13.3.2

```
        .cpu    cortex-a53
        .fpu    neon-fp-armv8
        .syntax unified         @ modern syntax
        .equ    STDOUT,1
        .equ    NUL,0
        .text
        .align  2
        .global writeStr
        .type   writeStr, %function
writeStr:
        sub     sp, sp, 16      @ space for saving regs
        str     r4, [sp, 0]     @ save r4
        str     r5, [sp, 4]     @      r5
        str     fp, [sp, 8]     @      fp
        str     lr, [sp, 12]    @      lr
        add     fp, sp, 12      @ set our frame pointer

        mov     r4, r0          @ r4 = string pointer
        mov     r5, 0           @ r5 = count
whileLoop:
        ldrb    r3, [r4]        @ get a char
        cmp     r3, NUL         @ end of string?
        beq     allDone         @ yes, all done

        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
        add     r5, r5, 1       @ count++
        b       whileLoop       @ back to top
allDone:
        mov     r0, r5          @ return count;
        ldr     r4, [sp, 0]     @ restore r4
        ldr     r5, [sp, 4]     @      r5
        ldr     fp, [sp, 8]     @         fp
        ldr     lr, [sp, 12]    @         lr
        add     sp, sp, 16      @ restore sp
        bx      lr              @ return
```

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

@ Constant program data
        .section .rodata
        .align   2
theString:
        .asciz        "Hello World.\n"

@ The program
        .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

        ldr     r0, theStringAddr
        bl      writeStr

        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

theStringAddr:
        .word    theString
```

13.3.3

```
        .cpu    cortex-a53
        .fpu    neon-fp-armv8
        .syntax unified         @ modern syntax
@ Useful source code constants
        .equ    STDIN,0
        .equ    NUL,0
        .equ    LF,10     @ '\n' under Linux
        .text
        .align  2
        .global readLn
        .type   readLn, %function
readLn:
        sub     sp, sp, 16      @ space for saving regs
        str     r4, [sp, 4]     @ save r4
        str     r5, [sp, 8]     @      r5
        str     fp, [sp, 12]    @      fp
        str     lr, [sp, 16]    @      lr
        add     fp, sp, 12      @ set our frame pointer
        mov     r4, r0          @ r4 = string pointer
        mov     r5, 0           @ r5 = count
        mov     r0, STDIN       @ read from keyboard
        mov     r1, r4          @ address of current storage
        mov     r2, 1           @ read 1 byte
        bl      read
whileLoop:
        ldrb    r3, [r4]        @ get just read char
        cmp     r3, LF          @ end of input?
        beq     endOfString     @ yes, input done
        add     r4, r4, 1       @ no, increment pointer var
        add     r5, r5, 1       @ count++
        mov     r0, STDIN       @ read from keyboard
        mov     r1, r4          @ address of current storage
        mov     r2, #1          @ read 1 byte
        bl      read
        b       whileLoop       @ and check for end
endOfString:
        mov     r0, NUL         @ string terminator
        strb    r0, [r4]        @ write over '\n'
        mov     r0, r5          @ return count;
        ldr     r4, [sp, 4]     @ restore r4
        ldr     r5, [sp, 8]     @      r5
        ldr     fp, [sp, 12]    @      fp
        ldr     lr, [sp, 16]    @      lr
        add     sp, sp, 16      @ space for saving regs
        bx      lr              @ return
```

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

@ Constant for assembler
        .equ    nBytes,50  @ amount of memory for string
@ Constant program data
        .section .rodata
        .align  2
prompt:
        .asciz        "Enter some text: "

@ The program
        .text
        .align  2
        .global main
        .type   main, %function
main:
        sub     sp, sp, 16      @ space for saving regs
                                @ (keeping 8-byte sp align)
        str     r4, [sp, 4]     @ save r4
        str     fp, [sp, 8]     @      fp
        str     lr, [sp, 12]    @      lr
        add     fp, sp, 12      @ set our frame pointer
        mov     r0, nBytes      @ get memory from heap
        bl      malloc
        mov     r4, r0          @ pointer to new memory
        ldr     r0, promptAddr  @ prompt user
        bl      writeStr
        mov     r0, r4          @ get user input
        bl      readLn
        mov     r0, r4          @ echo user input
        bl      writeStr
        mov     r0, r4          @ free heap memory
        bl      free
        mov     r0, 0           @ return 0;
        ldr     r4, [sp, 4]     @ restore r4
        ldr     fp, [sp, 8]     @         fp
        ldr     lr, [sp, 12]    @         lr
        add     sp, sp, 16      @         sp
        bx      lr              @ return

promptAddr:
        .word    prompt
```

