**Лабораторная работа №3**

**ВВОД-ВЫВОД НА АССЕМБЛЕРЕ**

**Студент гр. ИКПИ-22**

**Нестеренко Н.С.**

**Постановка задачи**

**ЦЕЛЬ РАБОТЫ**:

Реализовать целочисленные вычисления, сделанные в лабораторной работе №1 или №2, полностью используя язык Ассемблера для организации корректного ввода-вывода информации:

• исходные данные должны вводиться с проверкой правильности вводимых символов;

• входные данные и результат должны быть проверены на область допустимых значений;

• при наличии ошибки должно быть выдано соответствующее сообщение.

**ОГРАНИЧЕНИЕ:**

• передачу параметров между модулями (функциями) реализовать через глобальные переменные

• нельзя использовать Cи-функции ввода/вывода (scanf/printf и т.п.)

**ПОРЯДОК РАБОТЫ:**

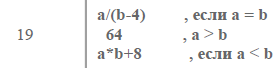
• внимательно изучить свой вариант ввода-вывода целочисленной информации применительно к решаемой задаче;

• написать на языке Ассемблера программу ввода исходных данных (с контролем допустимого диапазона), и вывода полученного результата;

• встроить вызов уже разработанного и отлаженного ранее в лабораторной работе ASM-модуля в программу;

• произвести тестовые проверки, отметить нормальные и аномальные результаты, сделать анализ результатов.

**ВАРИАНТ ЗАДАНИЯ:**



**Разработка алгоритма**

1. Входные данные:

(int16\_asm)

signed short (word): a, b

(uint16\_asm)

unsigned short (word): a, b

Ввод с клавиатуры в окне терминала (реализация на asm)

1. Выходные данные:

(int16\_asm)

signed int (double word): res

(uint16\_asm)

signed int (double word): res

Вывод в окно терминала результатов вычислений на ассемблере

**Таблица идентификаторов**

|  |  |  |  |
| --- | --- | --- | --- |
| **N** | **Обозначение в задаче** | **Размер** | **Назначение** |
| **1** | **a (int16\_asm)** | signed short (word)  16 бит | **Входные данные** |
| **2** | **b (int16\_asm)** |
| **3** | **a (uint16\_asm)** | unsigned short (word)  16 бит |
| **4** | **b (uint16\_asm)** |
| **5** | **res (int16\_asm)** | signed int (double word)  32 бит | **Выходные данные** |
| **6** | **res (uint16\_asm)** |

**Контрольный расчет**

Результаты вычислений приведены ниже в таблице вычислений.

|  |  |  |  |
| --- | --- | --- | --- |
| **Назначение набора данных**  **(int)** | **Набор данных**  **(a, b)** | **Результаты вычислений**  **(res)** | |
| **си** | **asm** |
| **Контрольный набор** | -34 56 | -1896 | -1896 |
| -32000 0 | 8 | 8 |
| 444 444 | 1 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Назначение набора данных**  **(uint)** | **Набор данных**  **(a, b)** | **Результаты вычислений**  **(res)** | |
| **си** | **asm** |
| **Контрольный набор** | 34 56 | 1912 | 1912 |
| 123 456 | 56096 | 56096 |
| 1 65000 | 65008 | 65008 |

**Программа на языке asm (nasm)**

section .data

    a *dw* 0

    b *dw* 0

    res *dq* 0

    ma *db* 0

    mb *db* 0

    fres *dq* 0

    i *db* 0

    msg *db* '1 - int (-32768 to 32767), 2 - uint(0 to 65535)', 0xa, 'input mode, a, b: '

    lenmsg equ $-msg

    dbz *db* 'exception: divizion by zero', 0xa

    lendbz equ $-dbz

    errmsg *db* 'exception: wrong input', 0xa

    lenerrmsg equ $-errmsg

    resl *db* 'result: '

    lenresl equ $-resl

    s *db* 8 dup(0)

section .text

    global main

    global input\_int

    global input\_uint

    global int16\_asm

    global uint16\_asm

    global exit

    global err\_exit

    ;вариант 19

    ;(a == b) : a/(b-4)

