

Учреждение образования  
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»  
Кафедра информатики

Отчет по лабораторной работе №3

Работа со знаковыми числами

Выполнил: студент гр. 853501  
Астрашаб В.В.

Руководитель: ст. преподаватель  
Шиманский В.В.

Минск 2019

## СОДЕРЖАНИЕ

1. Введение
2. Постановка задачи
3. Программная реализация
4. Выводы

Литература

Приложение 1. Текст программы

## 1. Введение

Целью данной работы является изучить следующий материал:

- 1) Знаковые и беззнаковые числа.
- 2) Что такое дополнительный код. Представление чисел в дополнительном коде.
- 3) Команда NEG.
- 4) Команды IMUL и IDIV и их отличие от команд MUL и DIV.
- 5) Команды CBW и CWD и их использование.
- 6) Алгоритмы ввода и вывода знаковых десятичных чисел.

## 2. Постановка задачи

### 2.1. Текст задания

Если  $a^2 \leq b^3$  то

Если  $c * d = a / b$  то

Результат =  $a \text{ XOR } b$

Иначе

Результат = найти 2 наибольших среди  $a, b, c, d$  и перемножить

Иначе

Результат =  $a * b + c/d$

### 2.2. Условие задания

Модифицировать программы второй лабораторной работы таким образом, чтобы можно было вводить и выводить знаковые числа.

### 3. Программная реализация

3.1 Для реализации я использовал программу DOSBox на ОС Windows 10. DOSBox содержит компоненты TurboAssembler, TurboLink и TurboDebugger для компиляции, линковки и отладки программы.

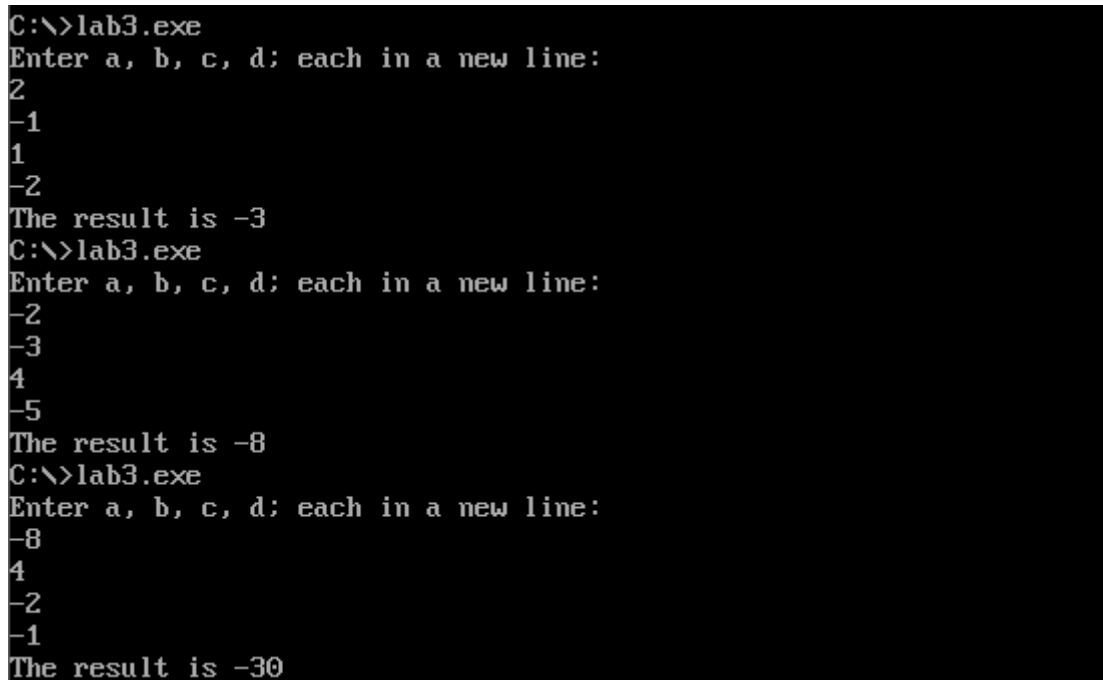
3.2 Значения переменных устанавливаются при вводе пользователем с клавиатуры. Программа разбита при помощи меток на несколько логических частей, каждая из которых выполняет определенную ветку условия.

