Salwaa Mumtaazah D.

23/516172/PA/22060

CSB

**Assignment 1**

**<https://github.com/Salwaa1209/SalwaaMumtaazahDarmanastri-SKJ-Lab.git>**

1.6.2 First Task: C++ Code to Assembly

1. Write a Simple C++ Program

Write a C++ program that adds two integers.

A screenshot of a computer program

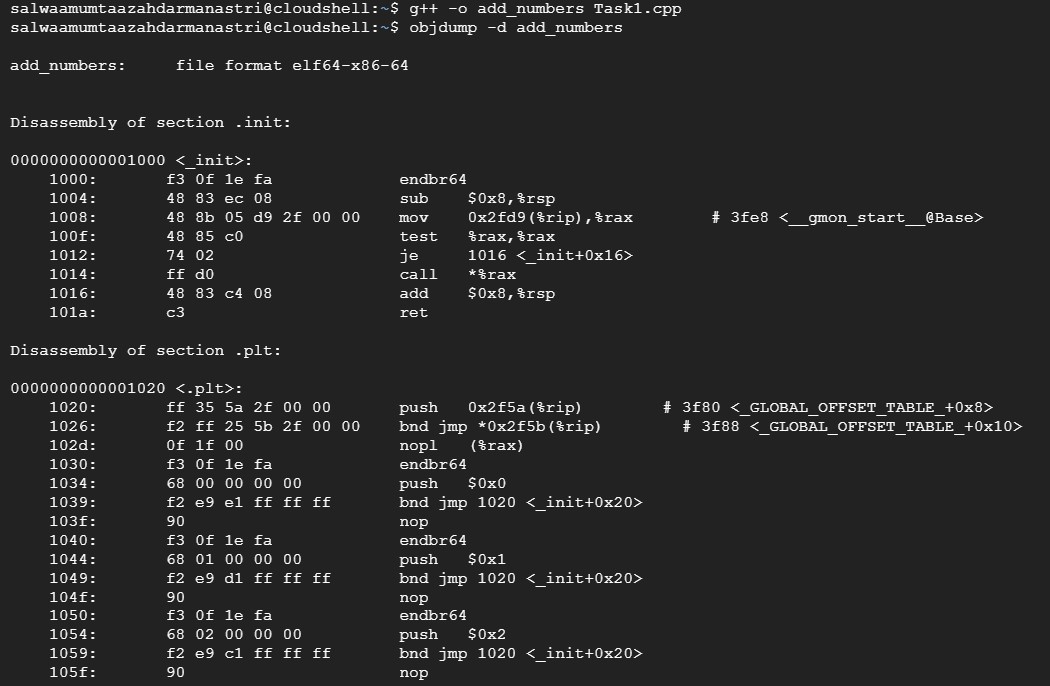
Description automatically generated

3. Disassemble the Code (10 points)

Disassemble the compiled executable to view the generated assembly code. Use the ‘objdump‘ command as follows:

objdump -d add\_numbers

This will display the assembly code corresponding to the compiled binary.



1050: f3 0f 1e fa endbr64

1054: 68 02 00 00 00 push $0x2

1059: f2 e9 c1 ff ff ff bnd jmp 1020 <\_init+0x20>

105f: 90 nop

1060: f3 0f 1e fa endbr64

1064: 68 03 00 00 00 push $0x3

1069: f2 e9 b1 ff ff ff bnd jmp 1020 <\_init+0x20>

106f: 90 nop

1070: f3 0f 1e fa endbr64

1074: 68 04 00 00 00 push $0x4

1079: f2 e9 a1 ff ff ff bnd jmp 1020 <\_init+0x20>

107f: 90 nop

1080: f3 0f 1e fa endbr64

1084: 68 05 00 00 00 push $0x5

1089: f2 e9 91 ff ff ff bnd jmp 1020 <\_init+0x20>

108f: 90 nop

1090: f3 0f 1e fa endbr64

1094: 68 06 00 00 00 push $0x6

1099: f2 e9 81 ff ff ff bnd jmp 1020 <\_init+0x20>

109f: 90 nop

Disassembly of section .plt.got:

00000000000010a0 <\_\_cxa\_finalize@plt>:

10a0: f3 0f 1e fa endbr64

10a4: f2 ff 25 1d 2f 00 00 bnd jmp \*0x2f1d(%rip) # 3fc8 <\_\_cxa\_finalize@GLIBC\_2.2.5>

