Міністерство освіти і науки України   
Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Кафедра автоматизації проектування енергетичних процесів і систем

Лабораторна робота №6 з дисципліни «Операційні системи»  
**Система команд. Арифметичні команди. Обчислення цілочисельних арифметичних виразів з використанням команд MUL, IMUL, DIV, IDIV, ADD, ADC, INC, SUB, SBB, DEC, NEG, CBW, CWD**

**Варіант: 15**

Виконав студент групи ТР-12

Каркушевський Владислав

Перевірила д.т.н., проф. Левченко Л. О.

КИЇВ 2022

**Мета роботи** : програмування блоку обчислення арифметичних виразів на прикладі реалізації математичних формул з використанням арифметичних команд асемблера за індивідуальними завданнями.

**Теоритична частина**:

**Команди переміщення даних:**

* MOV

Схема команди: MOV приймальник, джерело Призначення: пересилання даних між регістрами або регістрами та пам’яттю. Команда має обмеження:

• копіювання здійснюється з другого операнда у перший;

• значення другого операнда не змінююється;

• обидва оператора не можуть бути з пам’яті;

• лише один з операндів може бути сегментним, приймальником не може бути регістр CS; не можна пересилати сегментні регістри

**Арифметичні команди:**

* ADD (Команда складання)
* ADC (Команда складання двох операндів з урахуванням перенесенням з молодшого розряда)
* SUB (Команда віднімання)
* SBB (Команда віднімання із зайняттям)
* MUL (Команда множення двох цілих двійковихчисел без урахування знаку)
* IMUL (Команда множення двох цілих двійкових чисел з урахуванням знаку)
* DIV (Команда беззнакового ділення)
* ІDIV (Команда знакового цілочисельного ділення ІDIV)

1. Команда складання **ADD** (ADDition)

**Синтаксис**: ADD приймальник, джерело

**Призначення**: складання двох операндів джерела та приймальника розмірністю байт, слово або подвійне слово, записати результат складання за адресою першого операнда, встановити прапорці.

Виконання команди додавання **впливає на стан прапорців**: CF (Carry)– перенесення (зі старшого знакового розряду; наприклад, для байта в 9-му розряді при виконанні команди додавання з’явилася 1), PF (Parity)– парність, AF (Auxiliary)- допоміжне перенесення, ZF (Zero)– нуль, SF (Sign)– мінус, OF (Overflow)– переповнювання.

Команда ADD використовується для **складання двох цілочисельних операндів**. Якщо результат складання виходить за межі першого операнда (виникає переповнювання), то врахувати цю ситуацію слід шляхом аналізу прапорця CF і подальшого можливого застосування команди ADC.

1. Команда складання двох операндів з урахуванням перенесенням з молодшого розряда **ADC**

**Синтаксис: ADС приймальник, джерело.**

**Впливає** на прапорці AF, CF, OF, PF, SF, ZF

Команда ADC **використовується** при складанні багаторозрядних двійкових чисел.

Команда ADC додає **вміст прапорця** перенесення CF (0 або 1) до першого операнда - приймальника, а потім додає до приймальника другий операнд – джерело.

1. Команда віднімання SUB

**Синтаксис: SUB операнд\_1, операнд\_2**

**Впливає** на прапорці AF, CF, OF, PF, SF, ZF

Команда **призначена** для віднімання цілочисельних операндів або для віднімання молодших частин значень багатобайтних операндів.

Виконання команди віднімання **впливає на стан прапорців**: CF – перенесення (зі старшого знакового розряду; наприклад, для байта в 9- му розряді при виконанні команди додавання з’явилася 1), PF – парність, AF - допоміжне перенесення, ZF – нуль, SF – мінус, OF – переповнювання. При відніманні прапорець CF діє як ознака зайняття.

1. Команда віднімання із зайняттям SBB

**Синтаксис: SBB операнд\_1, операнд\_2**

Команда **призначена** для виконання цілочисельного віднімання старших частин значень багатобайтних операндів з урахуванням можливого попереднього зайняття при відніманні молодших частин значень цих операндів, коли виконувалося попереднє віднімання командами SBB та SUB (за станом прапорця перенесення CF).

1. Команда множення двох цілих двійковихчисел без урахування знаку MUL

**Синтаксис: MUL множник\_1**

Команда перемножує два цілих числа без знаку.