    ;(a > b) : 64

    ;(a < b) : a\*b+8

main:

    mov rbp, rsp

    call clear

    ; вывести сообщение

    mov edx, lenmsg ; длина строки

    mov ecx, msg ; строка

    mov ebx, 1 ; stdout

    mov eax, 4 ; sys\_write()

    int 0x80   ; call kernel

    ; прочитать ввод

    mov eax, 3 ; sys\_read()

    mov ebx, 0 ; stdin

    mov ecx, s ; куда читаем

    mov edx, 128 ; сколько читаем

    int 0x80

    ; выбор типа данных

    @choice:

    cmp *byte*[ecx], '1'

    je input\_int

    cmp *byte*[ecx], '2'

    je input\_uint

    call err\_exit

; очистка регистров

clear:

    xor eax, eax

    xor ecx, ecx

    xor ebx, ebx

    xor edx, edx

    ret

; ввод unsigned int

input\_uint:

    inc ecx

    cmp *byte*[ecx], 0x20

    jne input\_uint

    inc ecx

    jmp @string\_to\_uintA

    @string\_to\_uintA:

    ; посимвольный перевод строки в число

    mov eax, 10

    movzx ebx, *word*[a]

    mul ebx

    cwd

    mov *word*[a], ax

    mov ebx, eax

    movzx eax, *byte*[ecx]

    sub eax, '0'

    ; проверки

    cmp eax, 0

    jb err\_exit

    cmp eax, 9

    ja err\_exit

    add ebx, eax

    add *word*[a], ax

    cmp ebx, [a]

    jne err\_exit

    ; цикл пока s[i] != ' '

    inc ecx

    cmp *byte*[ecx], 0x20

    jne @string\_to\_uintA

    inc ecx

    je @string\_to\_uintB

    @string\_to\_uintB:

    ; посимвольный перевод строки в число

    mov eax, 10

    movzx ebx, *word*[b]

    mul ebx

    cwd

    mov *word*[b], ax

    mov ebx, eax

    movzx eax, *byte*[ecx]

    sub eax, '0'

    ; проверки

    cmp eax, 0

    jb err\_exit

    cmp eax, 9

    ja err\_exit

    add ebx, eax

    add *word*[b], ax

    cmp ebx, [b]

    jne err\_exit

    ; цикл пока s[i] != '\n'

    inc ecx

    cmp *byte*[ecx], 0x20

    je err\_exit

    cmp *byte*[ecx], 0xa

    jne @string\_to\_uintB

    call division\_by\_zero

    call uint16\_asm

    call printres

; ввод signed int

input\_int:

    inc ecx

    cmp *byte*[ecx], 0x20

    jne input\_int

    inc ecx

    cmp *byte*[ecx], 0x2D

    je @minus\_A

    jmp @string\_to\_intA

    @minus\_A:

    mov ax, 1

    mov [ma], ax

    inc ecx

    cmp *byte*[ecx], 0x2D

    je @minus\_A

    jmp @string\_to\_intA

    @string\_to\_intA:

    ; посимвольный перевод строки в число

    mov eax, 10

    movsx ebx, *word*[a]

    mul ebx

    cwd

    mov *word*[a], ax

    mov ebx, eax

    movzx eax, *byte*[ecx]

    sub eax, '0'