10ab: 0f 1f 44 00 00 nopl 0x0(%rax,%rax,1)

Disassembly of section .plt.sec:

00000000000010b0 <\_ZNSirsERi@plt>:

10b0: f3 0f 1e fa endbr64

10b4: f2 ff 25 d5 2e 00 00 bnd jmp \*0x2ed5(%rip) # 3f90 <\_ZNSirsERi@GLIBCXX\_3.4>

10bb: 0f 1f 44 00 00 nopl 0x0(%rax,%rax,1)

00000000000010c0 <\_\_cxa\_atexit@plt>:

10c0: f3 0f 1e fa endbr64

10c4: f2 ff 25 cd 2e 00 00 bnd jmp \*0x2ecd(%rip) # 3f98 <\_\_cxa\_atexit@GLIBC\_2.2.5>

10cb: 0f 1f 44 00 00 nopl 0x0(%rax,%rax,1)

00000000000010d0 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>:

10d0: f3 0f 1e fa endbr64

10d4: f2 ff 25 c5 2e 00 00 bnd jmp \*0x2ec5(%rip) # 3fa0 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@GLIBCXX\_3.4>

10db: 0f 1f 44 00 00 nopl 0x0(%rax,%rax,1)

00000000000010e0 <\_ZNSolsEPFRSoS\_E@plt>:

10e0: f3 0f 1e fa endbr64

10e4: f2 ff 25 bd 2e 00 00 bnd jmp \*0x2ebd(%rip) # 3fa8 <\_ZNSolsEPFRSoS\_E@GLIBCXX\_3.4>

10eb: 0f 1f 44 00 00 nopl 0x0(%rax,%rax,1)

00000000000010f0 <\_\_stack\_chk\_fail@plt>:

10f0: f3 0f 1e fa endbr64

10f4: f2 ff 25 b5 2e 00 00 bnd jmp \*0x2eb5(%rip) # 3fb0 <\_\_stack\_chk\_fail@GLIBC\_2.4>

10fb: 0f 1f 44 00 00 nopl 0x0(%rax,%rax,1)

0000000000001100 <\_ZNSt8ios\_base4InitC1Ev@plt>:

1100: f3 0f 1e fa endbr64

1104: f2 ff 25 ad 2e 00 00 bnd jmp \*0x2ead(%rip) # 3fb8 <\_ZNSt8ios\_base4InitC1Ev@GLIBCXX\_3.4>

110b: 0f 1f 44 00 00 nopl 0x0(%rax,%rax,1)

0000000000001110 <\_ZNSolsEi@plt>:

1110: f3 0f 1e fa endbr64

1114: f2 ff 25 a5 2e 00 00 bnd jmp \*0x2ea5(%rip) # 3fc0 <\_ZNSolsEi@GLIBCXX\_3.4>

111b: 0f 1f 44 00 00 nopl 0x0(%rax,%rax,1)

Disassembly of section .text:

0000000000001120 <\_start>:

1120: f3 0f 1e fa endbr64

1124: 31 ed xor %ebp,%ebp

1126: 49 89 d1 mov %rdx,%r9

1129: 5e pop %rsi

112a: 48 89 e2 mov %rsp,%rdx

112d: 48 83 e4 f0 and $0xfffffffffffffff0,%rsp

1131: 50 push %rax

1132: 54 push %rsp

1133: 45 31 c0 xor %r8d,%r8d

1136: 31 c9 xor %ecx,%ecx

1138: 48 8d 3d e8 00 00 00 lea 0xe8(%rip),%rdi # 1227 <main>

113f: ff 15 93 2e 00 00 call \*0x2e93(%rip) # 3fd8 <\_\_libc\_start\_main@GLIBC\_2.34>

1145: f4 hlt

1146: 66 2e 0f 1f 84 00 00 cs nopw 0x0(%rax,%rax,1)

114d: 00 00 00

0000000000001150 <deregister\_tm\_clones>:

1150: 48 8d 3d b9 2e 00 00 lea 0x2eb9(%rip),%rdi # 4010 <\_\_TMC\_END\_\_>

1157: 48 8d 05 b2 2e 00 00 lea 0x2eb2(%rip),%rax # 4010 <\_\_TMC\_END\_\_>

115e: 48 39 f8 cmp %rdi,%rax

1161: 74 15 je 1178 <deregister\_tm\_clones+0x28>

1163: 48 8b 05 76 2e 00 00 mov 0x2e76(%rip),%rax # 3fe0 <\_ITM\_deregisterTMCloneTable@Base>