Команда MUL сприймає старший біт в якості біта даних , а не як біт знака. Алгоритм роботи команди залежить від формату операнда команди і вимагає явної вказівки місцеположення тільки одного співмножника, який може бути розташований в пам'яті або в регістрі (перший співмножник). Місцеположення другого співмножника фіксовано і залежить від розміру першого співмножника:

• якщо операнд, вказаний в команді, — байт, то другий співмножник повинен розташовуватися в AL;

• якщо операнд, вказаний в команді, — слово, то другий співмножник повинен розташовуватися в АХ;

• якщо операнд, вказаний в команді, — подвійне слово, то другий співмножник повинен розташовуватися в EAX.

Результат множення поміщається також у фіксоване місце, яке визначається розміром співмножників:

• при перемноженні байтів результат поміщається в АХ;

• при перемноженні слів результат поміщається в пару DX:AX (молодші розряди - в AX, старші - в DX) і встанавлюються прапорці переповнення и перенесення;

• при перемноженні подвійних слів результат поміщається в пару ЕDX:EAX (молодші розряди - в ЕAX, старші - в ЕDX) і встанавлюються прапорці переповнення и перенесення;

Після виконання команди MUL **прапорці** CF і OF дорівнюються 0, якщо старша половина множення дорівнюється нулю; інакше обидва ці прапорці дорівнюються 1

1. Команда множення двох цілих двійкових чисел з урахуванням знаку IMUL

**Синтаксис: ІMUL множник\_1**

Команда виконує цілочисельне множення операндів з урахуванням їх знакових розрядів. Команда сприймає старші(перші ліворуч) біти чисел в якості знаків (0 – позитивне число, 1 – негативне число). Для виконання цієї операції необхідно наявність двох співмножників. Розміщення і задання їх місцеположення в команді залежить від форми вживаної команди множення, яка, у свою чергу, визначається моделлю мікропроцесора.

Алгоритм роботи залежить від форми команди (однооперандна, двохоперандна або трьохопреандна)

1. Команда беззнакового ділення DIV

**Ситаксис: DIV дільник (делитель)**

Команда призначена для ділення двох двійкових беззнакових чисел.

Для команди необхідно задати два операнди — ділимого(делимого) і дільника(делителя), беззнакове ділиме ділить націло дільник. Ділиме задається неявно і розмір його залежить від розміру дільника, який вказується в команді:

• якщо дільник розміром в байт, то ділиме повинно бути розташовано в регістрі АХ. Після операції частка (частное) поміщається в AL, а залишок — в AH;

• якщо дільник розміром в слово, то ділиме повинно бути розташовано в парі регістрів DX:AX, причому молодша частина ділимого знаходиться в АХ. Після операції частка поміщається в АХ, а залишок — в DX;

• якщо дільник розміром в подвійне слово, то ділиме повинно бути розташовано в парі регістрів EDX:EAX, причому молодша частина ділимого знаходиться в EAX. Після операції частка поміщається в EAX, а залишок — в EDX.

Ділення на нуль викликає переривання.

1. Команда знакового цілочисельного ділення ІDIV

**Синтаксис: ІDIV дільник (делитель)**

Команда призначена для ділення двох двійкових чисел зі знаком, ділить знакове ділиме націло на знаковий дільник. Команда ІDIV сприймає в якості знака старші (перші ліворуч) біти (1- від’ємне число, 0 – позитивне число). Ділення на нуль викликає переривання.

Алгоритм роботи подібний до команди DIV.

**Завдання**

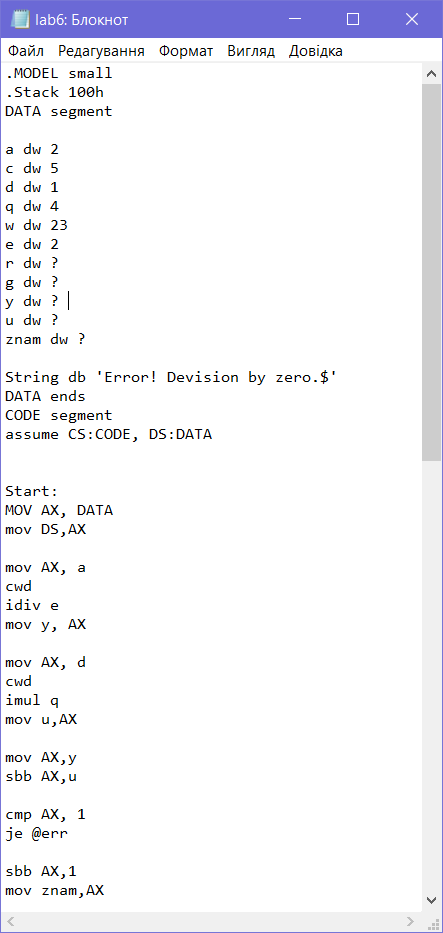
1. Підготуйте теоретичну частину щодо використання команд, призначених для цілочисельних арифметичних виразів.

