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

Кафедра ЕОМ

Звіт

з лабораторної роботи №3 з дисципліни:
“Системного програмування”
на тему: «Програмування задач лінійної структури. Обчислення виразів.»
Варіант: № 24.

Виконав:
ст. групи КІ-203
Ширий Б. І.
Перевірила:
стар. вик. кафедри ЕОМ
Ногаль М. В.

Львів – 2023

МЕТА РОБОТИ:

Вивчити способи задання констант та змінних в Асемблері та набути навиків використання арифметичних команд над даними різного розміру.

ЗАВДАННЯ:

УМОВА ЗАВДАННЯ:

1. Створити *.exe програму, яка реалізовує обчислення, заданого варіантом виразу і зберігає результат в пам'яті. Вхідні операнди A, B, C, D, E, F вважати знаковими і довжиною в байтах, згідно з індексу; К – константа, довжина якої визначається значенням(згідно варіанту). Для її опису слід використати директиву EQU.
2. За допомогою Debug, відслідкувати правильність виконання програми (продемонструвати результати проміжних та кінцевих обчислень).
3. Скласти звіт про виконану роботу з приведенням тексту програми та коментарів до неї.
4. Дати відповідь на контрольні запитання.

A, B, C, D, E, F - знакові операнди, довжиною в байтах, згідно з індексу, значення К подано у 16-му форматі. Відповідно до 24 варіанту мій вираз має вигляд:

$$X = K - \frac{B_2}{C_1} + D_3 + E_2 \cdot F_2,$$

де $K_4 = 74569024$.

ВИКОНАННЯ:

У таблиці 1 навів пояснення до дій заданих виразом.

Таблиця 1. Пояснення до обчислень.

Дія	**Пояснення**
$\frac{B_2}{C_1} = \Pi_1^{BC}$	два байти поділити на один байт, результат – один байт
$K_4 - A_{BC} = \Pi_4^{KBC}$	чотири байти відняти один байт, результат – чотири байти
$\Pi_4^{KBC} + D_3 = \Pi_4^{KBCD}$	чотири байти плюс три байти, результат – чотири байти
$E_2 * F_2 = \Pi_4^{EF}$	два байти помножити на два байти, результат – чотири байти
$\Pi_4^{KBCD} + \Pi_4^{EF} = X_4$	чотири байти додати чотири байти, результат – чотири байти

Написав текст, де кожне обчислення пояснене коментарями, відповідно до заданого завдання, який навів на лістингу 2.

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
K EQU 74569024h
B dw 15h
Cc db 7h
D dd 2003h
E dw 10h
F dw 24h
X dd ?
Message db 'X = K - B / C + D + E * F =                ', 13, 10
NumberOfCharsToWrite dd $-Message
format db '%d', 0
hConsoleOutput dd 0
NumberOfCharsWritten dd 0

.code
start:
;X = K - B / C + D + E * F

; B / C ---> eax
movsx   eax, B          ; eax = B
movsx   ecx, Cc         ; ecx = Cc
cdq
idiv    ecx                     ;15h / 4h = 3h

; K - ecx ---> ecx
mov     edx, K          ; edx = K
sub     edx, eax; 74569024h - 3h = 74569021h

add     edx, D          ; 74569021h + 2003h = 7456B024h

; E * F ---> eax
movsx   eax, E          ; eax = E
movsx   ecx, F          ; ecx = F
imul    eax, ecx; 10h * 24h = 240h

add     edx, eax; 7456B024h + 240h = 7456B264h
mov     X, edx          ; X = edx (результат)

push X
push offset format
push offset [Message+28]
call wsprintfA
push -11
call GetStdHandle
mov hConsoleOutput, eax
push 0
push offset NumberOfCharsWritten
push NumberOfCharsToWrite
push offset Message
push hConsoleOutput
call WriteConsoleA
push 0
call ExitProcess
end start
```

ВИВІД КОНСОЛІ:

Запустивши програму отримаємо вивід, зображений на рисунку 1.

Перевіримо чи збігається результат у 16- системі із виведеним у 10-вій:

$7456B264_{16} =$

$= 7 \cdot 16^7 + 4 \cdot 16^6 + 5 \cdot 16^5 + 6 \cdot 16^4 + 11 \cdot 16^3 + 2 \cdot 16^2 + 6 \cdot 16^1 + 4 \cdot 16^0 =$

$= 1951838820_{10}$

Отож, як бачимо вивід збігається поясненнями з коментарів – це означає, хо вивід був здійснений правильно.

ВИСНОВКИ:

Протягом виконання лабораторної роботи здійснив обчислення заданого виразу за допомогою MASM32. Таким чином, вивчив способи задання констант та змінних в Асемблері та набув навичок використання арифметичних команд над даними різного розміру.

КОНТРОЛЬНІ ПИТАННЯ:

ЯК ВИЗНАЧАЮТЬСЯ В АСЕМБЛЕРІ ДАНІ ДОВЖИНОЮ В 2 БАЙТИ? ЯК ВОНИ ЗАПИСУЮТЬСЯ У ПАМ'ЯТЬ?

Дані довжиною в 2 байти (16-бітні дані) в Асемблері можна визначити за допомогою різноманітних директив або за допомогою відповідних операндів.

DW - директива, яка вказує, що наступні дані повинні бути збережені як слово (2 байти).

