НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра технічної кібернетики

Звіти до комп'ютерних практикумів з кредитного модуля «Програмування мовою Асемблер»

Прийняв
доцент кафедри ТК
Лісовиченко О. І.
"..." ..... 2020 р.

Виконав
студент групи ІП-91
Кінчур В. В.

Київ 2020

Комп’ютерний практикум №2

Тема: засоби обміну даними

Завдання:

1. Написати процедуру введення і перетворення цілого числа.
2. Виконати математичну дію над числом (-88).
3. Перевести число в рядок та вивести його на екран.

Текст програми (NASM):

```
global _start                    ; start point for linker

section .data
        msg db "Enter number: ",  10
        len equ $ - msg
section .bss
        buffer resb 7
        dummy  resb 1

section .text
_start:

        mov ebx, 1
        mov eax, 4
        mov edx, len
        mov ecx, msg
        int 80h

_clear_buffer:
        mov ecx, 7              ; buffer length
.loop:
        mov byte [buffer+ecx], 0    ; fill with \0
        loop .loop

_read_input:
        mov ebx, 2              ; stdin
        mov edx, 7              ; maximal length of input
        mov ecx, buffer           ; place str to input variable
```

```
_flush_stdin:
        mov eax, 3              ; sys_read
        int 80h                 ; call kernel
        cmp byte [ecx+eax-1], 10    ; compare last char in stdin with \n
        je _main
        mov edx, 1
        mov ecx, dummy
        jmp _flush_stdin

_main:
        push buffer
        call _atoi              ; result in ax
        add esp, 4

        sub ax, 88              ; operation (var. 11)
        jo _start               ; met overflow

        mov edi, buffer
        push ax
        call _print_num
        add esp, 2

_exit:
        mov ebx, 0
        mov eax, 1
        int 80h

;----atoi(const char* str) -> int (ax)
```

```
;---- ax - output
;---- bl - current char
;---- dl - sign
_atoi:
        enter 0, 0
        xor eax, eax
        xor ebx, ebx
        xor edx, edx
        mov ecx, [ebp+8]          ; pointer to first char in input

.loop:
        mov bl, byte [ecx]        ; current char
        cmp bl, 0                 ; \0
        je .done
        cmp bl, 10                ; \n
        je .done

        cmp bl, 45                ; check for minus
        je .check_sign
        cmp bl, 48                ; less than '0'
        jl .error
        cmp bl, 57                ;  greater than '9'
        jg .error

.valid_digit:
        sub bl, 48                ; get digit
        imul ax, word 10
        jo .error
        add ax, bx
```

```
                jo .error
                jmp .new_iteration

.check_sign:
                cmp ecx, [ebp+8]        ; check is it first char
                jne .error
                mov dl, 1

.new_iteration:
                inc ecx
                jmp .loop

.error:
                leave
                add esp, 8
                jmp _start              ; prompt for a new number

.done:
                cmp dl, 0               ; check for sign
                je .exit
                neg ax
                jo .error

.exit:
                leave
                ret

;-- print_num(int) -> void
;-- print 16bit integer to stdin
```

```
_print_num:
        enter 0, 0

        cmp word [ebp+8], 0         ; check sign
        jge .init

        mov byte [edi], '-'
        inc edi
        neg word [ebp+8]            ; make input positive

.init:
        mov ax, [ebp+8]
        mov bx, 10

.loop:
        xor dx, dx
        div bx
        add dl, 48                  ; '0'
        push dx
        test ax, ax
        jnz .loop

.stack:
        pop ax
        stosb
        cmp esp, ebp                ; check stack is empty
        jne .stack

.print:
```

```
        mov ecx, buffer
        mov edx, edi
        sub edx, buffer
        mov ebx, 1
        mov eax, 4
        int 80h

.exit:
        leave
        ret
```

Введені та отримані результати

Тестування програми №1

```
Enter number:
121
33[vvkin@host105 asm-labs]$
```

Тестування програми №2

```
Enter number:
abcd
Enter number:
-11-2
Enter number:
--9
Enter number:
88
0[vvkin@host105 asm-labs]$
```

