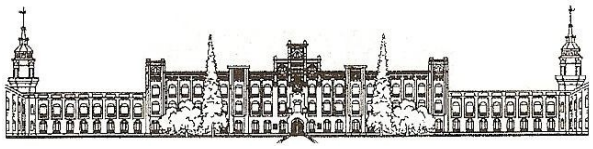

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА СИСТЕМНОГО ПРОГРАМУВАННЯ ТА СПЕЦІАЛІЗОВАНИХ КОМП’ЮТЕРНИХ СИСТЕМ

Лабораторна робота №1

з дисципліни «Архітектура для програмістів»

Виконав студент групи: КВ-22

ПІБ: Крутогуз Максим Ігорович

Перевірив: Молчанов О. А.

Київ 2025

Трансляція мов високого рівня у мови низького рівня. Частина 1

Мета лабораторної роботи наступна:

- ознайомитись із роботою сучасних трансляторів на прикладі трансляції невеликої програми, що містить різні конструкції керування потоком виконання і написаної мовою програмування високого рівня, у код цієї програми мовою низького рівня;

- навчитись виокремлювати синтаксичні/семантичні конструкції програми, записаної мовою високого рівня, у відповідній їй програмі, записаній мовою низького рівня на прикладі мови високого рівня С

Загальне завдання та термін виконання

Завдання лабораторної роботи складається з трьох пунктів:

1. Реалізувати програму сортування масиву згідно із варіантом мовою С (інформація про варіанти наведена в п. 4). Результатом виконання цього пункту є лістинг програми 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. Результатом виконання даного пункту буде лістинг асемблерного коду програми із коментарями в коді, в яких наведено відповідний код програми, записаною мовою С (приклад. див. в п. 3.2).

3. . Розібратись і вміти пояснити, що виконують ті чи інші команди мови асемблера, що будуть присутні в коді мовою асемблера, отриманого в другому пункті загального завдання; вміти пояснити зв’язок між кодом мовою С та кодом мовою асемблера.

Варіант #12

Задано двовимірний масив (матрицю) цілих чисел A[m][n]. Відсортувати окремо кожен рядок масиву алгоритмом No3 методу вставки (з лінійним пошуком справа з використанням бар’єру) за не- збільшенням.

Код 1 — Лістинг мовою C

```c
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#define N 5
#define M 6

void fillArray(int*, int, int);
void sortArray(int*, int, int);
void printArray(int*, int, int);

int main() {
    srand(time(NULL));
    int A[M][N+1];

    fillArray(&A[0][0], M, N + 1);

    printArray(&A[0][0], M, N + 1);

    sortArray(&A[0][0], M, N + 1);

    printArray(&A[0][0], M, N + 1);

    return 0;
}

void fillArray(int *arr, int m, int n) {
    for (int i = 0; i < m; i++) {
        for (int j = 1; j < n; j++) {
            arr[i * n + j] = rand() % 100;
        }
    }
}

void sortArray(int *arr, int m, int n) {
    for (int i = 0; i < m; i++) {
        for (int j = 2; j < n; j++) {
            arr[i*n + 0] = arr[i*n + j];
            int k = j;
            while (arr[i*n + 0] < arr[i*n + k - 1]) {
                arr[i* n + k] = arr[i*n + k - 1];
                k--;
            }

            arr[i * n + k] = arr[i * n + 0];
        }
    }
}
```

```c
void printArray(int *arr, int m, int n) {
    for (int i = 0; i < m; i++) {
        for (int j = 1; j < n; j++) {
            printf("%3d", arr[i * n + j]);
        }
        printf("\n");
    }
    printf("\n");
}
```

