

**“AZƏRBAYCAN HAVA YOLLARI” CJSC NATIONAL AVIATION ACADEMY**

**Individual Work № : 10**

**Topic:** **Convert C code to assembly language**

**Subject: OS2**

**Teacher: Mamed Shahmaliyev**

**Group: 1459i Student: Yusifov Rəvan**

**Date: Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_**

**­ Baku 2022**

Convert C/C++ code to assembly language

We use g++ compiler to turn provided C code into assembly language. To see the assembly code generated by the C compiler, we can use the “-S” option on the command line: 

**Syntax:**

***$ gcc -S filename.c***

This will cause gcc to run the compiler, generating an assembly file. Suppose we write a C code and store it in a file name “geeks.c” .

C

*// C code stored in geeks.c file*

*#include <stdio.h>*

*// global string*

*char s[] = "NAA";*

*// Driver Code*

*int main()*

*{*

*// Declaring variables*

*int a = 2000, b =17;*

*// Printing statement*

*printf("%s %d \n", s, a+b);*

*}*

**Running the command:**

***$ gcc -S geeks.c***

This will cause gcc to run the compiler, generating an assembly file geeks.s, and go no further. (Normally it would then invoke the assembler to generate an object- code file.)  
The assembly-code file contains various declarations including the set of lines:

CPP

*.section    \_\_TEXT, \_\_text, regular, pure\_instructions*

*.macosx\_version\_min 10, 12*

*.global    \_main*

*.align    4, 0x90*

*\_main:                                  ## @main*

*.cfi\_startproc*

*## BB#0:*

*pushq    %rbp*

*Ltmp0:*

*.cfi\_def\_cfa\_offset 16*

*Ltmp1:*

*.cfi\_offset %rbp, -16*

*movq    %rsp, %rbp*

*Ltmp2:*

*.cfi\_def\_cfa\_register %rbp*

*subq    $16, %rsp*

*leaq    L\_.str(%rip), %rdi*

*leaq    \_s(%rip), %rsi*

*movl    $2000, -4(%rbp)         ## imm = 0x7D0*

*movl    $17, -8(%rbp)*

*movl    -4(%rbp), %eax*

*addl    -8(%rbp), %eax*

*movl    %eax, %edx*

*movb    $0, %al*

*callq    \_printf*

*xorl    %edx, %edx*

*movl    %eax, -12(%rbp)         ## 4-byte Spill*

*movl    %edx, %eax*

*addq    $16, %rsp*

*popq    %rbp*

*retq*

*.cfi\_endproc*

*.section    \_\_DATA, \_\_data*

*.global    \_s                      ## @s*

*\_s:*

*.asciz    "NAA"*

*.section    \_\_TEXT, \_\_cstring, cstring\_literals*

*L\_.str:                                 ## @.str*

*.asciz    "%s %d \n"*

*.subsections\_via\_symbols*

Each indented line in the above code corresponds to a single machine instruction. For example, the pushq instruction indicates that the contents of register %rbp should be pushed onto the program stack. All information about local variable names or data types has been stripped away. We still see a reference to the global   
variable s[]= “NAA”, since the compiler has not yet determined where in memory this variable will be stored.