НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»
ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Лабораторна робота № 1.1

з дисципліни

«Архітектура для програмістів»

Тема:

"ТРАНСЛЯЦІЯ МОВ ВИСОКОГО РІВНЯ У МОВИ НИЗЬКОГО РІВНЯ. Ч.1"

Виконав: студент III курсу
ФПМ групи КВ-94
Кувашов Я.Р.
Перевірив: Молчанов О.А.

Київ 2022

Загальне завдання

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.

Завдання за варіантом 10

Відсортувати побічну діагональ масиву алгоритмом No2 методу обмінів («бульбашкове сортування» з використанням «прапорця») за незбільшенням.

Лістинг програми мовою С

```
void sort(int n, int  A[n][n])
{   int tmp,i,j;
int R = n-1;
_Bool flag = 1 ;
while (flag == 1)
          { flag = 0;

     for(  i = 0;i<R;i++)
         if(A[i+1][n-2-i]>A[i][n-1-i])
                     {
              tmp = A[i+1][n-2-i];
A[i+1][n-2-i] = A[i][n-1-i];
 A[i][n-1-i] = tmp;
flag = 1;                          }
```

```
            R--;
              }
}
```

Лістинг програми мовою асемблера

```
 sort:
        push    rbp
        mov     rbp, rsp
        push    rbx
        mov     DWORD PTR [rbp-52], edi
        mov     QWORD PTR [rbp-64], rsi
; function starts
        mov     eax, DWORD PTR [rbp-52]
        movsx   rdx, eax
        sub     rdx, 1
        mov     QWORD PTR [rbp-40], rdx
        movsx   rdx, eax
        mov     rcx, rdx
        mov     ebx, 0
        mov     edx, DWORD PTR [rbp-52]     ; initialize n
        sub     edx, 1                      ; n-1
        mov     DWORD PTR [rbp-24], edx     ; R = n-1
        mov     BYTE PTR [rbp-25], 1        ; _Bool flag = 1
        jmp     .L2                         ; goto while loop

.L6:
; while (flag == 1)  loop starts
        mov     BYTE PTR [rbp-25], 0        ; flag = 0
; main loop for( int i = 0;i<n-1;i++) condition
        mov     DWORD PTR [rbp-20], 0       ; int i = 0
        jmp     .L3                         ; jump to check statement
.L5:
; main loop for( int i = 0;i<n-1;i++) starts
;if(A[i+1][n-2-i]>A[i][n-1-i])
        mov     edx, DWORD PTR [rbp-20]
        add     edx, 1
        movsx   rcx, edx
        movsx   rdx, eax
        imul    rdx, rcx
        lea     rcx, [0+rdx*4]
        mov     rdx, QWORD PTR [rbp-64]
        add     rcx, rdx
        mov     edx, DWORD PTR [rbp-52]
        sub     edx, 2
        sub     edx, DWORD PTR [rbp-20]
        movsx   rdx, edx
```

```
        mov     ecx, DWORD PTR [rcx+rdx*4]
        mov     edx, DWORD PTR [rbp-20]
        movsx   rsi, edx
        movsx   rdx, eax
        imul    rdx, rsi
        lea     rsi, [0+rdx*4]
        mov     rdx, QWORD PTR [rbp-64]
        add     rsi, rdx
        mov     edx, DWORD PTR [rbp-52]
        sub     edx, 1
        sub     edx, DWORD PTR [rbp-20]
        movsx   rdx, edx
        mov     edx, DWORD PTR [rsi+rdx*4]
        cmp     ecx, edx
        jle     .L4
; if true branch start
;tmp = A[i+1][n-2-i]
        mov     edx, DWORD PTR [rbp-20]
        add     edx, 1
        movsx   rcx, edx
        movsx   rdx, eax
        imul    rdx, rcx
        lea     rcx, [0+rdx*4]
        mov     rdx, QWORD PTR [rbp-64]
        add     rcx, rdx
        mov     edx, DWORD PTR [rbp-52]
        sub     edx, 2
        sub     edx, DWORD PTR [rbp-20]
        movsx   rdx, edx
        mov     edx, DWORD PTR [rcx+rdx*4]
        mov     DWORD PTR [rbp-44], edx
;A[i+1][n-2-i] = A[i][n-1-i]
        mov     edx, DWORD PTR [rbp-20]  A[i][]
        movsx   rcx, edx
        movsx   rdx, eax
        imul    rdx, rcx
        lea     rcx, [0+rdx*4]
        mov     rdx, QWORD PTR [rbp-64]
        add     rcx, rdx
        mov     edx, DWORD PTR [rbp-52]
        sub     edx, 1                      ; n-1
        sub     edx, DWORD PTR [rbp-20]     ; A[i][n-1-i]
        mov     esi, DWORD PTR [rbp-20]     ; put i into ESI
        add     esi, 1                      ; i+1
        movsx   rdi, esi                    ; A[i+1]
        movsx   rsi, eax
        imul    rsi, rdi
        lea     rdi, [0+rsi*4]
        mov     rsi, QWORD PTR [rbp-64]
        add     rsi, rdi
        mov     edi, DWORD PTR [rbp-52]     ; put n into EDI
```

```
        sub     edi, 2                      ; n-2
        sub     edi, DWORD PTR [rbp-20]     ; A[n-2-i]
        movsx   rdx, edx
        mov     ecx, DWORD PTR [rcx+rdx*4]
        movsx   rdx, edi
        mov     DWORD PTR [rsi+rdx*4], ecx
; A[i][n-1-i] = tmp;
        mov     edx, DWORD PTR [rbp-20]     ;A[i][]
        movsx   rcx, edx
        movsx   rdx, eax
        imul    rdx, rcx
        lea     rcx, [0+rdx*4]
        mov     rdx, QWORD PTR [rbp-64]
        lea     rsi, [rcx+rdx]
        mov     edx, DWORD PTR [rbp-52]     ; put n to edx
        sub     edx, 1                      ; n-1
        sub     edx, DWORD PTR [rbp-20]     ; A[i][n-1-i]
        movsx   rdx, edx
        mov     ecx, DWORD PTR [rbp-44]     ; A[i][n-1-i] = tmp
        mov     DWORD PTR [rsi+rdx*4], ecx
        mov     BYTE PTR [rbp-25], 1        ;   flag = 1
; if true branch end
.L4:
        add     DWORD PTR [rbp-20], 1       ; i++
.L3:
        mov     edx, DWORD PTR [rbp-20]     ; write i to edx
        cmp     edx, DWORD PTR [rbp-24]     ; i<R
        jl      .L5                         ; goto main loop body
; main loop for( int i = 0;i<n-1;i++) ends
        sub     DWORD PTR [rbp-24], 1       ; R--
; while (flag == 1)  loop ends
; while (flag == 1)  loop condition
.L2:
        cmp     BYTE PTR [rbp-25], 0        ; compare flag with 0
       jne     .L6                         ; if flag !=0 ( equivalent to
                                  while(1)) jump to label 6 (loop body)
        nop                                 ; no operation
        nop                                 ; no operation
        mov     rbx, QWORD PTR [rbp-8]
         leave                               ; copy ebp to esp, set value
                                           of  esp before fuction starts
        ret                                 ; return to programme
; the end of function
```