116a: 48 85 c0 test %rax,%rax

116d: 74 09 je 1178 <deregister\_tm\_clones+0x28>

116f: ff e0 jmp \*%rax

1171: 0f 1f 80 00 00 00 00 nopl 0x0(%rax)

1178: c3 ret

1179: 0f 1f 80 00 00 00 00 nopl 0x0(%rax)

0000000000001180 <register\_tm\_clones>:

1180: 48 8d 3d 89 2e 00 00 lea 0x2e89(%rip),%rdi # 4010 <\_\_TMC\_END\_\_>

1187: 48 8d 35 82 2e 00 00 lea 0x2e82(%rip),%rsi # 4010 <\_\_TMC\_END\_\_>

118e: 48 29 fe sub %rdi,%rsi

1191: 48 89 f0 mov %rsi,%rax

1194: 48 c1 ee 3f shr $0x3f,%rsi

1198: 48 c1 f8 03 sar $0x3,%rax

119c: 48 01 c6 add %rax,%rsi

119f: 48 d1 fe sar %rsi

11a2: 74 14 je 11b8 <register\_tm\_clones+0x38>

11a4: 48 8b 05 45 2e 00 00 mov 0x2e45(%rip),%rax # 3ff0 <\_ITM\_registerTMCloneTable@Base>

11ab: 48 85 c0 test %rax,%rax

11ae: 74 08 je 11b8 <register\_tm\_clones+0x38>

11b0: ff e0 jmp \*%rax

11b2: 66 0f 1f 44 00 00 nopw 0x0(%rax,%rax,1)

11b8: c3 ret

11b9: 0f 1f 80 00 00 00 00 nopl 0x0(%rax)

00000000000011c0 <\_\_do\_global\_dtors\_aux>:

11c0: f3 0f 1e fa endbr64

11c4: 80 3d ad 30 00 00 00 cmpb $0x0,0x30ad(%rip) # 4278 <completed.0>

11cb: 75 2b jne 11f8 <\_\_do\_global\_dtors\_aux+0x38>

11cd: 55 push %rbp

11ce: 48 83 3d f2 2d 00 00 cmpq $0x0,0x2df2(%rip) # 3fc8 <\_\_cxa\_finalize@GLIBC\_2.2.5>

11d5: 00

11d6: 48 89 e5 mov %rsp,%rbp

11d9: 74 0c je 11e7 <\_\_do\_global\_dtors\_aux+0x27>

11db: 48 8b 3d 26 2e 00 00 mov 0x2e26(%rip),%rdi # 4008 <\_\_dso\_handle>

11e2: e8 b9 fe ff ff call 10a0 <\_\_cxa\_finalize@plt>

11e7: e8 64 ff ff ff call 1150 <deregister\_tm\_clones>

11ec: c6 05 85 30 00 00 01 movb $0x1,0x3085(%rip) # 4278 <completed.0>

11f3: 5d pop %rbp

11f4: c3 ret

11f5: 0f 1f 00 nopl (%rax)

11f8: c3 ret

11f9: 0f 1f 80 00 00 00 00 nopl 0x0(%rax)

0000000000001200 <frame\_dummy>:

1200: f3 0f 1e fa endbr64

1204: e9 77 ff ff ff jmp 1180 <register\_tm\_clones>

0000000000001209 <\_Z10addNumbersii>:

1209: f3 0f 1e fa endbr64

120d: 55 push %rbp

120e: 48 89 e5 mov %rsp,%rbp

1211: 89 7d ec mov %edi,-0x14(%rbp)

1214: 89 75 e8 mov %esi,-0x18(%rbp)

1217: 8b 55 ec mov -0x14(%rbp),%edx

121a: 8b 45 e8 mov -0x18(%rbp),%eax

121d: 01 d0 add %edx,%eax

121f: 89 45 fc mov %eax,-0x4(%rbp)

1222: 8b 45 fc mov -0x4(%rbp),%eax

1225: 5d pop %rbp

1226: c3 ret

0000000000001227 <main>:

1227: f3 0f 1e fa endbr64

122b: 55 push %rbp

122c: 48 89 e5 mov %rsp,%rbp

122f: 48 83 ec 20 sub $0x20,%rsp

1233: 64 48 8b 04 25 28 00 mov %fs:0x28,%rax

123a: 00 00

123c: 48 89 45 f8 mov %rax,-0x8(%rbp)

