Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра системного програмування
і спеціалізованих комп'ютерних системи

Лабораторна робота №1

з дисципліни
«Архітектура комп'ютерів 2. Програмне забезпечення»

Виконав:
Студент групи КВ-83
Лазуткін Олег

Перевірив:
Молчанов О. А.

Київ – 2021

Загальне завдання

1. Реалізувати програму сортування масиву згідно із варіантом мовою С.
2. Виконати трансляцію програми, написаної мовою С, в асемблерний код за допомогою **gcc** й встановити семантичну відповідність між командами мови С та командами одержаного асемблерного коду, додавши відповідні коментарі з поясненням.

Варіант № 12

Задано двовимірний масив (матрицю) цілих чисел $A[m,n]$. Відсортувати окремо кожен рядок масиву алгоритмом №3 методу вставки (з лінійним пошуком справа з використанням бар'єру) за незбільшенням.

1. Лістинг програми мовою С

```
void sort(int m, int n, int array[m][n])
{
    int j;
    for (int row = 0; row < m; row++)
    {
        for (int column = 2; column < n; column++)
        {
            array[row][0] = array[row][column];
            j = column;
            while (array[row][0] > array[row][j - 1])
            {
                array[row][j] = array[row][j - 1];
                j = j - 1;
            }
            array[row][j] = array[row][0];
        }
    }
}
```

2. Лістинг програми мовою асемблера з поясненнями

```
        .text
        .globl  sort
        .def    sort;   .scl    2;      .type   32;     .endef
        .seh_proc       sort
sort:
        push    rbp
        .seh_pushreg    rbp
        mov     rbp, rsp
        .seh_setframe   rbp, 0
        sub     rsp, 32
        .seh_stackalloc 32
        .seh_endprologue
        // start comments
        // associate actuals with formals
        mov     DWORD PTR 16[rbp], ecx          // associate actual value with formal m
        mov     DWORD PTR 24[rbp], edx          // associate actual value with formal n
        mov     QWORD PTR 32[rbp], r8       // associate actual value with formal array
        mov     eax, DWORD PTR 24[rbp]
        movsx   rdx, eax
        sub     rdx, 1
        mov     QWORD PTR -24[rbp], rdx
        movsx   rdx, eax
        mov     r9, rdx
        mov     r10d, 0
        // function body start
        // first for loop start
        mov     DWORD PTR -8[rbp], 0        // initilize row: int row = 0
        jmp     .L2                         // jump to first for loop condition check
.L7:
        // first for loop body start
        // second for loop start
        mov     DWORD PTR -12[rbp], 2       // initilize column: int column = 0
        jmp     .L3                         // jump to second for loop condition check
.L6:
        // second for loop body start
        mov     edx, DWORD PTR -8[rbp]
        movsx   rcx, edx
        movsx   rdx, eax
        imul    rdx, rcx
        lea     rcx, 0[0+rdx*4]
        mov     rdx, QWORD PTR 32[rbp]
        add     rcx, rdx
        mov     edx, DWORD PTR -8[rbp]
        movsx   r8, edx
        movsx   rdx, eax
        imul    rdx, r8
        lea     r8, 0[0+rdx*4]
        mov     rdx, QWORD PTR 32[rbp]
        add     r8, rdx
        mov     edx, DWORD PTR -12[rbp]
        movsx   rdx, edx
```

```
        mov     edx, DWORD PTR [r8+rdx*4]
        mov     DWORD PTR [rcx], edx            // array[row][0] = array[row][column]
        mov     edx, DWORD PTR -12[rbp]
        mov     DWORD PTR -4[rbp], edx          // j = column
        // while loop start
        jmp     .L4                             // jump to while loop condition check
.L5:
        // while loop body start
        mov     edx, DWORD PTR -8[rbp]
        movsx   rcx, edx
        movsx   rdx, eax
        imul    rdx, rcx
        lea     rcx, 0[0+rdx*4]
        mov     rdx, QWORD PTR 32[rbp]
        lea     r8, [rcx+rdx]
        mov     edx, DWORD PTR -8[rbp]
        movsx   rcx, edx
        movsx   rdx, eax
        imul    rdx, rcx
        lea     rcx, 0[0+rdx*4]
        mov     rdx, QWORD PTR 32[rbp]
        add     rcx, rdx
        mov     edx, DWORD PTR -4[rbp]
        sub     edx, 1
        movsx   rdx, edx
        mov     ecx, DWORD PTR [rcx+rdx*4]
        mov     edx, DWORD PTR -4[rbp]
        movsx   rdx, edx
        mov     DWORD PTR [r8+rdx*4], ecx       // array[row][j] = array[row][j - 1]
        sub     DWORD PTR -4[rbp], 1            // j = j-1
        // while loop body end
.L4:
        // while loop condition start
        mov     edx, DWORD PTR -8[rbp]
        movsx   rcx, edx
        movsx   rdx, eax
        imul    rdx, rcx
        lea     rcx, 0[0+rdx*4]
        mov     rdx, QWORD PTR 32[rbp]
        add     rdx, rcx
        mov     ecx, DWORD PTR [rdx]
        mov     edx, DWORD PTR -8[rbp]
        movsx   r8, edx
        movsx   rdx, eax
        imul    rdx, r8
        lea     r8, 0[0+rdx*4]
        mov     rdx, QWORD PTR 32[rbp]
        add     r8, rdx
        mov     edx, DWORD PTR -4[rbp]
        sub     edx, 1
        movsx   rdx, edx
        mov     edx, DWORD PTR [r8+rdx*4]
        cmp     ecx, edx                        // array[row][0] > array[row][j - 1]
        jg      .L5
```

```
        // while loop condition end
        // while loop end
        mov     edx, DWORD PTR -8[rbp]
        movsx   rcx, edx
        movsx   rdx, eax
        imul    rdx, rcx
        lea     rcx, 0[0+rdx*4]
        mov     rdx, QWORD PTR 32[rbp]
        lea     r8, [rcx+rdx]
        mov     edx, DWORD PTR -8[rbp]
        movsx   rcx, edx
        movsx   rdx, eax
        imul    rdx, rcx
        lea     rcx, 0[0+rdx*4]
        mov     rdx, QWORD PTR 32[rbp]
        add     rdx, rcx
        mov     ecx, DWORD PTR [rdx]
        mov     edx, DWORD PTR -4[rbp]
        movsx   rdx, edx
        mov     DWORD PTR [r8+rdx*4], ecx       // array[row][j] = array[row][0]
        // second for loop body end
        add     DWORD PTR -12[rbp], 1           // increment column++
.L3:
        // second for loop condition start
        mov     edx, DWORD PTR -12[rbp]
        cmp     edx, DWORD PTR 24[rbp]]         // column < n
        jg      .L6
        // second for loop condition end
        // second for loop end
        // first for loop body end
        add     DWORD PTR -8[rbp], 1            // increment row++
.L2:
        // first for loop condition start
        mov     edx, DWORD PTR -8[rbp]
        cmp     edx, DWORD PTR 16[rbp]          // row < m
        jl      .L7
        // first for loop condition end
        // first for loop end
        nop
        add     rsp, 32
        pop     rbp
        ret
        // function body end
```