    ; проверки

    cmp eax, 0

    jb err\_exit

    cmp eax, 9

    ja err\_exit

    add *word*[a], ax

    add ebx, eax

    cmp ebx, [a]

    jne err\_exit

    ; цикл пока s[i] != ' '

    inc ecx

    cmp *byte*[ecx], 0x20

    jne @string\_to\_intA

    inc ecx

    mov ax, [ma]

    cmp ax, 1

    je @minus\_A\_true

    movsx rax, *dword*[a]

    mov rbx, -32768

    cmp rax, rbx

    jl err\_exit

    mov rbx, 32767

    cmp rax, rbx

    jg err\_exit

    cmp *byte*[ecx], 0x2D

    je @minus\_B

    jmp @string\_to\_intB

    @minus\_A\_true:

    mov eax, [a]

    neg eax

    mov [a], eax

    movsx rax, *dword*[a]

    mov rbx, -32768

    cmp rax, rbx

    jl err\_exit

    mov rbx, 32767

    cmp rax, rbx

    jg err\_exit

    cmp *byte*[ecx], 0x2D

    je @minus\_B

    jmp @string\_to\_intB

    @minus\_B:

    mov ax, 1

    mov [mb], ax

    inc ecx

    cmp *byte*[ecx], 0x2D

    je @minus\_B

    jmp @string\_to\_intB

    @string\_to\_intB:

    ; посимвольный перевод строки в число

    mov eax, 10

    movsx ebx, *word*[b]

    mul ebx

    cwd

    mov *word*[b], ax

    mov ebx, eax

    movzx eax, *byte*[ecx]

    sub eax, '0'

    ; проверки

    cmp eax, 0

    jb err\_exit

    cmp eax, 9

    ja err\_exit

    add *word*[b], ax

    add ebx, eax

    cmp ebx, [b]

    jne err\_exit

    ; цикл пока s[i] != '\n'

    inc ecx

    cmp *byte*[ecx], 0x20

    je err\_exit

    cmp *byte*[ecx], 0xa

    jne @string\_to\_intB

    mov ax, [mb]

    cmp ax, 1

    je @minus\_B\_true

    movsx rax, *dword*[b]

    mov rbx, -32768

    cmp rax, rbx

    jl err\_exit

    mov rbx, 32767

    cmp rax, rbx

    jg err\_exit

    call division\_by\_zero

    call int16\_asm

    call printres

    @minus\_B\_true:

    mov eax, [b]

    neg eax

    mov [b], eax

    movsx rax, *dword*[b]

    mov rbx, -32768

    cmp rax, rbx

    jl err\_exit

    mov rbx, 32767

    cmp rax, rbx

    jg err\_exit

    call division\_by\_zero

    call int16\_asm

    call printres

; проверка деления на ноль

division\_by\_zero:

    xor ax, ax

    movsx ax, [b]

    sbb ax, 4

    cmp ax, 0

    je @dbz\_true

    ret

    @dbz\_true:

    mov edx, lendbz ; длина строки

    mov ecx, dbz ; строка

    mov ebx, 1 ; stdout

    mov eax, 4 ; sys\_write()

    int 0x80   ; call kernel

    call exit

; вывод ошибки

err\_exit:

    mov edx, lenerrmsg ; длина строки

    mov ecx, errmsg ; строка

    mov ebx, 1 ; stdout

    mov eax, 4 ; sys\_write()

    int 0x80   ; call kernel

    call exit

; выход

exit:

    mov eax, 1 ; sys\_exit()

    int 0x80   ; call kernel

; вывод результата

printres:

    call clear

    mov edx, lenresl ; длина строки

    mov ecx, resl ; строка

    mov ebx, 1 ; stdout

    mov eax, 4 ; sys\_write()

    int 0x80   ; call kernel

    call clear

    mov rax, [res]

    mov [fres], rax

    cmp rax, 0

    jl @print\_minus

    mov ax, [s]

    mov ax, 0

    mov [s], ax

    jmp @print\_else

    @print\_minus:

    mov rax, [fres]

    neg rax

    mov [fres], rax

    mov ecx, '-'

    mov [s], ecx

    mov edx, 1 ; длина строки

    mov ecx, s ; строка

    mov ebx, 1 ; stdout

    mov eax, 4 ; sys\_write()

    int 0x80   ; call kernel

    mov ax, [s]

    mov ax, 0

    mov [s], ax

    jmp @print\_else

    @print\_else:

