

Міністерство освіти і науки України
Національний університет „Львівська політехніка”
Кафедра “Спеціалізованих комп’ютерних систем”



Звіт
до лабораторної роботи № 4
з дисципліни
Системне програмування

Дослідження роботи команд переходів. Програмування задач з використанням
алгоритмів розгалуження.

Варіант: 19

Виконав:

ст. гр. КІ-207

Шаповал Віталій

Перевірив:

Асистент катедри ЕОМ

Максимів М. Р.

Мета: освоїти використання команд порівняння, умовного та безумовного переходів. Набути вміння використовувати арифметичні команди над знаковими даними та команди логічних операцій.

Завдання:

1. Створити *.exe програму, яка реалізовує обчислення, заданого варіантом виразу. Вхідні дані слід вважати цілими числами зі знаком, розміром один байт. Результат обчислення виразу повинен записуватися у пам'ять. Уникнути випадку некоректних обчислень при діленні на нуль та при переповненні розрядної сітки (вивести відповідне текстове повідомлення).
2. За допомогою Debug, відслідкувати правильність виконання програми (продемонструвати результати проміжних та кінцевих обчислень) та проаналізувати отримані результати для різних вхідних даних.
3. Скласти звіт про виконану роботу з приведенням тексту програми та коментарів до неї.
4. Дати відповідь на контрольні запитання.

Варіант: 19

19	$X = \begin{cases} -4 * b/a + 2, & a > b, \\ -57, & a = b, \\ (a - b)/b, & a < b. \end{cases}$
-----------	--

Виконання:

Код:

```
.686
.model flat, stdcall
option casemap:none
include \masm32\include\windows.inc
include \masm32\include\kernel32.inc
include \masm32\include\user32.inc
includelib \masm32\lib\kernel32.lib
includelib \masm32\lib\user32.lib

.data

A    db  -50
B    db  -10
X    dw   0

Hello db 13, 10
```

```

        db ' X = (a-b)/b      if a < b', 13, 10
        db ' X = -57          if a = b', 13, 10
        db ' X = -4b/a        if a > b', 13, 10
Operands  db 13, 10, ' A =      B =      ', 13, 10
NumberOfCharsToWrite_Hello dd $-Hello
Error      db 13, 10, ' Error - divide by zero!', 13, 10
NumberOfCharsToWrite_Error dd $-Error
Result     db 13, 10, ' X =          ', 13, 10
NumberOfCharsToWrite_Result dd $-Result
format db '%hd', 0
hConsoleOutput dd 0
NumberOfCharsWritten dd 0

```

```

.code
start:
;Вивід повідомлення Hello
mov al, A
cbw
push ax
push offset format
push offset [Operands+8]
call wsprintfA
mov al, B
cbw
push ax
push offset format
push offset [Operands+17]
call wsprintfA
push -11
call GetStdHandle
mov hConsoleOutput, eax
push 0
push offset NumberOfCharsWritten
push NumberOfCharsToWrite_Hello
push offset Hello
push hConsoleOutput
call WriteConsoleA

```

```

;перевірка на рівність A та B
mov al, A
cmp al, B
jne A_ne_B
mov X, -57
jmp Output_Result

```

```

A_ne_B:
jg A_g_B
cmp B, 0
je Output_Error
;Обчислення X при a<b
mov al, A
sub al, B
cbw

```

```
idiv B
mov X, ax
jmp Output_Result

A_g_B:
cmp A, 0
je Output_Error
;обчислення X при a>b
mov al, B
cbw
mov bx, ax
mov ax, -4
imul ax, bx
idiv A
mov X, ax
jmp Output_Result

;вивід результату
Output_Result:
push X
push offset format
push offset [Result+8]
call wsprintfA
push offset NumberOfCharsWritten
push NumberOfCharsToWrite_Result
push offset Result
push hConsoleOutput
call WriteConsoleA
jmp exit

;вивід повідомлення про ділення на 0
Output_Error:
push offset NumberOfCharsWritten
push NumberOfCharsToWrite_Error
push offset Error
push hConsoleOutput
call WriteConsoleA
jmp exit

;вихід з програми
exit:
push 0
call ExitProcess
end start
```

