Лабораторна робота №2

Тема: **ДОСЛІДЖЕННЯ НА МІКРОПРОЦЕСОРНИХ ПРИСТРОЯХ АЛГОРИТМІВ АРИФМЕТИЧНИХ ОПЕРАЦІЙ З ОДНОБАЙТОВИМИ І МНОГОБАЙТОВИМИ ЦІЛИМИ ЧИСЛАМИ (ДОДАВАННЯ ТА ВІДНІМАННЯ)**

Мета: ознайомлення з принципами програмування на мові АСЕМБЛЕР МП 1810, із використанням налагоджених засобів, з дослідженням алгоритмів арифметичних операцій додавання і віднімання однобайтових і многобайтових цілих чисел. Ознайомитися з принципами введення та виведення арифметичних операцій додавання і віднімання однобайтових і многобайтових цілих чисел на екран на мові Turbo Assembler.

1 Теоретичні відомості

1.1 Правила арифметичних операцій над числами

Арифметичні операції над цілими числами в будь-який позиційній системі числення виконують за тими же правилам, що й у десятковій арифметиці, оскільки всі вони грунтуються на загальних правилах виконання операцій над відповідними поліномами. При цьому використовують ті таблиці додавання (віднімання) і множення, що мають місце в конкретній системі числення. Зокрема, для двійкової системи діють правила:

0+0=0;	0-0=0;	0×0=0;
0+1=1;	1-0=1;	0×1=0;
1+0=1;	1-1=0;	1×0=0;
1+1=10;	10-1=1;	1×1=1.

Ці правила не містять операцій ділення двійкових цифр, оскільки їх виконання на відміну від інших операцій не може бути зведене до дій над окремими цифрами. При додаванні в двійковому розряді двох одиниць відбувається *перенос* із даного розряду в наступний, більш старший, а при

вирахуванні в двійковому розряді одиниці з нуля відбувається *позика* в даний розряд із більш старшого розряду, у результаті в даному розряді встановлюється одиниця.

При виконанні арифметичних дій виникає проблема ідентифікації від'ємних чисел. Для зображення знака числа приділяється спеціальний знаковий розряд (звичайно старший розряд числа) *S*. Зображення знака "+" у цьому розряді прийнято кодувати для двійкових чисел цифрою 0, а знака "-" - цифрою 1. При цьому, зображення числа із знаком містять тільки двійкові цифри, але припускається, що число складається з двох частин: *знакової* та *цифрової*.

Якщо цифрова частина додатних і від'ємних чисел містить завжди абсолютне значення числа, то такий засіб представлення знакових чисел називають *прямим кодом*. Обробка поданих у прямому коді чисел потребує окремих операцій над цифровою і знаковою частинами. Виконання операцій додавання і віднімання, призводить до появи двох представлень нуля: +0 і -0 (наприклад, +0=00000000; -0=10000000). Ці недоліки прямого коду стримують його виковикористовують в основному *додаткові коди*.

Зміст використання додаткового коду полягає в тому, що, по-перше, арифметичні операції віднімання і додавання чисел у додаткових кодах зводяться до операції алгебраїчного підсумовування (віднімання замінюється додаванням зменшуваного з додатковим кодом, що віднімається); по-друге, обробка знакової й цифрової частин чисел при додаванні провадиться за тими самими правилами, причому правильний знак результату формується автоматично.

У мікропроцесорі 1810 розрядність формату даних може бути подана як n=8N, де N - кількість байтів (N≥1), операції додавання (віднімання) чисел для випадків N=1 і N=2 виконуються за допомогою відповідних команд (див. програми 2.1 і 2.2).

