Лабораторна робота №3

**org 100h** – оголошується адреса розміщення програми в памяті

**.model smal**l — директива організації памяти(буде використана тільки внутрішня память процесора)

**.data** — сегмент даних ( оголошується змінні та масив )

**mass dd 12, 34, 56, 78, 90** ; оголошення масиву чисел

**dess dd 64 dup(?)**  ; масив слів довжиною 64 розряди

**lns dd 10** ; кількість циклів під час перебору елементів масива

**adx dd ?** ; змінні під які оголошується пам’ять

**abx dd ?**  ; save adress from bx

**xw dd ?**

**wr dd ?** ;кінцевий результат

**.code**

**mov ax, 3**

**int 10h**

**mov ax, 1003h**

**mov bx, 0**

**int 10h**

**push ax** зберігаємо попереднє значення регістру ах у стеку

**mov ax, 0b800h** записуємо до регістру ах 16 значення

**mov xw, 01h ; load xw number 1** — записуємо 16 значення до неявно оголошенної змінної xw

**xor si, si** ; обнуляємо регістр si , за допомогою логічної команди xor(будемо використовувати його як лічильник адреси , при доступі до елементів масиву)

mov cx, lns ; обявляємо кількість елементів у масиві , заносимо значення змінної lns до регістра лічильника cx(є лічильником за замовчуванням)

**cycle:** - оголошуємо мітку початку циклу при виконані тіла цикла, значення сх буде зменшуватись на 1

**mov ax, mass[si]** зчитуємо перший елемент масива , який знаходиться за адресою 0 , бо si= 0. та заносимо до регістру ах

**add ax, xw** - додаємо до зчитаного значення ах (0b800h) + перший елемент масива mass[si]

**mov dess[si], ax**  заносимо значення ах до масиву dess , за адресою 0 , за допомогою команди

**add si,4** ; додаємо до адреси si 4 біти

**loop cycle** повертаємось до наступного елементу цикла

**mov cx, lns** — заносимо кількість виконання циклів у сх

**xor si, si**  — обнуляємо регістр si

**result:** - мітка циклу

**mov ax,dess[si]** — значення масиву dess заносимо у регістр ах

**mov wr, ax** — переносимо це значення у змінну wr

**call view** — викликаємо підпрограму view

**add si, 4** — додаємо до лічильника адреси 4

**cmp si, 16** — порівняння si з 16 , якщо si більше 16 перейти на мітку js stop

**js stop** — примусово зупинити цикл

**loop result** — повернення до наступного елементу цикла (щоб взяти інший елемент)

**jmp stop**

**view:** — ім’я підпрограми

**mov bx, wr** — перенесення значення змінної wr у регістр bx

Організація нового цикла, який виводить значення регістру на екран

(внутрішній цикл виводу 8 розрядів bl на екран)

**mov cx, 8** —

**print: mov ah, 2** —

**mov dl, '0'** —

**test bl, 10000000b** —

**jz zero** — порівняння з нулем

**mov dl, '1'** —

**zero: int 21h** —

**shl bl, 1** —

**loop print**  —

**mov dl, ' '** —

**int 21h** — переривання програми для того щоб відобразити значення на екран

**ret** — оператор , що повертає виконання у точку входу(завершення тіла підпрограми)

**stop:** — примусова зупинка

**mov ah, 0** — обнуляємо значення регістра ah ,заносимо вектор переривання завершення основної програми

**int 16h** — виклакаємо переривання зупинки програми

**ret** — оператор що вказує на завершення основної програми