Результат виконання:

Дослідження правильності виконання програми для $A < B$

```
55 ;перевірка на рівність A та B
56 mov al, A
57 cmp al, B
58 jne A_ne_B
59 mov X, -57
60 jmp Output_Result
61
62 A_ne_B:
63 jg A_g_B
64 cmp B, 0
65 je Output_Error
66 ;Обчислення X при a<b
67 mov al, A
68 sub al, B
69 cbw
70 idiv B
71 mov X, ax
72 jmp Output_Result
73
```

Скрин 1. Розставлені брекпойнти

Після виконання `cmp` вміст регістру EFL змінився на

EFL = 00000287 = 0000 0000 0000 0000 0000 0010 1000 0111,

`jne` перевіряє прапорець ZF, якому відповідає 6-ий біт EFL, який рівний 0, отже A і B – нерівні, тому `jne` переміщає нас до `A_ne_B`

`jg` перевіряє прапорець SF, якому відповідає 7-ий біт EFL, який рівний 1, отже прапорець знаходиться в стані NG, тому $A < B$, і `jg` ігнорується.

```
64 cmp B, 0
65 je Output_Error ≤ 4ms elapsed
66 ;Обчислення X при a<b
```

Скрин 2. Перевірка на $B == 0$

Після виконання `cmp B, 0` вміст регістру EFL змінився на

EFL = 00000286 = 0000 0000 0000 0000 0000 0010 1000 0110,

`je` перевіряє прапорець ZF, який рівний 0, отже $B \neq 0$, тому `je` ігнорується.

Name	Value
A	206 'O'
B	246 'ц'
al	206 'O'
ax	206
X	0

```
66 ;Обчислення X при a<b
67 mov al, A
68 sub al, B ≤ 2ms elapsed
69 cbw
70 idiv B
71 mov X, ax
```

Скрин 3. Поміщення A в регістр al

67	mov al, A	Name	Value
68	sub al, B	A	206 'O'
69	cbw ≤ 3ms elapsed	B	246 'ц'
70	idiv B	al	216 'Ш'
71	mov X, ax	ax	216
72	jmp Output_Result	X	0

Скрин 4. А – В

67	mov al, A	Name	Value
68	sub al, B	A	206 'O'
69	cbw	B	246 'ц'
70	idiv B ≤ 2ms elapsed	al	216 'Ш'
71	mov X, ax	ax	65496
72	jmp Output_Result	X	0

Скрин 5. Розширення al до ax

67	mov al, A	Name	Value
68	sub al, B	A	206 'O'
69	cbw	B	246 'ц'
70	idiv B	al	4 '\x4'
71	mov X, ax	ax	4
72	jmp Output_Result ≤ 2ms	X	4

Скрин 6. А / В та поміщення результату з регістру ax у змінну X

```
PS C:\Users\UsVeR\Documents\MASM\lab4> .\lab4.exe

X = (a-b)/b    if a < b
X = -57        if a = b
X = -4b/a      if a > b

A = -50 B = -10

X = 4
```

Скрин 7. Результат виконання програми

Перевірка правильності виконання для A == B

```
55 ;перевірка на рівність A та B
56 mov al, A
57 cmp al, B
58 jne A_ne_B
59 mov X, -57
60 jmp Output_Result
61
62 A_ne_B:
```

Скрин 8. Розставлені брекпойнти

55	;перевірка на рівність A та B	Name	Value
56	mov al, A	A	246 'ц'
57	cmp al, B ≤ 1ms elapsed	B	246 'ц'
58	jne A_ne_B	al	246 'ц'
59	mov X, -57	X	0
60	jmp Output_Result	Add item to watch	

Скрин 9. Переміщення A до регістру al

55	;перевірка на рівність A та B
56	mov al, A
57	cmp al, B
58	jne A_ne_B ≤ 2ms elapsed
59	mov X, -57
60	jmp Output_Result

