

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

Отчет по лабораторной работе №2

Вычисления с вводом данных и выводом результата

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) Что такое сегмент, сегменты данных, кода и стека. Сегментные регистры. Что такое смещение. Команды LEA и OFFSET.
- 2) Что такое стек и как он используется. Команды PUSH и POP.
- 3) Что такое подпрограмма, использование подпрограмм в ассемблере. Команды CALL и RET.
- 4) Понятие дальнего перехода и дальнего вызова подпрограммы.
- 5) Команда INT.
- 6) Функции 21h прерывания для ввода и вывода символа и строки.
- 7) Организация циклов в Ассемблере. Команда LOOP.
- 8) Алгоритмы ввода двоичного, десятичного и шестнадцатеричного чисел.
- 9) Алгоритмы вывода двоичного, десятичного и шестнадцатеричного чисел.

2. Постановка задачи

2.1. Текст задания

Если $a^2 \neq b^3$ то

Если $c * d = a / b$ то

Результат = $a \text{ XOR } b$

Иначе

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

Иначе

Результат = $a * b + c/d$

2.2. Условие задания

Для программы, разработанной в лабораторной работе 1, написать подпрограммы для ввода и вывода десятичных чисел.

В главной программе необходимо ввести числа при помощи этих подпрограмм, выполнить расчеты согласно варианту задания по лабораторной работе 1 и вывести результат на экран.

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.

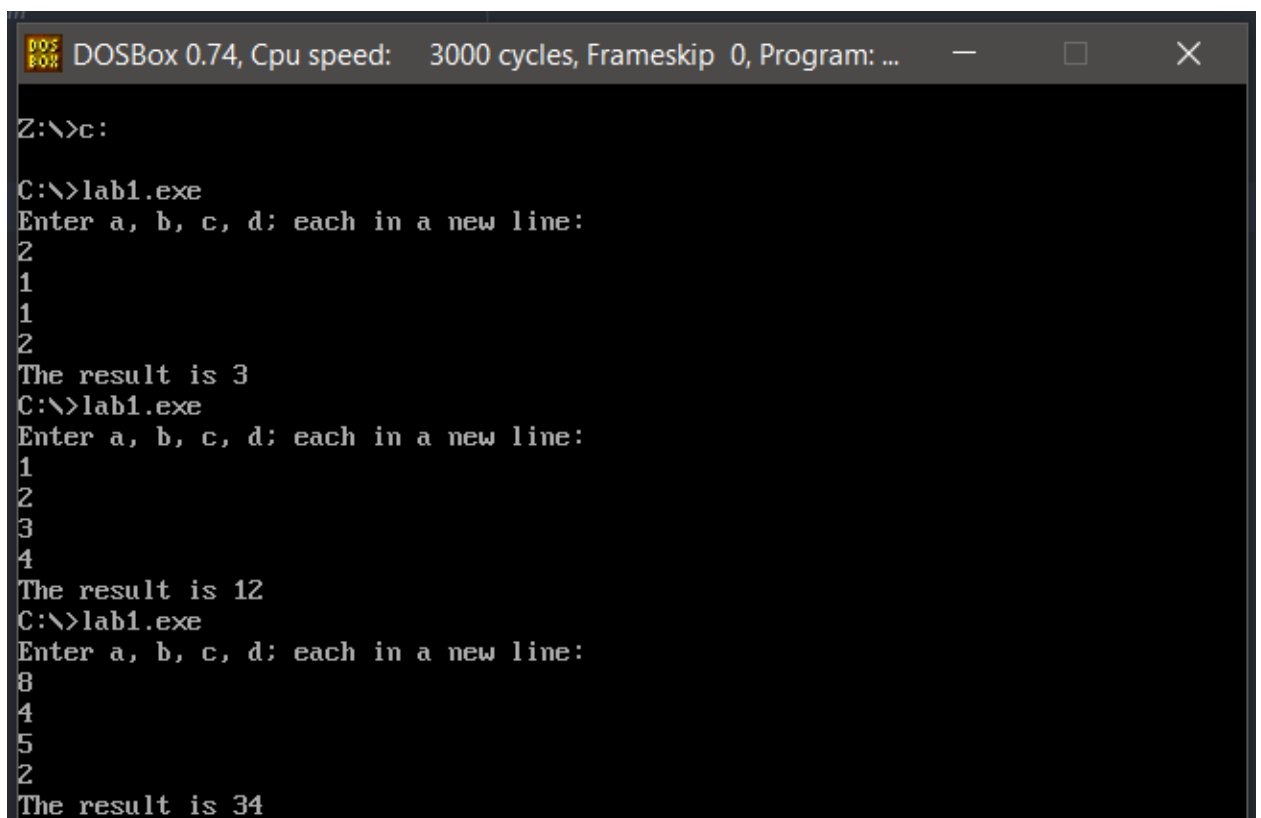
3.2 Значения переменных устанавливаются при вводе пользователем с клавиатуры. Программа разбита при помощи меток на несколько логических частей, каждая из которых выполняет определенную ветку условия.

