**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 ветвления в программах на языке ассемблера.**

|  |  |
| --- | --- |
| Студент гр. 2384 | Валеева А.А. |
| Преподаватель | Морозов С.М. |

Санкт-Петербург

2023

**ЦЕЛЬ РАБОТЫ.**

Изучить ветвление в языке ассемблера, написать программу, вычисляющая значения функции по заданным параметрам.

**ЗАДАНИЕ.**

Вариант 4

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-

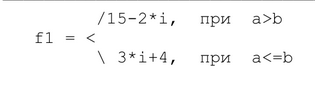
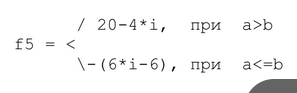
из табл.2 по цифрам шифра индивидуального задания (n1.n2.n3).

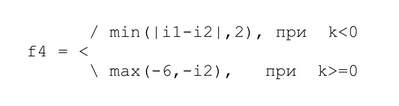
Значения a,b,i,k являются исходными данными, которые должны

выбираться студентом самостоятельно и задаваться в процессе ис-

полнения программы в режиме отладки. При этом следует рассмот-

реть все возможные комбинации параметров a,b и k, позволяющие

проверить различные маршруты выполнения программы.

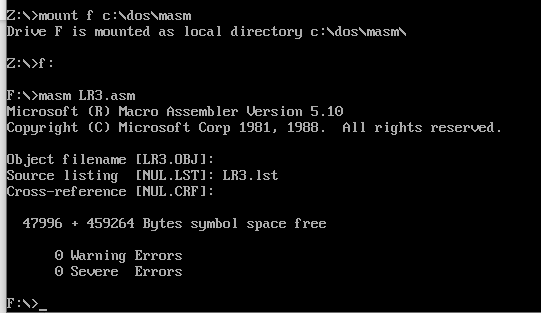


**ХОД РАБОТЫ.**

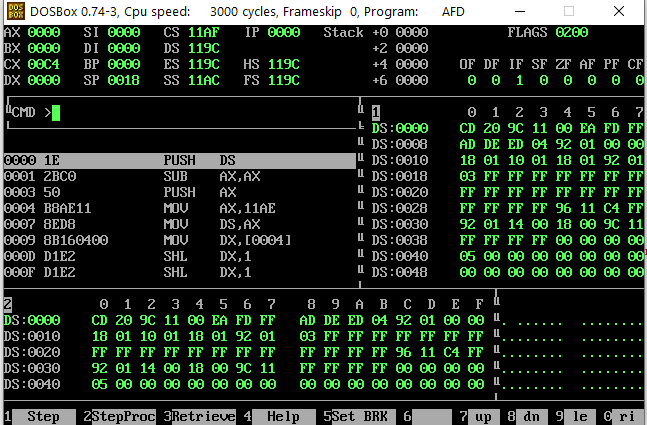
1. В файле «lr3.asm» написана программа на языке ассемблера.

* Под сегмент стека выделено 12 слов
* В сегменте данных создано 7 переменных, каждая из которых занимает 1 слово
* В коде для реализации ветвления использованы: «cmp arg1, arg2» для сравнения, jle, jge и jmp для перемещения IP между строками программы.

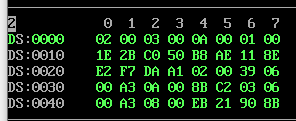
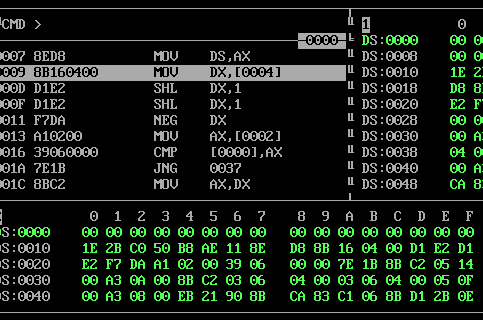
1. В целях минимизации количества операторов в программе i1 и i2 вычислялись следующим образом:
2. Перед основными вычислениями в регистр dx было сохранено значение (-4i)
3. Если (a > b), то сначала, используя регистр ax, вычисляем значение i2 = -4i – 20, которое было получено с помощью и прибавления 20 ( add ax, 20), результат сохраним в i2(mov i2, ax). Затем заново копируем (-4i) в ax Затем дважды прибавляем i к значению в ax, теперь ax = -2i; прибавляем 15 и сохраняем результат в i1.
4. Если (a <= b), то копируем (-4i) в сx, затем прибавляем 6, отнимаем дважды i и сохраняем полученный результат в i2. Также промежуточное значение -4i+6 было сохранено в dх, теперь для меньшего количества действий используем его. Изменяем знак с помощью neg, отнимаем i и прибавляем 10. Получаем в dx 3i+4, это значение сохраним в i1.
5. Для вычисления модуля используется ветвление. Сначала проверяем является ли число положительным. Если да, то обходим следующий шаг с помощью jge. Иначе с помощью neg получаем противоположное.
6. В Dosbox протранслирована программа с созданием файла листинга «lr3.lst». Для этого:
7. Смонтирован диск с помощью команды mount f c:\dos\masm, f:. С помощью команды «masm LR3.asm» исходный код был протранслирован с созданием файла листинга. Ошибок в процессе не выявлено.



1. С помощью «link LR3.obj» скомпилирован файл «lr3.exe» с созданием файла карты памяти.
2. Командой afd LR3.exe запускаем отладку



1. С помощью F8 вводим значения a, b, i, k.