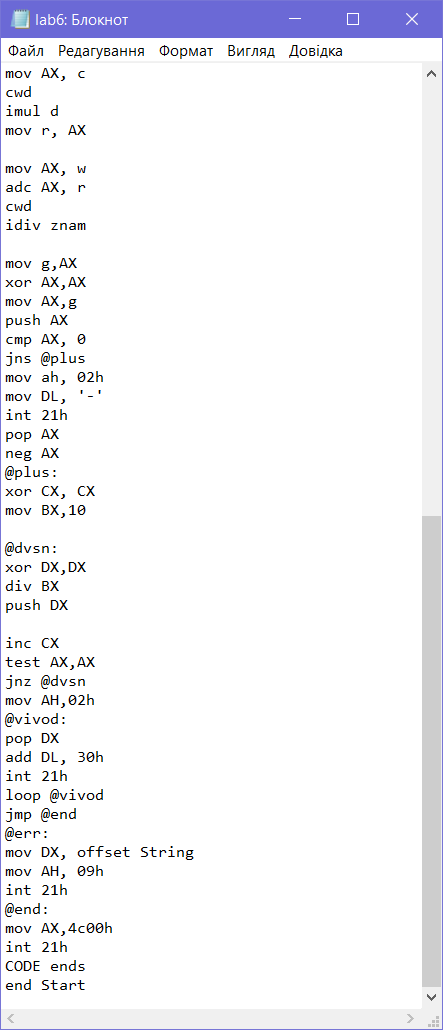
2. Підготуйте і налагодьте програму для обчислення простих формул за зразком. Продемонструйте роботу програми під керуванням налагоджувача, прокоментуйте вміст обчислюваних змінних

**Варіант 15**



**Код програми:**

****

****

**Розглянемо детальніше, та пояснимо основні дії:**

**.MODEL small** ;модель пам’яті для ехе-файлів

**.Stack 100h** ;виділяємо PSP 256 байт

**DATA segment** ;початок сегменту даних

**a dw 2** ;вихідні числа які використовуються в формулі

**c dw 5**

**d dw 1**

**q dw 4**

**w dw 23**

**e dw 2** ;та змінні для збереження

**r dw ?**

**g dw ?**

**y dw ?**

**u dw ?**

**znam dw ?**

**String db 'Error! Devision by zero.$'** ;повідомлення, яке виводиться коли ;знаменник = 0

**DATA ends** ;кінець сегменту даних

**CODE segment** ;початок сегменту коду

**assume CS:CODE, DS:DATA** ;директива assume повідомляє, що регістр cs ; вказує на початок сегменту коду, ds – сегменту даних

