ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ "САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М. А. БОНЧ-БРУЕВИЧА"

Факультет инфокоммуникационных сетей и систем

Кафедра программной инженерии и вычислительной техники

ЛАБОРАТОРНАЯ РАБОТА №3

«ВВОД-ВЫВОД НА АССЕМБЛЕРЕ»

по дисциплине «Машинно-зависимые языки программирования»

Выполнил:

студент 2 курса

дневного отделения

группы ИКПИ-23

Даненко Д. А.

Санкт-Петербург 2023

А. Постановка задачи

Реализовать целочисленные вычисления, сделанные в лабораторной работе №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.

Реализация задачи должна быть выполнена для Linux.

Задание 2-рой лабораторной работы:

11 , a = b
a*a/b , если a < b
b*a/11 , если a > b

Б. Таблица идентификаторов

N	Обозначение в задаче	Идентификатор	Назначение
1	A (signed int)	a	Входные данные
2	B (signed int)	b	
3	—	msg_1 db 'Input values:', 0xa, 0	Промежуточные данные
4	—	msg_2 db 'a = ', 0	
5	—	msg_3 db 'b = ', 0	
6	—	msg_4 db 'Result:', 0xa, 0	
7	—	msg_5 db 0xa, 'ERROR!	
8	Результат	res	Выходные данные

В. Таблица результатов

Результаты вычислений приведены ниже в таблице вычислений.

Тип	A	B	X
Signed Int	**32767**	**-32768**	**32766**
	-32768	**32767**	**97609914**