```
my_data dw 2003h
```

Це збереже значення 2003h у двох байтах у пам'яті.

AX, BX, CX, DX - регістри загального призначення, які можуть бути використані для збереження 16-бітних даних.

```
mov ax, 2003h
```

Це збереже значення 2003h у регістрі AX.

У пам'яті 16-бітні дані зберігаються в двох порядках байтів, залежно від порядку байтів, який використовується на конкретній архітектурі процесора.

У більшості систем порядок байтів визначається за допомогою так званої "малої" або "великої" ендіанів. У "малій" ендіанів порядок байтів є зворотним, тобто менш значущий байт зберігається в меншому адресі, а більш значущий байт - в більшому. У "великій" ендіанів порядок байтів є прямим, тобто більш значущий байт зберігається в меншому адресі, а менш значущий байт - в більшому.

ЧИМ ВІДРІЗНЯЮТЬСЯ ДИРЕКТИВА EQU ВІД DW ?

Директива EQU та DW використовуються в асемблерному програмуванні, але мають різні призначення.

Директива EQU встановлює символьну константу, яка може бути використана в програмі замість числового значення. Наприклад, EQU LENGTH 10 встановлює константу LENGTH зі значенням 10.

Директива DW, з іншого боку, використовується для резервування місця в пам'яті для зберігання двобайтових цілих чисел. Наприклад, DW 2003 зарезервує два байти пам'яті для зберігання числа 2003.

Отже, директива EQU використовується для створення символьних констант, тоді як DW - для резервування місця в пам'яті для чисел.

ЯКІ АРИФМЕТИЧНІ КОМАНДИ Є В МОВІ АСЕМБЛЕР?

У мові Асемблер існують такі арифметичні команди, які наведені у таблиці 2.

Таблиця 2. Арифметичні команди асемблера.

Команда	Функціонал
MUL	Команда множення, яка примножує регістр ax на те, що стоїть після неї. Результат заноситься в ax.
DIV	Команда ділення, яка ділить регістр ax на те, що стоїть після неї. Результат заноситься в ax.
ADD	Команда додавання - додавши два числа, результат заноситься в перший регістр.
SUB	Команда віднімання - віднімаючи два числа, результат заноситься в перший регістр.
INC	Збільшення значення на 1.
DEC	Зменшення значення на 1.

Ці команди дозволяють виконувати базові арифметичні операції в мові Асемблер. Додатково до цих команд можуть бути доступні команди для роботи з дійсними числами, такі як FADD, FSUB, FMUL та FDIV.

ПОЯСНИТИ ВИНИКНЕННЯ ПЕРЕПОВНЕННЯ І ПЕРЕНОСУ ПРИ ВИКОНАННІ АРИФМЕТИЧНИХ КОМАНД.

Переповнення виникає, коли результат арифметичної операції великий для розрядності регістра, в якому зберігається результат. Наприклад, при додаванні двох беззнакових чисел, які займають 8 біт, якщо результат перевищує 255, відбувається переповнення.

Перенос виникає при виконанні операцій додавання та віднімання зі знаком. Він виникає, коли результат операції не може бути збережений в розрядності регістра, наприклад, при додаванні двох чисел зі знаком, що займають 8 біт, якщо результат перевищує 127 або менше -128, відбувається перенос.

ЯКІ ФОРМАТИ ОПЕРАЦІЙ МНОЖЕННЯ І ДІЛЕННЯ ВИ ЗНАЄТЕ?

Загалом, у системному програмуванні є кілька форматів операцій множення і ділення, які навів у таблиці 3:

Таблиця 3. Формати множення та ділення.

Формат	**Пояснення**
MUL	Формат операції множення, який використовується для множення двох беззнакових чисел.
IMUL	Формат операції множення, який використовується для множення двох знакових чисел.
DIV	Формат операції ділення, який використовується для ділення беззнакового числа.
IDIV	Формат операції ділення, який використовується для ділення знакового числа.

У кожному з цих форматів операцій передбачені відповідні команди процесора, які виконують відповідні операції.

ЩО РОБИТЬ КОМАНДА CMP?

Команда асемблера CMP використовується для порівняння двох значень. Вона віднімає одне значення від іншого і встановлює прапорець процесора відповідно до результату порівняння. Якщо перше значення більше за друге, то прапорець CF встановлюється в 1, а якщо навпаки, то прапорець CF - 0. Якщо значення рівні, то прапорець ZF встановлюється в 1. Результат порівняння

можна використовувати для прийняття рішень в програмі, наприклад, для виконання різних операцій в залежності від того, яке з двох значень більше.