

Міністерство освіти і науки, молоді та спорту України
Національний університет “Львівська політехніка”

Кафедра ЕОМ



Звіт

з лабораторної роботи №4 з дисципліни:
“Системного програмування”

на тему: «Дослідження роботи команд переходів.

Програмування задач з використанням алгоритмів розгалуження.»

Варіант: № 24.

Виконав:

ст. групи КІ-203

Ширий Б. І.

Перевірила:

стар. вик. кафедри ЕОМ

Ногаль М. В.

МЕТА РОБОТИ:

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

ЗАВДАННЯ:

УМОВА 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.

A, B - знакові операнди, розміром один байт.

Для 24 варіанту наведена ось такий вираз:

$$X = \begin{cases} \frac{A}{B} - 1, & A < B, \\ 10 - A, & A = B, \\ \frac{-2 \cdot B - 5}{A}, & A > B. \end{cases}$$

ВИКОНАННЯ:

Отож, написав текст Асемблера згідно свого варіанту, який навів у лістингу 1.

Лістинг 1. Текст програми.

```
.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 10
B db 10
X dw 0
Hello db 13, 10, ' X = A/B - 1 if A < B', 13, 10
      db ' X = 10 - A if A = B', 13, 10
      db ' X = (-2 * B - 5)/A if A > B', 13, 10
Operands db 13, 10, ' A =      B =      ', 13, 10
NumberOfCharsToWrite_Hello dd $-Hello
Error db 13, 10, ' Error - Devided 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
; X = A == B ? (10 - A) : goto AnotB
    movsx    edx, A      ; edx = A
    movsx    eax, B      ; eax = B
    cmp      edx, eax    ; edx == eax
    jne      AnotB       ; якщо ні, то goto AnotB
    movsx    ecx, A      ; ecx = A
    mov      edx, 10     ; edx = 10
    sub      edx, ecx     ; edx = 10 - A
    jmp      AnotB
```

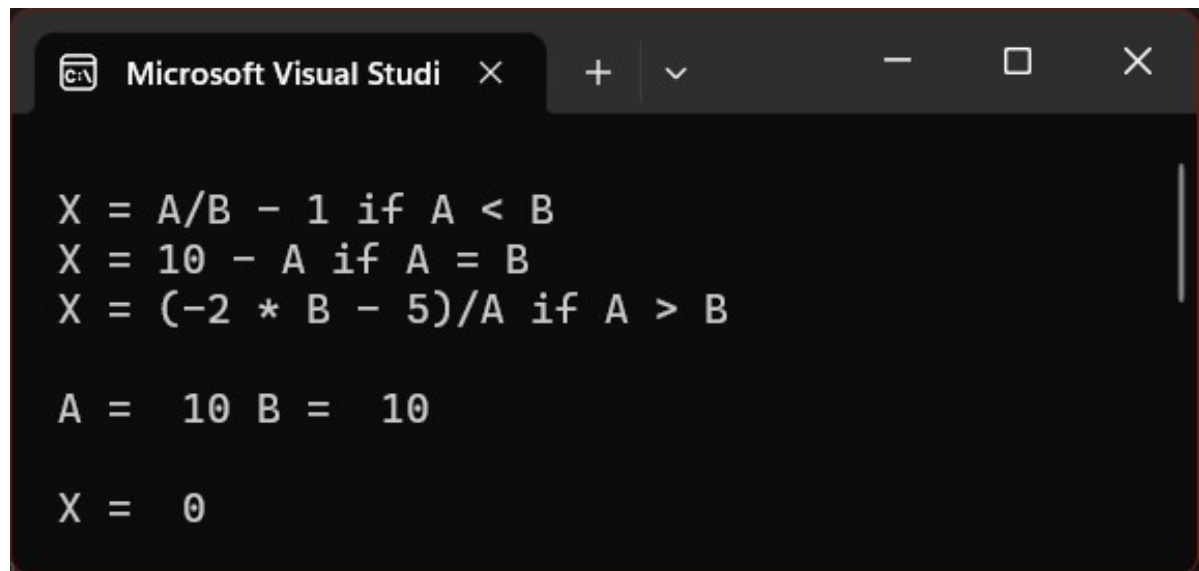
```

        mov     X, dx      ; X = dx
        jmp     GoToOutput ; goto GoToOutput
AnotB:
; X = A < B ? (A/B + 1) : goto AnotLessB
        movsx   eax, A      ; eax = A
        movsx   ecx, B      ; ecx = B
        cmp     eax, ecx    ; eax < ecx
        jge     AnotLessB   ; якщо ні, то goto AnotLessB
        cmp     ecx, 0      ; ecx == 0
        je      OutputError ; якщо так, то goto OutputError
        movsx   eax, A      ; eax = A
        movsx   ecx, B      ; ecx = B
        cdq
        idiv    ecx         ; A / B
        sub     eax, 1      ; A/B + 1
        mov     X, ax       ; X = ax
        jmp     GoToOutput  ; goto GoToOutput
AnotLessB:
; X = A < B ? ((-2 * B - 5)/A) : goto GoToOutput
        movsx   eax, A      ; eax = A
        movsx   ecx, B      ; ecx = B
        cmp     eax, ecx    ; eax > ecx
        jle     GoToOutput  ; якщо ні, то goto GoToOutput
        cmp     eax, 0      ; eax == 0
        je      OutputError ; якщо так, то goto OutputError
        movsx   edx, B      ; edx = B
        imul    eax, edx, -2; -2 * B
        sub     eax, 5      ; -2 * B - 5
        movsx   ecx, A      ; ecx = A
        cdq
        idiv    ecx         ; (-2 * B - 5) / A
        mov     X, ax       ; X = ax
;вивід результату
GoToOutput:
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 нуль
OutputError:
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

