**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра математического обеспечения и применения ЭВМ**

**ОТЧЕТ**

**по лабораторной работе №3**

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

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

|  |  |
| --- | --- |
| Студент гр. 3344 | Тукалкин В.А. |
| Преподаватель | Фирсов М.А. |

Санкт-Петербург

2024

**Цель работы**

Изучение ветвления на языке ассемблер Intel X86.

**Задание**

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

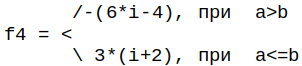
а) значения i1 = fn1(a,b,i) и i2 = fn2(a,b,i);

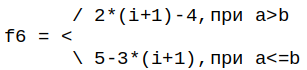
b) значения res= fn3(i1,i2,k),

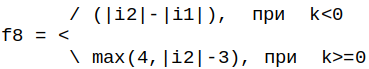
где вид функций fn1, fn2 определяется из табл. 1, а функции fn3 — из табл.2 по цифрам шифра индивидуального задания (n1.n2.n3).

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

Номер варианта: 20. Шифр задания: 4.6.8. Формулы для f1, f2, f3:







Замечания (требования):

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) Случаи, когда число-результат выходит за пределы одного слова, учитывать не требуется.

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

6) В коде должна быть выделена главная вычислительная часть, именно её нужно минимизировать. В главную часть не входят команды для инициализации сегментного регистра и для корректного завершения программы.

**Основные теоретические положения**

Для организации ветвления в ассемблере используются команды для безусловных и условных переходов:

* JMP (Jump): Безусловный переход.
* JNZ (Jump if Not Zero): Условный переход, если значение в регистре не равно нулю.
* JZ (Jump if Zero): Условный переход, если значение в регистре равно нулю.
* JE (Jump if Equal): Условный переход, если два значения равны.
* JNE (Jump if Not Equal): Условный переход, если два значения не равны.
* JG (Jump if Greater): Условный переход, если первое значение больше или равно второго.
* JGE (Jump if Greater or Equal): Условный переход, если первое значение больше второго.
* JL (Jump if Less): Условный переход, если первое значение меньше второго.
* JA (Jump if Above): Условный переход, если первое значение больше второго, не включая равенство.
* JB (Jump if Below): Условный переход, если первое значение меньше второго, не включая равенство.
* JBE (Jump if Below or Equal): Условный переход, если первое значение меньше или равно второму.

**Выполнение работы**

1. Преобразование формул, по которым будут выполняться вычисления, потому что нельзя использовать операцию умножения:

f4 = 4 – i<<1 – i<<2, при a>b

6 + i<<1 + i, при a<=b

f6 = 2 – i<<1 - i, при a<=b

i<<1 - 2, при a>b

f8 = |i2| - |i1|, при k<0

max(4, |i2| - 3), при k>=0

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.

3. Разработана программа, в коде выделены части, посвящённые вычислению итоговых значений (3 штуки) и ключевых промежуточных значений, написаны комментарии к каждой команде. Итоговое число команд составило: 37. Код представлен в приложении 1.

4. Программа была протестирована, с помощью 4 тестов, результаты представлены в таблице (приложение 2).

**Вывод**

В результате выполнения лабораторной работы было изучено ветвление в программе на языке ассемблер.

**Приложение**

Код программы lr3.asm:

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

AStack SEGMENT STACK

DW 32 DUP(0)

AStack ENDS

DATA SEGMENT

a DW 0

b DW 0

i DW 0

k DW -1

i1 DW 0

i2 DW 0

res DW 0

DATA ENDS

CODE SEGMENT

Main PROC FAR

mov ax,data ;загрузка сегментного

mov ds,ax ;регистра данных.

;Вычисление f4 и f6

mov ax,a ;присвоние ax a

mov cx,i ;присвоние cx i

cmp ax,b ;сравнение ax и b

shl cx,1 ;битовый сдвиг на 2, теперь в cx хранится i\*2

jg a\_greater ;переход (a>=b)

;f4, если a<=b

mov i1,6 ;присвоние i1 6

add cx,i ;прибавить в cx i, теперь в cx хранится i\*3

add i1,cx ;прибавить к i1 cx (6+i\*3 = 3\*(i+2))

;f6, если a<=b

mov i2,2 ;присвоние в i2 2

sub i2,cx ;вычитание из i2 cx (2-3\*i = 5-3\*(i+1))

jmp f8 ;переход к f8

a\_greater:

;f6, если a>b

mov i2,cx ;присвоение i2 cx (i\*2)

sub i2,2 ;i\*2-2 = 2\*(i+1)-4

;f4, если a>b

mov i1,4 ;присвоение i1 4

add cx,i ;сложение с cx i (cx=3\*i)

shl cx,1 ;битовый сдвиг на 2, теперь в cx хранится i\*6

sub i1,cx ;вычитание (4-i\*2-i\*4 = 4-6\*i)

f8:

;Вычисление f8

mov ax,k ;присвоение в ax k

mov cx,i1 ;присвоение в cx i1

mov dx,i2 ;присвоение в dx i2

pos\_i2:

neg dx ;изменение знака для взятия модуля

jl pos\_i2 ;перемещение если dx (i2) меньше 0

mov res,dx ;присвоение в res dx

cmp k,0 ;сравнение k и 0 (k<0)

jl k\_smaller ;перемещение

;k>=0

sub res,3 ;вычитание из res 3

cmp res,4 ;сравнение res и 4 (res<4)

jl res\_smaller;переход

jmp finish ;переход на завершение

res\_smaller:

mov res,4 ;присвоение в res 4

jmp finish ;переход к завершению

k\_smaller:

;k<0

pos\_i1:

neg cx ;изменение знака числа cx (i1)

jl pos\_i1 ;переход если cx (i1) отрицательное

sub res,cx ;вычитание cx (i1) из res

finish:

int 20h ;завершение программы

Main ENDP

CODE ENDS

END Main

Приложение 2

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | a | b | I | k | i1 (f4) | i2 (f6) | res (f8) | Условия |
| 1 | 0 | 0 | 0 | 0 | 6 | 2 | 4 | a<=b, k>=0 |
| 2 | 0 | 0 | 0 | -1 | 6 | 2 | -4 | a<=b, k<0 |
| 3 | 1 | 0 | 0 | 0 | 4 | -2 | 4 | a>b, k>=0 |
| 4 | 1 | 0 | 0 | -1 | 4 | -2 | -2 | a>b, k<0 |