Скрин 10. A – B та внесення результату до регістру прапорців, без зміни змінних

Після виконання директиви `cmp al, B` вміст регістру EFL змінився на:

EFL = 00000246 = 0000 0000 0000 0000 0000 0010 0100 0110

`jne` перевіряє прапорець ZF, якому відповідає 6-ий біт EFL, який рівний 1, отже A і B – рівні, тому `jne` не переміщає нас до `A_ne_B`:

55	;перевірка на рівність A та B
56	mov al, A
57	cmp al, B
58	jne A_ne_B
59	mov X, -57 ≤ 3ms elapsed
60	jmp Output_Result

Скрин 11. `Jne` не спрацьовує

55	;перевірка на рівність A та B	Name	Value
56	mov al, A	A	246 'ц'
57	cmp al, B	B	246 'ц'
58	jne A_ne_B	al	246 'ц'
59	mov X, -57	X	65479
60	jmp Output_Result ≤ 1ms elapsed	Add item to watch	

Скрин 12. Поміщення до результату X константи -57

```

Microsoft Visual Studio Debug Console

X = (a-b)/b    if a < b
X = -57        if a = b
X = -4b/a      if a > b

A = -10 B = -10

X = -57

```

Скрин 13. Результат виконання програми

Перевірка правильності виконання програми для $A > B$:

```
55 ;перевірка на рівність A та B
56 mov al, A
57 cmp al, B
58 jne A_ne_B
59 mov X, -57
60 jmp Output_Result
61
62 A_ne_B:
63 jg A_g_B
64 cmp B, 0
65 je Output_Error
```

Скрин 14. Розставлені брекпойнти

```
74 A_g_B:
75 cmp A, 0
76 je Output_Error
77 ;обчислення X при a>b
78 mov al, B
79 cbw
80 mov bx, ax
81 mov ax, -4
82 imul ax, bx
83 idiv A
84 mov X, ax
85 jmp Output_Result
86
87 ;вивід результату
```

Скрин 15. Розставлені брекпойнти

```
55 ;перевірка на рівність A та B
56 mov al, A
57 cmp al, B
58 jne A_ne_B
59 mov X, -57
60 jmp Output_Result
61
```

Скрин 16. A-B та занесення результату в регістр, без впливу на змінні

Після виконання `cmp` вміст регістру EFL змінився на

EFL = 00000217 = 0000 0000 0000 0000 0000 0010 0001 0111

`jne` перевіряє прапорець ZF, якому відповідає 6-ий біт EFL, який рівний 0, отже A і B – нерівні, тому `jne` переміщає нас до `A_ne_B`

```
61
62 A_ne_B:
63 jg A_g_B ≤ 1ms elapsed
64 cmp B, 0
65 je Output_Error
66 ;Обчислення X при a<b
```

Скрин 17. Jne спрацьовує та переміщає до A_ne_B

јг перевіряє прапорець SF, якому відповідає 7-ий біт EFL, який рівний 0, отже прапорець знаходиться в стані NG, тому $A > B$, і јг перекидає програму до A_g_B.

```

74      A_g_B:
75      cmp A, 0
76      jg Output_Error ≤ 2ms elapsed
77      ;обчислення X при a>b
78      mov al, B
79      cbw
80      mov bx, ax
81      mov ax, -4
82      imul ax, bx
83      idiv A
84      mov X, ax
85      jmp Output_Result

```

Скрин 18. Јг спрацьовує та переміщає до A_g_B

Після виконання `cmp A, 0` вміст регістру EFL змінився на

$EFL = 00000202 = 0000\ 0000\ 0000\ 0000\ 0000\ 0010\ 0000\ 0010$,

је перевіряє прапорець ZF, який рівний 0, отже $A \neq 0$, тому је ігнорується.

```

77      ;обчислення X при a>b
78      mov al, B ≤ 3ms elapsed
79      cbw
80      mov bx, ax
81      mov ax, -4
82      imul ax, bx
83      idiv A
84      mov X, ax
85      jmp Output_Result