1240: 31 c0 xor %eax,%eax

1242: 48 8d 05 bb 0d 00 00 lea 0xdbb(%rip),%rax # 2004 <\_IO\_stdin\_used+0x4>

1249: 48 89 c6 mov %rax,%rsi

124c: 48 8d 05 ed 2d 00 00 lea 0x2ded(%rip),%rax # 4040 <\_ZSt4cout@GLIBCXX\_3.4>

1253: 48 89 c7 mov %rax,%rdi

1256: e8 75 fe ff ff call 10d0 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

125b: 48 8d 45 ec lea -0x14(%rbp),%rax

125f: 48 89 c6 mov %rax,%rsi

1262: 48 8d 05 f7 2e 00 00 lea 0x2ef7(%rip),%rax # 4160 <\_ZSt3cin@GLIBCXX\_3.4>

1269: 48 89 c7 mov %rax,%rdi

126c: e8 3f fe ff ff call 10b0 <\_ZNSirsERi@plt>

1271: 48 89 c2 mov %rax,%rdx

1274: 48 8d 45 f0 lea -0x10(%rbp),%rax

1278: 48 89 c6 mov %rax,%rsi

127b: 48 89 d7 mov %rdx,%rdi

127e: e8 2d fe ff ff call 10b0 <\_ZNSirsERi@plt>

1283: 8b 55 f0 mov -0x10(%rbp),%edx

1286: 8b 45 ec mov -0x14(%rbp),%eax

1289: 89 d6 mov %edx,%esi

128b: 89 c7 mov %eax,%edi

128d: e8 77 ff ff ff call 1209 <\_Z10addNumbersii>

1292: 89 45 f4 mov %eax,-0xc(%rbp)

1295: 48 8d 05 7a 0d 00 00 lea 0xd7a(%rip),%rax # 2016 <\_IO\_stdin\_used+0x16>

129c: 48 89 c6 mov %rax,%rsi

129f: 48 8d 05 9a 2d 00 00 lea 0x2d9a(%rip),%rax # 4040 <\_ZSt4cout@GLIBCXX\_3.4>

12a6: 48 89 c7 mov %rax,%rdi

12a9: e8 22 fe ff ff call 10d0 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

12ae: 48 89 c2 mov %rax,%rdx

12b1: 8b 45 ec mov -0x14(%rbp),%eax

12b4: 89 c6 mov %eax,%esi

12b6: 48 89 d7 mov %rdx,%rdi

12b9: e8 52 fe ff ff call 1110 <\_ZNSolsEi@plt>

12be: 48 89 c2 mov %rax,%rdx

12c1: 48 8d 05 5a 0d 00 00 lea 0xd5a(%rip),%rax # 2022 <\_IO\_stdin\_used+0x22>

12c8: 48 89 c6 mov %rax,%rsi

12cb: 48 89 d7 mov %rdx,%rdi

12ce: e8 fd fd ff ff call 10d0 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

12d3: 48 89 c2 mov %rax,%rdx

12d6: 8b 45 f0 mov -0x10(%rbp),%eax

12d9: 89 c6 mov %eax,%esi

12db: 48 89 d7 mov %rdx,%rdi

12de: e8 2d fe ff ff call 1110 <\_ZNSolsEi@plt>

12e3: 48 89 c2 mov %rax,%rdx

12e6: 48 8d 05 3b 0d 00 00 lea 0xd3b(%rip),%rax # 2028 <\_IO\_stdin\_used+0x28>

12ed: 48 89 c6 mov %rax,%rsi

12f0: 48 89 d7 mov %rdx,%rdi

12f3: e8 d8 fd ff ff call 10d0 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

12f8: 48 89 c2 mov %rax,%rdx

12fb: 8b 45 f4 mov -0xc(%rbp),%eax

12fe: 89 c6 mov %eax,%esi

1300: 48 89 d7 mov %rdx,%rdi

1303: e8 08 fe ff ff call 1110 <\_ZNSolsEi@plt>

1308: 48 8b 15 c1 2c 00 00 mov 0x2cc1(%rip),%rdx # 3fd0 <\_ZSt4endlIcSt11char\_traitsIcEERSt13basic\_ostreamIT\_T0\_ES6\_@GLIBCXX\_3.4>

