НАЦІОНАЛЬНИЙ ТЕХНІЧНЙ 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
доцент кафедри ІПІ
Лісовиченко О.І.
"22" квітня 2024р.

Виконав
Студент групи ІП-24
Піддубний Б.С.

Київ 2024

Комп’ютерний практикум №2

Тема: Засоби обміну даними

Завдання:

1. Написати програму з використанням 2-х процедур:
    a. Процедура введення і перетворення цілого числа. Після цього треба виконати математичну дію над числом (номер завдання вибирати за номером у списку Classroom).
    b. 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.).

Варіант 22: операція (· 2)

Текст програми:

```
.8086

.model small


.stack 100h


.data

    header_msg          db 10, '--- Multiplier by 2 ---', 10, 10, 13, '$'

    prompt_msg          db 'Enter your number (from -32 768 to 32 767): $'

    result_msg          db 'Result is: $'

    overflow_err_msg    db 'Error: Overflow happened.', 10, 13, '$'

    non_int_err_msg     db 'Error: You have entered non-integer value.', 10, 13,
'$'

    incorrect_input_msg db 'Error: Incorrect input number.', 10, 13, '$'

    new_line            db 10, 13, "$"
```

```
    buffer          db 7, ?, 7 dup (?)
    num             dw 0
    minus_sign      db 0

.code
    start:
                    mov  ax, DGROUP
                    mov  ds, ax

    ; print header
                    lea  dx, header_msg
                    call print_line

    ; read number
                    call process_number


    ; print result
                    lea  dx, new_line
                    call print_line
                    lea  dx, result_msg
                    call print_line
                    call print_number

    ; terminate program
```

```
                mov  ax, 4c00h
                int  21h

print_line proc
                mov  ah, 09h
                int  21h
                ret
print_line endp

process_number proc
   process_input:
                xor  ax, ax
                xor  dx, dx
                mov  minus_sign, 0

   ; print prompt
                lea  dx, prompt_msg
                call print_line

   ; receive user input using DOS function 21h, interruption 0Ah
                lea  dx, buffer
                mov  ah, 0ah
                int  21h

   ; Process numerical input (digits 0-9)
```

```
                xor  ax, ax
                mov  cl, buffer + 1          ; Set the loop counter to the number
of characters to process
                mov  si, 2                   ; Set the SI index to the first character
of number
                cmp  buffer + 2, '-'
                jne  process_digit

        ; Handle negative sign (if present)
                inc  si                      ; Increment SI to point to the first digit
after the minus sign
                dec  cl                      ; Decrement CL to account for the minus
sign (one less digit to process)
                mov  minus_sign, 1

        ; Convert ASCII to integer and check for errors
        process_digit:
                mov  bl, buffer[si]
                cmp  bl, '0'
                jb   handle_non_int
                cmp  bl, '9'
                ja   handle_non_int
                sub  bl, '0'
                mov  dx, 10
                imul dx
                jo   handle_overflow
                add  ax, bx
```

```
                jo   handle_overflow
                inc  si
                loop process_digit
                jmp  check_sign

handle_overflow:
                lea  dx, new_line
                call print_line
                lea  dx, overflow_err_msg
                call print_line
                jmp  process_input

handle_non_int:
                lea  dx, new_line
                call print_line
                lea  dx, non_int_err_msg
                call print_line
                jmp  process_input

check_sign:
                cmp  minus_sign, 1
                jne  perform_math_operation
                neg  ax

perform_math_operation:
```

```
                mov  dx, 2
                imul dx
                jo   handle_overflow
                mov  num, ax

                ret
process_number endp

print_number proc
                mov  bx, num
                or   bx, bx
                jns  m1
                mov  al, '-'
                int  29h
                neg  bx

   m1:
                mov  ax, bx
                xor  cx, cx
                mov  bx, 10

   m2:
                xor  dx, dx
                div  bx
                add  dl, '0'
```

