**Лабораторна робота №4**

**Арифметичні та логічні операції процесора**  
Мета: Отримання знань про логічні команди і команди зсувів, а також придбання практичних навичок щодо застосування команд побітовій обробки при програмуванні на мові Асемблер.  
 **Основні теоретичні відомості.**

Команди побітовій обробки даних маніпулюють групами бітів в регістрах або комірках пам'яті.  
Існує 3 групи команд побітовій обробки:  
- Логічні команди;  
- Команди зсуву;  
- Команди циклічного зсуву.  
   
**1. Логічні команди**  
До логічних відносяться команди AND, OR, XOR, TEST, NOT.

Формат перших чотирьох команд: *команда отримувач, джерело*  
Формат команди NOT: *NOT отримувач*Логічні команди обробляють один байт або слово в регістрі або в пам'яті і встановлюють прапори CF, OF, PF, SF, ZF (прапор AF не визначено). При цьому обробка байта (слова) здійснюється безпосередньо біт за бітом.  
Логічні операції часто використовуються для установки в двійкові розряди необхідних значень. Існують наступні способи отримання подібних результатів:  
- Для установки в даний розряд 0 необхідно логічно помножити розряд на 0 (команда  
AND);  
- Для установки в даний розряд 1 необхідно логічно скласти розряд з 1 (команда OR).

Розглянемо дії логічних команд:  
**AND**: Якщо обидва з порівнюваних бітів отримувача і джерела рівні одиниці, то результуючий біт дорівнює одиниці; у всіх інших випадках результат дорівнює нулю.  
**OR**: Якщо хоча б один з двох порівнюваних бітів отримувача і джерела дорівнює одиниці, то результуючий біт приймача дорівнює одиниці; якщо порівнювані біти дорівнюють нулю, то результат дорівнює нулю.  
**XOR**: Якщо один з порівнюваних бітів дорівнює нулю, а інший дорівнює одиниці, то результуючий біт дорівнює одиниці; якщо обидва біта однакові, то результат дорівнює нулю.  
**NOT**: Встановлює зворотне значення бітів у байті або слові, регістрі або комірці пам'яті.

**TEST**: Діє як AND, встановлює прапори, але не змінює біти.  
*Приклади*,  
MOV AL, 11000101B MOV BH, 01011100B  
AND AL, BH; встановлює в AL 010001000  
OR CL, CL; встановлює прапори CF і ZF  
  
**2. Команди зсуву**  
Команди зсуву здійснюють зсув 8 - або 16-бітового вмісту регістрів на одну або кілька позицій вліво або вправо.

Формат: *команда отримувач, лічильник*  
У цих командах вміст отримувача зсувається на величину, що задається лічильником.  
При цьому лічильник може бути цифрою 1 або значенням без знаку в регістрі CL.  
Команди зсуву діляться на команди логічного, арифметичного і циклічного зсуву.  
   
***2.1. Команди арифметичного зсуву*****SAL / SAR** - зрушити арифметично вліво / вправо.  
Команда SAL не зберігається знак операнда, але заносить 1 в прапор переповнення CF і 0 у біт, що звільняється. Команда SAR зберігає знак операнда, заносячи його при кожному зсуві в старший біт операнда. Наприклад,  
MOV BL, 10110100B  
MOV AL, BL; нехай CF = 1  
SAL AL, 1; AL = 01101000, CF = 1  
MOV AL, BL  
SAR AL, 1; AL = 11011010, CF = 0  
  
***2.2. Команди логічного зсуву***:  
**SHL / SHR** - зрушити логічно вліво / вправо.  
Команда SHL ідентична команді SAL. Команда SHR аналогічна SAR, але зрушує операнд вправо, заносячи 0 в старший біт операнда. Приклади,  
MOV BL, 10110100B  
MOV AL, BL; нехай CF = 1  
SHL AL, 1; AL = 01101000, CF = 1  
MOV AL, BL  
SHR AL, 1; AL = 01011010, CF = 0  
Оскільки зсув операнда на 1 біт подвоює значення операнда, а зрушення на 1 біт вправо зменшує його вдвічі, то команди зсуву можна використовувати в якості команд швидкого множення і ділення.  
Приклади,  
MOV CL, 2  
SHL AX, CL; помножити число без знаку на 4  
  
**2.3. Команди циклічного зсуву**:  
**ROL / ROR** - зрушити циклічно вліво / вправо.  
При виконанні цих команд біт, що вийшов за межі операнда, входить до нього з іншого кінця і поміщається у прапор переносу. Приклади,  
MOV BL, 10110100B; нехай CF = 1  
MOV AL, BL  
ROL AL, 1; AL = 01101001, CF = 1  
  