130f: 48 89 d6 mov %rdx,%rsi

1312: 48 89 c7 mov %rax,%rdi

1315: e8 c6 fd ff ff call 10e0 <\_ZNSolsEPFRSoS\_E@plt>

131a: b8 00 00 00 00 mov $0x0,%eax

131f: 48 8b 55 f8 mov -0x8(%rbp),%rdx

1323: 64 48 2b 14 25 28 00 sub %fs:0x28,%rdx

132a: 00 00

132c: 74 05 je 1333 <main+0x10c>

132e: e8 bd fd ff ff call 10f0 <\_\_stack\_chk\_fail@plt>

1333: c9 leave

1334: c3 ret

0000000000001335 <\_Z41\_\_static\_initialization\_and\_destruction\_0ii>:

1335: f3 0f 1e fa endbr64

1339: 55 push %rbp

133a: 48 89 e5 mov %rsp,%rbp

133d: 48 83 ec 10 sub $0x10,%rsp

1341: 89 7d fc mov %edi,-0x4(%rbp)

1344: 89 75 f8 mov %esi,-0x8(%rbp)

1347: 83 7d fc 01 cmpl $0x1,-0x4(%rbp)

134b: 75 3b jne 1388 <\_Z41\_\_static\_initialization\_and\_destruction\_0ii+0x53>

134d: 81 7d f8 ff ff 00 00 cmpl $0xffff,-0x8(%rbp)

1354: 75 32 jne 1388 <\_Z41\_\_static\_initialization\_and\_destruction\_0ii+0x53>

1356: 48 8d 05 1c 2f 00 00 lea 0x2f1c(%rip),%rax # 4279 <\_ZStL8\_\_ioinit>

135d: 48 89 c7 mov %rax,%rdi

1360: e8 9b fd ff ff call 1100 <\_ZNSt8ios\_base4InitC1Ev@plt>

1365: 48 8d 05 9c 2c 00 00 lea 0x2c9c(%rip),%rax # 4008 <\_\_dso\_handle>

136c: 48 89 c2 mov %rax,%rdx

136f: 48 8d 05 03 2f 00 00 lea 0x2f03(%rip),%rax # 4279 <\_ZStL8\_\_ioinit>

1376: 48 89 c6 mov %rax,%rsi

1379: 48 8b 05 78 2c 00 00 mov 0x2c78(%rip),%rax # 3ff8 <\_ZNSt8ios\_base4InitD1Ev@GLIBCXX\_3.4>

1380: 48 89 c7 mov %rax,%rdi

1383: e8 38 fd ff ff call 10c0 <\_\_cxa\_atexit@plt>

1388: 90 nop

1389: c9 leave

138a: c3 ret

000000000000138b <\_GLOBAL\_\_sub\_I\_\_Z10addNumbersii>:

138b: f3 0f 1e fa endbr64

138f: 55 push %rbp

1390: 48 89 e5 mov %rsp,%rbp

1393: be ff ff 00 00 mov $0xffff,%esi

1398: bf 01 00 00 00 mov $0x1,%edi

139d: e8 93 ff ff ff call 1335 <\_Z41\_\_static\_initialization\_and\_destruction\_0ii>

13a2: 5d pop %rbp

13a3: c3 ret

Disassembly of section .fini:

00000000000013a4 <\_fini>:

13a4: f3 0f 1e fa endbr64

13a8: 48 83 ec 08 sub $0x8,%rsp

13ac: 48 83 c4 08 add $0x8,%rsp

13b0: c3 ret

1.6.3 Second Task: Assembly to C++

1. Analyze the Provided Assembly Code

Consider the following assembly code (for illustration purposes; it may not compile directly):

section .data

num1 dw 5

num2 dw 10

result dw 0

section .text

global \_start

\_start:

mov ax, [num1]

imul ax, [num2]

mov [result], ax

; Exit the program

mov eax, 1

xor ebx, ebx

int 0x80

Explanation:

The code is divided into four sections: Data Segment, Code Segment, Execution Code, and Program Exit.

* **Data Segment**

This section defines the program's variables. The code defines two labels, num1 and num2, which store the values 5 and 10, respectively. The label result is initialized to 0 and stores the result of the multiplication.

* **Code Segment**

This section contains the executable code. The label \_start is defined as the entry point of the program, where execution begins.

