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:

```
.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

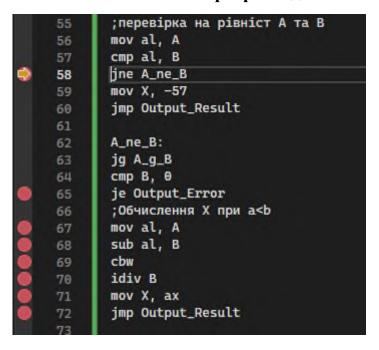
Hello db 13, 10
```

```
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
           db 13, 10, ' Error - divide by zero!', 13, 10
NumberOfCharsToWrite_Error dd $-Error
           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 рівніст А та В
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 правильности виконання програми для А < В

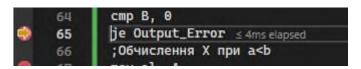


Скрин 1. Розставлені брекпойнти

Після виконання стр вміст регістру EFL змінився на EFL = 00000287 = 0000 0000 0000 0000 0000 0010 1000 0111,

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

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



Скрин 2. Перевірка на B == 0

Після виконання стр B, 0 вміст регістру EFL змінився на EFL = 00000286 = 0000 0000 0000 0000 0010 1000 0110,

је перевіряє прапорець ZF, який рівний 0, отже B != 0, тому је ігнорується.

66	;Обчислення X при a <b< th=""><th>Name</th><th>Value</th></b<>	Name	Value
67	mov al, A	n A	206 'O'
68	sub al, B ≤ 2ms elapsed	B	246 'ц'
69	cbw	al	206 'O'
70	idiv B	ax	206
71	mov X, ax	ín X	0

Скрин 3. Поміщення A в регістр al

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

Скрин 4. А – В

67	mov al, A	Name	Value
68	sub al, B		206 'O'
69	cbw	🔂 B	246 'ц'
9 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 ах

67	mov al, A	Name	Value
68	sub al, B	A	206 'O'
69	cbw	B	246 'ц'
9 70	idiv B	al	4 '\x4'
71	mov X, ax	Ø ax	4
72	jmp Output_Result ≤2	2ms 📦 X	4

Скрин 6. А / В та поміщення результату з регістру ах у змінну Х

Скрин 7. Результат виконання програми

Перевірка павильности виконання для А == В

```
55 ;перевірка на рівніст A та B

56 mov al, A

57 cmp al, B

58 jne A_ne_B

60 jmp Output_Result

61 A_ne_B:
```

Скрин 8. Розставлені брекпойнти

55	;перевірка на рівніст А та В	Name	Value
6	mov al, A	<u> </u>	246 'ц'
	cmp al, B ≤ 1ms elapsed	∰ B	246 'ц'
58	jne A_ne_B	Ø al	246 'ц'
9 59	mov X, −57		0
6 9	jmp Output_Result	Add item to watch	

Скрин 9. Переміщення A до регістру al

55	;перевірка на рівніст А та В
9 56	mov al, A
9 57	cmp al, B
€ 58	jne A_ne_B ≤ 2ms elapsed
59	mov X, −57
60	jmp Output_Result

Скрин 10. А – В та внесення результату до регістру прапорців, без зміни змінних

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

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

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

```
55 ;перевірка на рівніст А та В

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. Јпе не спрацьовує

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

Скрин 12. Поміщення до результату Х 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 правильности виконання програми для A > B:

```
55 ;перевірка на рівніст A та B
mov al, A
cmp al, B
jne A_ne_B
mov X, -57
jmp Output_Result
61
62 A_ne_B:
jg A_g_B
cmp B, 0
je Output_Error
```

Скрин 14. Розставлені брекпойнти

```
74
       A_g_B:
       стр А, О
       je Output_Error
       ;обчислення Х при а>b
77
78
       mov al, B
       cbw
79
80
       mov bx, ax
       mov ax, -4
       imul ax, bx
       idív A
83
       mov X, ax
84
       jmp Output_Result
86
       ;вивід результату
```

Скрин 15. Розставлені брекпойнти

```
55 ;перевірка на рівніст А та В
56 mov al, А
57 cmp al, В
јпе А_пе_В
59 mov X, −57
60 jmp Output_Result
```

Скрин 16. А-В та занесення результату в регістр, без впливу на змінні

Після виконання стр вміст регістру EFL змінився на

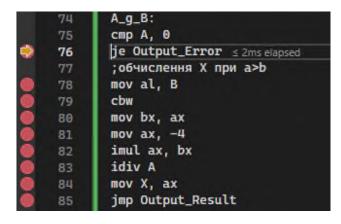
```
EFL = 00000217 = 0000 0000 0000 0000 0010 0011 0111
```