```

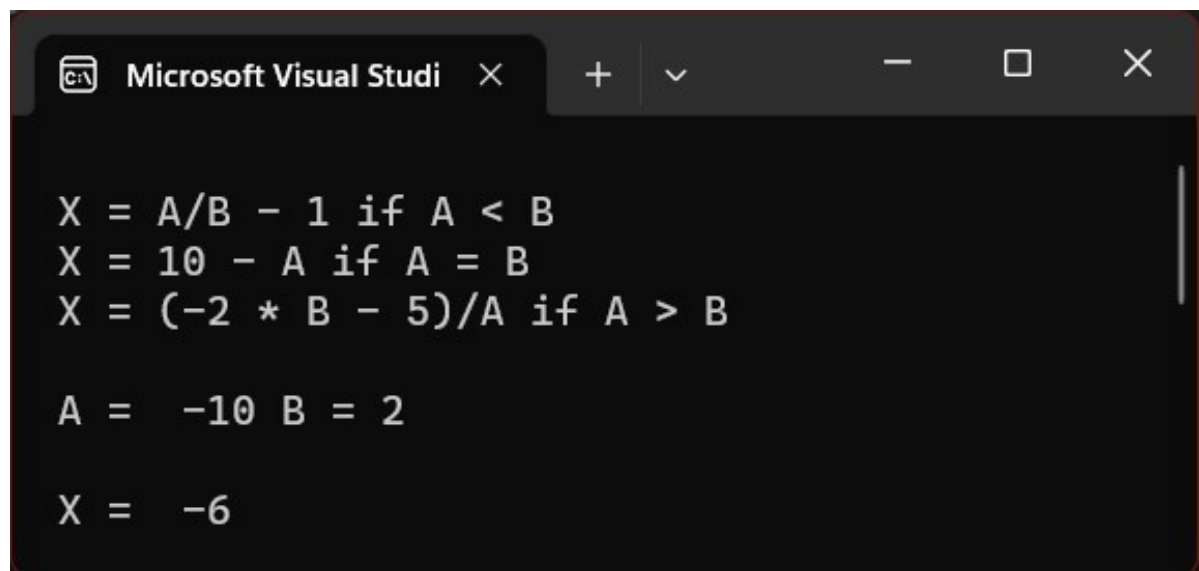
ВИВІД КОНСОЛІ:

Отож, запуслав програму з різними вхідними даними, щоб перевірити її на коректність. Вивід консолі зображений на рисунках 1, 2, 3, 4 та 5.