3.3 Ввод значений знаковых переменных с клавиатуры происходит с помощью подпрограммы ReadSigned. Ответ со знаком выводится на экран с помощью подпрограммы WriteSigned.

#### 3.4 Примеры:

##### 3.4.1 Отладка веток

Тест с отрицательными числами для всех веток задания



```
C:\>lab3.exe
Enter a, b, c, d; each in a new line:
2
-1
1
-2
The result is -3
C:\>lab3.exe
Enter a, b, c, d; each in a new line:
-2
-3
4
-5
The result is -8
C:\>lab3.exe
Enter a, b, c, d; each in a new line:
-8
4
-2
-1
The result is -30
```

Результат: Все ветки работают корректно, ответы верные, ввод и вывод отрицательных чисел работает корректно.

### 3.4.2 Проверка на ввод недопустимых данных

Попытки ввода любых символов, отличных от цифр десятичной системы счисления (кроме минуса в самом начале числа), а также попытки ввода значений чисел, выходящих за диапазон (-32768...32767) (диапазон знаковой переменной размером слово).

```
Enter a, b, c, d; each in a new line:
12a
The input is invalid. Please try again
-456b
The input is invalid. Please try again
--
The input is invalid. Please try again
-2345-
The input is invalid. Please try again
34-
The input is invalid. Please try again
c
The input is invalid. Please try again
s
```

Результат: Программа запрещает ввод недопустимых символов.

```
Enter a, b, c, d; each in a new line:
32768
The input is invalid. Please try again
123456
The input is invalid. Please try again
-32769
The input is invalid. Please try again
-39234
The input is invalid. Please try again
```

Результат: Не допускается ввод чисел, не входящих в диапазон допустимых значений.

```
Enter a, b, c, d; each in a new line:
1
-1234
5
6
Error. The result is out of bounds.

C:\>lab3.exe
Enter a, b, c, d; each in a new line:
1
2
-2345
5432
Error. The result is out of bounds.
```

Результат: не допускается выход за границы типа знакового byte во время выполнения программы.

```
Enter a, b, c, d; each in a new line:
-1
0
2
3
Error. Division by zero.

C:\>lab3.exe
Enter a, b, c, d; each in a new line:
-1
1
2
0
Error. Division by zero.
```

Результат: не допускается деление на 0 во время выполнения программы.

В случае некорректного ввода производится повторный ввод значения соответствующей переменной.

#### 4. Выводы

На практике я изучил и опробовал, в соответствии с поставленной задачей: представление знаковых и беззнаковых чисел в регистрах, дополнительный код, команду NEG, команды IMUL и IDIV и их отличие от команд MUL и DIV, команды CBW и CWD и их использование, алгоритмы ввода и вывода знаковых десятичных чисел.

В процессе выполнения лабораторной работы я использовал DOSBox и его компоненты TurboLink и TurboAssembler.

Для ввода и вывода знаковых чисел я создал специальные подпрограммы, в которых использовал стек в качестве структуры для временного хранения данных. Данные подпрограммы сохраняют регистры в ходе своей работы.

Для примеров я сделал скриншоты из программы DOSBox.



## Литература

1. Юров В.И. – «Assembler. Учебник для вузов. 2-ое издание, 2003 год».
2. Юров В.И. – «Assembler. Практикум. 2-ое издание, 2006 год».
3. Калашников О.А. – «Ассемблер - это просто. 2-ое издание, 2011 год».

