# Exercises

Consider the assembly code which the compiler generates for a C function. Explain what each assembly instruction does and describe what data is in the register.

1. ;;;5 void fn(int8\_t \* a, int32\_t \* b, float \* c) {

000000 b5f0 PUSH {r4-r7,lr}

**Storing the value of r4,r5,r6 since it will be changed during this function execution.**

1. 000002 b085 SUB sp,sp,#0x14

**Changing the stack pointer as we push r4,r5,46.**

1. 000004 4604 MOV r4,r0

**Saving value of r4 in r0**

1. 000006 460d MOV r5,r1

**Saving value of r5 in r1**

1. 000008 4616 MOV r6,r2

**Saving value of r6 in r2**

1. ;;;6 volatile int8\_t a1, a2;

;;;7 volatile int32\_t b1, b2;

;;;8 volatile float c1, c2;

;;;9

;;;10 a1 = 15;

00000a 270f MOVS r7,#0xf

**Declaring the variable initializing its value as a1=15**

1. ;;;11 a2 = -14;

00000c 200d MOVS r0,#0xd

**initializing register value with 14**

1. 00000e 43c0 MVNS r0,r0

**Storing negation of 14 i.e -14**

1. 000010 9004 STR r1,[sp,#0x10]

**Store the value of r1 i.e -14 in stack**

1. ;;;12 \*a = a1\*a2;

0x00000214 4668 MOV r0,sp

**Moving stack pointer value in r0**

1. 0x00000216 7D00 LDRB r0,[r0,#0x14]

**Load value of r0+#0x14 to r0**

1. 0x00000218 4669 MOV r1,sp

**Moving stack pointer value in r1**

1. 0x0000021A 7C09 LDRB r1,[r1,#0x10]

**Load value of r1+#0x10 to r1**

1. 0x0000021C 4348 MULS r0,r1,r0

**Multiply r1 and r0 register value and move it in r0**

1. 0x0000021E B240 SXTB r0,r0

**Cyclic rotate the value of r0**

1. 0x00000220 7020 STRB r0,[r4,#0x00]

**Store value of r0 at r4**

1. ;;;13

;;;14 b1 = 15;

00001a 200f MOVS r0,#0xf

**Move const value of 15 in register r0**

1. 00001c 9003 STR r0,[sp,#0xc]

**Store value of r0 at sp+#0xc**

1. ;;;15 b2 = -14;

00001e 200d MOVS r0,#0xd

**Move const value of 14 in register r0**

1. 000020 43c0 MVNS r0,r0

**Negation of 14 i.e -14**

1. 000022 9002 STR r0,[sp,#8]

**Store value of r0 at sp+#8**

1. ;;;16 \*b = b1\*b2;

000024 9902 LDR r1,[sp,#8]

**Load value of r1+#8(i.e b2) to r1**

1. 000026 9803 LDR r0,[sp,#0xc]

**Load value of r0+#0xc(i.e b1) to r0**

1. 000028 4348 MULS r0,r1,r0

**Mulitiply r1 and r0 (i.e b1 and b2)**

1. 00002a 6028 STR r0,[r5,#0]

**Store value of r0 at r5+#0**

1. ;;;17

;;;18 c1 = 15;

00002c 4809 LDR r0,|L1.84|

**Load value of L1.84 to r0**

1. 00002e 9001 STR r0,[sp,#4]

**Store value of r0 at sp+#4**

1. ;;;19 c2 = -14;

000030 4809 LDR r0,|L1.88|

**Load value of L1.88 to r0**

1. 000032 9000 STR r0,[sp,#0]

**Store value of r0 at sp+#0**

1. ;;;20 \*c = c1\*c2;

000034 9900 LDR r1,[sp,#0]

**Load value of sp to r1**

1. 000036 9801 LDR r0,[sp,#4]

**Load value of sp+#4 to r0**

1. 000038 f7fffffe BL \_\_aeabi\_fmul

**Branch handling**

1. 00003c 6030 STR r0,[r6,#0]

**Store value of r0 at r6+#0**

1. ;;;21

;;;22 }

00003e b005 ADD sp,sp,#0x14

**Add value stored at sp and sp+#0x14 and update value of sp.**

Deallocate the stack space for this function.

1. 000040 bdf0 POP {r4-r7,pc}

***Restore the value of r4,r5,r6 to origin so pop the stack of that.***