НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

*Звіти до комп’ютерних практикумів дисципліни*

«Системне програмне забезпечення»

**Прийняв Виконав**

**доцент кафедри ІПІ Студент групи ІП-21**

**Лісовиченко О.І. Скрипець О.О.**

**“06” травня 2024 р.**

Київ – 2024

**Комп’ютерний практикум №2**

**Тема:** засоби обміну даними.

**Завдання**:

Написати програму з використанням 2-х процедур:

1. Процедура введення і перетворення цілого числа. Після цього треба виконати

математичну дію над числом (номер завдання вибирати за номером у списку

(Classroom) - Таблиця 2.1).

2. Процедура переведення отриманого результату в рядок та виведення його на

екран.

2. Програма повинна мати захист від некоректного введення вхідних даних (символи,

переповнення, ділення на 0 і т.і.)

Текст програми

stack\_seg segment para stack 'stack'

db 64 dup ( 'STACK' )

stack\_seg ends

data\_seg segment para public 'data'

valid\_flag db 1

num\_value dw 0

number dw -96

temp\_val dw 0

minus\_flag dw 0

prompt\_input db 0dh, 0ah,'enter number in range of -32767 to 32767: $'

output\_msg db 0dh, 0ah, 'output: $'

error\_input\_msg db 0dh, 0ah, 'the number you entered is too large$'

error\_minus\_msg db 0dh, 0ah, 'the result of is less than -32767$'

error\_char\_msg db 0dh, 0ah, 'wrong character$'

error\_empty\_msg db 0dh, 0ah, 'empty input$'

data\_seg ends

code\_seg segment para public 'code'

main\_process proc

assume cs: code\_seg, ds: data\_seg, ss: stack\_seg

push ds

xor ax, ax ; обнулення

push ax

mov ax, data\_seg

mov ds, ax

main\_cycle:

call read\_num

call subtract\_96

cmp valid\_flag,1

jne not\_eq ; перейти якщо не дорівнює

jmp end\_if

not\_eq:

xor ax, ax

mov num\_value, ax

mov temp\_val, ax

mov minus\_flag, ax

mov valid\_flag, 1

jmp main\_cycle

end\_if:

call print\_num

jmp main\_cycle

end:

mov ah, 4ch

int 21h ; переривання, завершення програми

main\_process endp

subtract\_96 proc

mov ax, number

add num\_value, ax

jo error\_minus ; переходить на помилкук якщо відбулось переповненя

ret

error\_minus:

mov valid\_flag, 0

lea dx, error\_minus\_msg

mov ah, 09h ; копіює значення в регістр, вказщує на функцію що виводить рядок

int 21h ; викликає цю функцію

ret

subtract\_96 endp

read\_num proc

xor ax, ax ; обнулення

mov num\_value, ax ; обнулення

mov temp\_val, ax ; обнулення

mov minus\_flag, ax ; обнулення

lea dx, prompt\_input ; завантажує адресу в dx

mov ah, 09h

int 21h

mov bx, 10 ; буде використовуватись для множення

mov cx, 5 ; буде використовуватись для контролю циклу

read\_cycle:

xor ax, ax

mov ah, 01h ; читання символу з клави

int 21h

cmp al, 13 ; якщо натиснено ентер(ASCII-код 13) - зупинка

je stop

cmp al, 48 ; якщо введено число

jl check\_sign ; перейти якщо код символу менше 48 (що відповідає ‘0’)

cmp al, 57

ja error\_char ; перейти якщо код символу більше 57 (що відповідає ‘9’)

sub al, '0' ; перетворення кода в число

sub ah, ah

mov temp\_val, ax

mov ax, num\_value

imul bx ; зсув в ліво на один (множення на 10)

jo error\_input ; якщо переповнення

add ax, temp\_val

jo error\_input

mov num\_value, ax

loop read\_cycle ; для проходження по циклу поки сх не буде 0

cmp minus\_flag, 1

je make\_minus

ret

stop:

cmp cx, 5

je error\_empty

mov cx, 0

cmp minus\_flag, 1

je make\_minus

ret

make\_minus:

neg num\_value

ret

check\_sign:

cmp al, '-'

jne error\_char

cmp cx, 5

jne error\_char

inc minus\_flag

cmp minus\_flag, 2 ; якщо 2 мінуси

jne read\_cycle

jmp error\_char

error\_input:

mov valid\_flag, 0

lea dx, error\_input\_msg

mov ah, 09h

int 21h

jmp end

error\_char:

mov valid\_flag, 0

lea dx, error\_char\_msg

mov ah, 09h

int 21h

jmp end

error\_empty:

mov valid\_flag, 0

lea dx, error\_empty\_msg

mov ah, 09h

int 21h

end: ret

read\_num endp

print\_num proc

lea dx, output\_msg ; виведення повідомлення

mov ah, 09h

int 21h

mov bx, num\_value

or bx, bx ; встановлення Sign Flag для перевірки від'ємності

jns positive ; перехід бо додатній

mov al, '-'

int 29h ; вивести мінус

neg bx

positive:

mov ax, bx

xor cx, cx

mov bx, 10

convert: ; перетворення числа в рядок

xor dx, dx

div bx

add dl, '0'

push dx

inc cx

test ax, ax ;=0

jnz convert

print:

pop ax ; виведення цифр зі стеку

int 29h

loop print

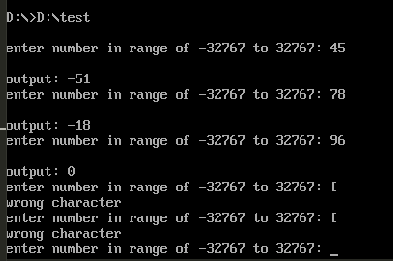
ret

print\_num endp

code\_seg ends

end main\_process

**Виведення:**



45-96=-51

78-96=-18

96-96=0

**Висновок:**

У цій лабораторній роботі ми розглянули створення програми на асемблері, яка використовує дві процедури для введення та перетворення цілого числа, виконання математичної дії над ним, а також перетворення результату в рядок та його виведення на екран.

Ця робота також підкреслила важливість ретельного тестування та налагодження коду, щоб забезпечити його коректну роботу в усіх можливих сценаріях використання. Загалом, ця лабораторна робота була корисною для поглиблення наших знань про програмування на асемблері та розуміння того, як можна ефективно структурувати та організувати код на цій мові.