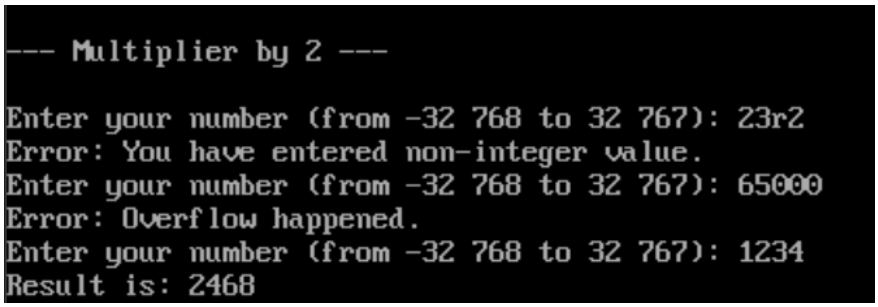
```
        push dx
        inc  cx
        test ax, ax
        jnz  m2

    m3:
        pop  ax
        int  29h
        loop m3

        ret
print_number endp

end start
```

Введені та отримані результати



Вміст .lst файлу:

```
Turbo Assembler    Version 4.1    04/22/24 10:42:50          Page 1
test.asm



                1                       .8086
                2  0000                .model small
                3
                4  0000                .stack 100h
                5
                6  0000                .data
                7  0000  0A 2D 2D 2D 20 4D      75+   header_msg    db 10,        '---
Multiplier by 2 ---', 10, 10, 13, '$'
                8        6C 74 69 70 6C 69     65+
                9        72 20 62 79 20 32     20+
               10        2D 2D 2D 0A 0A 0D 24
               11  001C  45 6E 74 65 72 20 79+    prompt_msg          db 'Enter   your
number (from -32 768 to 32 767): $'
               12        6F 75 72 20 6E 75     6D+
               13        62 65 72 20 28 66     72+
               14        6F 6D 20 2D 33 32     20+
               15        37 36 38 20 74 6F     20+
               16        33 32 20 37 36 37     29+
               17        3A 20 24
               18  0049  52 65 73 75 6C 74  20+    result_msg      db 'Result        is: $'
```

```
19      69 73 3A 20 24
20 0055 45 72 72 6F 72 3A 20+   overflow_err_msg   db 'Error:
Overflow happened.', 10, 13, '$'
21      4F 76 65 72 66 6C   6F+
22      77 20 68 61 70 70   65+
23      6E 65 64 2E 0A 0D   24
24 0071 45 72 72 6F 72 3A 20+   non_int_err_msg    db 'Error:You
have entered non-integer value.', 10, 13, '$'
25      59 6F 75 20 68 61   76+
26      65 20 65 6E 74 65   72+
27      65 64 20 6E 6F 6E   2D+
28      69 6E 74 65 67 65   72+
29      20 76 61 6C 75 65   2E+
30      0A 0D 24
31 009E 45 72 72 6F 72 3A 20+   incorrect_input_msg db 'Error:
Incorrect input      number.', 10, 13, '$'
32      49 6E 63 6F 72 72   65+
33      63 74 20 69 6E 70   75+
34      74 20 6E 75 6D 62   65+
35      72 2E 0A 0D 24
36 00BF 0A 0D 24                new_line        db 10, 13,      "$"
37
38 00C2 07 ?? 07*(??)           buffer                  db 7, ?, 7 dup
(?)
39 00CB 0000                    num             dw 0
40 00CD 00                      minus_sign      db 0
41
```

```
42 00CE                 .code
43 0000                 start:
44 0000  B8 0000s                       mov  ax, DGROUP
45 0003  8E D8                      mov  ds, ax
46
47                      ; print header
48 0005  BA 0000r                       lea          dx,
header_msg
49 0008  E8 0017                        call print_line
50
51                      ; readnumber
52 000B  E8 0019                        call
process_number
53
54
55                      ; print result
56 000E  BA 00BFr                       lea  dx, new_line
57 0011  E8 000E                        call print_line
```

```
58 0014  BA 0049r                          lea  dx, result_msg
59 0017  E8 0008                           call print_line
60 001A  E8 008A                           call print_number
61
62                          ; terminate program
63 001D  B8 4C00                           mov  ax, 4c00h
64 0020  CD 21                        int  21h
65
66 0022                  print_lineproc
67 0022  B4 09                        mov  ah, 09h
68 0024  CD 21                        int  21h
69 0026  C3                           ret
70 0027                  print_lineendp
71
72 0027                  process_number proc
73 0027                      process_input:
74 0027  33 C0                       xor  ax, ax
75 0029  33 D2                       xor  dx, dx
76 002B  C6 06 00CDr 00                    mov  minus_sign, 0
77
```

```
                        ; print prompt
 0030  BA 001Cr                    lea dx, prompt_msg
 0033  E8 FFEC                     call print_line

                        ; receive user input using DOS function 21h, interruption 0Ah
 0036  BA 00C2r                    lea  dx, buffer
 0039  B4 0A                       mov  ah, 0ah
 003B  CD 21                       int  21h

                        ; Process numerical input (digits 0-9)