## Приложение 1. Текст программы

```
model small
.stack 100h
.data
a dw ?
b dw ?
c dw ?
d dw ?
ae dw ?
be dw ?
ce dw ?
de dw ?
b_cube dw ?
Digits dw 0
Incorrect db 'The input is invalid. Please try again$'
Result db 'The result is $'
Prompt db 'Enter a, b, c, d; each in a new line:$'
overflowString db 'Error. The result is out of
bounds.$'
DivisionByZero db 'Error. Division by zero.$'
.code

Interrupt proc
    mov ah, 4Ch
    int 21h
    ret
Interrupt endp

HandleOverflow proc
; Overflow Error message
```

```
    lea dx, OverflowString
    mov ah, 09h
    int 21h
    mov dl, 0Ah
    mov ah, 02h
    int 21h
    call Interrupt
HandleOverflow endp
```

```
HandleDivisionByZero proc
; Division by zero Error message
    lea dx, DivisionByZero
    mov ah, 09h
    int 21h
    mov dl, 0Ah
    mov ah, 02h
    int 21h
    call Interrupt
    ret
HandleDivisionByZero endp
```

```
Read proc
; Read a number to bx
Retry:
; Set bx to 0
    xor bx, bx
```

```
CycleRead:
; Read a character
```

```
mov ah, 01h
int 21h
```

```
; Check for Enter
```

```
cmp al, 0Dh
jz EndCycle
```

```
; Check for bad characters
```

```
cmp al, '0'
jc BadChar
cmp al, 3Ah
jnc BadChar
```

```
; Convert character to the number and add to the result
```

```
xor cx, cx
mov cl, al
sub cx, '0'
mov ax, bx
mov dx, 10
mul dx
jc BadChar
add ax, cx
jc BadChar
mov bx, ax
jmp CycleRead
```

```
; Write a warning about incorrect input
```

```
BadChar:
```

```
mov dl, 0dh
mov ah, 02h
int 21h
```

```
mov dl, 0Ah
mov ah, 02h
int 21h
lea dx, Incorrect
mov ah, 09h
int 21h
mov dl, 0Ah
mov ah, 02h
int 21h
jmp Retry
```

EndCycle:

```
ret
```

Read endp

ReadSigned proc

```
mov ax, ax
```

```
mov cx, cx
```

```
mov dx, dx
```

```
; Read a signed number to bx
```

RetrySigned:

```
; Set bx to 0
```

```
xor bx, bx
```

```
xor si, si
```

CycleReadSigned:

```
; Read a character
```

```
mov ah, 01h
```

```
int 21h
```

```
; Check for Enter
cmp al, 0Dh
jz EndCycleSigned
```

```
; If symbol is -, si == 0 and bx == 0, then si := 1
cmp al, '-'
jnz SkipSigned
cmp si, 0
jnz SkipSigned
cmp bx, 0
jnz SkipSigned
mov si, 1
jmp CycleReadSigned
```

SkipSigned:

```
; Check for bad characters
cmp al, '0'
jc BadCharSigned
cmp al, 3Ah
jnc BadCharSigned
```

```
; Convert character to the number and add to the result
xor cx, cx
mov cl, al
sub cx, '0'
mov ax, bx
mov dx, 10
imul dx
jo BadCharSigned
add ax, cx
jo BadCharSigned
```

```
mov bx, ax
jmp CycleReadSigned
```

```
; Write a warning about incorrect input
```

```
BadCharSigned:
```

```
mov dl, 0dh
mov ah, 02h
int 21h
mov dl, 0Ah
mov ah, 02h
int 21h
lea dx, Incorrect
mov ah, 09h
int 21h
mov dl, 0Ah
mov ah, 02h
int 21h
jmp RetrySigned
```

```
EndCycleSigned:
```

```
; If si == 1, bx := -bx
```

```
cmp si, 1
jnz ReturnSigned
neg bx
```

```
ReturnSigned:
```

```
mov ax, ae
mov cx, ce
mov dx, de
ret
```

```
ReadSigned endp
```

```

write proc
; Write a number from ax to console
; Fill the stack with digits
PushCycle:
    mov bx, 10
    xor dx, dx
    div bx
    push dx
    inc Digits
    cmp al, 0
    jnz PushCycle

    mov cx, Digits

; Print digits from the stack to console
PopCycle:
    pop dx
    add dl, '0'
    mov ah, 02h
    int 21h
    loop PopCycle

    ret
write endp

```

```

writeSigned proc
; Write a signed number from ax to console
    mov ae, ax

```



```
mov be, bx
mov ce, cx
mov de, dx
; If the number is negative, print - and take the
absolute value
test ax, 10000000000000000b
jz Positive
mov bx, ax
xor dx, dx
mov dl, '-'
mov ah, 02h
int 21h
mov ax, bx
neg ax
```

Positive:

```
call write
mov ax, ae
mov bx, be
mov cx, ce
mov dx, de
ret
writeSigned endp
```