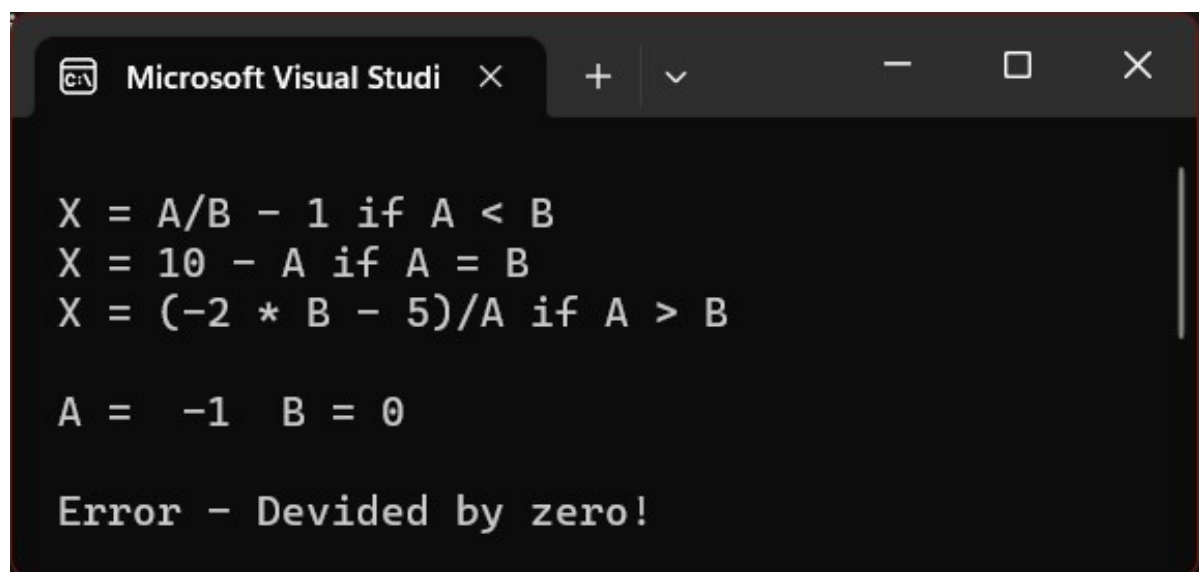
```
Microsoft Visual Studi X + v - □ X  
  
X = A/B - 1 if A < B  
X = 10 - A if A = B  
X = (-2 * B - 5)/A if A > B  
  
A = 10 B = 10  
  
X = 0
```

Рис. 1. Вивід, коли змінні рівні.



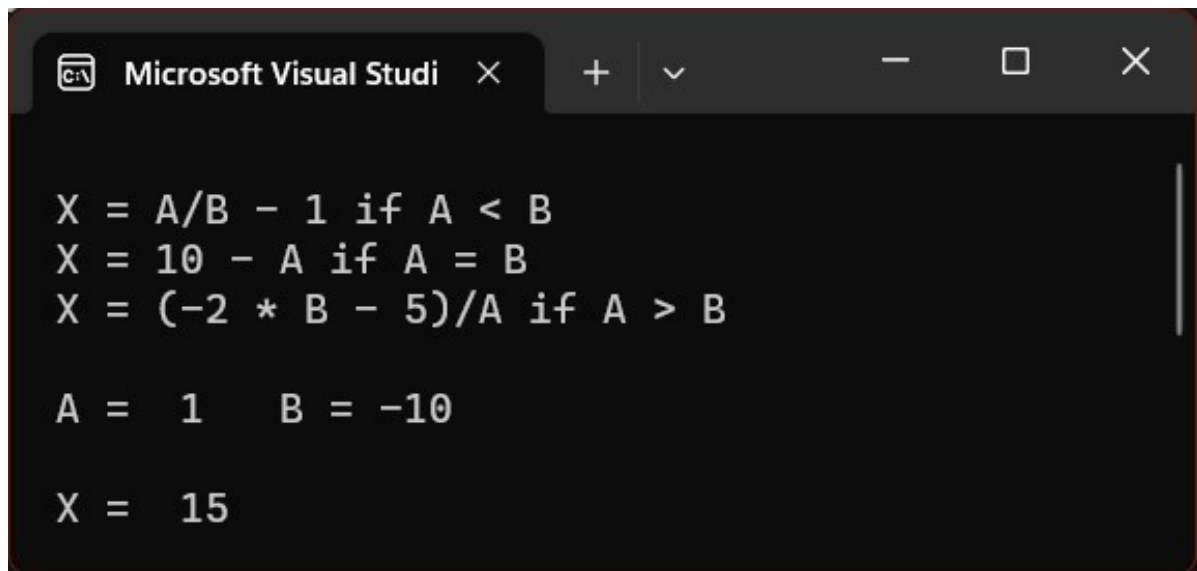
```
Microsoft Visual Studi X + v - □ X  
  
X = A/B - 1 if A < B  
X = 10 - A if A = B  
X = (-2 * B - 5)/A if A > B  
  
A = -10 B = 2  
  
X = -6
```

Рис. 2. Вивід, коли змінна А менша за змінну В.