Unsigned Word	**65535**	**65534**	**390433699**
	65535	**65535**	**11**

Г. Программа

INT32

```
%define STDIN 0;   ввод - клава
%define STDOUT 1;  вывод- экран
%define SYS_WRITE  0x01
%define SYS_READ   0x00

section .data
;                       <<<   DATA  >>>
      ;-------------------------------------------------------

      msg_1 db 'Input values:', 0xa, 0
      len_1 equ $ - msg_1

      msg_2 db 'a = ', 0
      len_2 equ $ - msg_2

      msg_3 db 'b = ', 0
      len_3 equ $ - msg_3

      msg_4 db 'Result:', 0xa, 0
      len_4 equ $ - msg_4

      msg_5 db 0xa, 'ERROR! Try again', 0xa
      len_5 equ $ - msg_5

      ;-------------------------------------------------------
      ;Буфер
      len_inp db 7            ;ввод max 6 символов (-32768) + '/n'
      len_out db 12           ;вывод max 11 символов (-2147483648) + '/n'
      len_cur db 0            ;реальная длина
      sign db 0               ;знак числа
```

```
	buffer times 12 db 0	;введенные символы

	;-----------------------------------------------------------
	;Переменные
	a dw 0
	b dw 0
	res dd 0

section .text
global _start

;			<<<   MAIN  >>>
_start:
	;-----------------------------------------------------------
	;Вывод 1-го сообщения
	mov rsi, msg_1		;message to write
	mov rdx,len_1		;message length
	call write

	;-----------------------------------------------------------
	;Вывод 2-го сообщения
	mov rsi, msg_2
	mov rdx, len_2
	call write

	;-----------------------------------------------------------
	;Ввод a
	call read
	call to_int		;преобразовать 'a' в число
	mov word [a], ax	; -> 'a'
	call test
	call buffer_cl		;очистка буфера

	;-----------------------------------------------------------
	;Вывод 3-го сообщения
	mov rsi, msg_3
	mov rdx,len_3
```

```
      call write

      ;-----------------------------------------------------------
      ;Ввод b
      call read
      call to_int
      mov word [b], ax        ; -> 'b'
      call test
      call buffer_cl

      ;-----------------------------------------------------------
      ;Рассчет значений
      call culculate          ; -> 'res'
      call test
      call to_str             ;преобразовать 'res' в строку

      ;-----------------------------------------------------------
      ;Вывод результата
      mov rsi, buffer
      mov dl, byte [len_cur]
      call write

      ;-----------------------------------------------------------
      ;Выод из программы
      call exit


;                     <<<   FUNC  >>>
test:
      ret
;-------------------------------------------------------------
;Функция вывода
write:
      mov rax, SYS_WRITE  ;sys_write()
      mov rdi, STDOUT   ;file descriptor (stdout)
      syscall
      ret
```

```
;-------------------------------------------------------------
;Функция ввода
read:
      mov eax, SYS_READ   ;sys_read()
      mov rdi, STDIN    ;file descriptor (stdin)
      mov rsi, buffer   ;message to write
      mov edx, len_inp  ;message length
      syscall           ;execute read(0, buffer, buf_size)
      dec al
      mov byte [len_cur], al
      ret

;-------------------------------------------------------------
;Очистка буфера
buffer_cl:
      xor ecx, ecx
      mov cl, byte [len_cur]
      mov eax, 0
      clear:
      mov byte [buffer + eax], ''
      inc eax
      loop clear
