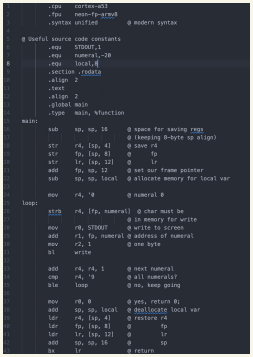
10.4.1

```
        .cpu    cortex-a53
        .fpu    neon-fp-armv8
        .syntax unified         @ modern syntax

@ Useful source code constant
        .equ    STDOUT,1

@ Constant program data
        .section  .rodata
        .align  2
helloMsg:
        .asciz   "Hello, Bob!\n"
        .equ    helloLngth,.-helloMsg

@ Program code
        .text
        .align  2
        .global main
        .type   main, %function
main:
        sub     sp, sp, 8       @ space for fp, lr
        str     fp, [sp, 0]     @ save fp
        str     lr, [sp, 4]     @   and lr
        add     fp, sp, 4       @ set our frame pointer

        mov     r0, STDOUT      @ file number to write to
        ldr     r1, helloMsgAddr   @ pointer to message
        mov     r2, helloLngth  @ number of bytes to write
        bl      write           @ write the message

        mov     r0, 0           @ return 0;
        ldr     fp, [sp, 0]     @ restore caller fp
        ldr     lr, [sp, 4]     @       lr
        add     sp, sp, 8       @   and sp
        bx      lr              @ return

        .align  2
helloMsgAddr:
        .word   helloMsg
```

10.6.1

```
        .cpu    cortex-a53
        .fpu    neon-fp-armv8
        .syntax unified         @ modern syntax
        .equ    STDIN,0
        .equ    STDOUT,1
        .equ    char1,-8
        .equ    char2,-7
        .equ    char3,-6
        .equ    char4,-5
        .equ    nChars,4
        .equ    local,8
        .section  .rodata
        .align  2
promptMsg:
        .asciz  "Enter four characters: "
        .equ    promptLngth,.-promptMsg
responseMsg:
        .asciz  "You entered: "
        .equ    responseLngth,.-responseMsg
        .text
        .align  2
        .global main
        .type   main, %function
main:
        sub     sp, sp, 8       @ space for fp, lr
        str     fp, [sp, 0]     @ save fp
        str     lr, [sp, 4]     @   and lr
        add     fp, sp, 4       @ set our frame pointer
        sub     sp, sp, local   @ allocate memory for local var
        mov     r0, STDOUT      @ prompt user for input
        ldr     r1, promptMsgAddr
        mov     r2, promptLngth
        bl      write
        mov     r0, STDIN       @ from keyboard
        add     r1, fp, char1   @ address of 1st char
        mov     r2, 1           @ one char
        bl      read
        mov     r0, STDIN       @ from keyboard
        add     r1, fp, char2   @ address of 2nd char
        mov     r2, 1           @ one char
        bl      read
        mov     r0, STDIN       @ from keyboard
        add     r1, fp, char3   @ address of 3rd char
        mov     r2, 1           @ one char
        bl      read
        mov     r0, STDIN       @ from keyboard
        add     r1, fp, char4   @ address of 4th char
        mov     r2, 1           @ one char
        bl      read
        mov     r0, STDOUT      @ nice message for user
        ldr     r1, responseMsgAddr
        mov     r2, responseLngth
        bl      write
        mov     r0, STDOUT      @ echo user's characters
        add     r1, fp, char1   @ address of 1st char
        mov     r2, nChars      @ all four characters
        bl      write
        mov     r0, 0           @ return 0;
        add     sp, sp, local   @ delete local vars
        ldr     fp, [sp, 0]     @ restore caller fp
        ldr     lr, [sp, 4]     @       lr
        add     sp, sp, 8       @   and sp
        bx      lr              @ return
        .align  2
promptMsgAddr:
        .word   promptMsg
responseMsgAddr:
        .word   responseMsg
```