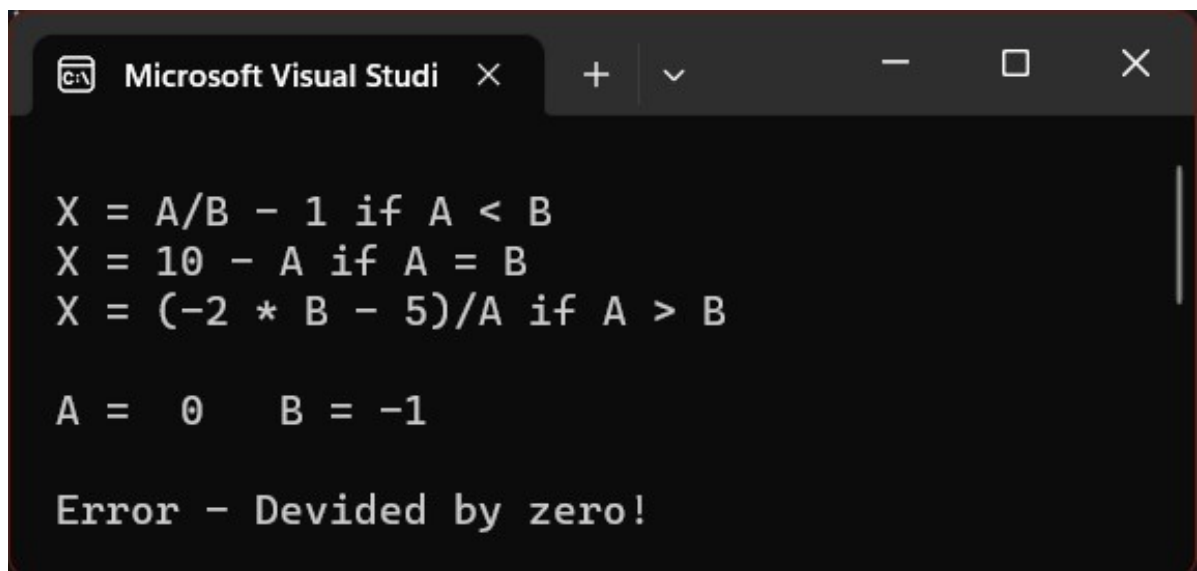
```
Microsoft Visual Studi X + v - □ X  
  
X = A/B - 1 if A < B  
X = 10 - A if A = B  
X = (-2 * B - 5)/A if A > B  
  
A = -1 B = 0  
  
Error - Devided by zero!
```

Рис. 3. Вивід, коли змінна А менша за змінну В з діленням на нуль.



```
Microsoft Visual Studi X + - □ X  
  
X = A/B - 1 if A < B  
X = 10 - A if A = B  
X = (-2 * B - 5)/A if A > B  
  
A = 1 B = -10  
  
X = 15
```

Рис. 4. Вивід, коли змінна A більша за змінну B.



```
Microsoft Visual Studi X + - □ X  
  
X = A/B - 1 if A < B  
X = 10 - A if A = B  
X = (-2 * B - 5)/A if A > B  
  
A = 0 B = -1  
  
Error - Devided by zero!
```

Рис. 5. Вивід, коли змінна A більша за змінну B з діленням на нуль.

ВИСНОВКИ:

Під час виконання лабораторної роботи успішно освоїв використання команд порівняння, умовного та безумовного переходів, а також набув навичок використання арифметичних команд над знаковими даними та команд логічних операцій. Застосування цих команд дозволяє ефективно програмувати різні задачі, що вимагають розгалуження та прийняття рішень на основі різних перевірок. Отже, виконання лабораторної роботи було корисним для розвитку навичок програмування на мові Асемблер та дозволило поглибити мої знання з цієї області.

КОНТРОЛЬНІ ПИТАННЯ:

ПОНЯТТЯ ПРО МАШИННІ КОДИ, БАЙТ СПОСОБУ АДРЕСАЦІЇ ТА СПОСОБИ АДРЕСАЦІЇ.

Машинний код - це набір інструкцій, які комп'ютер може виконувати безпосередньо. Ці інструкції записуються у вигляді двійкових чисел, які комп'ютер може розуміти. Байт - це мінімальна адресована одиниця пам'яті, яка зазвичай складається з 8 бітів.

Способи адресації вказують, як операнд (дані, що обробляються в інструкції) знаходяться у пам'яті. Є кілька типів способів адресації, які наведені у таблиці:

| Спосіб | Пояснення |
|----------------------|---|
| Безпосередній | Операнд вказується прямо в інструкції. |
| Неперервна | Операнд знаходиться за фіксованою адресою в пам'яті. |
| Регістрова | Операнд знаходиться в регістрі процесора. |
| Пряма | Операнд вказується за допомогою адреси в пам'яті. |
| Операційна | Операнд знаходиться в пам'яті за допомогою обчислень, які виконує процесор. |

Кожен спосіб адресації має свої переваги та недоліки, і їх використовують у залежності від потреб програми та архітектури процесора.