```

Watch 1

Search (Ctrl+E) 🔍 Search Depth:

Name	Value
A	4 '\x4'
B	255 'я'
al	4 '\x4'
X	0
Add item to watch	

Скрин 19. Је не спрацьовує, програма продовжує послідовне виконання

```

77      ;обчислення X при a>b
78      mov al, B
79      cbw ≤ 1ms elapsed
80      mov bx, ax
81      mov ax, -4
82      imul ax, bx
83      idiv A
84      mov X, ax
85      jmp Output_Result

```

Watch 1

Search (Ctrl+E) 🔍 Search D

Name	Value
A	4 '\x4'
B	255 'я'
al	255 'я'
X	0
Add item to watch	

Скрин 20. Поміщення B в регістр al

```

77      ;обчислення X при a>b
78      mov al, B
79      cbw
80      mov bx, ax ≤ 2ms elapsed
81      mov ax, -4
82      imul ax, bx
83      idiv A
84      mov X, ax
85      jmp Output_Result

```

Search (Ctrl+E) 🔍 Search Dep

Name	Value
A	4 '\x4'
B	255 'я'
al	255 'я'
X	0
ax	65535
bx	28672
Add item to watch	

Скрин 21. Розширення al до ax

77	;обчислення X при a>b	Name	Value
78	mov al, B	A	4 '\x4'
79	cbw	B	255 'я'
80	mov bx, ax	al	255 'я'
81	mov ax, -4 ≤ 1ms elapsed	X	0
82	imul ax, bx	ax	65535
83	idiv A	bx	65535
84	mov X, ax	Add item to watch	

Скрин 22. Переміщення вмісту ax до bx

77	;обчислення X при a>b	Search (Ctrl+E) Search Depth:	
78	mov al, B	Name	Value
79	cbw	A	4 '\x4'
80	mov bx, ax	B	255 'я'
81	mov ax, -4	al	252 'ь'
82	imul ax, bx ≤ 2ms elapsed	X	0
83	idiv A	ax	65532
84	mov X, ax	bx	65535
85	jmp Output_Result	Add item to watch	

Скрин 23. Поміщення константи -4 до ax

77	;обчислення X при a>b	Search (Ctrl+E) Search Depth:	
78	mov al, B	Name	Value
79	cbw	A	4 '\x4'
80	mov bx, ax	B	255 'я'
81	mov ax, -4	al	4 '\x4'
82	imul ax, bx	X	0
83	idiv A ≤ 2ms elapsed	ax	4
84	mov X, ax	bx	65535
85	jmp Output_Result	Add item to watch	

Скрин 24. -4 * B

77	;обчислення X при a>b	Search (Ctrl+E) Search Depth:	
78	mov al, B	Name	Value
79	cbw	A	4 '\x4'
80	mov bx, ax	B	255 'я'
81	mov ax, -4	al	1 '\x1'
82	imul ax, bx	X	0
83	idiv A	ax	1
84	mov X, ax ≤ 1ms elapsed	bx	65535
85	jmp Output_Result	Add item to watch	

Скрин 25. Вміст ax ділиться на A

77	;обчислення X при a>b	Name	Value
78	mov al, B	A	4 '\x4'
79	cbw	B	255 'я'
80	mov bx, ax	al	1 '\x1'
81	mov ax, -4	X	1
82	imul ax, bx	ax	1
83	idiv A	bx	65535
84	mov X, ax	Add item to watch	
85	jmp Output_Result ≤ 1ms elapsed		

Скрин 26. Переміщення вмісту ax до X

```
X = (a-b)/b    if a < b
X = -57        if a = b
X = -4b/a      if a > b

A = 4    B = -1

X = 1
```

Скрин 27. Результат виконання програми

Висновок: виконавши цю лабораторну роботу я освоїв використання команд порівняння, умовного та безумовного переходів та набув вміння використовувати арифметичні команди над знаковими даними та команди логічних операцій.