 003D  33 C0                       xor  ax, ax
 003F  8A 0E 00C3r                 mov  cl, buffer + 1     ; Set the loop counter to the number +
                        of  characters to process
 0043  BE 0002                     mov  si, 2              ; Set the SI index to the first +
                        character of number
 0046  80 3E 00C4r 2D              cmp  buffer + 2, '-'
 004B  75 08                       jne  process_digit

                        ; Handle negative sign  (if present)
 004D  46                          inc  si                 ; Increment SI to point to the first +
                        digit after the minus sign
```

```
 99  004E  FE C9                                    dec  cl
   ; Decrement CL to account for the    +
100                          minus sign       (one less digit       to process)
101 0050  C6 06 00CDr 01                           mov  minus_sign, 1
102
103                          ; Convert ASCII to integer and check      for
errors
104 0055                     process_digit:
105 0055  8A 9C 00C2r                              mov  bl, buffer[si]
106 0059  80 FB 30                                 cmp  bl, '0'
107 005C  72 27                                 jb   handle_non_int
108 005E  80 FB 39                                 cmp  bl, '9'
109 0061  77 22                                 ja   handle_non_int
110 0063  80 EB 30                                 sub  bl, '0'
111 0066  BA 000A                                  mov  dx, 10
112 0069  F7 EA                                 imul dx
113 006B  70 0A                                 jo   handle_overflow
114 006D  03 C3                                 add  ax, bx
```

```
0006F  70 06                    jo   handle_overflow
0071  46                        inc  si
0072  E2 E1                     loop process_digit
0074  EB 1D 90                  jmp  check_sign

0077                    handle_overflow:
0077  BA 00BFr                  lea  dx, new_line
007A  E8 FFA5                   call print_line
007D  BA 0055r                  lea  dx, overflow_err_msg
0080  E8 FF9F                   call print_line
0083  EB A2                     jmp  process_input

0085                    handle_non_int:
0085  BA 00BFr                  lea  dx, new_line
0088  E8 FF97                   call print_line
008B  BA 0071r                  lea  dx, non_int_err_msg
008E  E8 FF91                   call print_line
0091  EB 94                     jmp  process_input
```

```
134 0093                    check_sign:
135 0093  80 3E 00CDr 01                cmp  minus_sign, 1
136 0098  75 02                      jne
perform_math_operation
137 009A  F7 D8                         neg  ax
138
139 009C                    perform_math_operation:
140 009C  BA 0002                       mov  dx, 2
141 009F  F7 EA                         imul dx
142 00A1  70 D4                         jo
handle_overflow
143 00A3  A3 00CBr                      mov  num, ax
144
145 00A6  C3                         ret
146 00A7                    process_number endp
147
148 00A7                    print_number proc
149 00A7  8B 1E 00CBr                   mov  bx, num
150 00AB  0B DB                         or   bx, bx
151 00AD  79 06                         jns  m1
152 00AF  B0 2D                         mov  al, '-'
153 00B1  CD 29                         int  29h
154 00B3  F7 DB                         neg  bx
155
156 00B5                    m1:
157 00B5  8B C3                         mov  ax, bx
```

```
158 00B7  33 C9                 xor  cx, cx
159 00B9  BB 000A                    mov  bx, 10
160
161 00BC                  m2:
162 00BC  33 D2                       xor  dx, dx
163 00BE  F7 F3                 div  bx
164 00C0  80 C2 30                    add  dl, '0'
165 00C3  52                    push dx
166 00C4  41                    inc  cx
167 00C5  85 C0                 test ax, ax
168 00C7  75 F3                 jnz  m2
169
170 00C9                  m3:
171 00C9  58                    pop  ax
```

```
172 00CA CD 29                          int  29h
173 00CC E2 FB                          loop m3
174
175 00CE C3                         ret
176 00CF                    print_number endp
177
178                         end start
```

