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-82
Іваненко Олександр Андрійович

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.

Завдання за варіантом 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_mat(int a[3][4], int m, int n) {
    int l;
    for(int k = 0; k < m; k++){
        for (int i = 2; i < n+1; i++){
            a[k][0] = a[k][i];
            l = i;
            while (a[k][0] < a[k][l-1]){
                a[k][l] = a[k][l - 1];
                l--;
            }
            a[k][l] = a[k][0];
        }
    }
}
```

2. Лістинг програми мовою асемблера з поясненнями

```
sort_mat:
        push   rbp
        .seh_pushreg rbp
        mov    rbp, rsp
        .seh_setframerbp, 0
        sub    rsp, 16
        .seh_stackalloc     16
        .seh_endprologue
        // start comments
      // associate actuals with formals
        mov    QWORD PTR 16[rbp], rcx   // associate actual value with formal a
        mov    DWORD PTR 24[rbp], edx   // associate actual value with formal m
        mov    DWORD PTR 32[rbp], r8d   // associate actual value with formal n
        //function body

        //first loop start
        mov    DWORD PTR -8[rbp], 0  //int k = 0;
        jmp    .L7 //jump to first loop condition check

.L12:  //body first loop
       //second loop start
        mov    DWORD PTR -12[rbp], 2 //int i = 2;
        jmp    .L8 //jump to second loop condition check

.L11: //body second loop
        //a[k][0] = a[k][i];
        //a[k][0]
        mov    eax, DWORD PTR -8[rbp] //save value of k to eax
        cdqe // extend value in EAX to RAX preserving sign
        sal    rax, 4 //left shift on 4 bytes
        mov    rdx, rax //save RAX value to RDX
        mov    rax, QWORD PTR 16[rbp] //save arr start address to RAX
```

```
        lea    rcx, [rdx+rax]  // calculate and save shift in arr to RCX

        //a[k][i]
        mov    eax, DWORD PTR -8[rbp] //save value of k to eax
        cdqe // extend value in EAX to RAX preserving sign
        sal    rax, 4 //left shift on 4 bytes
        mov    rdx, rax //save RAX value to RDX
        mov    rax, QWORD PTR 16[rbp] //save arr start address to RAX
        add    rdx, rax //save RAX value to RDX
        mov    eax, DWORD PTR -12[rbp] //save i to EAX
        cdqe
        mov    eax, DWORD PTR [rcx+rax*4] //save value of a[k][i] to EAX
                                    //a[k][0] = a[k][i];
        mov    DWORD PTR [rdx], eax //save value of EAX to a[k][0]

        //l = i;
        mov    eax, DWORD PTR -12[rbp] //save i to EAX
                                    //l = i;
        mov    DWORD PTR -4[rbp], eax  //save eax to l

        //while loop start
        jmp    .L9 //jump to while loop condition

.L10: //while body

        //a[k][l] = a[k][l - 1];
        mov    eax, DWORD PTR -8[rbp] //save value of k to eax
        cdqe // extend value in EAX to RAX preserving sign
        sal    rax, 4 //left shift on 4 bytes
        mov    rdx, rax //save RAX value to RDX
        mov    rax, QWORD PTR 16[rbp] //save arr start address to RAX
        lea    rcx, [rdx+rax] // calculate and save shift in arr to RCX
```

```
        mov    eax, DWORD PTR -4[rbp] //save l value to eax
        lea    r8d, -1[rax] // calculate and save shift in arr to R8X

        mov    eax, DWORD PTR -8[rbp] //save value of k to eax
        cdqe // extend value in EAX to RAX preserving sign
        sal    rax, 4 //left shift on 4 bytes
        mov    rdx, rax //save RAX value to RDX
        mov    rax, QWORD PTR 16[rbp] //save arr start address to RAX
        add    rdx, rax // create address of certain arr[k]
        movsx  rax, r8d // move r8d value to rax
        mov    ecx, DWORD PTR [rcx+rax*4] //save value of a[k][l - 1] to ECX
        mov    eax, DWORD PTR -4[rbp] //save l value to eax
        cdqe
                                        //a[k][l] = a[k][l - 1];
        mov    DWORD PTR [rdx+rax*4], ecx //save value of a[k][l - 1] to a[k][l]

        //l--;
        sub    DWORD PTR -4[rbp], 1
        //while body end

.L9://while condition

        mov    eax, DWORD PTR -8[rbp] //save value of k to eax
        cdqe // extend value in EAX to RAX preserving sign
        sal    rax, 4 //left shift on 4 bytes
        mov    rdx, rax  //save RAX value to RDX
        mov    rax, QWORD PTR 16[rbp] //save arr start address to RAX
        add    rax, rdx // create address of certain arr[k]
        mov    ecx, DWORD PTR [rax]  // create address of certain arr[k][0]

        mov    eax, DWORD PTR -8[rbp] //save value of k to eax
        cdqe // extend value in EAX to RAX preserving sign
        sal    rax, 4 //left shift on 4 bytes
```

```
        mov    rdx, rax //save RAX value to RDX
        mov    rax, QWORD PTR 16[rbp] //save arr start address to RAX
        add    rdx, rax // create address of certain arr[k]
        mov    eax, DWORD PTR -4[rbp] // move l to eax
        sub    eax, 1 // l - 1
        cdqe
        mov    eax, DWORD PTR [rdx+rax*4] // create address of certain arr[k][l-1]
        cmp    ecx, eax // a[k][0] < a[k][l-1]
        jl     .L10 //jump to while body if ecx 'a[k][0]' < eax 'a[k][l-1]'
        //while loop end

        //a[k][l] = a[k][0];
        mov    eax, DWORD PTR -8[rbp] //save value of k to eax
        cdqe // extend value in EAX to RAX preserving sign
        sal    rax, 4 //left shift on 4 bytes
        mov    rdx, rax //save RAX value to RDX
        mov    rax, QWORD PTR 16[rbp] //save arr start address to RAX
        add    rax, rdx // create address of certain arr[k]
        mov    edx, DWORD PTR -8[rbp] //save value of k to edx
        movsx  rdx, edx //save to rdx value of edx
        mov    rcx, rdx
        sal    rcx, 4 //left shift rcx on 4 bytes
        mov    rdx, QWORD PTR 16[rbp] //save arr start address to RDX
        add    rdx, rcx //create address of a[k][l]
        mov    ecx, DWORD PTR [rax] //save to ecx value a[k][0];
        mov    eax, DWORD PTR -4[rbp]
        cdqe
                //a[k][l]
        mov    DWORD PTR [rdx+rax*4], ecx //a[k][l] = a[k][0];
        //body second loop end
        add    DWORD PTR -12[rbp], 1 //i++;
.L8: //second loop condition
        mov    eax, DWORD PTR 32[rbp] // save value of i to eax
```

```
                                    // i < n+1
        cmp    eax, DWORD PTR -12[rbp]  // compare eax(i value) and formal n
        jge    .L11 // jump to body second loop if i < n+1
        add    DWORD PTR -8[rbp], 1 // k++;
      //second loop end
      //body first loop end

.L7: //first loop condition
        mov    eax, DWORD PTR -8[rbp]  // save value of k to eax
                                    // k < m
        cmp    eax, DWORD PTR 24[rbp]  // compare eax(k value) and formal m
        jl     .L12      // jump to body first loop if k < m
        //first loop end

        nop
        add    rsp, 16
        pop    rbp
        ret        //end function
        .seh_endproc
```