* **Execution Code**

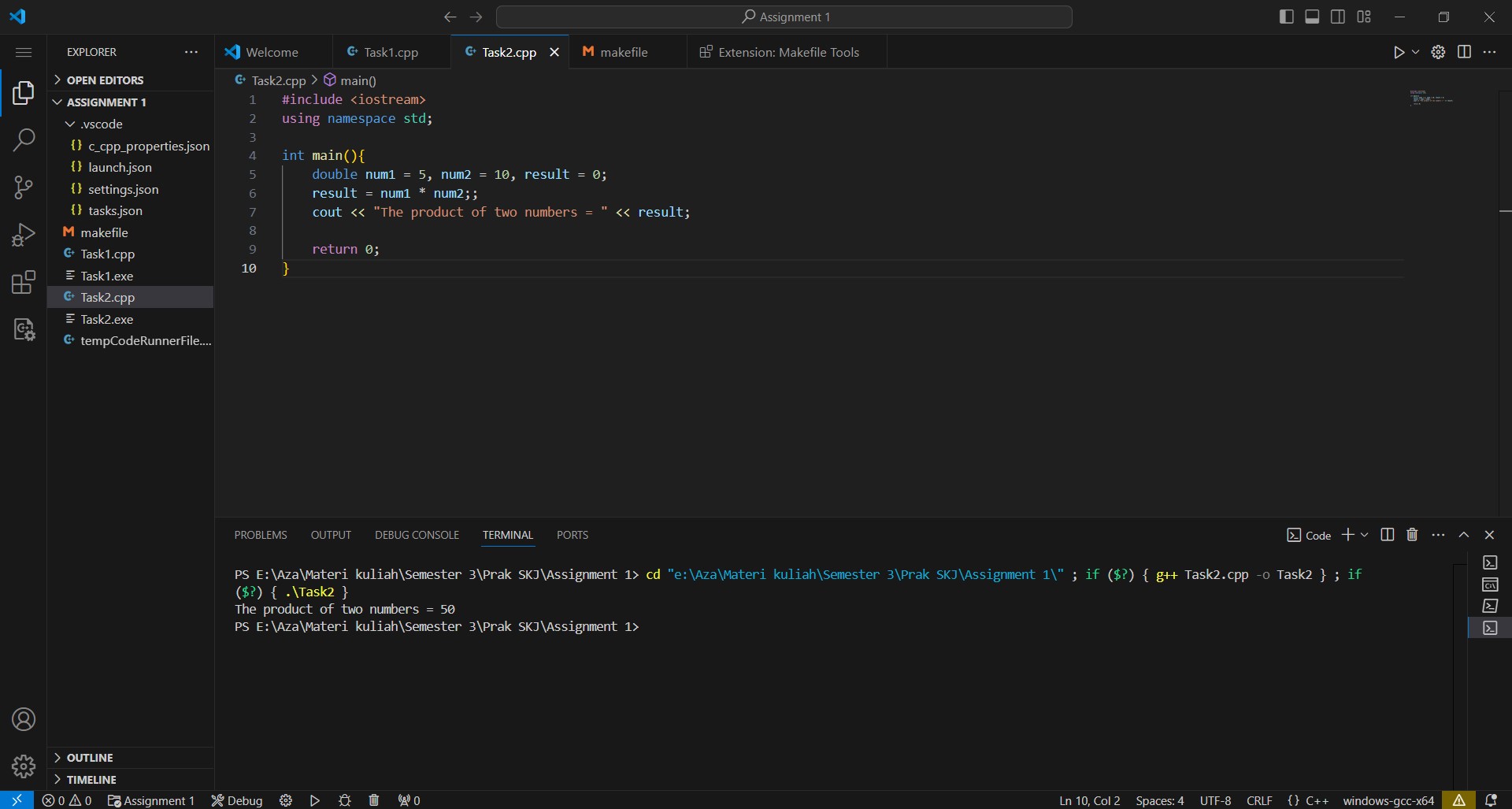
This section contains the instructions that perform the multiplication and store the result. The code loads the values of num1 and num2 into registers, multiplies them using the imul instruction, and stores the result in the result label. The result is then stored in memory.

* **Program Exit**

This section contains the instructions that exit the program. The code sets the exit status code to 0 and triggers a system call using the interrupt instruction int 0x80.

2. Write the Equivalent C++ Code (10 points)

Based on the provided assembly code, write a C++ program that performs the same functionality. The C++ program should produce the same result as the assembly code.



c

Write a Makefile for Task 1 and Task 2