Програма 2.1

```
TITLE  Складання цілих беззнакових чисел
; Визначення сегмента стека
SSEG SEGMENT PARA STACK 'STACK'
DB 256 DUP(0)
SSEG ENDS
; Визначення сегмента даних для доданків та результату
DSEG SEGMENT PARA PUBLIC 'DATA'
SLOG1 DW 1 DUP (0) ; Доданок 1, довжина слово
SLOG2 DW 1 DUP (0) ; Доданок 2, довжина слово
SUM DD 2 DUP(0) ; Результат, довжина 3 байта
DSEG ENDS
; Визначення сегмента  коду програми
CSEG SEGMENT PARA PUBLIC 'CODE'
      ASSUME CS:CSEG,DS:DSEG,SS:SSEG
SUMMA PROC FAR
mov ax,DSEG ; Визначення адресу сегмента DATA
mov ds,ax ; Пересилання в сегментний регістр DS
Start: clc ; Очищення прапора переносу
mov cx,0
mov ax,SLOG1 ; Додавання SLOG1 и SLOG2
add ax,SLOG2
mov bx,OFFSET SUM ; Отримання зміщення і
mov [bx],ax ; збереження результату складання
jnc Lmem ; Був перенос? Ні - перехід на мітку
inc cx ; Так - створити старший байт результату
Lmem: mov [bx+2],cx ; Зберегти старший байт результату
jmp Start
SUMMA ENDP
CSEG ENDS
END SUMMA
```

Програма 2.2

```
TITLE Віднімання цілих беззнакових чисел
; Визначення сегмента стека
SSEG SEGMENT PARA STACK 'STACK'
DB 256 DUP(0)
SSEG ENDS
; Визначення сегмента даних для даних та результату
;
DSEG SEGMENT PARA PUBLIC 'DATA'
UMEN DW 1 DUP (0) ; зменшуване, довжина слово
VICH DW 1 DUP (0) ; від'ємник, довжина слово
RAZ DW 1 DUP(0) ; результат, довжина слово
DSEG ENDS
; Визначення сегмента  коду програми
CSEG SEGMENT PARA PUBLIC 'CODE'
ASSUME CS:CSEG,DS:DSEG,SS:SSEG
SUBST PROC FAR
mov ax,DSEG ; Визначення адресу сегмента DATA
mov ds,ax ; Пересилання в сегментний регістр DS
Start: clc ; Очищення прапора переносу
mov cx,0
mov ax,UMEN ; Віднімання
sub ax,VICH
mov bx,OFFSET RAZ ; Отримання зміщення і
mov [bx],ax ; збереження результату
jmp Start
SUBST ENDP
CSEG ENDS
      END SUBST
```

1.2 Правила введення/виведення інформацій на Turbo Assembler

1.2.1 ASCII код

Для цілей стандартизації в мікрокомп'ютерах використовується американський національний стандартний код для обміну інформацією ASCII (American National Standard Code for Information Interchange). [Читається як "аськи" код (прим. перекладача)]. Саме з цієї причини комбінація біт 01000001 позначає букву А. Наявність стандартної коди полегшує обмін даними між різними пристроями комп'ютера. 8-бітовий розширений ASCII-код, використовуваний в PC забезпечує представлення 256 символів, включаючи символи для національних алфавітів. У Додатку 2 приведений список символів ASCII кодів.

1.2.2 Двійкові числа

Оскільки комп'ютер може розрізнити лише нульовий і одиничний стан біта, то він працює системі числення з основою 2 або в двійковій системі. Фактично біт успадкував свою назву від англійського "Binary digit" (двійкова цифра). Поєднанням двійкових цифр (бітів) можна представити будь-яке значення. Значення двійкового числа визначається відносною позицією кожного біта і наявністю одиничних бітів. Нижче показано восьмибітове число що містить всі одиничні біти:

Позиційні ваги: 128 64 32 16 8 4 2 1

Включені біти: 1 1 1 1 1 1 1 1

1.2.3 Перетворення ASCII-формату в двійковий формат

Виконання арифметичних операцій над числами в ASCII або BCD форматах зручно лише для коротких полів. В більшості випадків для арифметичних операцій використовується перетворення в двійковий формат. Практично простіше перетворення з ascii-формату безпосередньо

в двійковий формат, чим перетворення з ASCII- в bcd-формат і, потім, в двійковий формат: Метод перетворення базується на тому, що ascii-формат має основу 10, а комп'ютер виконує арифметичні операції лише над числами з основою 2. Процедура перетворення полягає в наступному:

1. Починають з найправішого байта числа в ascii-форматі і обробляють справа наліво.
2. Видаляють трійки з лівих шест. цифр кожного ascii-байта.
3. Умножають ascii-цифрі на 1, 10, 100 (шест.1, A, 64) і так далі і складають результати.

