МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний Технічний Університет України
«Київський Політехнічний Інститут»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №5
з дисципліни «Системне програмування – 1»
на тему: «Програмування множення чисел підвищеної розрядності»

Виконав:
студент 2-го курсу ФІОТ
групи ІВ-71
Мазан Я. В.
Перевірив:
Старший викладач
Порєв В. М.

Київ – 2019

Мета:
Навчитися програмувати на асемблері множення чисел підвищеної розрядності, а також закріпити навички програмування власних процедур у модульному проекті.

Варіант завдання:
N=9
n = N*2+30 = 48

Код програми:

main.asm:

```
%include "io.inc"

%include "multiply.asm"


%macro PRINT_LONG 2 ; length, dword long_number
mov ebp, %1
mov ecx, 0
%%print:
dec ebp
PRINT_HEX 4, [%2 + ecx*4]
PRINT_STRING " "
inc ecx
cmp ebp, 0
jnz %%print
%endmacro


section .bss
buffer : resd 9
factorial : resd 9
factorial_squared : resd 18
test1 : resd 51
res_test1 : resd 102
;test2 : resd 51
res_test2 : resd 52
test3 : resd 51
res_test3 : resd 102
section .data
test21 dd 0xffffffff
var dd 1
inscription db "Лабораторна №5. Виконав Мазан Ян, група ІВ-71", 0
section .text
global CMAIN
CMAIN:
mov ebp, esp; for correct debugging
;write your code here
xor eax, eax
mov dword [factorial + 8*4], 1
mov ebx, 1
@factorial:
push 9
push factorial
push dword [var]
push buffer
call MulFactorial
add esp, 16
mov eax, dword [var]
inc eax
mov dword [var], eax
cmp eax, 48
```

```
jle @factorial
mov esi, 0
@initTest1:
mov dword [test1 + 4*esi], 0xffffffff
inc esi
cmp esi, 51
jl @initTest1
@initTest3:
mov dword [test3], 0xc0 ; 0b11000000
PRINT_STRING inscription
NEWLINE
NEWLINE
;incorrect
push 51
push test1
push test1
push res_test1
call MulN_x_N
add esp, 16
NEWLINE
PRINT_STRING "test1: "
PRINT_LONG 102, res_test1
;correct
push 51
push test1
push test21
push res_test2
call MulN_x_32
add esp, 16
NEWLINE
NEWLINE
PRINT_STRING "test2: "
PRINT_LONG 52, res_test2
;correct
push 51
push test1
push test3
push res_test3
call MulN_x_N
add esp, 16
NEWLINE
NEWLINE
PRINT_STRING "test3: "
PRINT_LONG 102, res_test3
NEWLINE
NEWLINE
PRINT_STRING "n! = "
PRINT_LONG 9, factorial
push 9
push factorial
push factorial
push factorial_squared
call MulN_x_N
add esp, 16
NEWLINE
PRINT_STRING "(n!)^2 = "
PRINT_LONG 18, factorial_squared
ret
```

multiplication.asm:

```
%include "io.inc"

section .bss
    i : resd 1
    j : resd 1
section .text
global mulN_x_32
global mulN_x_N
```

```
MulN_x_32:                          ; size, A, B 32 bit, res
   push ebp
   mov ebp, esp

   mov esi, dword [ebp + 20]            ; size
   mov edi, dword [ebp + 16]            ; A
   mov ebx, dword [ebp + 12]            ; B 32 bit
   mov ecx, dword [ebp + 8]             ; res
   dec esi

   @mulN_x_32:
      mov eax, dword [edi + 4*esi]
      mul dword [ebx]
      add dword [ecx + 4*esi + 4], eax
      add dword [ecx + 4*esi], edx

      dec esi
      cmp esi, 0
      jge @mulN_x_32
   leave
   ret 16

MulN_x_N:                           ; size, A, B, res
   push ebp
   mov ebp, esp

   mov esi, dword [ebp + 20]            ; size
   dec esi                          ; Making pointer to point on start of
   push esi                          ; save pointer on the last element
   mov dword [i], esi                   ; i - iterates over A's dwords
   mov dword [j], esi                   ; j - iterates over B's dwords
                                    ; base pointer: edi = i + j
   mov esi, dword [ebp + 16]            ; A
   mov ebx, dword [ebp + 12]            ; B
   mov edi, dword [ebp + 8]             ; result

   @outerCycle:
      pop dword [j]
      push dword [j]
      @innerCycle:
         mov ecx, dword [i]
         mov eax, dword [esi + 4*ecx]    ; eax = dword [A + i]
         mov ecx, dword [j]
         mul dword [ebx + 4*ecx]         ; eax *= dword [B + j]
         add ecx, dword [i]
         add dword [edi + 4*ecx + 4], eax; res = dword [eax + i + j + 4], eax
         adc dword [edi + 4*ecx], edx    ; res = dword [eax + i + j], edx
         push ecx
         pushf
         @addcarryflag:                  ; spread carry flag over all other digit numbers
            popf
            adc dword [edi + 4*ecx - 4], 0
            pushf
            dec ecx
            cmp ecx, 0
            jge @addcarryflag
         popf
         pop ecx
         mov eax, dword [j]
         dec eax
         mov dword [j], eax
         cmp eax, 0
         jge @innerCycle
      mov eax, dword [i]
      dec eax
      mov dword [i], eax
      cmp eax, 0
      jge @outerCycle
   leave
   ret 16

MulFactorial:                       ; size, A, B 32 bit, res
   push ebp
   mov ebp, esp

   mov esi, dword [ebp + 20]            ; A's size
   mov edi, dword [ebp + 16]            ; A
   mov ebx, dword [ebp + 12]            ; B 32 bit
   mov ecx, dword [ebp + 8]             ; buffer
```

```
push ecx                        ; save ecx if was used before
push esi
dec esi
@mulFact:
   mov eax, dword [edi + 4*esi]
   mul ebx
   add dword [ecx + 4*esi], eax
   add dword [ecx + 4*esi - 4], edx
   dec esi
   cmp esi, 0
   jnz @mulFact

pop esi
dec esi
@swap:                          ; mov result into A
   mov edx, dword [ecx + 4*esi]
   mov dword [ecx + 4*esi], 0
   mov dword [edi + esi*4], edx
   dec esi
   cmp esi, 0
   jge @swap
pop ecx
leave
ret 16
```

Висновок:

Під час виконання лабораторної роботи були покращені навички написання власних модулів, роботи з циклами, а також були закріпленні основні навички в операціях множення чисел з підвищеною розрядністю.