**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: Представление и обработка целых чисел. Организация ветвящихся процессов.**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 1304 |  | Чернякова В.А. |
| Преподаватель |  | Кирьянчиков В.А. |

Санкт-Петербург

2022

## Цель работы.

Разработать на языке Ассемблера программу, которая по заданным целочисленным значениям параметров вычисляет значения функций.

## Задание.

Разработать на языке Ассемблера программу, которая по заданным целочисленным значениям параметров a, b, i, k вычисляет:

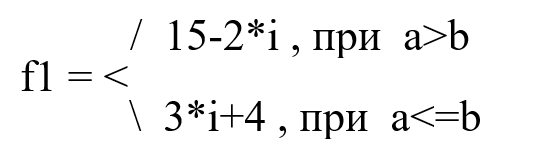
а) значения функций i1 = f1(a,b,i) и i2 = f2(a,b,i);

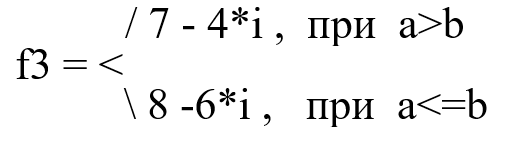
b) значения результирующей функции res = f3(i1, i2, k),

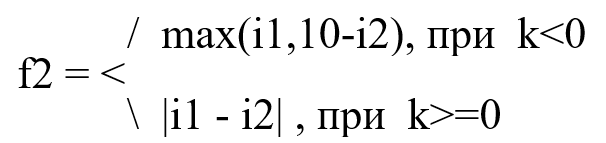
где вид функций f1 и f2 определяется из табл. 2, а функции f3 - из табл.3 по цифрам шифра индивидуального задания (n1, n2, n3), приведенным в табл.4.

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 всевозможные комбинации параметров a, b и k, позволяющие проверить различные маршруты выполнения программы, а также различные знаки параметров a и b.

Вариант 2







## Ход работы.

Были реализованы функции из каталога заданий, соответствующие 2 варианту. Реализованная программа протранслирована с различными тестовыми данными.

Для выполнения данного задания были использованы такие команды общего назначения как:

Команды передачи данных.

1) mov – присваивание

Двоичные арифметические команды.

2) add - сложение

3) sub - вычитание

4) cmp – сравнение

5) neg – смена знака

Команды побитового сдвига.

1) sal - арифметический сдвиг влево команды передачи управления.

2) jmp – команда безусловного перехода

3) Int - вызов программного прерывания

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

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

Преобразования:

1. Перед ветвлением в регистры были записаны значения: i, 2\*i, 3\*i.
2. Выражение 8-6\*i можно получить из 3\*i+4:

3\*i+4 => осуществляем побитовый сдвиг влево (команда shl) => 6\*i+8

6\*i+8 => меняем знаки (команда neg) => -6\*i-8

-6\*i-8 => добавляем 16 для получения необходимого выражения (команда add) =>8-6\*i

1. Выражение 7-4\*i можно получить из 15-2\*i:

15-2\*i => осуществляем побитовый сдвиг влево (команда shl) => 30-4\*i

30-4\*i => вычитаем 23 для получения необходимого выражения (команда sub) =>7-4\*i

Код написанной программы, отвечающей требованиям лабораторной работы согласно варианту, представлена в приложении. Программа протестирована с различными значениями, результат выполнения тестирования записан в таблице 1.

Таблица 1 – результат тестирования программы.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№ теста** | **Тестируемый**  **случай** | **Функции для данного случая** | **Данные** | |
| **Входные** | **Выходные** |
| 1 | a > b  k < 0 | f1 = 15-2\*i  f3 = 7-4\*i  f2 = max(i1,i2) | a = 56  b = 1  k = -15  i = 1 | f1 = 13  f3 = 3  f2 = 13 |
| 2 | a > b  k > 0 | f1 = 15-2\*i  f3 = 7-4\*i  f2 = abs(i1-i2) | a = -10  b = -46  k = 56  i = 2 | f1 = 11  f3 = -1  f2 = 12 |
| 3 | a > b  k = 0 | f1 = 15-2\*i  f3 = 7-4\*i  f2 = abs(i1-i2) | a = 23  b = -9  k = 0  i = 3 | f1 = 9  f3 = -5  f2 = 14 |
| 4 | a <= b  k < 0 | f1 = 3\*i+4  f3 = 8-6\*i  f2 = max(i1,i2) | a = 45  b = 345  k = -10  i = 1 | f1 = 7  f3 = 2  f2 = 7 |
| 5 | a <= b  k > 0 | f1 = 3\*i+4  f3 = 8-6\*i  f2 = abs(i1-i2) | a = -7  b = 34  k = 7  i = 3 | f1 = 13  f3 = -10  f2 = 23 |
| 6 | a <= b  k = 0 | f1 = 3\*i+4  f3 = 8-6\*i  f2 = abs(i1-i2) | a = 34  b = 34  k = 0  i = 2 | f1 = 10  f3 = -4  f2 = 14 |

## Вывод.

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

# ПРИЛОЖЕНИЕ.

**Lb3.ASM**

Исходный код программы.