Код 2 — Лістинг мовою асемблера

```
sortArray:
        push    rbp
        mov     rbp, rsp
        ; Saving values of params into a stack
        mov     QWORD PTR [rbp-24], rdi ; matrix adress
        mov     DWORD PTR [rbp-28], esi ; m param
       mov     DWORD PTR [rbp-32], edx ; n param sortArray(int *arr,int m,int n)
        ; function body start
        ; i loop start
        mov     DWORD PTR [rbp-4], 0 ; i = 0
        jmp     .L9
.L14:
        ; j loop start
        mov     DWORD PTR [rbp-8], 2 ; j = 2
        jmp     .L10
.L13:
        mov     eax, DWORD PTR [rbp-4] ; eax = i
        imul    eax, DWORD PTR [rbp-32] ; eax = eax * n
        mov     edx, eax ; edx = eax
        mov     eax, DWORD PTR [rbp-8] ; eax = j
        add     eax, edx ; eax = eax + edx
        cdqe ; rax = eax (sign-extend of EAX)
         lea     rdx, [0+rax*4] ; getting address of displacement in array: 4*(i
* n + j)
        mov     rax, QWORD PTR [rbp-24] ; rax = arr
         lea     rcx, [rdx+rax] ; getting address of element in matrix in memory
(address in the right side of expression)
        mov     eax, DWORD PTR [rbp-4] ; eax = i
        imul    eax, DWORD PTR [rbp-32] ; eax = eax * n
        cdqe ;rax = eax (sign-extend of EAX)
         lea     rdx, [0+rax*4] ; getting address of displacement in array: 4*(i
* n + 0)
        mov     rax, QWORD PTR [rbp-24] ; rax = arr
        add     rdx, rax ; rdx = rdx + rax
        mov     eax, DWORD PTR [rcx] ; eax = arr[i * n + j]
          mov     DWORD PTR [rdx], eax ; arr[i * n + 0] = eax arr[i*n + 0] =
arr[i*n + j]
        mov     eax, DWORD PTR [rbp-8] ; eax = j
        mov     DWORD PTR [rbp-12], eax ; k = eax int k = j
        ; while loop start
        jmp     .L11
```

```
.L12:
        mov     eax, DWORD PTR [rbp-4] ; eax = i
        imul    eax, DWORD PTR [rbp-32] ; eax = eax * n
        mov     edx, eax ; edx = eax
        mov     eax, DWORD PTR [rbp-12] ; eax = k
        add     eax, edx ; eax = eax + edx : i * n + k
        cdqe ;rax = eax (sign-extend of EAX)
        sal     rax, 2 ; rax = 4*rax
        lea     rdx, [rax-4] ; the displacement i * n + k - 1
        mov     rax, QWORD PTR [rbp-24] ; rax = arr
        lea     rcx, [rdx+rax] ; address of arr[i * n + k – 1] in memory
        mov     eax, DWORD PTR [rbp-4] ; eax = i
        imul    eax, DWORD PTR [rbp-32] ; eax = eax * n
        mov     edx, eax ; edx = eax
        mov     eax, DWORD PTR [rbp-12] ; eax = k
        add     eax, edx ; eax = i * n + k
        cdqe ; rax = eax (sign-extend of EAX)
        lea     rdx, [0+rax*4] ; load effective adrress of arr[i * n + k]
        mov     rax, QWORD PTR [rbp-24] ; rax = arr
        add     rdx, rax ; arr[i * n + k] in memory
        mov     eax, DWORD PTR [rcx] ; eax =  arr[i * n + k – 1]
         mov      DWORD PTR [rdx], eax ; arr[i * n + k] = eax arr[i* n + k] =
arr[i*n + k - 1];
        sub     DWORD PTR [rbp-12], 1 ; k--
.L11:
        mov     eax, DWORD PTR [rbp-4]
        imul    eax, DWORD PTR [rbp-32]
        cdqe
        lea     rdx, [0+rax*4]
        mov     rax, QWORD PTR [rbp-24]
        add     rax, rdx
        mov     edx, DWORD PTR [rax] ; address in memory of arr[i*n + 0]
        mov     eax, DWORD PTR [rbp-4]
        imul    eax, DWORD PTR [rbp-32]
        mov     ecx, eax
        mov     eax, DWORD PTR [rbp-12]
        add     eax, ecx
        cdqe
        sal     rax, 2
        lea     rcx, [rax-4]
        mov     rax, QWORD PTR [rbp-24]
        add     rax, rcx
        mov     eax, DWORD PTR [rax] ; address in memory of arr[i*n + k - 1]
        cmp     edx, eax ; loop while condition check
        jl      .L12
        ; while loop end while (arr[i*n + 0] < arr[i*n + k - 1])
        mov     eax, DWORD PTR [rbp-4]
        imul    eax, DWORD PTR [rbp-32]
        cdqe
        lea     rdx, [0+rax*4]
        mov     rax, QWORD PTR [rbp-24]
        lea     rcx, [rdx+rax] ; adress in memory of arr[i * n + 0]
        mov     eax, DWORD PTR [rbp-4]
        imul    eax, DWORD PTR [rbp-32]
```

```
        mov     edx, eax
        mov     eax, DWORD PTR [rbp-12]
        add     eax, edx
        cdqe
        lea     rdx, [0+rax*4]
        mov     rax, QWORD PTR [rbp-24]
        add     rdx, rax
        mov     eax, DWORD PTR [rcx] ; getting value from arr[i * n + 0]
         mov      DWORD PTR [rdx], eax ; assigning got value into arr[i* n + k]
arr[i * n + k] = arr[i * n + 0]
        add     DWORD PTR [rbp-8], 1 ; increment of j
.L10:
        mov     eax, DWORD PTR [rbp-8]
        cmp     eax, DWORD PTR [rbp-32] ; compare j < n
        jl      .L13
        ; j loop end for (int j = 2; j < n; j++)
        add     DWORD PTR [rbp-4], 1 ; increment of i
.L9:
        mov     eax, DWORD PTR [rbp-4]
        cmp     eax, DWORD PTR [rbp-28] ; compare i < m
        jl      .L14
        ; i loop end for (int i = 0; i < m; i++)

        nop
        nop
        pop     rbp
        ret
        ; function end
```

Тестування:

```
65 60 46 24 96
 8 21 54 23 58
94 64 55 78 82
50 56 33 94  4
79 64 68 40 14
88 84 78 98 36

24 46 60 65 96
 8 21 23 54 58
55 64 78 82 94
 4 33 50 56 94
14 40 64 68 79
36 78 84 88 98
```