Для прикладу розглянемо перетворення числа 1234 з ascii-формату в двійковий формат:

	Десяткове	Шестнадцяткове
4 x 1 =	4	4
3 x 10 =	30	1E
2 x 100 =	200	C8
1 x 1000 =	1000	3E8
Результат:		04D2

Перевірте, що шест.04d2 дійсно відповідає десятковому 1234. В програмі 2.3 в процедурі B10asbi виконується перетворення ascii-числа 1234 в двійковий формат. У прикладі передбачається, що довжина ascii-числа дорівнює 4 і вона записана в полі ASCLEN. Для ініціалізації адреса ascii-поля Ascval-1 заноситься в регістр SI, а довжина - в регістр BX. Команда по мітці B20 пересилає ascii-байт в регістр AL:

```
MOV  AL,[SI+BX]
```

Тут використовується адреса Ascval-1 плюс вміст регістра BX (4), тобто виходить адреса Ascval+3 (найправіший байт поля ASCVAL). У кожному циклі вміст регістра BX зменшується на 1, що приводить до звернення до наступного зліва байта. Для даної адресації можна використовувати регістр BX, але не CX, і, отже, не можна застосовувати команду LOOP. У кожному циклі відбувається також множення поля Mult10 на 10, що дає в результаті множники 1,10,100 і так далі Такий

прийом застосований для більшій продуктивності множник можна зберігати в регістрі SI або DI.

Программа 2.3. Перетворення ASCII и двійкового форматів.

```
TITLE   EXCONV (COM) Перетворення ASCII та двійкових форматів
CODESG  SEGMENT PARA 'Code'
        ASSUME  CS:CODESG,DS:CODESG,SS:CODESG
        ORG     100H
BEGIN:  JMP     SHORT MAIN
; ----------------------------------------
ASCVAL  DB      '1234'              ;Елементи даних
BINVAL  DB      0
ASCLEN  DB      4
MULT10  DB      1
; ----------------------------------------
MAIN    PROC    NEAR                ;Основна процедура:
        CALL    B10ASBI             ;Викликати перетворення ASCII
        CALL    C10BIAS             ;Викликати перетворення двійкове
        RET
MAIN    ENDP
;               -------------------------------
;               Перетворення ASCII в двійкове:
;               -------------------------------
B10ASBI PROC
        MOV     CX,10               ;Фактор множення
        LEA     SI,ASCVAL-1         ;Адрес ASCVAL
        MOV     BX,ASCLEN           ;Длина ASCVAL
B20:
        MOV     AL,[SI+BX]          ;Вибрати ASCII-символ
        AND     AX,000FH            ;Очистити зону трійки
        MUL     MULT10              ;Помножити на фактор 10
        ADD     BINVAL,AX           ;Додати к двійковому
        MOV     AX,MULT10           ;Обчислити наступний
        MUL     CX                  ; фактор множення
        MOV     MULT10,AX
        DEC     BX                  ;останій ASCII-символ?
        JNZ     B20                 ; Ні - продовжити
        RET
B10ASBI ENDP

;               -------------------
;               Перетворення двійкового в ASCII:
;               -------------------
C10BIAS PROC
        MOV     CX,0010             ;Фактор ділення
        LEA     SI,ASCVAL+3         ;Адрес ASCVAL
        MOV     AX,BINVAL           ;Загрузити дв. число
C20:
        CMP     AX,0010             ;Значення меньше  10?
        JB      C30                 ; Да - вийти
        XOR     DX,DX               ;Очистити частину часного
        DIV     CX                  ;Поділити на 10

        OR      DL,30H
        MOV     [SI],DL             ;Записати ASCII-символ
        OEC     SI
        JMP     C20
        MOV     [SI],AL             ; как ASCII-символ C30:
```

```
        OR      AL,30H              ;Записати останнє частне

        RET
C10BIAS ENDP

CODESG  ENDS
        END     BEGIN
```