**Start:** ;точка входу

**MOV AX, DATA** ;ініціалізація регістру ds

**mov DS,AX** ;адресою сегмента даних

**mov AX, a** ; a🡪AX

**cwd** ;команда перетворення слова у подвійне слово

**idiv e ;**а/2 → AX, знакове ділення

**mov y, AX** ; AX🡪y

**mov AX, d ;** d🡪AX

**cwd** ;команда перетворення слова у подвійне слово

**imul q** ; d\*4 🡪AX

**mov u,AX** ;AX🡪u

**mov AX,y** ; y🡪AX

**sbb AX,u** ; AX-u

**cmp AX, 1** ;перевіряємо, чи ах=1

**je @err** ;якщо так, то виводимо повідомлення про помилку

**sbb AX,1** ; a/2-4\*d-1 🡪AX

**mov znam,AX ;** AX🡪znam

**mov AX, c ;** AX🡪c

**cwd** ;команда перетворення слова у подвійне слово

**imul d ;** c\*d🡪AX

**mov r, AX ;** AX🡪r

**mov AX, w** ; w🡪AX

**adc AX, r** ; c\*d+23🡪AX

**cwd**;команда перетворення слова у подвійне слово

**idiv znam ;** (c\*d+23)/(a/2-4\*d-1)🡪AX

**mov g,AX** ;AX🡪g

**xor AX,AX** ; обнуляємо регістр ах

**mov AX,g ;** g → ax

**push AX** ;зберігаємо в стек

**cmp AX, 0** ;порівнюємо ах з 0

**jns @plus** ;перевіряємо число позитивне чи від’ємне

**mov ah, 02h** ;перевіряємо число позитивне чи від’ємне

**mov DL, '-'**

**int 21h**; виклик DOS - функція переривання

**pop AX** ;вивантажуємо значення зі стеку

**neg AX** ;перетворюємо від’ємне число у позитивне

**@plus:**

**xor CX, CX** ;лічильник остач

**mov BX,10** ;завантажуємо основу 10-ї системи числення

**@dvsn:**

**xor DX,DX** ;остача при ділені

**div BX ;**виконуємо ділення на 10

**push DX** ;результат у зворотньому порядку утворює число у

;новій системі счислення

**inc CX** ;збільшуємо лічильник

**test AX,AX** ;перевіряємо, чи ціла частина = 0

**jnz @dvsn** ;ділимо, поки ціла частина не дорівнює нулю

**mov AH,02h** ;виводимо остачу

**@vivod:**

**pop DX** ;дістаємо зі стеку збережені числа

**add DL, 30h** ;переводимо їх в ASCII-коди

**int 21h**; виклик DOS - функція переривання

**loop @vivod** ;повторюємо, поки СХ не дорівнює 0

**jmp @end** ;вихід

**@err:** ;виведення повідомленняпро помилку

**mov DX, offset String**

**mov AH, 09h**; Функція DOS 9h означає виведення на екран

**int 21h**; виклик DOS - функція переривання

**@end:** ;коректно завершуємо нашу програму, виходимо

**mov AX,4c00h**

**int 21h** ; виклик DOS - функція переривання

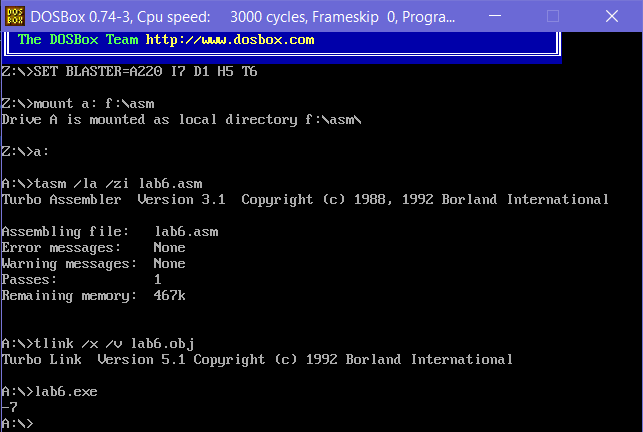
**CODE ends** ;кінець сегменту коду

**end Start** ;кінець програми

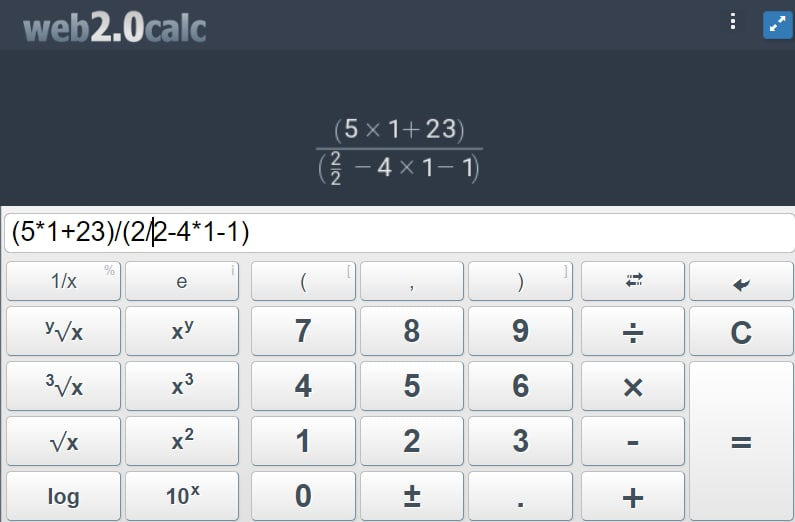
Запустимо **транслятор** та створими лістинг(\*.lst) та об’єктний файл(\*.obj) за допомогою команди: **tasm /la /zi lab6.asm**

