i,k, размером 1 слово, вычисляет:

а) значения i1 = fn1(a,b,i) и i2 = fn2(a,b,i);

b) значения res= fn3(i1,i2,k),

где вид функций fn1,fn2 определяется из табл. 1, а функции fn3 — из табл.2 по цифрам шифра индивидуального задания (n1.n2.n3).

Значения a,b,i,k являются исходными данными, которые должны выбираться студентом самостоятельно и задаваться в процессе исполнения программы в режиме отладки. При этом следует рассмотреть все возможные комбинации параметров a,b и k, позволяющие проверить различные маршруты выполнения программы.

Замечания:

1) при разработке программы не использовать фрагменты, представленные на ЯВУ, в частности, для ввода-вывода данных. Исходные данные должны вводиться, а результаты контролироваться в режиме отладки;

2) при вычислении функций fn1 и fn2 вместо операции умножения следует использовать арифметический сдвиг и, возможно, сложение;

3) не использовать процедуры (в том числе при вычислении функций fn1 и fn2);

4) при разработке программы следует минимизировать длину кода; для этого могут потребоваться преобразования формул и введение промежуточных результатов.

Для выполнения задания:

1. Преобразовать формулы, по которым будут выполняться вычисления, согласно замечанию 5.

2. В табличном процессоре (например, LibreOffice Calc) создать документ, в котором в 4 ячейки вводятся входные данные (a, b, i, k), в 3 ячейках появляются результаты i1, i2, res, вычисленные по исходным формулам, и ещё в одной ячейке — res, вычисленный по преобразованной формуле. Проверить, что результаты совпадают для разных входных данных.

3. Разработать программу; в коде должны быть выделены части, посвящённые вычислению итоговых значений (3 штуки) и ключевых промежуточных значений, если таковые были определены при преобразовании формул. Код должен быть с подробными комментариями: после каждой вычислительной команды должно быть указано полученное к текущему моменту значение. Подсчитать количество команд в главной вычислительной части и указать в комментарии.

4. Протестировать программу, оформив таблицу с входными данными и результатами. Тесты должны проверять все пути выполнения вычислений, и их точно не менее 4.

Вариант 24. Шифр 5.7.5

/20-4\*i, при a>b

F5 = <

\ -(6\*i-6), при a<=b

/-(4\*i-5), при a>b

f7 = <

\ 10-3\*i, при a<=b

/ min(|i1|,6), при k=0

F5 = <

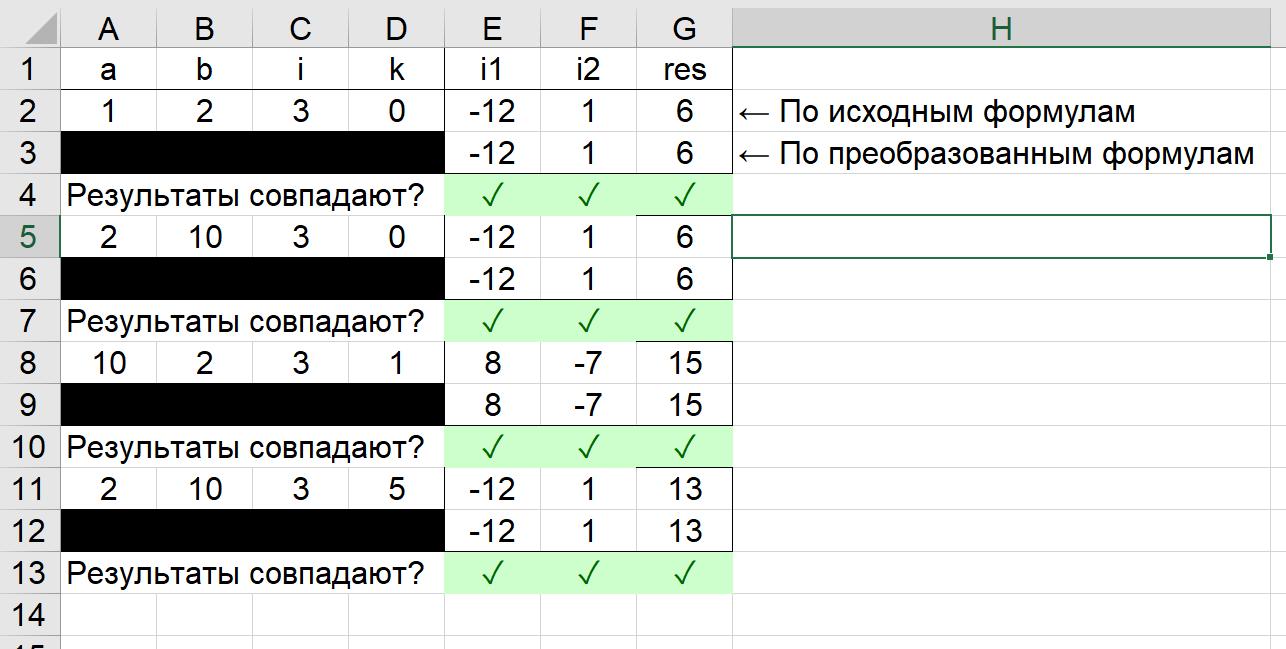
\ |i1|+|i2|, при k/=0

**Выполнение работы.**

1. В связи с тем, что формула для вычисления первой функции f5 при a > b походит на формулу функции f7 при a<b, было принято решение сократить ее:

-(4\*i-5) == i1-15, где i1 – результат выполнения первой функции f5.