1.2.4 Перетворення двійкового формату в ASCII-формат

Для того, щоб відобразити на екрані арифметичний результат, необхідно перетворити його в ASCII-формат. Дана операція включає в себе процес, зворотний попередньому. Замість множення використовується ділення двійкового числа на 10 (0A), доки результат не буде менше 10. Залишки, які лежать в межах від 0 до 9, утворюють число в ASCII-форматі. В якості прикладу розглянемо перетворення 4D2H назад в десятковий формат:

		Частка	Залишок
4D2	:A	7B	4
7B	:A	C	3
C	:A	1	2

Так, як останнє число 1 менше ніж 0AH, то операція завершена. Залишки разом з останньою часткою утворюють результат в ASCII- форматі, записують справа наліво – 1234. Всі залишки і остання частка повинні записуватися в пам’ять з трійками, тобто 31323334.

В програмі 2.3 процедура C10bias перетворить шест.4d2 (результат обчислення в процедурі B10asbi) в ascii-число 1234.

2 Порядок виконання роботи

2.1 Напишіть текст програми згідно свого варіанту представлений в Таб.2.1. Якщо N≥10, то V=N (де N - номер списку в журналі, V – номер варіанту), інакше, якщо 11≤N≤20, то V=N-10, або 21≤N≤30, то V=N-20, або 31≤N≤40, то V=N-30 . *A, B, C* – вважати в десятковій формі.

2.2 Повторіть повний цикл створення програми для Турбо Асемблера див. Лабораторну роботу 1 рис.1.1.

2.3 Запустіть програму на виконання.

2.4 Проаналізуйте та збережіть результат програми.

Таблиця 2.1

№ варіанту	завдання
1	Написати програму для обрахунку *C=A+B* , де *A*, *B* – додатні числа, *A* та *B* ввести з клавіатури, *C* – вивести на екран.
2	Написати програму для обрахунку *C=A+B* , де *A>B*, *A* та *B* ввести з клавіатури, *C* – вивести на екран.
3	Написати програму для обрахунку *C=A-B* , де *A*, *B* – додатні числа, *A* та *B* ввести з клавіатури, *C* – вивести на екран
4	Написати програму для обрахунку *C=A-B* , де A>B, A та *B* ввести з клавіатури, *C* – вивести на екран.
5	Написати програму для обрахунку *C=A+B* , де *A,B* – можуть бути, як від'ємні, так і додатні, *A* та *B* ввести з клавіатури, *C* – вивести на екран.
6	Написати програму для обрахунку *C=A-B* , де *A,B* – можуть бути, як від'ємні, так і додатні, *A* та *B* ввести з клавіатури, *C* – вивести на екран.
7	Написати програму для обрахунку *C=2*(A+B)* , де *A*, *B* –

	додатні числа, *A* та *B* ввести з клавіатури, *C* – вивести на екран.
8	Написати програму для обрахунку *C=(A+B)/2* , де *A>B*, *A* та *B* ввести з клавіатури, *C* – вивести на екран.
9	Написати програму для обрахунку *C=2*(A-B)* , де *A*, *B* – додатні числа, *A* та *B* ввести з клавіатури, *C* – вивести на екран
10	Написати програму для обрахунку *C=(A-B)/2* , де *A>B*, А та *B* ввести з клавіатури, *C* – вивести на екран.

3 Зміст звіту

3.1 Назва та мета роботи

3.2 Блок-схема алгоритму програми свого варіанту

3.3 Текс програми

3.4 Результат виконання програми

3.5 Висновки по роботі

4 Контрольні питання

4.1 Назвіть порядок перетворення ASCII-формату в двійковий формат?

4.2 Назвіть порядок перетворення двійкового формату в ASCII-формату?

4.3 Для яких цілей застосовується ASCII-код?

4.4 Які групи коамнд для МП 1810ВМ86 ви знаєте?

4.5 Назвіть призначення регістрів AX, BX, CX, DX, SP, BP, DI, SI, IP, F, CS, DS,SS, ES?

4.6 За якими правилами виконують арифметичні операції в двійковій системі?

4.7 В який системі числення здійснюються арифметичні операції на комп’ютері?