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

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

Скрин 17. Јпе спрацьовує та переміщає до А_ne_В

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



Скрин 18. Је спрацьовує та переміщає до А_g_В

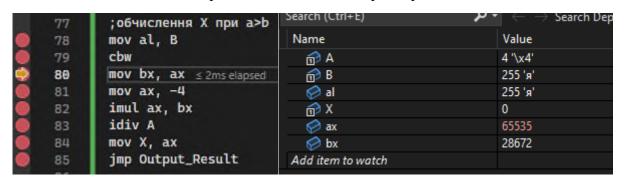
Після виконання стр A, 0 вміст регістру EFL змінився на EFL = 00000202 = 0000 0000 0000 0000 0010 0000 0010, је перевіряє прапорець ZF, який рівний 0, отже A !=0, тому је ігнорується.

77	;обчислення X при a>b	Watch 1	
78 79	mov al, B ≤3ms elapsed cbw	Search (Ctrl+E)	$oldsymbol{ ho}_{oldsymbol{ au}} \leftarrow ightarrow ext{Search Depth:}$
80	mov bx, ax	Name	Value
81	mov ax, -4	n A	4 '\x4'
82	imul ax, bx	∰ B	255 'я'
83	idiv A	al	4 '\x4'
84	mov X, ax	í X	0
85	jmp Output_Result	Add item to watch	

Скрин 19. Је не спрацьовує, програма продовжує послідовне виконання

77	77 ;обчислення X при a>b 78 mov al, B 79 cbw ≤1mselapsed	Watch 1		
79		Search (Ctrl+E)	$oldsymbol{ ho}_{ullet} \leftarrow ightarrow Search ($	
80	mov bx, ax	Name	Value	
81	mov ax, -4	n A	4 '\x4'	
82	imul ax, bx	on B	255 'я'	
	idiv A mov X, ax jmp Output_Result	Ø al	255 'я'	
		Add item to watch	0	
85		7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7		

Скрин 20. Поміщення B в регістр al



Скрин 21. Розширення al до ax

77	;обчислення X при a>b	Name	Value
78	mov al, B	<u> </u>	4 '\x4'
79	cbw	f B	255 'я'
9 80	mov bx, ax	🥏 al	255 'я'
9 81	mov ax, -4 ≤ 1ms elapsed	of X	0
82	imul ax, bx		65535
83	idîv A	bx	65535
84	mov X, ax	Add item to watch	

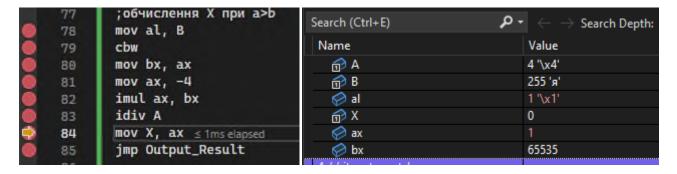
Скрин 22. Переміщення вмісту ах до bx

77	;обчислення X при a>b mov al, B	Search (Ctrl+E)	$ullet \ \leftarrow \ o$ Search Depth:
79	cbw	Name	Value
89	mov bx, ax	<u> </u>	4 '\x4'
81	mov ax, -4	⊕ B	255 'я'
₿ 82	imul ax, bx ≤ 2ms elapsed	<i>⊜</i> al	252 'ь'
83	idiv A	n X	0
		🥟 ax	65532
84	mov X, ax	🔗 bx	65535
85	jmp Output_Result	Add item to watch	

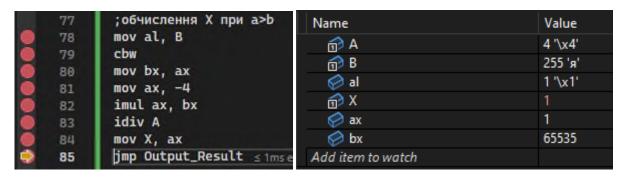
Скрин 23. Поміщення константи -4 до ах

77	;обчислення X при a>b	Search (Chi+L)	Sealch
78	mov al, B	Name	Value
79	cbw	o A	4 '\x4'
80	mov bx, ax	B ☐ B	255 'я'
81	mov ax, -4		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 * В



Скрин 25. Вміст ах ділиться на А



Скрин 26. Переміщення вмісту ах до Х

```
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.