**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****.**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 3388 |  | Беннер В.А. |
| 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 общего назначения как:

Команды передачи данных.

1. Mov – присваивание

Двоичные арифметические команды.

1. Add - сложение
2. Sub - вычитание
3. Cmp – сравнение
4. Neg — смена знака

Команды побитового сдвига.

1. Sal - арифметический сдвиг влево

Команды передачи управления.

1. Jmp - безусловный переход
2. Int - вызов программного прерывания
3. Jge(jump greater equal) - выполняет короткий переход, если первый операнд больше второго операнда или равен ему при выполнении операции сравнения с помощью команды cmp
4. **Jg(jump greater) -** выполняет короткий переход, если первый операнд больше второго операнда при выполнении операции сравнения с помощью команды cmp.
5. Jne(jump negative equal) - выполняет короткий переход, если первый операнд не равен второму операнду при выполнении операции сравнения с помощью команды cmp.
   1. Выполнение работы

1. Из таблицы получен вариант набора функций, которые необходимо реализовать, формулы преобразованы в соответствии с замечанием 5.

/15-2\*i, при a>b

f1 = <

\3\*i+4, при a<=b

/4-6\*i, при a>b

f4 = <

\ 3\*i+4+2, 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. Результаты занесены в таблицу 1.

Таблица 1 - результат выполнения набора функций.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № теста | Тестируемый случай | Функции для данного случая | Данные | |
| входные | выходные |
| 1 | a > b  k = 0 | f1 = 15 – 2\*i  f2 = -(6\*i – 4)  f3 = abs(f1 + f2) | a = 7, b = 3  k = 0  i = 2 | f1 = 11 = 000B  f2 = -8 = FFF8  f3 = 3 = 0003 |
| 2 | a > b  k != 0 | f1 = 15 – 2\*i  f2 = -(6\*i – 4)  f3 = min(f1, f2) | a = 7, b = 3  k = 1  i = 3 | f1 = 9 = 0009  f2 = -14 = FFF2  f3 = -14 = FFF2 |
| 3 | a <= b  k = 0 | f1 = 3\*i + 4  f2 = 3\*(i + 2)  f3 = abs(f1 + f2) | a = 5, b = 5  k = 0  i = 2 | f1 = 10 = 000A  f2 = 12 = 000C  f3 = 22 = 0016 |
| 4 | a <= b  k != 0 | f1 = 3\*i + 4  f2 = 3\*(i + 2)  f3 = min(f1, f2) | a = 3, b = 5  k = 1  i = 3 | f1 = 13 = 000D  f2 = 15 = 000F  f3 = 13 = 000D |

* 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

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

AStack SEGMENT STACK

DW 32 DUP(0)

AStack ENDS

DATA SEGMENT

i DW 2

a DW 7

b DW 3

k DW 0

i1 DW ? ;f1

i2 DW ? ;f4

res DW ? ;f3

DATA ENDS

CODE SEGMENT

Main PROC FAR

mov AX,DATA

mov DS,AX

;Вычисление f1 и f2

mov ax,a ;заносим значение а в ах

mov cx,i ;заносим i в cx

sal cx,1 ;cx = i\*2

cmp ax,b ;Сравнение значений a и b

jg A1 ;если a>b то на A1

;если a<=b

add cx,i ;cx = 2\*i + i = 3\*i

add cx,4 ;cx = 3\*i + 4

mov i1,cx ;сохранение результата f1 в i1

add cx,4 ;cx = 3\*i + 4 + 2 = 3(i + 2)

mov i2,cx ;сохраняем рез-т f2 d i2

jmp A2 ;Пропускаем следующие шаги идем к А2

A1: ;если a>b

mov i1,15 ;i1 = 15

sub i1,cx ;i1 = i1 - cx = 15 - 2\*i, сохраняем результат в i1

mov ax,cx ;ax = 2\*i

sal cx,1 ;cx:=2\*i\*2

add cx,ax ;cx = 4\*i + 2\*i = 6\*i

mov i2,4 ;ax = 4

sub i2,cx ;ax = ax - cx = 4 - 6\*i, сохраняем результат в i2

;Вычисление f3

A2:

mov ax,k ;ax = k

cmp ax,0 ;сравниваем k и 0

JNe B1 ;если k не равно 0 то перйти на B1

;решение при к = 0

mov dx,i1 ;dx = i1

add dx,i2 ;dx = i1 + i2

ABS\_F3:

neg dx

js ABS\_F3 ;взятие модуля

mov res,dx ;res = dx

jmp B2

B1:

;если k не равно 0

mov ax,i1 ;ax = i1

cmp ax,i2

JGe C2 ;если i1 >= i2 то перейти на C2

mov res,ax ;res = ax

jmp B2

C2:

mov res,bx ;если i1 >= i2

B2:

mov ax, i1

mov bx, i2

mov cx, res

pop ds

mov ah, 4ch

int 21h

Main ENDP

CODE ENDS

END Main