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

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

Теоретична частина

6.1 Команди переміщення даних

6.1.1 MOVe operand Пересилання операнда

MOVe operand to/from system registers Пересилання операнда у системні регістри (або з них)

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 змінюються;
- 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:

```
MOV  ES, DS
```

Треба розписати:

```
MOV  AX, DS
MOV  ES, AX
```

- не можна напряму ініціалізувати сегмент даних

```
DSEG SEGMENT
.....................
        MOV DS, DSEG
```

Треба розписати:

```
        MOV AX, DSEG
        MOV DS, AX
```

- довжина обох операндів повинна бути однаковою.

Команда Mov має розширену можливість: для випадку, коли довжина операндів різна використовується директива заначення типу:

Тип **PTR** *вираз*

Оператор PTR може використовуватися з елементами даних, мітками інструкцій. Він використовує специфікатори типів Byte, Word, Dword, Tbyte для явного зазначення типу DB, DW, DD, DQ, DT для змінних. Він також використовує специфікатори типів NEAR, FAR, PROC для явного зазначення відстані до мітки перреходу. Таким чином, *тип* – це специфікатор типу, наприклад, BYTE, а *вираз* – це змінна або константа.

Наприклад,

```
BYTEA      DB   22H
```

```
                DB   35H
        WORDA DW   2672H      ; у памяті зберігається у вигляді 7226H
        ……….
MOV BYTE PTR WORDA, 05; Розмістити 05 у перший байт WORDA
MOV AX, WORD PTR BYTEA;  Розмістити в AX два байти (2235H) з
 BYTEA
```

6.2 Арифметичні команди

6.2.1 Команда складання ADD (ADDition)

***Синтаксис*: ADD приймальник, джерело**

Символьний код: *ADD регістр/пам'ять, регістр/пам'ять/безпосереднє значення*

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

Ця операція є коректною при використанні операндів: регістр-регістр, регістр-пам'ять, пам'ять-пам'ять, пам'ять-безпосереднє значення.

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. Наприклад, складемо значення в регістрі AX і області пам'яті CHH. При складанні слід врахувати можливість переповнювання.

```
ADD AX, CHH   ;  додати значення з області пам'яті CHH до регістру AX
ADD AX, CH    ;   додати регістр до регістру, результат записати в AX.
ADD EBX, DBLWORD   ;   додати подвійне слово з пам'яті до регістру.
ADD BL, 10              ;    додати 10 до молодшої частини регістра BX.
```

6.2.2 Команда складання двох операндів з урахуванням перенесенням з молодшого розряда ADC (ADDition with Carry)

***Синтаксис*: ADC приймальник, джерело.**

Символьний код: *ADC регістр/пам'ять, регістр/пам'ять/безпосереднє значення*

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

Результат заноситься у перший операнд, в залежності від результату встановлюються прапорці.

Логіка роботи команди: <приймальник>=<приймальник>+<джерело>+<CF>

Команда ADC використовується *при складанні багаторозрядних двійкових чисел*. Її можна використовувати як самостійно, так і спільно з командою ADD. При спільному використанні команди ADC з командою ADD складання молодших байтів/слів/подвойних слів здійснюється командою ADD, а вже старші байти/слова/подвойні слова складаються командою ADC, що враховує перенесення з молодших розрядів в старші. Тобто команда ADC додає вміст прапорця перенесення CF (0 або 1) до першого операнда - приймальника, а потім додає до приймальника другий операнд – джерело.

6.2.3 Команда віднімання SUB (SUBtract)

Синтаксис: SUB операнд_1, операнд_2

Символьний код: *SUB регістр/пам'ять, регістр/пам'ять/безпосереднє значення*

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

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

Віднімання здійснюється за методом складання з двійковим доповненням: для другого операнда встановлюється додатковий код (біти інвертуються +1), а потім відбувається складання з першим операндом. *Операнд_2 віднімається від операнда_1, результат записується в операнд_1.*

Команда SUB діє як приймальник = приймальник – джерело

SUB BL, 10; з регістру BL відняти значення 10, результат занести в BL

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.

6.2.4 Команда віднімання із зайняттям (заемом) SBB (SuBtract with Borrow) або віднімання з перенесенням

Синтаксис: SBB операнд_1, операнд_2

Символьний код: SBB регістр/пам'ять, регістр/пам'ять/безпосереднє значення

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).