12.2.1



12.2.2

```
1         .cpu    cortex-a53
2         .fpu    neon-fp-armv8
3         .syntax unified         @ modern syntax
4
5     @ The program
6         .text
7         .align  2
8         .global main
9         .type   main, %function
10    main:
11        sub     sp, sp, 16      @ space for saving regs
12                                @ (keeping 8-byte sp align)
13        str     r4, [sp, 4]     @ save r4
14        str     fp, [sp, 8]     @      fp
15        str     lr, [sp, 12]    @      lr
16        add     fp, sp, 12      @ set our frame pointer
17        sub     sp, sp, local   @ allocate memory for local var
18
19        mov     r4, 'A          @ beginning of alphabet
20    loop:
21        strb    r4, [fp, alpha] @ char must be
22                                @ in memory for write
23        mov     r0, STDOUT      @ write to screen
24        add     r1, fp, alpha   @ address of alpha
25        mov     r2, 1           @ one byte
26        bl      write
27
28        add     r4, r4, 1       @ next alpha
29        cmp     r4, 'Z          @ whole alphabet?
30        ble     loop            @ no, keep going
31
32        mov     r0, 0           @ yes, return 0;
33        add     sp, sp, local   @ deallocate local var
34        ldr     r4, [sp, 4]     @ restore r4
35        ldr     fp, [sp, 8]     @         fp
36        ldr     lr, [sp, 12]    @         lr
37        add     sp, sp, 16      @         sp
38        bx      lr              @ return
```

12.4.1

```
1         .cpu    cortex-a53
2         .fpu    neon-fp-armv8
3         .syntax unified
4         .equ    STDOUT,1
5         .equ    STDIN,0
6         .equ    NUL,0
7         .equ    response,-20
8         .equ    local,8
9         .section .rodata
10        .align  2
11    prompt:
12        .asciz  "Enter a single character: "
13        .align  2
14    numeral:
15        .asciz  "You entered a numeral.\n"
16        .align  2
17    other:
18        .asciz  "You entered a non-numeric character.\n"
19        .text
20        .align  2
21        .global main
22        .type   main, %function
23    main:
24        sub     sp, sp, 16      @ space for saving regs
25                                @ (keeping 8-byte sp align)
26        str     r4, [sp, 4]     @ save r4
27        str     fp, [sp, 8]     @      fp
28        str     lr, [sp, 12]    @      lr
29        add     fp, sp, 12      @ set our frame pointer
30        sub     sp, sp, local   @ local variable
31        ldr     r4, promptAddr  @ prompt user
32    promptLoop:
33        ldrb    r3, [r4]        @ get a char
34        cmp     r3, NUL         @ end of string?
35        beq     getResponse     @ yes, get response
36        mov     r0, STDOUT      @ no, write to screen
37        mov     r1, r4          @ address of current char
38        mov     r2, 1           @ write 1 byte
39        bl      write
40        add     r4, r4, 1       @ increment pointer var
41        b       promptLoop      @ back to top
```

```
getResponse:
        mov     r0, STDIN       @ from keyboard
        add     r1, fp, response  @ address of response
        mov     r2, 2           @ one char plus enter key
        bl      read

        ldrb    r3, [fp, response] @ load response
        cmp     r3, '9          @ check high end
        bhi     notNumeral      @ >'9', other char
        cmp     r3, '0          @ check low end
        blo     notNumeral      @ <'0', other char

        ldr     r4, numeralAddr @ "You entered a numeral."
numLoop:
        ldrb    r3, [r4]        @ get a char
        cmp     r3, NUL         @ end of string?
        beq     endThen         @ yes, end of then block
        mov     r0, STDOUT      @ no, write to screen
        mov     r1, r4          @ address of current char
        mov     r2, 1           @ write 1 byte
        bl      write
        add     r4, r4, 1       @ increment pointer var
        b       numLoop         @ back to top
endThen:
        b       endElse         @ branch over else block

notNumeral:
        ldr     r4, otherAddr   @ "You entered some other character."
notNumLoop:
        ldrb    r3, [r4]        @ get a char
        cmp     r3, NUL         @ end of string?
        beq     endElse         @ yes, end of if-else
        mov     r0, STDOUT      @ no, write to screen
        mov     r1, r4          @ address of current char
        mov     r2, 1           @ write 1 byte
        bl      write
        add     r4, r4, 1       @ increment pointer var
        b       notNumLoop      @ back to top
```

```
endElse:
        mov     r0, 0           @ return 0;
        add     sp, sp, local   @ deallocate local var
        ldr     r4, [sp, 4]     @ restore r4
        ldr     fp, [sp, 8]     @         fp
        ldr     lr, [sp, 12]    @         lr
        add     sp, sp, 16      @         sp
        bx      lr              @ return

        .align      2
promptAddr:
        .word   prompt
numeralAddr:
        .word   numeral
otherAddr:
        .word   other
```