2. С помощью программы Microsoft Excel были вычислены тестовые значения и проверена замена формулы. Проверка проводилась на четырех различных случаях, каждый из которых задействует разные варианты функций программы. (см. Рисунок 1)

Рисунок 1

3. Была разработана программа, полностью соответствующая условию: код разбит на части, посвященные вычислению итоговых значений. В коде присутствуют подробные комментарии. Код указан в приложении.

4. Программа протестирована на всех возможных путях выполнения вычислений. Результаты занесены в Табл. 1

Табл. 1

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| a = 1  b = 2  i = 3  k = 0 | i1 = FFF4 i2 = 0001 res = 0006 |
| a = 2  b = 10  i = 3  k = 0 | i1 = FFF4 i2 = 0001 res = 0006 |
| a = 10  b = 2  i = 3  k = 1 | i1 = 0008  i2 = FFF9  res = 000F |
| a = 10  b = 2  i = 3  k = 0 | i1 = 0008  i2 = FFF9  res = 0006 |

**Вывод.**

Были изучены принципы разработки программ на языке ассемблера с использованием условных переходов, арифметических операций и работы с числами со знаком. Реализована программа, в которой применены полученные знания на практике.

# **ПРИЛОЖЕНИЕ**

Файл LR3.ASM

; Стек программы

AStack SEGMENT STACK

DW 12 DUP(?)

AStack ENDS

; Данные программы

DATA SEGMENT

a DW 1 ; Входное значение a

b DW 2 ; Входное значение b

i DW 3 ; Входное значение i

k DW 0 ; Входное значение k

i1 DW 0 ; Результат fn1

i2 DW 0 ; Результат fn2

res DW 0 ; Результат fn3

DATA ENDS

; Код программы

CODE SEGMENT

ASSUME CS:CODE, DS:DATA, SS:AStack

Main PROC FAR

push DS

sub AX,AX

push AX

mov AX,DATA

mov DS,AX

; ---- Вычисление i1 = fn1(a, b, i) ----

mov dx, i

shl dx, 1 ;2\*i

shl dx, 1 ;4\*i

mov ax, a

cmp ax, b

jg fn1\_greater ; Если a > b, переход к fn1\_greater

; Если a <= b:

; fn1 = -(6\*i - 6) = -6\*i + 6

mov cx, dx ; cx = 4\*i

add cx, i ; 5\*i

add cx, i ; 6\*i

neg cx ; -6\*i

add cx, 6 ; -6\*i + 6

mov i1, cx

mov bx, i1 ; bx = i1

get\_abs\_i1:

neg bx

js get\_abs\_i1 ; bx = |i1|

store\_i2:

; ---- Вычисление i2 = fn2(a, b, i) ----

; Если a <= b:

; fn2 = 10 - 3\*i

sub dx, i ; 3\*i

mov i2, 10

sub i2, dx ; 10 - 3\*i

jmp store\_res

fn1\_greater:

; fn1 = 20 - 4\*i

mov i1, 20

sub i1, dx ; i1 = 20 - 4\*i

fn2\_greater:

; fn2 = -(4\*i - 5) = -4\*i + 5 = i1 - 15

mov cx, i1

sub cx, 15 ; i2 = i1 - 15

mov i2, cx

store\_res:

; ---- Вычисление res = fn3(i1, i2, k) ----

cmp k, 0

je fn3\_if\_k\_zero

; Если k != 0:

; res = |i1| + |i2|

mov ax, i2 ;ax = i2

get\_abs\_i2:

neg ax

js get\_abs\_i2 ;ax = |i2|

find\_res:

add ax, bx ; |i1| + |i2|

mov res, ax ; res = ax

jmp end\_program

fn3\_if\_k\_zero:

; res = min(|i1|, 6)

cmp bx, 6

jle set\_res\_i1 ; Если |i1| <= 6, то res = |i1|

mov bx, 6 ; Иначе res = 6

set\_res\_i1:

mov res, bx ; res = bx

end\_program:

; Завершение программы Количество строк в главной вычислительной части: 40

mov ax, 4C00h

int 21h

Main ENDP

CODE ENDS

END Main