Команда ***SBB*** спочатку віднімає вміст прапорця CF з операнда_1 та віднімає з операнда_1 операнд_2: приймальник = приймальник - джерело - перенесення.

6.2.5 Команда множення двох цілих двійковихчисел без урахування знаку MUL (MULtiply)

Команда перемножує два цілих числа без знаку.

Синтаксис: **MUL множник_1**

Символьний код: MUL регістр/пам’ять.

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 **AX**;
- 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 AX;
- 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 EDX:EAX (молодші розряди - в EAX, старші - в EDX) і встанавлюються прапорці переповнення и перенесення;

Контролювати розмір результату зручно, використовуючи прапорці CF або OF (прапорці AF, PF, SF, ZF –не визначено). Після виконання команди MUL прапорці CF і OF дорівнюються 0, якщо старша половина множення дорівнюється нулю; інакше обидва ці прапорці дорівнюються 1.

Множення і ділення це одні з найповільніших операцій процесорів сімейства 80x86 (особливо 8086 і 8088). Множення і ділення на константи, що часто зустрічаються, швидше відбувається при зсуві розрядів.

6.2.6 Команда множення двох цілих двійкових чисел з урахуванням знаку IMUL (Integer MULtiply)

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. Так, для мікропроцесора i8086 можлива тільки одноопернадна форма команди, для подальших моделей мікропроцесорів додатково можна використовувати двох- і трьохоперандні форми цієї команди.

Синтаксис: **IMUL множник_1**

IMUL множник_1, множник_2
IMUL результат, множник_1, множник_2

Символьний код: IMUL регістр/пам'ять
IMUL регістр, безпосереднє значення (для 80286 і вище)
IMUL регістр, регістр, безпосереднє значення (для 80286 і вище)
IMUL регістр, регістр/пам'ять (для 80386 та вище)

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

1. Як і для команди MUL**,** вважається, що 2-й співмножник розташовується в регістрі AL (для операнда в команді - байта), в AX (для операнда в команді – слова), в EAX (для операнда –подвійне слово). *Результат множення для команди з одним операндом розташовується в AX* (при перемноженні двох байтів), в пару DX:AX (при перемноженні слів), в пару EDX:EAX (при перемноженні подвійних слів).
   Наприклад,
   IMUL DL ; перемножити DL на AL зі знаком
   IMUL MEM_WORD; перемножити вміст комірки пам'яті на AX зі ;знаком

Три інших формати використовують будь-який 16- або 32-розрядний регістр загального призначення, розміри елементів даних повинні бути однаковими.

2. Перший операнд (регістр) містить співмножник (множимое), саме в ньому з'являється результат множення, 2- операнд – множник (множитель), який має безпосереднє значення.
   Наприклад, IMUL DX, 456 ; перемножити DX на 456
   IMUL BX, 32
   IMUL CX, 50

3. Перший операнд (регістр) вказує, де повинен знаходитися результат множеня; другий операнд (регістр або адреса пам'яті) містить співмножник (множимое), третій операнд містить безпосереднє значення.
   Наприклад, IMUL CX, DX, 56
   IMUL EBX, подівйне_слово_у_пам'яті, 10

4. Перший операнд (регістр) містить співмножник (множимое), саме у ньому з'являється результат; другий операнд (регістр або адреса пам'яті) - множник (множитель). Наприклад,
   IMUL DX, слово_у_пам'яті
   IMUL EBX, EDX

Команда imul встановлює в нуль прапорці OF і CF, якщо розмір результату відповідає регістру призначення. Якщо ці прапори відмінні від нуля, то це означає, що результат дуже великий для відведених йому регістром призначення рамок і необхідно вказати більший за розміром регістр для успішного завершення даної операції множення.

Інші прапорці AF, PF, SF, ZF є невизначеними.

6.2.7 Команда беззнакового ділення DIV (DIVide unsigned)

Команда призначена для ділення двох двійкових беззнакових чисел.

Синтаксис: **DIV дільник (делитель)**

Смвольний код: DIV регістр/пам'ять

Алгоритм роботи:

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 AX. 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 AX. Після операції *частка* поміщається в **AX**, 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**.

Наприклад,
```
DIV BH    ; байт
DIV CX    ; слово
DIV ECX   ; подвійне слово
```
Ділення на нуль викликає переривання.