13.3.4

```
        .cpu    cortex-a53
        .fpu    neon-fp-armv8
        .syntax unified         @ modern syntax
        .equ    STDIN,0
        .equ    NUL,0
        .equ    LF,10     @ '\n' under Linux
        .text
        .align  2
        .global readLn
        .type   readLn, %function
readLn:
        sub     sp, sp, 24      @ space for saving regs
        str     r4, [sp, 4]     @ save r4
        str     r5, [sp, 8]     @      r5
        str     r6, [sp,12]     @      r6
        str     fp, [sp, 16]    @      fp
        str     lr, [sp, 20]    @      lr
        add     fp, sp, 20      @ set our frame pointer
        mov     r4, r0          @ r4 = string pointer
        mov     r5, 0           @ r5 = count
        mov     r6, r1          @ r6 = max chars
        sub     r6, 1           @ for NUL
        mov     r0, STDIN       @ read from keyboard
        mov     r1, r4          @ address of current storage
        mov     r2, 1           @ read 1 byte
        bl      read
whileLoop:
        ldrb    r3, [r4]        @ get just read char
        cmp     r3, LF          @ end of input?
        beq     endOfString     @ yes, input done
        cmp     r5, r6          @ max chars?
        bge     ignore          @ yes, ignore rest
        add     r4, r4, 1       @ no, increment pointer var
        add     r5, r5, 1       @ count++
ignore:
        mov     r0, STDIN       @ read from keyboard
        mov     r1, r4          @ address of current storage
        mov     r2, 1           @ read 1 byte
        bl      read
        b       whileLoop       @ and check for end
endOfString:
        mov     r0, NUL         @ string terminator
        strb    r0, [r4]        @ write over '\n'
        mov     r0, r5          @ return count;
        ldr     r4, [sp, 4]     @ restore r4
        ldr     r5, [sp, 8]     @      r5
        ldr     r6, [sp,12]     @      r6
        ldr     fp, [sp, 16]    @      fp
        ldr     lr, [sp, 20]    @      lr
        add     sp, sp, 24      @      sp
        bx      lr              @ return
```

13.3.5

```
        .cpu    cortex-a53
        .fpu    neon-fp-armv8
        .syntax unified         @ modern syntax
        .equ    nBytes,5  @ amount of memory for string
        .section .rodata
        .align  2
prompt:
        .asciz        "Enter some text: "
        .text
        .align  2
        .global main
        .type   main, %function
main:
        sub     sp, sp, 16      @ space for saving regs
                                @ (keeping 8-byte sp align)
        str     r4, [sp, 4]     @ save r4
        str     fp, [sp, 8]     @      fp
        str     lr, [sp, 12]    @      lr
        add     fp, sp, 12      @ set our frame pointer
        mov     r0, nBytes      @ get memory from heap
        bl      malloc
        mov     r4, r0          @ pointer to new memory
        ldr     r0, promptAddr  @ prompt user
        bl      writeStr
        mov     r0, r4          @ get user input
        mov     r1, nBytes      @ limit input size
        bl      readLn
        mov     r0, r4          @ echo user input
        bl      writeStr
        mov     r0, r4          @ free heap memory
        bl      free
        mov     r0, 0           @ return 0;
        ldr     r4, [sp, 4]     @ restore r4
        ldr     fp, [sp, 8]     @         fp
        ldr     lr, [sp, 12]    @         lr
        add     sp, sp, 16      @         sp
        bx      lr              @ return

promptAddr:
        .word    prompt
```

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

@ Useful source code constants
        .equ    STDOUT,1

@ Constant program data
        .section .rodata
        .align   2
theChar:
        .ascii        "\n"

@ The code
        .text
        .align  2
        .global newLine
        .type   newLine, %function
newLine:
        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      @ write to screen
        ldr     r1, theCharAddr @ address of newline char
        mov     r2, 1           @ write 1 byte
        bl      write

        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

theCharAddr:
        .word   theChar
```