    call clear

    mov rax, [fres]

    mov rbx, 10

    cqo

    div rbx

    mov [fres], rax

    movzx ecx, *byte*[i]

    add edx, '0'

    mov [s + ecx], edx

    movzx edx, *byte*[i]

    inc edx

    mov [i], edx

    mov rcx, [fres]

    cmp rcx, 0

    jne @print\_else

    jmp @final\_print

    @final\_print:

    call clear

    movzx eax, *byte*[i]

    dec eax

    mov [i], eax

    mov ebx, s

    add ebx, eax

    mov edx, 1 ; длина строки

    mov ecx, ebx ; строка

    mov ebx, 1 ; stdout

    mov eax, 4 ; sys\_write()

    int 0x80   ; call kernel

    movzx eax, *byte*[i]

    cmp eax, 0

    jne @final\_print

    mov eax, 0xa

    mov [s], eax

    mov edx, 1 ; длина строки

    mov ecx, s ; строка

    mov ebx, 1 ; stdout

    mov eax, 4 ; sys\_write()

    int 0x80   ; call kernel

    call exit

int16\_asm:

    movsx rax, *dword*[b]

    mov rbx, -32768

    cmp rax, rbx

    jl err\_exit

    mov rbx, 32767

    cmp rax, rbx

    jg err\_exit

    xor eax, eax

    xor ecx, ecx

    xor ebx, ebx

    xor edx, edx

    mov ax, [a]

    mov cx, [b]

    cmp ax, cx

    je @int\_equal

    jg @int\_great

    jl @int\_less

    @int\_equal:

    xor eax, eax

    xor ecx, ecx

    mov ax, [b]

    cwde

    sbb eax, 4

    mov ecx, eax

    xor eax, eax

    mov ax, [a]

    cwde

    cdq

    idiv ecx

    mov [res], eax

    jmp @int\_exit

    @int\_great:

    xor eax, eax

    mov eax, 64

    mov [res], eax

    jmp @int\_exit

    @int\_less:

    cwde

    xor ecx, ecx

    mov ecx, eax

    mov eax, [b]

    cwde

    imul ecx

    adc eax, 8

    mov [res], eax

    jmp @int\_exit

    @int\_exit:

    ret

uint16\_asm:

    xor eax, eax

    xor ecx, ecx

    xor ebx, ebx

    xor edx, edx

    mov ax, [a]

    mov cx, [b]

    cmp ax, cx

    je @uint\_equal

    ja @uint\_great

    jb @uint\_less

    @uint\_equal:

    xor eax, eax

    xor ecx, ecx

    mov ax, [b]

    cwd

    sbb eax, 4

    mov ecx, eax

    xor eax, eax

    mov ax, [a]

    cwd

    cdq

    idiv ecx

    mov [res], eax

    jmp @uint\_exit

    @uint\_great:

    xor eax, eax

    mov eax, 64

    mov [res], eax

    jmp @uint\_exit

    @uint\_less:

    cwd

    xor rcx, rcx

    mov rcx, rax

    movzx rax, *word*[b]

    cdq

    imul rcx

    adc rax, 8

    mov [res], rax

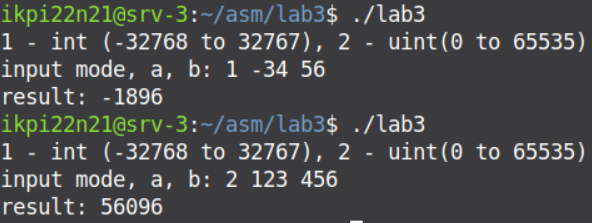
    jmp @uint\_exit

    @uint\_exit:

    ret

**Результаты работы программы**

Далее предоставлены результаты выполнения программы с выбранным для тестирования контрольным набором исходных данных.



**Выводы**

Тот факт, что результаты расчетов на ассемблере, совпадают с результатами расчетов лабораторной работы №2, свидетельствуют о том, что программа составлена правильно.