3.3 Ввод значений переменных с клавиатуры происходит с помощью подпрограммы Read. Ответ выводится на экран с помощью подпрограммы Write.

3.4 Примеры:

3.4.1 Отладка веток

Тест для всех веток задания



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
Z:\>c:
C:\>lab1.exe
Enter a, b, c, d; each in a new line:
2
1
1
2
The result is 3
C:\>lab1.exe
Enter a, b, c, d; each in a new line:
1
2
3
4
The result is 12
C:\>lab1.exe
Enter a, b, c, d; each in a new line:
8
4
5
2
The result is 34
```

Результат: Все ветки работают корректно, ответы верные.

3.4.2 Проверка на ввод недопустимых данных

Попытки ввода любых символов, отличных от цифр десятичной системы счисления, а также попытки ввода значений числа больше, чем 65535 (максимальное значение переменной размером слово).

```
Enter a, b, c, d: each in a new line:
a
The input is invalid. Please try again
b
The input is invalid. Please try again
c
The input is invalid. Please try again
&
The input is invalid. Please try again
)
The input is invalid. Please try again
!
The input is invalid. Please try again
α
The input is invalid. Please try again
P
The input is invalid. Please try again
S
The input is invalid. Please try again
```

Результат: Программа запрещает ввод недопустимых символов.

```
87293
The input is invalid. Please try again
84384
The input is invalid. Please try again
77892
The input is invalid. Please try again
555555
The input is invalid. Please try again
98765
The input is invalid. Please try again
65536
The input is invalid. Please try again
```

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

В случае некорректного ввода производится повторный ввод значения соответствующей переменной.

4. Выводы

На практике я изучил и опробовал, в соответствии с поставленной задачей: сегменты, команды LEA и OFFSET, работу со стеком, команды PUSH и POP, использование подпрограмм, команды CALL и RET, понятие дальнего перехода и вызова подпрограммы, команду INT, функции прерывания 21h, организацию циклов и команду LOOP, алгоритмы ввода и вывода в разных системах счисления.

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

Для ввода и вывода данных я создал специальные подпрограммы, в которых использовал стек в качестве структуры для временного хранения данных.

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 ?
Digits dw 0
Incorrect db 'The input is invalid. Please try again$'
Result db 'The result is $'
Prompt db 'Enter a, b, c, d; each in a new line:$'

.code

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, 40h

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

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
```

```
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
```

Start:

```
mov ax, @data
mov ds, ax
```

; Read the variables

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

```
call Read
mov a, bx
```

```
call Read
mov b, bx
```

```
call Read
mov c, bx
```

```
call Read
mov d, bx
```

; calculate b^3 and move to bx

```
mov ax, b
mul b
mul b
mov bx, ax
```

; calculate a^2 and move to ax

```

mov ax, a
mul ax

; if  $a^2 \neq b^3$  jump to Cond2
cmp ax, bx
jnz Cond2

; else  $res = a*b + c/d$ 
; calculate  $a*b$  and save to bx
mov ax, a
mul b
mov bx, ax

; calculate  $c/d$  and save to ax
xor dx, dx
mov ax, c
div d

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

jmp Final

Cond2:
; calculate  $c*d$  and save to bx
mov ax, c
mul d
mov bx, ax

; calculate  $a/b$  and save to ax
mov ax, a

```

```

xor dx, dx
div 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
; choose two biggest from a, b, c, d (bubble sort until
a and b contain biggest numbers)
mov ax, c
cmp ax, d
jc Swap_1
Return_1:
mov ax, b
cmp ax, c
jc Swap_2
Return_2:
mov ax, a
cmp ax, b
jc Swap_3
Return_3:
mov ax, c
cmp ax, d
jc Swap_4
Return_4:
mov ax, b
cmp ax, c
jc 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:

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

mov ax, a

mul b

jmp Final

Equals:

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

mov ax, a

xor ax, b


```
jmp Final
```

```
Final:
```

```
; write the result
```

```
mov bx, ax
```

```
lea dx, Result
```

```
mov ah, 09h
```

```
int 21h
```

```
mov ax, bx
```

```
call write
```

```
mov ah, 4Ch
```

```
int 21h
```

```
end Start
```