PrintPrompt proc

```
lea dx, Prompt
mov ah, 09h
int 21h
mov dl, 0Ah
mov ah, 02h
int 21h
```

```
    ret
PrintPrompt endp
```

```
PrintResult proc
; Print ax after Result contents
    mov bx, ax
    lea dx, Result
    mov ah, 09h
    int 21h
    mov ax, bx

    call writeSigned
    ret
PrintResult endp
```

```
Read_abcd proc
    call ReadSigned
    mov a, bx

    call ReadSigned
    mov b, bx

    call ReadSigned
    mov c, bx

    call ReadSigned
    mov d, bx

    ret
```

```
Read_abcd endp
```

```
Power proc  
; ax := bx^cx  
mov ax, 1  
cmp cx, 0  
jz PowerEnd
```

```
PowerLoop:  
imul bx  
jo PowerOverflow  
loop PowerLoop
```

```
jmp PowerEnd
```

```
PowerOverflow:  
call HandleOverflow
```

```
PowerEnd:  
ret  
Power endp
```

```
Swap_a_b proc  
mov ax, a  
mov bx, b  
mov a, bx  
mov b, ax  
ret  
Swap_a_b endp
```

```
Swap_b_c proc
```

```
    mov ax, b
```

```
    mov bx, c
```

```
    mov b, bx
```

```
    mov c, ax
```

```
    ret
```

```
Swap_b_c endp
```

```
Swap_c_d proc
```

```
    mov ax, c
```

```
    mov bx, d
```

```
    mov c, bx
```

```
    mov d, ax
```

```
    ret
```

```
Swap_c_d endp
```

```
Sort_abcd proc
```

```
; choose two biggest from a, b, c, d (bubble sort until  
a and b contain biggest numbers)
```

```
    mov ax, c
```

```
    cmp ax, d
```

```
    js Swap_1
```

```
Return_1:
```

```
    mov ax, b
```

```
    cmp ax, c
```

```
    js Swap_2
```

```
Return_2:
```

```
    mov ax, a
```

```
    cmp ax, b
    js Swap_3
Return_3:
    mov ax, c
    cmp ax, d
    js Swap_4
Return_4:
    mov ax, b
    cmp ax, c
    js Swap_5
Return_5:
    jmp End_sort
```

```
Swap_1:
    call Swap_c_d
    jmp Return_1
```

```
Swap_2:
    call Swap_b_c
    jmp Return_2
```

```
Swap_3:
    call Swap_a_b
    jmp Return_3
```

```
Swap_4:
    call Swap_c_d
    jmp Return_4
```

```
Swap_5:
    call Swap_b_c
```

```
    jmp Return_5
```

```
End_sort:
```

```
    ret
```

```
Sort_abcd endp
```

```
Start:
```

```
    mov ax, @data
```

```
    mov ds, ax
```

```
; Read the variables
```

```
    call PrintPrompt
```

```
    call Read_abcd
```

```
; calculate  $b^3$  and move to b_cube
```

```
    mov bx, b
```

```
    mov cx, 3
```

```
    call Power
```

```
    mov b_cube, ax
```

```
; calculate  $a^2$  and move to ax
```

```
    mov bx, a
```

```
    mov cx, 2
```

```
    call Power
```

```
; if  $a^2 \neq b^3$  jump to Cond2
```

```
    cmp ax, b_cube
```

```
    jnz Cond2
```

```
; else res =  $a*b + c/d$ 
```

; calculate a\*b and save to bx

mov ax, a

imul b

jo overflow

mov bx, ax

; calculate c/d and save to ax

mov ax, c

mov dx, d

cmp dx, 0

jz DivByZero

cwd

idiv d

; calculate a\*b + c/d and save to ax, finish the  
program

add ax, bx

jo overflow

jmp Final

Cond2:

; calculate c\*d and save to bx

mov ax, c

imul d

jo overflow

mov bx, ax

; calculate a/b and save to ax

mov ax, a

mov dx, b

cmp dx, 0

```
jz DivByZero
```

```
cwd
```

```
idiv b
```

```
; if c*d == a/b jump to Equals
```

```
cmp ax, bx
```

```
jz Equals
```

```
; else choose two biggest from a, b, c, d and multiply
```

```
call Sort_abcd
```

```
; multiply two greater numbers and save to ax, finish  
the program
```

```
mov ax, a
```

```
imul b
```

```
jo Overflow
```

```
jmp Final
```

Equals:

```
; calculate a xor b and move to ax, finish the program
```

```
mov ax, a
```

```
xor ax, b
```

```
jmp Final
```

Final:

```
call PrintResult
```

```
call Interrupt
```

Overflow:

```
call HandleOverflow
```



DivByZero:

call HandleDivisionByZero

end Start