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) {
   ;;;6 volatile int8_t a1, a2;
   ;;;7 volatile int32_t b1, b2;
   ;;;8 volatile float c1, c2;
   - *0x00000200 B570 PUSH {r4-r6, lr}*
     The value of the registers are being pushed on the stack.
     For R4 – R6, it is because they are saved registers (ie. their values must be preserved across the function call). So, these value on the stack may be used later to restore them.
     And for LR, it is because this procedure might be called from other procedures, so the original LR must be saved before overwriting.
   - *0x00000202 B086 SUB sp, sp, #0x18*
     Allocate stack space for the function
   - *0x00000204 4604 MOV r4, r0*
     Copy R0 to R4
   - *0x00000206 460D MOV r5, r1*
     Copy R1 to R5
   - *0x00000208 4616 MOV r6, r2*
     Copy R2 to R6

2. ;;;10 a1 = 15;
   - *0x0000020A 210F MOVS r1, #0x0F*
     Copy 0xF to R1
   - *0x0000020C 9105 STR r1, [sp, #0x14]*
     Store value of R1 to space allocated to the variable a1 on the stack

3. ;;;11 a2 = -14;
   - *0x0000020E 210D MOVS r1, #0x0D*
     Copy 0xD to R1
   - *0x00000210 43C9 MVNS r1, r1*
     Copy the value of R1 negated (ie. Bitwise NOT of R1) to R1
   - *0x00000212 9104 STR r1, [sp, #0x10]*
     Store the value of R1 to space allocated to variable a2 on the stack

4. ;;;12 *a = a1*a2;
   - *0x00000214 4668 MOV r0, sp*
     Copy SP to R0
   - *0x00000216 7D00 LDRB r0, [r0, #0x14]*
     Copy value from address stored in R0 with offset 0x14 to R0
   - *0x00000218 4669 MOV r1, sp*
     Copy SP to R1
   - *0x0000021A 7C09 LDRB r1, [r1, #0x10]*
     Copy value from address stored in R1 with offset 0x10 to R1
   - *0x0000021C 4348 MULS r0, r1, r0*
     Multiply the value stored in R1 and R0 and store the result in R0
   - *0x0000021E B240 SXTB r0, r0*
     Sign extend R0
   - *0x00000220 7020 STRB r0, [r4, #0x00]*
     Store the value in R0 at address stored in R4

5. ;;;14 b1 = 15;
   - *0x00000222 200F MOVS r0, #0x0F*
     Copy 0xF to R0
   - *0x00000224 9003 STR r0, [sp, #0x0C]*
     Store the value in R0 to address pointed by SP with offset 0x0C (ie, the memory allocated to b1 in the stack frame)

6. ;;;15 b2 = -14;
   - *0x00000226 200D MOVS r0, #0x0D*
     Copy 0xD to R0
   - *0x00000228 43C0 MVNS r0, r0*
     Copy the value of R0 negated (ie. Bitwise NOT of R0) to R0
   - *0x0000022A 9002 STR r0, [sp, #0x08]*
     Store the value in R0 to address pointed by SP with offset 0x08 (ie, the memory allocated to b2 in the stack frame)

7. ;;;16 *b = b1*b2;
   - *0x0000022C 9902 LDR r1, [sp, #0x08]*
     Load the value stored at memory address in SP with offset 0x08 to R1 (basically load b1 to R1)
   - *0x0000022E 9803 LDR r0, [sp, #0x0C]*
     Load the value stored at memory address in SP with offset 0x0C to R0 (basically load b2 to R0)
   - *0x00000230 4348 MULS r0, r1, r0*
     Multiply the values in registers R0 and R1 and store the result in R0

- *0x00000232 6028 STR r0, [r5, #0x00]*
  Store the value in R0 to address in R5 with offset 0 (basically store the product of b1 and b2 in the address pointed to by b)

8. ;;;18 c1 = 15;
   - *0x00000234 4805 LDR r0, [pc, #20] ; @0x0000024C*
     C1 is float so, its value is stored in the program code itself. Load that value from the code (using address in PC and an offset) to register R0
   - *0x00000236 9001 STR r0, [sp, #0x04]*
     Store the value in R0 to the address pointed by SP with offset 0x04 (ie, the memory allocated to c1 in the stack frame)

9. ;;;19 c2 = -14;
   - *0x00000238 4805 LDR r0,[pc,#20] ; @0x00000250*
     C2 is float so, its value is stored in the program code itself. Load that value from the code (using address in PC and an offset) to register R0
   - *0x0000023A 9000 STR r0,[sp,#0x00]*
     Store the value in R0 to the address pointed by SP with offset 0x00 (ie, the memory allocated to c2 in the stack frame)

10. ;;;20 *c = c1*c2;
    - *0x0000023C 9900 LDR r1, [sp, #0x00]*
      Load the value stored at memory address in SP with offset 0x00 to R1 (basically load c2 to R1)
    - 0x0000023E 9801 LDR r0, [sp, #0x04]
      Load the value stored at memory address in SP with offset 0x04 to R0 (basically load c1 to R0)
    - *0x00000240 F000F810 BL.W __aeabi_fmul (0x00000264)*
      Call function __aeabi_fmul to multiply the floats
    - *0x00000244 6030 STR r0, [r6, #0x00]*
      Copy the product (in r0) to the address stored in r6 (address pointed to by c)

11. ;;;22 }
    - *0x00000246 B006 ADD sp, sp, #0x18*
      Deallocate the stack space for this function.
    - *0x00000248 BD70 POP {r4-r6, pc}*
      Restore the saved registers and program counter (so that next instruction after the function call would execute)