Тестування програми №3

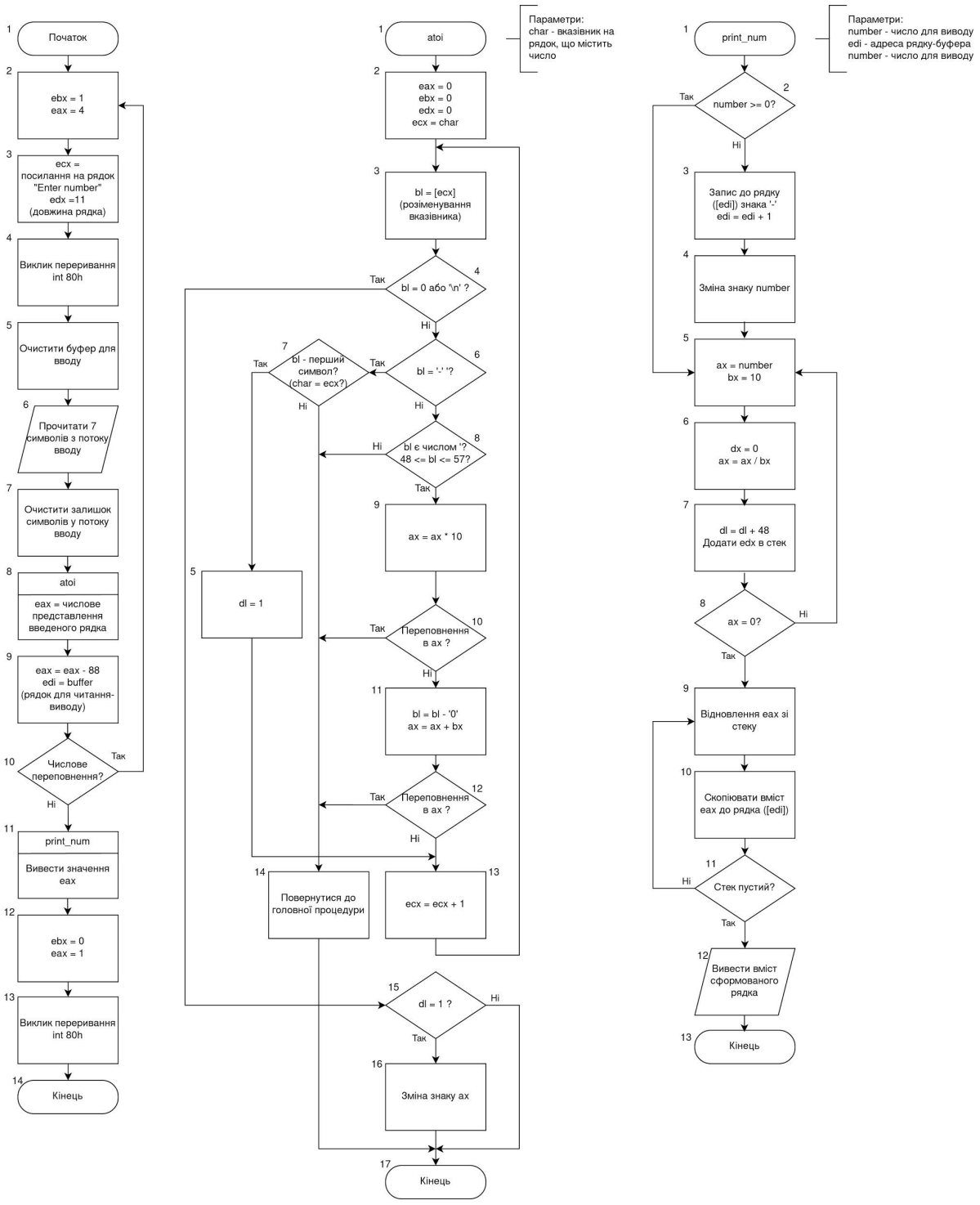
```
Enter number:
65536
Enter number:
32768
Enter number:
32767
32679[vvkin@host105 asm-labs]$
```

Схема функціонування програми

Див. додаток.

Висновок

1. Написав процедуру введення і перетворення цілого числа, обмеженого значеннями -32768 .. 32767 (16 бітне ціле знакове число).
2. Виконав математичну дію над числом (-88).
3. Перевів число в рядок та вивів його на екран.


1
Початок
2
ebx = 1
eax = 4
3
ecx =
посилання на рядок
"Enter number"
edx =11
(довжина рядка)
4
Виклик переривання
int 80h
5
Очистити буфер для
вводу
6
Прочитати 7
символів з потоку
вводу
7
Очистити залишок
символів у потоку
вводу
8
atoi
eax = числове
представлення
введеного рядка
9
eax = eax - 88
edi = buffer
(рядок для читання-
виводу)
10
Числове
переповнення?
Так
Ні
11
print_num
Вивести значення
eax
12
ebx = 0
eax = 1
13
Виклик переривання
int 80h
14
Кінець
1
atoi
Параметри:
char - вказівник на
рядок, що містить
число
2
eax = 0
ebx = 0
edx = 0
ecx = char
3
bl = [ecx]
(розіменування
вказівника)
4
bl = 0 або '\n' ?
Так
Ні
5
dl = 1
6
bl = '-' ?
Так
Ні
7
bl - перший
символ?
(char = ecx?)
Так
Ні
8
bl є числом ?
48 <= bl <= 57?
Ні
Так
9
ax = ax * 10
10
Переповнення
в ax ?
Так
Ні
11
bl = bl - '0'
ax = ax + bx
12
Переповнення
в ax ?
Так
Ні
13
ecx = ecx + 1
14
Повернутися до
головної процедури
15
dl = 1 ?
Ні
Так
16
Зміна знаку ax
17
Кінець
1
print_num
Параметри:
number - число для виводу
edi - адреса рядку-буфера
number - число для виводу
2
number >= 0?
Так
Ні
3
Запис до рядку
([edi]) знака '-'
edi = edi + 1
4
Зміна знаку number
5
ax = number
bx = 10
6
dx = 0
ax = ax / bx
7
dl = dl + 48
Додати edx в стек
8
ax = 0?
Ні
Так
9
Відновлення eax зі
стеку
10
Скопіювати вміст
eax до рядка ([edi])
11
Стек пустий?
Ні
Так
12
Вивести вміст
сформованого
рядка
13
Кінець