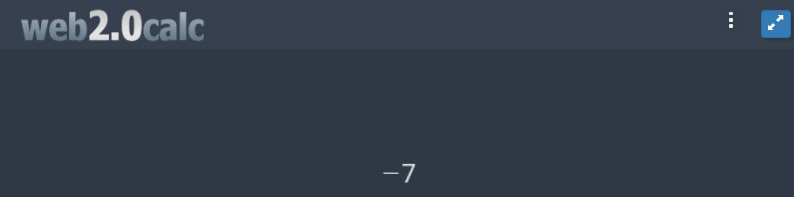
Запустимо **компоновщик** та створимо файл (\*.exe) за допомогою команди: **tlink /x /v lab6.obj**

Запустимо нашу програму ввівши команду **lab6.exe**

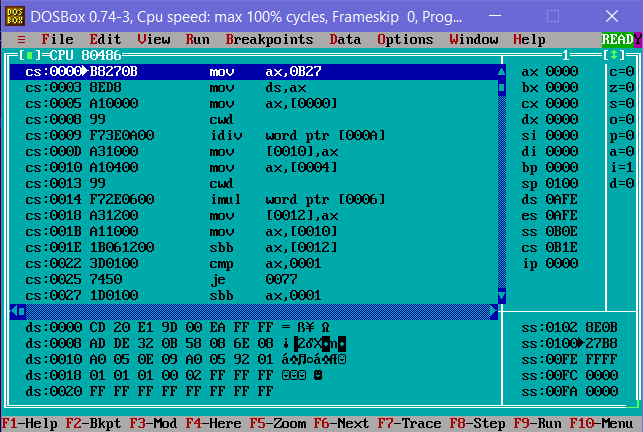
****

**Результат у калькуляторі:**

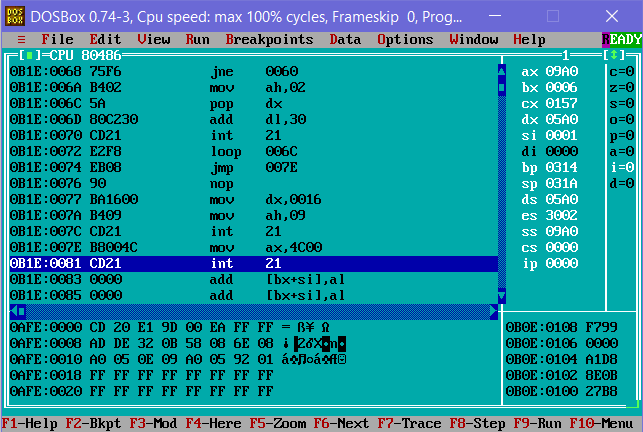


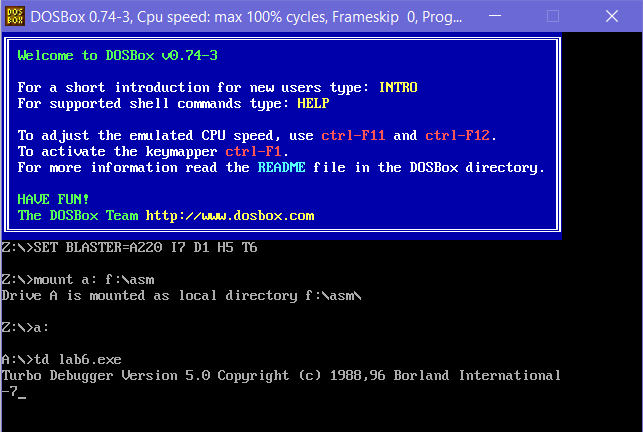


Запустимо **налагоджувач**

****

За допомогою клавіші **F8**(або **F7**) покроково виконуємо нашу програму і виводимо отримане словосполучення в консольному режимі





**Висновок:**

На цій лабораторній роботі було освоєно арифметичні команди. Було запрограмовано блок обчислення арифметичних виразів на прикладі реалізації математичних формул з використанням арифметичних команд асемблера за індивідуальними завданнями. Виконано тестові перевірки, проаналізовано результати. Виведено результат на екран.