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

AStack SEGMENT STACK

DW 12 DUP('!')

AStack ENDS

DATA SEGMENT

a DW 0

b DW 0

i DW 0

k DW 0

i1 DW 0 ;f1

i2 DW 0 ;f3

res DW 0 ;f2

DATA ENDS

CODE SEGMENT

Main PROC FAR

push DS

sub AX,AX

push AX

mov AX, DATA

mov DS, AX

mov CX,i ; cx = i

mov BX,CX

sal BX,1 ; bx = 2\*i

mov DX,BX

add DX,CX; dx = 3\*i ;

mov AX,a ; ax = a

cmp AX,b ; сравнение a и b

jg AGreaterB ; a>b 1) заменить на jа AGreaterB

;f1

add DX,4 ; dx = 3\*i+4

mov i1, DX

;f3

shl DX, 1; dx = 6\*i+8

neg DX ; dx = -6\*i-8

add DX,16 ; dx = 8-6\*i

mov i2,DX

jmp F2 ; выполнение безусловного перехода

AGreaterB:

;f1

neg BX ; bx = -2\*i

add BX,15; bx = 15-2\*i

mov i1,BX

;f3

shl BX, 1; bx = 30-4\*i

sub BX,23; bx = 7-4\*i

mov i2,BX

;f2

F2:

mov AX,k

cmp AX,0

jl KLESSZERO ; 2) заменить на jа KLESSZERO

mov AX,i1

sub AX,i2

L:

neg AX

js L

mov res,AX

jmp QUIT

KLESSZERO:

mov AX,i2

neg AX

add AX,10 ; ax = 10-i2

cmp i1,AX

jg i1\_GREATER\_i2 ; 3) заменить на jb i1\_GREATER\_i2

mov res,AX ; res = 10-i2

jmp QUIT ; 4) удалить команду

i1\_GREATER\_i2:

mov AX,i1

mov res,AX

QUIT:

int 20 ; завершение программы

Main ENDP

CODE ENDS

END Main

**Вопросы для защиты.**

*Внести указанные изменения (ошибки) 1-4 в код, рассматривая их по отдельности. Указать, почему и как они проявятся. Показать тесты, которые эти ошибки обнаружат и их результаты. Если приведенные в отчете тесты ошибку не находят, то внести дополнительный тест.*

## ЗАЩИТА.

1)Замена на ja AGreaterB

Команда *ja* осуществляет переход после команды *cmp op1, op2* если *op1* больше *op2*. При этом значение флагов *CF = 0* и *ZF = 0.*

Команда работает для чисел без знака. То есть корректное сравнение и дальнейшее срабатывание команды возможно только в том случае, если оба операнда одинакового знака.

*CF*(флаг переноса) = 0, то есть результат предыдущей операции умещается в приемник, переноса из старшего бита не происходит или заёма, как при вычитании.

*ZF*(флаг нуля) = 0, то есть результат предыдущей команды не 0.

ТЕСТ:

**a = 2310 = 001716**

**b = -910 = FFF716**

**i = 3**

**k = 0**

Аналитически *a > b* => должен произойти переход по метке *AGreaterB* и результат работы программы следующий: **i1 = 000916 = 916** и **i2 = FFFB16 = -510.**

Если запустить программу, то при выполнении операции сравнения *CMP AX, b* , где **AX = 001716=a** и **b = FFF716** , происходит сравнение чисел без учета знака, то есть *AX < b* из-за этого переход по метке *AGreaterB* не происходит, так как команда *ja*, следующая за *cmp*, сравнивает числа не учитывая их знак. В итоге **i1 = 000D16 = 1310 i2 = FFF616 = -1010.** Данный ответ неверный!

Если рассмотреть флаги, после выполнения *CMP AX, b СF = 1*, так как команда *CMP* осуществляется за счет вычитания, то есть происходит операция **001716-FFF716** для чего необходимо занимать у старшего бита => *CF* устанавливается в *1*. Из-за этого переход командой *ja* на соответствующую метку *AGreaterB* не происходит.

2)Замена на ja KLESSZERO

Команда *ja* осуществляет переход после команды *cmp op1, op2* если *op1* больше *op2.* При этом значение флагов *CF = 0* и *ZF = 0.*

Команда работает для чисел без знака. То есть корректное сравнение и дальнейшее срабатывание команды возможно только в том случае, если оба операнда одинакового знака.

*CF*(флаг переноса) = 0, то есть результат предыдущей операции умещается в приемник, переноса из старшего бита не происходит или заема, как при вычитании.

*ZF*(флаг нуля) = 0, то есть результат предыдущей команды не 0.