ЯКІ КОМАНДИ БЕЗУМОВНИХ ПЕРЕХОДІВ ВИ ЗНАЄТЕ?

У більшості архітектур процесорів існують команди безумовного переходу, які дозволяють програмі перейти до іншої частини коду без будь-яких обмежень або умов. Деякі з таких команд, навів у таблиці:

| Команда | Пояснення |
|-------------|--|
| JMP | Команда безумовного переходу, що дозволяє перейти до іншої частини коду, вказаної в операнді. |
| CALL | Команда використовується для виклику підпрограми, яка зазвичай розташовується в іншій частині коду. При виконанні команди CALL адреса наступної інструкції зберігається в стеку, щоб після завершення підпрограми програма могла продовжити виконання з цієї адреси. |
| RET | Команда використовується для повернення з підпрограми. Коли ця команда виконується, адреса наступної інструкції береться з верхнього рядка стеку і переходиться до цієї адреси. |

JUMP

Команда використовується в асемблері для безумовного переходу до мітки в іншій частині коду. Мітки використовуються для позначення конкретних рядків коду.

Ці команди безумовного переходу дозволяють програмі виконувати різні дії залежно від умов та даних, що оброблюються.

ЯКІ КОМАНДИ УМОВНИХ ПЕРЕХОДІВ Є В МОВІ АСЕМБЛЕР?

У мові Асемблер існують команди умовного переходу, які дозволяють програмі приймати рішення на основі значень, що знаходяться в регістрах або в пам'яті.

Приклади команд навів у таблиці:

| Команда | Умова |
|------------|--|
| <i>JE</i> | Якщо два значення рівні |
| <i>JNE</i> | Якщо два значення не рівні. |
| <i>JL</i> | Якщо перше значення менше за друге. |
| <i>JLE</i> | Якщо перше значення менше або дорівнює другому. |
| <i>JG</i> | Якщо перше значення більше за друге. |
| <i>JGE</i> | Якщо перше значення більше або дорівнює другому. |
| <i>JA</i> | Якщо перше значення більше за друге (для беззнакових чисел). |
| <i>JAE</i> | Якщо перше значення більше або дорівнює другому (для беззнакових чисел). |

Ці команди дозволяють програмі перевіряти різні умови та приймати рішення про подальші дії на основі результатів цих перевірок.

ЯКА РІЗНИЦЯ В ОРГАНІЗАЦІЇ УМОВНИХ ПЕРЕХОДІВ ДЛЯ ЗНАКОВИХ І БЕЗ ЗНАКОВИХ ДАНИХ?

Умовні переходи для знакових та беззнакових даних відрізняються у використанні команд порівняння та умовних переходів. Для беззнакових даних використовуються команди порівняння та переходів, які виконують операції порівняння без знаку, тобто порівнюють числа як позитивні значення, без урахування знаку. Для знакових даних використовуються команди переходу, які виконують операції порівняння зі знаком, тобто порівнюють числа як знакові значення. Наприклад, команда *JG* (Jump if Greater) виконає перехід, якщо перше значення є більшим за друге значення в знаковому форматі, а команда *JA* (Jump if Above) виконає перехід, якщо перше значення є більшим за друге значення в беззнаковому форматі.

ЩО РОБИТЬ КОМАНДА CMP?

Команда CMP у мові асемблера використовується для порівняння двох значень. Вона віднімає друге значення від першого значення, але не зберігає результат. Замість цього, вона встановлює прапорець стану процесора відповідно до результату порівняння. Якщо перше значення більше за друге, то встановлюється прапорець CF (Carry Flag), а якщо вони рівні, то встановлюється прапорець ZF (Zero Flag). Ці прапорці використовуються в подальших операціях, таких як команди безумовних та умовних переходів.

ЩО РОБИТЬ КОМАНДА TEST?

Команда TEST виконує логічне "І" (AND) для двох операндів. Вона не змінює значень операндів, а лише встановлює прапорці відповідно до результату операції. Якщо результат дорівнює 0, то встановлюється прапорець ZF (Zero Flag), що означає, що операнди містять нульові біти в зазначених позиціях. Операція TEST зазвичай використовується для перевірки наявності деяких прапорців в регістрі або для перевірки сумісності двох значень.