**ТЕСТИРОВАНИЕ**

Для тестирована работы программ, она была выполнена различных входных данных. Результаты представлены в таблице 1.

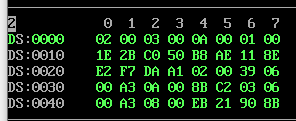
Таблица 1.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| a | b | i | k | res | Комментарий | i1 | i2 |
| 0002 | 0003 | 000А | 0001 | 36 (54 в 10 с/сч) | Верно | 22 (34 в 10с/сч) | FFCA(-54) |
| 0002 | 0003 | 000А | FFFF | 0002 | Верно | 22 | FFCA |
| 0005 | 0004 | 000F | 0003 | 28 (40 в 10 с/сч) | Верно | FFF1(-15 в 10 с/сч) | FFD8 (-40 в 10 с/сч) |
| 0005 | 0004 | 000F | FFFF | 0002 | Верно | FFF1 | FFD8 |

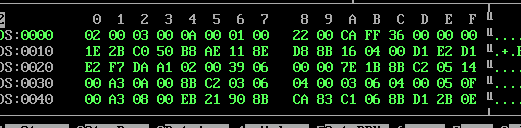
\*дополнительный код, отрицательное число

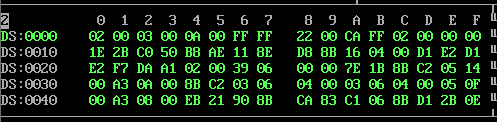
**Примеры тестов:**

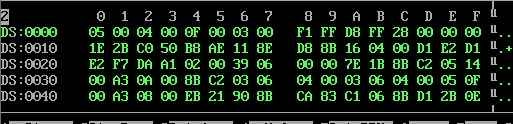
1. Ввожу данные в 16-с/сч:



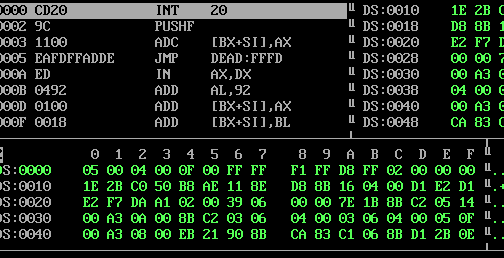
Получаю ответ в 16-с/сч:



1. Введенные и полученные данные в 16 с/сч:  
   
2. Введенные и полученные данные в 16с/сч:

=

1. Введенные значение и полученные данные в 16 с/сч:



**ВЫВОД.**

В ходе работы было изучено ветвление в языке ассемблера и написана программа, вычисляющая функции по заданным параметрам.

**ПРИЛОЖЕНИЕ А**

Название файла: LR3.asm

AStack SEGMENT STACK

DW 12 DUP(?)

AStack ENDS

DATA SEGMENT

a DW 0

b DW 0

i DW 0

k DW 0

i1 DW ?

i2 DW ?

res DW ?

DATA ENDS

CODE SEGMENT

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

Main PROC FAR

push DS ; подготовка сегмента стека и данных

sub AX, AX

push AX

mov AX, DATA

mov DS, AX

mov dx, i ;подготовка -4i для дальнейших вычислений

shl dx, 1 ; dx = 2 \* i = 2i

shl dx, 1 ; dx = 4i

neg dx ; dx = -4i

;проверка условия для ветвления

mov ax, b

cmp a, ax ; сравниваем а и б

jle F1LE; если а <= б то переходим к F1LE

; случай a > b

mov ax, dx ; ax = -4i

add ax, 20 ; ax = -4i + 20

mov i2, ax ; сохраняем i2 = -4i + 20

mov ax, dx ; ax = -4i

add ax, i ; ax = -4i + i = -3i

add ax, i; ax = -3i + i = -2i

add ax, 15 ; ax = 15 - 2i

mov i1, ax ; сохраняем i1 = 15 - 2i

jmp F1END ; переходим к концу первого оператора ветвления

F1LE: ; a <= b

mov cx, dx ; cx = -4i

add cx, 6 ; cx = -4i + 6

mov dx, cx ; dx = -4i + 6

sub cx, i; cx = -5i + 6

sub cx, i; cx = -6i + 6

mov i2, cx ; сохраняем i2 = -6i + 6

neg dx ; dx = -(-4i + 6) = 4i - 6

sub dx, i; dx = 3i - 6

add dx, 10; dx = 3i + 4

mov i1, dx ; сохраянем i1 = 3i + 4

F1END: ; конец первого ветвления

mov ax, i1 ; копируем i1 и i2 в РОН

mov bx, i2 ;?

F3:

cmp k, 0 ; если k < 0,

jl F3E ; то переходим к F3L (f3 lower)

neg bx

cmp bx, -6 ; если i2 >= -6

jge I1GE ; переходим к I1GE

mov res, -6 ; иначе res = -6

jmp FIN ; завершаем п

F3E:

sub ax, bx ; i1 - i2

F3L:

neg ax ; меняем знак i1-i2

js F3L ; если i1-i2 < 0, то переходим к F3L и меняем знак ещё раз (проверяет SF)

cmp ax, 2 ; if i1-i2>2 уходим (???)

jl I1G;

mov res, 2 ; иначе ответ 2

jmp FIN

I1G:

mov res, ax

jmp FIN

I1GE:

mov res, bx

FIN:

ret

Main ENDP

CODE ENDS

END Main