6.2.8 Команда знакового цілочисельного ділення IDIV (Integer DIVide signed)

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

Синтаксис: **IDIV дільник (делитель)**

Смвольний код: DIV регістр/пам'ять

Алгоритм роботи: Для команди необхідно задати два операнди — ділиме і дільник. Ділиме задається неявно, і розмір його залежить від розміру дільника, місцезнаходження якого вказується в команді:

- якщо *дільник* розміром в *байт*, то *ділиме* повинно бути розташовано в регістрі AX. Після операції *частка* 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 ax. Після операції *частка* поміщається в AX, 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;

Залишок завжди має знак ділимого. Знак частки залежить від стану знакових бітів (старших розрядів) ділимого і дільника.

Впливає на стан прапорців AF, CF, OF, PF, SF, ZF.

Для збільшення розрядності знакового ділимого використовується інструкція CBW (Convert Word to Doubleword)та MOVSX - команди перетворення слова у подвійне слово.

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.

```
; Лабораторна робота 4 «Системне програмування»
; Виконав П. І. П. гр. Дата ДД.ММ.РР
; Арифметичні команди
;Обчислення формули z=(x+y)*w-(w-x)/y

    Datas segment     ; Сегмент даних
        X dw 7        ; з вихідними
        Y dw 3        ; числами
        W dw 11       , і змінна
        Z dw ?        ; для збереження
    Datas ends

    Codes segment
    Assume cs:Codes, ds : Datas

        First:
        MOV ax, Datas
        mov ds,ax
        mov ax,X      ; x -> ax
        add ax,Y      ; x+y -> ax
        mul W         ; (x+y)*W -> ax
        sub ax,W      ; (x+y)*W-W -> ax
        mov Z,ax      ; ax -> Z'
        mov ax,X      ; x -> ax
        cwd
        div Y         ;x/y -> ax
        sub Z,ax      ; Z' -x/y -> Z"
        mov ah,4ch
        int 21h
```

```
Codes ends
    end First
```

Відповідно до варіантів підготуйте програми, що реалізують формули (етап перший). Обчисліть вирази у знакових форматах довжиною 8 та 16 біт, використовуючи вищенаведені арифметичні операції. Виконайте тестові перевірки. Проаналізуйте результат.
Етап другий - додайте виведення на екран результату обчислення.

Варіанти:

1) (2*c — d + 23) /(a/4 — 1);
2) (c + 4*d — 123)/(1 — a/2);
3) (-2*c + d*82)/(a/4 — 1);
4) (2*c + d — 52)/(a/4 + 1);
5) (c/4 — d*62)/(a*a + 1);
6) (-2*c — d + 53)/(a/4 — 1);
7) (2*c — d/4)/(a*a + 1);
8) (2 + c — d*23)/(2*a*a — 1);
9) (2*c — d/3)/(b — a/4);
10) (4*c + d — 1)/(c — a/2);
11) (2*c — d*42)/(c + a — 1);
12) (25/c — d + 2)/(b + a*a-1);
13) (c — d/2 + 33)/(2*a*a-1);
14) (4*c — d/2 + 23)/(a*a — 1)
15) (c*d + 23)/(a/2 -4*d- 1);
16) (c/d + 3*a/2)/(c — a + 1);
17) (2*c + d*51)/(d — a — 1);
18) (2*c + d/4 + 23)/(a*a — 1);
19) (2*c — d/2 +1)/(a*a+7);
20) (2*c/d + 2)/(d — a*a — 1);
21) (12/c — d*4 +73)/(a*a+1);
22) (2*c/a — d*d)/(d + a — 1);
23) (-53/a + d -4*a)/(1+a*b);
24) (-15*a + b — a/4)/(b*a -1);
25) (-25/a +c-b*a)/(1+c*b/2);
26) (4*a — 1 + b/2)/(b*c — 5);
27) (8*b + 1 — c)/(a/2+ b*c);
28) (4*a — b — 1)/(c/b + a);
29) (4*b/c — 1)/(12*c+a — b);
30) (b + c*b — a/4)/(a*b — 1);

Приклад. Обчислити вираз

(25/c — d + 2)/(d + a*a-1);

Контрольні питання

1. Прокоментуйте вміст усіх полів сегмента даних у вашій програмі.
2. Поясніть призначення псевдокоманд у вашій програмі.
3. Поясніть вигляд даних, як використаються в програмі.
4. Поясніть відмінність у роботі команд знакової і беззнакової арифметики (MUL / IMUL, DIV / IDIV). Яка логіка роботи цих команд?
5. Які використовуються регістри при роботі команд множення та ділення?