      mov byte [len_cur], 0
      ret

;-------------------------------------------------------------
;Функции преобраования в число
to_int:
      mov byte [sign], 0
      mov edx, 0                    ;edx -> адрес строки
      mov bl, [buffer + edx]        ;bl  -> первый символ строки
      cmp bl,'-'
      jne int_no_sign
      inc edx                 ;'+' адрес строки
      dec al                  ;'-' длина строки
      mov byte [len_cur], al
```

```
	mov byte [sign], 1

int_no_sign:
	xor ecx, ecx
	mov di,10		;di -> мноитель
	mov cl, [len_cur]	;cx -> счётчик цикла
	;jecxz studw_error		;длина = 0 -> ошибка

	xor rax,rax
	xor rbx,rbx

next_symb:
	mov bl, [buffer + edx]	;bl -> символ строки
	inc edx			;'+' адрес
	cmp bl,'0'
	jl error		;код символа < '0' -> ошибка
	cmp bl,'9'
	jg error		;код символа > '9' -> ошибка
	sub bl,'0'		;преобразование в число
	push rdx
	mul di			;ax -> ax * 10
	pop rdx
	jc error		;резульатат > 2^16  -> ошибка
	add ax, bx
	jc error		;переполнение     -> ошибка
	loop next_symb		;цикл
	mov dl, byte [sign]
	test dl, dl
	jz int_plus
	cmp ax, 32768		;модуль отр.числа  <= 32768
	ja error		;больше           -> ошибка
	neg ax
	ret

int_plus:
	cmp ax,32767		;пол.число  <= 32767
	ja error		;больше           -> ошибка
```

```
      ret

;---------------------------------------------------------------
;Функции преобраования в строку
to_str:
      mov byte [sign], 0
      mov eax, dword [res]
      mov ecx, 0
      test eax, eax           ;проверка знака
      mov ebx, 10
      jns str_no_sign
      mov byte [sign], 1      ;добавление знака
      neg eax
str_no_sign:
      xor edx, edx            ;обнуление старшей части двойного слова
      div ebx                       ;edx:eax / ebx
      add dx,'0'                    ;преобразование в код символа
      push dx                       ;сохранение в стек
      inc ecx
      test eax, eax           ;проверка AX
      jnz str_no_sign         ;частное не 0 -> икл
if_sing_str:
      mov dl, byte [sign]
      test dl, dl
      jz str
      mov byte [buffer + eax], '-'
      inc eax
str:                          ;икл из стека
      pop dx
      mov byte [buffer + eax],dl
      inc eax                       ;'+' адрес буфера
      loop str
      mov byte [buffer + eax], 0x0a
      inc eax
      mov byte [len_cur], al
      ret
```

```
;-------------------------------------------------------------
;Функция вывода ошибки
error:
      mov rsi, msg_5
      mov rdx,len_5
      call write
      call exit

;-------------------------------------------------------------
;Функция выода из прораммы
exit:
      mov ebx, 0
      mov rax,60              ;sys_exit()
      syscall


;                       <<<   CALC  >>>
culculate:
      xor rax, rax
      xor rbx, rbx
      xor rcx, rcx
      xor rdx, rdx
      ; a*b/11    , a > b
      ; 11  , a = b
      ; a*a/b , a < b
      mov ax, word [a]
      mov bx, word [b]
      cmp ax, bx
      ja @bg
      je @eq
      imul   eax        ; edx:eax = a*a
       idiv   ebx             ; eax = a*a/b
      jmp @end_if
      ; a > b
      @bg:
            imul   ebx        ; eax:edx = a*b
            mov   ecx,  11    ; ecx = 11
```

```
        idiv   ecx       ; eax:edx / eax
        jmp @end_if
    ; a = b
    @eq:
        mov eax, 11
    @end_if:
    mov dword [res], eax
    ret
```