Symbol Table

Symbol Name	Type	Value
??DATE	Text	"04/22/24"
??FILENAME	Text	"test "
??TIME	Text	"10:42:50"
??VERSION	Number	040A
@32BIT	Text	0
@CODE	Text	_TEXT
@CODESIZE	Text	0
@CPU	Text	0101H
@CURSEG	Text	_TEXT
@DATA	Text	DGROUP
@DATASIZE	Text	0
@FILENAME	Text	TEST
@INTERFACE	Text	000H
@MODEL	Text	2
@STACK	Text	DGROUP
@WORDSIZE	Text	2
BUFFER	Byte	DGROUP:00C2

CHECK_SIGN Near _TEXT:0093

HANDLE_NON_INT Near _TEXT:0085

HANDLE_OVERFLOW Near _TEXT:0077

HEADER_MSG Byte DGROUP:0000

INCORRECT_INPUT_MSG Byte DGROUP:009E

M1 Near _TEXT:00B5

M2 Near _TEXT:00BC

M3 Near _TEXT:00C9

MINUS_SIGN Byte DGROUP:00CD

NEW_LINE Byte DGROUP:00BF

NON_INT_ERR_MSG Byte DGROUP:0071

NUM Word DGROUP:00CB

OVERFLOW_ERR_MSG Byte DGROUP:0055

PERFORM_MATH_OPERATION Near _TEXT:009C

PRINT_LINE Near _TEXT:0022

PRINT_NUMBER Near _TEXT:00A7

PROCESS_DIGIT Near _TEXT:0055

PROCESS_INPUT Near _TEXT:0027

PROCESS_NUMBER Near _TEXT:0027

PROMPT_MSG Byte DGROUP:001C

RESULT_MSG Byte DGROUP:0049

START Near _TEXT:0000

Groups & Segments Bit Size Align Combine Class

```
DGROUP                  Group
 STACK                  16  0100 Para     Stack          STACK
 _DATA                  16  00CE Word     Public  DATA
_TEXT                   16  00CF Word     Public  CODE
```

Вміст .map файлу:

```
 Start  Stop   Length Name             Class

 00000H 000CEH 000CFH _TEXT           CODE
 000D0H 0019DH 000CEH _DATA            DATA
 001A0H 0029FH 00100H STACK           STACK

 Program entry point at 0000:0000
```

Висновок:

Під час виконання комп'ютерного практикуму я розробив програму, що використовує дві процедури для обміну даними:

1. Перша процедура вводить і перетворює ціле число, а потім виконує математичну операцію над ним (в моєму випадку, множення на 2).
2. Друга процедура перетворює отриманий результат в рядок і виводить його на екран.

Особливу увагу було приділено захисту від некоректного введення вхідних даних. Програма оброблює нецифрові символи, переповнення та некоректно введене число. Ця робота демонструє ефективне використання процедур для обміну даними і обробки помилок, що є важливими аспектами розробки програмного забезпечення. Завдяки цьому, програма може бути легко модифікована або розширена для виконання інших завдань або обробки інших типів даних.