ТЕСТ:

**a = -1010 = FFF616**

**b = -4610 = FFD216**

**k = 5610 = 003816**

**i = 2**

**i1 = 000B16 = 1110**

**i2 = FFFF16 = -110**

При выполнении операции сравнения *cmp AX, 0* где **AX = 003816 = k** выполняется условие первый операнд *AX* больше второго *0*, что необходимо для срабатывания перехода с помощью *ja* на метку *KLESSZERO*, то есть выбранная команда не является корректной для данного сравнения. Работают команды, описанные в метке *KLESSZERO* **res** становится равным **000B16 = 1110**, то есть отрабатывает функция *max(i1, 10-i2)*, хотя аналитически и при корректной работе программы должно выполняться *abs(i1-i2)* и значение **res=000С16 = 1210.**

Если рассмотреть флаги, то *CF = 0*. Это объясняется тем, что при выполнении *cmp AX, 0* **(003816 - 000016**) не требуется переноса из старшего бита или занимания при вычитании. *ZF = 0*, так как результат вычитания не является нулевым результатом. Из-за этого благодаря *ja* происходит переход на метку *KLESSZERO*.

Если значение *k* будет отрицательным, то функция отработает корректно за счет того, что сравнение без знаковое, а любое отрицательное число без учета знака всегда больше *0* => срабатывание *ja*.

3)Замена на jb i1\_Greater\_i2

Команда *jb* осуществляет переход после команды *cmp op1, op2* если *op1<op2*. При этом значение флага *CF = 1*.

Работает для чисел без знака. То есть при сравнении знак не учитывается.

*CF*(флаг переноса) = *1*, то есть результат предыдущей операции не умещается в приемник, произошел перенос из старшего бита или заём, как при вычитании.

ТЕСТЫ:

**Первый тест.**

**a = 5610 = 003816**

**b = 110 = 000116**

**k = -1510 = FFF116**

**i = 1**

**i1 = 000D16 = 1310**

**i2 = 000316 = 310**

Аналитически и при правильной работе программы *k < 0* из этого следует, что **res** будет строится на основе выбора максимума между *i1* и *10-i2*. Итоговое значение **res** должно быть равно **res= 000D16=1310**, так как *i1 > 10-i2.*

При выполнении команды *cmp i1,AX*(то есть сравниваются **000D16=1310** и **10-i2=000A16-000316=000716=710**) так как первый операнд не меньше второго переход с помощью *jb* по метке *i1\_GREATER\_i2* не происходит. То есть выбранная команда не является корректной для данного сравнения. Из-за этого считается, что *10-i2* больше, чем *i1*, хотя это не так и **res = 000A16-000316 = 000716 = 710**. Данный ответ неверный!

Если рассматривать относительно флагов, то *CF* после выполнения *cmp i1, AX* становится равным *0*. Это происходит из-за того, что при вычитании из **i1=000D16=1310** значения **10-i2=000A16-000316=000716=710** результат умещается в приемник и заем бита не требуется. Поэтому переход по *ja* на необходимую метку не происходит.

**Второй тест.**

**a = 4510 = 002D16**

**b = 34510 = 015916**

**k = -1010 = FFF616**

**i = 1**

**i1 = 000716 = 710**

**i2 = 000216 = 210**

Аналитически и при правильной работе программы *k < 0* из этого следует, что **res** будет строится на основе выбора максимума между *i1* и *10-i2*. Итоговое значение **res** должно быть равно **res= 000816=810**, так как *i1 < 10-i2.*

При выполнении команды *cmp i1, AX*(то есть сравниваются **000716 = 710** и **10-i2=000A16-000216=000816=810**) так как первый операнд меньше второго переход с помощью *jb* по метке *i1\_GREATER\_i2* происходит, то есть выбранная команда не является корректной для данного сравнения. Из-за этого считается, что *i1* больше, чем *10-i2*, хотя это не так и **res = i1 = 000716 = 710**. Данный ответ неверный!

Если рассматривать относительно флагов, то *CF* после выполнения c*mp i1, AX* становится равным *1*. Это происходит из-за того, что при вычитании из **i1=000716=710** значения **10-i2=000A16-000216=000816=810** приходится занимать у старшего бита. Поэтому переход по *ja* на метку происходит, но в данном случае этот переход не является необходимым.

4)Удаление команды jmp QUIT

Команда *jmp* – команда безусловного перехода. Выполняет переход всегда и в любом случае туда, куда этот самый переход указан.

При удалении команды *jmp QUIT* переход к метке *QUIT,* отвечающий за завершение работы программы, не происходит.

ТЕСТ:

**a = 4510 = 002D16**

**b = 34510 = 015916**

**k = -1010 = FFF616**

**i = 1**

**i1 = 000716 = 710**

**i2 = 000216 = 210**

**10 – i2 = 001016 – 000216 = 000816 = 810**

Аналитически и при корректной работе программы **res = 000816 = 810.** То есть был выбран максимум между *(i1, 10-i2).* При выбранных тестовых значениях максимум – *10-i2*.

После выполнения сравнения *cmp i1,AX* сравнение **000716** и **000816** переход по *jg* на метку *i1\_GREATER\_i2* не происходит, так как *i1<AX*. В r**es** записывается значение, хранимое в *AX,* а именно **000816 = 810**. Но из-за того, что была удалена команда *jmp QUIT* переход на метку не осуществляется, программа не завершает работу и продолжает дальше выполнять команды. Начинают работать команды, представленные в метке *i1\_GREATER\_i2*, так как ничто не препятствует их выполнению. Происходит *mov AX,i1* и *mov res,AX*. Таким образом **res=000716=710**. Данный ответ является неправильным!