UINT16

```
%define STDIN 0;   ввод - клава
%define STDOUT 1;  вывод- экран
%define SYS_WRITE  0x01
%define SYS_READ   0x00

section .data
;                       <<<   DATA  >>>
    ;-------------------------------------------------------------
    ;Сообщения
    msg_1 db 'Input values:', 0xa, 0
    len_1 equ $ - msg_1

    msg_2 db 'a = ', 0
    len_2 equ $ - msg_2

    msg_3 db 'b = ', 0
    len_3 equ $ - msg_3

    msg_4 db 'Result:', 0xa, 0
    len_4 equ $ - msg_4

    msg_5 db 0xa, 'ERROR! Try again', 0xa
    len_5 equ $ - msg_5

    ;-------------------------------------------------------------
    ;Буфер
    len_inp db 6            ;ввод max 5 символов (65535) + '/n'
    len_out db 11           ;вывод max 10 символов (4294967296) + '/n'
    len_cur db 0            ;реальная длина
```

```
	sign db 0		;знак числа

	buffer times 11 db 0	;введенные символы

	;-----------------------------------------------------------
	;Переменные
	a dw 0
	b dw 0
	res dd 0

section .text
global _start

;			<<<   MAIN  >>>
_start:
	;-----------------------------------------------------------
	;Вывод 1-го сообщения
	mov rsi, msg_1		;message to write
	mov rdx,len_1		;message length
	call write

	;-----------------------------------------------------------
	;Вывод 2-го сообщения
	mov rsi, msg_2
	mov rdx, len_2
	call write

	;-----------------------------------------------------------
	;Ввод a
	call read
	call to_int		;преобразовать 'a' в число
	mov word [a], ax	; -> 'a'
	call test
	call buffer_cl		;очистка буфера

	;-----------------------------------------------------------
	;Вывод 3-го сообщения
```

```
      mov rsi, msg_3
      mov rdx,len_3
      call write

      ;-------------------------------------------------------------
      ;Ввод b
      call read
      call to_int
      mov word [b], ax        ; -> 'b'
      call test
      call buffer_cl

      ;-------------------------------------------------------------
      ;Рассчет значений
      call culculate          ; -> 'res'
      call test
      call to_str             ;преобразовать 'res' в строку

      ;-------------------------------------------------------------
      ;Вывод результата
      mov rsi, buffer
      mov dl, byte [len_cur]
      call write

      ;-------------------------------------------------------------
      ;Выод из программы
      call exit


;                     <<<   FUNC  >>>
test:
      ret
;-------------------------------------------------------------
;Функция вывода
write:
      mov rax, SYS_WRITE  ;sys_write()
      mov rdi, STDOUT   ;file descriptor (stdout)
```

```
	syscall
	ret

;-------------------------------------------------------------
;Функция ввода
read:
	mov eax, SYS_READ   ;sys_read()
	mov rdi, STDIN    ;file descriptor (stdin)
	mov rsi, buffer   ;message to write
	mov edx, len_inp  ;message length
	syscall           ;execute read(0, buffer, buf_size)
	dec al
	mov byte [len_cur], al
	ret

;-------------------------------------------------------------
;Очистка буфера
buffer_cl:
	xor ecx, ecx
	mov cl, byte [len_cur]
	mov eax, 0
	clear:
	mov byte [buffer + eax], ''
	inc eax
	loop clear
	mov byte [len_cur], 0
	ret

;-------------------------------------------------------------
;Функции преобраования в число
to_int:
	mov edx, 0                    ;edx -> адрес строки
	mov di,10               ;di -> мноитель
	xor ecx, ecx
	mov cl, [len_cur]       ;cx -> счётчик цикла
	;jecxz studw_error            ;длина = 0 -> ошибка
	xor rax,rax
```

```
      xor rbx,rbx
next_symb:
      mov bl, [buffer + edx]  ;bl -> символ строки
      inc edx                 ;'+' адрес
      cmp bl,'0'
      jl error                ;код символа < '0' -> ошибка
      cmp bl,'9'
      jg error                ;код символа > '9' -> ошибка
      sub bl,'0'              ;преобразование в число
      push rdx
      mul di                  ;ax -> ax * 10
      pop rdx
      jc error                ;результат > 2^16  -> ошибка
      add ax, bx
      jc error                ;переполнение     -> ошибка
      loop next_symb                ;цикл
      cmp ax, 65535                 ;пол.число        <= 65535
      ja error                ;больше           -> ошибка
      ret

;---------------------------------------------------------------
;Функции преобраования в строку
to_str:
      mov byte [sign], 0
      mov eax, dword [res]
      mov ecx, 0
      test eax, eax           ;проверка знака
      mov ebx, 10
      jns str_no_sign
      mov byte [sign], 1      ;добавление знака
      neg eax
str_no_sign:
      xor edx, edx            ;обнуление старшей части двойного слова
      div ebx                       ;edx:eax / ebx
      add dx,'0'                    ;преобразование в код символа
      push dx                       ;сохранение в стек
      inc ecx
```

```
      test eax, eax           ;проверка AX
      jnz str_no_sign         ;частное не 0 -> икл
if_sing_str:
      mov dl, byte [sign]
      test dl, dl
      jz str
      mov byte [buffer + eax], '-'
      inc eax
str:                          ;икл из стека
      pop dx
      mov byte [buffer + eax],dl
      inc eax                       ;'+' адрес буфера
      loop str
      mov byte [buffer + eax], 0x0a
      inc eax
      mov byte [len_cur], al
      ret

;-----------------------------------------------------------------
;Функция вывода ошибки
error:
      mov rsi, msg_5
      mov rdx,len_5
      call write
      call exit

;-----------------------------------------------------------------
;Функция выода из прораммы
exit:
      mov ebx, 0
      mov rax,60              ;sys_exit()
      syscall


;                       <<<   CALC  >>>
culculate:
      xor rax, rax
```

```
xor rbx, rbx
xor rcx, rcx
xor rdx, rdx
; a*b/11    , a > b
; 11  , a = b
; a*a/b , a < b
mov ax, word [a]
mov bx, word [b]
cmp ax, bx
ja @bg
je @eq
imul   eax        ; edx:eax = a*a
 idiv   ebx             ; eax = a*a/b
jmp @end_if
; a > b
@bg:
      imul   ebx        ; eax:edx = a*b
      mov   ecx,  11    ; ecx = 11
       idiv   ecx       ; eax:edx / eax
       jmp @end_if
; a = b
@eq:
      mov eax, 11
@end_if:
mov dword [res], eax
ret
```

Д. Выводы

Тот факт, что результаты, выполненные на NASM, совпадают с результатами, полученными во второй лабораторной работе, свидетельствует о том, что программа составлена корректно.