***RСL / RСR*** - зрушити циклічно разом з прапором перенесення вліво / вправо.  
При виконанні цих команд здійснюється зсув, в протилежний кінець операнда поміщається значення прапора CF і потім в CF поміщається біт, що вийшов за межі операнда.  
Приклади,  
MOV BL, 10110100B; нехай CF = 1  
MOV AL, BL  
RCR AL, 1; AL = 11011010, CF = 0  
  
***2.4. Найпростіші способи введення символів з клавіатури і виведення символів на екран***.  
Для виведення символів на екран в поточній позиції курсору, необхідно:  
- Визначити область даних для виведення при цьому останнім символом області виведення повинен бути знак $;

- Занести в область даних для виведення необхідні дані;  
- Встановити в регістрі AH значення 09;  
- Занести в регістр DX адреса області даних для виводу;  
- Вказати команду переривання INT 21H.  
*Приклад.*Для виведення на екран тексту 'Введіть дані' необхідні наступні команди:

STR DB 'Введіть дані', '$'  
...

MOV AH, 09

LEA DX, STR

INT 21H  
Для введення послідовності символів з клавіатури необхідно:  
- Визначити область пам'яті в сегменті даних для символів, що вводять, причому довжина цієї області повинна бути на 2 байти більше максимальної довжини рядка з урахуванням того, що останній введений символ завжди буде 'RETURN' (код 0DH). Перший байт містить максимальну кількість символів, а другий байт буде містити число реально введених символів до натискання клавіші 'RETURN';  
- Встановити в регістрі AH значення 10;  
- Помістити в регістр DX адресу області пам'яті;  
- Вказати команду переривання INT 21H.  
Приклад.  
Для введення з клавіатури рядка максимальною довжиною в 50 символів необхідні наступні команди:  
STR DB 50,52 DUP (?)  
...

MOV AH, 10

LEA DX, STR

INT 21H  
  
**Приклад програми**  
  
Наступна програма вводить рядок символів латинського алфавіту і замінює усі малі літери на прописні, а прописні - на рядкові. Результат виводиться на екран. Наприклад, рядок TablE буде замінена на рядок tABLe і виведеться на екран.  
  
STACKSG SEGMENT PARA STACK 'stack'

DB 64 DUP (?)  
STACKSG ENDS  
DATASG SEGMENT PARA 'DATA'

STR DB 10,12 DUP ('')

DATASG ENDS  
CODESG SEGMENT PARA 'CODE'  
ASSUME CS: CODESG, DS: DATASG, SS: STACKSG

ENTRY PROC FAR

PUSH DS

SUB AX, AX

PUSH AX  
MOV AX, DATASG  
MOV DS, AX

; Введення вихідного рядка

MOV AH, 10  
LEA DX, STR

INT 21H

; Блок заміни символу 'A' на 'a'

MOV SI, 2

SUB CX, CX; CX = CX-CX

MOV CL, STR +1; довжина рядка  
M1: MOV AL, STR [SI]; занести в AL перший символ

CMP AL, 'A'; <'A'?

JB NEXT1

CMP AL, 'z';> 'z'?

JA NEXT1

CMP AL, 'a'; <'a'?

JB MARK1  
;  
XOR AL, 00100000B; замінити  
MOV STR [SI], AL; занести новий символ в рядок  
JMP SHORT NEXT1

MARK1: OR AL, 00100000B; замінити

MOV STR [SI], AL; занести новий символ в рядок  
NEXT1: INC SI; перейти до наступного символу (SI = SI +1)

LOOP M1  
; Друк результуючого рядка  
PRINT1:

MOV Ah, 09h

LEA DX, STR

SUB BX, BX

MOV BL, STR +1

MOV STR [BX +2], '$'

MOV STR, 0Ah

MOV STR +1,0 Dh

INT 21H  
RET

ENTRY ENDP

CODESG ENDS

END ENTRY  
  
**Варіанти завдань на лабораторну роботу**1. Ввести з терміналу рядок символів. Вивести на екран їх двійкове подання.  
2. Ввести з терміналу рядок символів. Вивести на екран кількість одиничних бітів третього символу рядка.  
3. Дано двійкове число. Вивести на екран його шістнадцяткове подання.  
4. Дано двійкове число. Вивести на екран його вісімкове подання.  
5. Ввести з терміналу рядок символів. Вивести на екран кількість нульових бітів другого символу рядка.  
6. Розробити програму швидкого множення числа довжиною подвійне слово.  
7. Розробити програму швидкого ділення числа без знака довжиною подвійне слово.