**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе№3**

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

Тема: **Изучение организации ветвлений в программах на языке ассемблера****.**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3388 |  | Сабалиров М.З. |
| Преподаватель |  | Фирсов М.А. |

Санкт-Петербург

2024

## **Цель работы.**

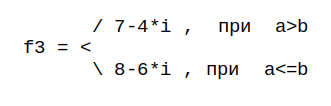
* 1. Разработать на языке Ассемблера программу, которая по заданным целочисленным значениям параметров вычисляет значения функций.
  2. Задание.
  3. Разработать на языке Ассемблер iX86 программу, которая по заданным целым значениям a,b,i,k, размером 1 слово, вычисляет:

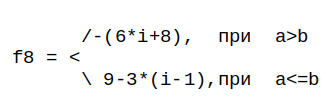
а) значения i1 = fn1(a,b,i) и i2 = fn2(a,b,i);

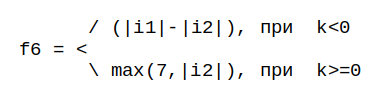
b) значения res= fn3(i1,i2,k),

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 нужные ячейки памяти в режиме отладки. При этом следует рассмотреть все возможные комбинации параметров a, b и k, позволяющие проверить различные маршруты выполнения программы.







* 1. Для выполнения задания:
  2. 1. Преобразовать формулы, по которым будут выполняться вычисления, согласно замечанию 5.
  3. 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.
  4. 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.
  5. 4. Протестировать программу, оформив таблицу с входными данными и результатами. Тесты должны проверять все пути выполнения вычислений, и их точно не менее 4.
  6. **Основные теоретические положения**

Для выполнения данного задания были использованы такие команды как:

1)Mov – присваивание

2)Add - сложение

3) Sub - вычитание

4) Cmp – сравнение

5) Neg – смена знака

6) Sal - арифметический сдвиг влево

7) Jmp - безусловный переход

8) Int - вызов программного прерывания

9) Jge(jump greater equal) - выполняет короткий переход, если первый операнд больше второго операнда или равен ему при выполнении операции сравнения с помощью команды cmp

**10) Jg(jump greater) -** выполняет короткий переход, если первый операнд больше второго операнда при выполнении операции сравнения с помощью команды cmp.

11) jl(jump less) - выполняет короткий переход, если первый операнд меньше второго операнда при выполнении операции сравнения с помощью команды CMP.

Cmp при сравнении выставляет следующие флаги:

1) Флаг нуля **ZF** устанавливается, если AX = BX

2) Флаг знака **SF** устанавливается, если результат отрицательный. При этом установка флага НЕ означает, что AX обязательно меньше BX.

* + Если ((SF = 0) и (OF = 1)) или ((SF = 1) и (OF = 0)), тогда AX меньше, чем BX
  + Если ((SF = 0) и (OF = 0)) или ((SF = 1) и (OF = 1)), тогда AX больше или равно BX

Таким образом, если флаги SF и OF не равны, то AX меньше, чем BX. Если эти флаги равны, тогда AX больше или равно BX

3) Флаг переполнения **OF** устанавливается, если при вычитании произойдет переполнение знакового бита.

* Флаг переноса **CF** устанавливается, если при вычитании AX — BX потребуется заимствование
  1. Выполнение работы

1. Из таблицы получен вариант набора функций, которые необходимо реализовать, приведенного в каталоге Задания.

/7-4\*i, при a>b

f3 = <

\-8-6\*i, при a<=b

/-3\*i+12, при a>b

f8 = <

\-6\*i+8, при a<=b

2. В табличном процессоре LibreOffice Calc создан 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. Проверено, что результаты совпадают для разных входных данных.

3. Программа разработана. Код написан с подробным комментированием.

4. Программа выполнена в пошаговом режиме под управлением отладчика с фиксацией значений используемых переменных. Результат можно посмотреть в таблице 1.

Таблица 1 - результат выполнения набора функций.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № теста | Тестируемый случай | Функции для данного случая | Данные | |
| входные | выходные |
| 1 | a > b  k = 0 | f1 : -4\*i + 7  f2 : -8 – 6\*i  f3 : res = max(7, |i2|) | a = 3  b = 1  k = 0  i = 2 | i1 = FFFF (-1)  i2 = FFEC (-20)  res = 0014 (20) |
| 2 | a > b  k != 0 | f1 : -4\*i + 7  f2 : -8 – 6\*i  f3 : res = |i1| + |i2| | a = 4  b = 2  k = 2  i = 1 | i1 = 0003  i2 = FFF2 (-14)  res = 000E (14) |
| 3 | a <= b  k = 0 | f1 : i1 = -6\*i + 8  f2 : i2 = 12 – 3\*i  f3 : res = max(7, |i2|) | a = 1  b = 1  k = 0  i = 2 | i1 = FFFC (-4)  i2 = 0006  res = 0007 |
| 4 | a <= b  k != 0 | f1 : i1 = -6\*i + 8  f2 : i2 = 12 – 3\*i  f3 : res = |i1| + |i2| | a = 1  b = 1  k = 2  i = 1 | i1 = 0002  i2 = 0009  res = 0009 |

* 1. Выводы

В ходе выполнения лабораторной работы были получены навыки разработки программы с заданными целочисленными значениями на языке программирования Ассемблер.

1. Приложение А  
   Исходный код программы

Название файла: lr3.asm

AStack SEGMENT STACK

DW 2 DUP(?)

AStack ENDS

DATA SEGMENT

a DW 3

b DW 1

k DW 0

i DW 2

i1 DW ?

i2 DW ?

res DW ?

DATA ENDS

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

mov ax,i ; ax = i

shl ax,1 ; ax = 2i

shl ax,1 ; ax = 4i

mov bx,ax ; bx = 4i

add bx,i ; bx = 5i

add bx,i ; bx = 6i

mov cx, a ; cx = a

cmp cx, b ; cx = b

jg AgB ; a > b

; a <= b

AleB:

; f1 : i1 = -6\*i + 8

mov i1, bx ; i1 = 6\*i

neg i1 ; i1 = -6\*i

add i1, 8 ; i1 = -6\*i + 8

; f2 : i2 = 12 - 3\*i

sub ax, i ; ax = 3\*i

neg ax ; ax = -3\*i

mov i2, ax ; i2 = -3\*i

add i2, 12 ; i2 = -3\*i + 12

jmp f3 ; перейти к f3

; a > b

AgB:

; f1 : -4\*i + 7

mov i1, 7 ; i1 = 7

sub i1, ax ; i1 = 7 -4\*i

; f2 : -8 - 6\*i

mov i2, -8 ; i2 = -8

sub i2, bx ; i2 = -8 - 6\*i

f3:

mov bx, i2 ; bx = i2

ABS\_I2: ; i2 = |i2|

neg bx

js ABS\_I2

mov res, bx ; res = |i2|

cmp k, 0

jl Kl0 ; k < 0, то перейти на Kl0

Kb0: ; k >= 0 : res = max(7, |i2|)

cmp bx, 7 ; |i2| > 7?

jg final ; если |i2| > 7 перейти к final

mov res, 7 ; res = 7

jmp final ; перейти к final

Kl0:

; k < 0 : res = |i1| + |i2|

mov ax, i1 ; ax = i1

ABS\_I1:

neg ax ; ax = |i1|

js ABS\_I1

add res, ax ; res = |i2| + |i1|

final:

mov ax, i1

mov bx, i2

mov cx, res

pop ds

mov ah,4ch

int 21h

Main ENDP

CODE ENDS

END Main