

13.3.2

```

1  .cpu    cortex-a53
2  .fpu    neon-fp-armv8
3  .syntax unified      @ modern syntax
4  .equ    STDOUT,1
5  .equ    NUL,0
6  .text
7  .align  2
8  .global writeStr
9  .type   writeStr, %function
10
11 writeStr:
12     sub    sp, sp, 16      @ space for saving regs
13     str    r4, [sp, 0]    @ save r4
14     str    r5, [sp, 4]    @   r5
15     str    fp, [sp, 8]    @   fp
16     str    lr, [sp, 12]   @   lr
17     add    fp, sp, 12     @ set our frame pointer
18
19     mov    r4, r0         @ r4 = string pointer
20     mov    r5, 0          @ r5 = count
21
22 whileLoop:
23     ldrb   r3, [r4]       @ get a char
24     cmp    r3, NUL        @ end of string?
25     beq    allDone       @ yes, all done
26
27     mov    r0, STDOUT     @ no, write to screen
28     mov    r1, r4         @ address of current char
29     mov    r2, 1          @ write 1 byte
30     bl     write
31
32     add    r4, r4, 1      @ increment pointer var
33     add    r5, r5, 1      @ count++
34     b      whileLoop     @ back to top
35
36 allDone:
37     mov    r0, r5        @ return count;
38     ldr    r4, [sp, 0]    @ restore r4
39     ldr    r5, [sp, 4]    @   r5
40     ldr    fp, [sp, 8]    @   fp
41     ldr    lr, [sp, 12]   @   lr
42     add    sp, sp, 16     @ restore sp
43     bx     lr            @ return

```

13.3.3

```

1  .cpu    cortex-a53
2  .fpu    neon-fp-armv8
3  .syntax unified      @ modern syntax
4
5 @ Constant program data
6 .section .rodata
7 .align  2
8
9 theString:
10     .asciz "Hello World.\n"
11
12 @ The program
13 .text
14 .align  2
15 .global main
16 .type   main, %function
17
18 main:
19     sub    sp, sp, 8      @ space for fp, lr
20     str    fp, [sp, 0]    @ save fp
21     str    lr, [sp, 4]    @   and lr
22     add    fp, sp, 4      @ set our frame pointer
23
24     ldr    r0, theStringAddr
25     bl     writeStr
26
27     mov    r0, 0          @ return 0;
28     ldr    fp, [sp, 0]    @ restore caller fp
29     ldr    lr, [sp, 4]    @   lr
30     add    sp, sp, 8      @   and sp
31     bx     lr            @ return
32
33 theStringAddr:
34     .word  theString

```

```

1  .cpu    cortex-a53
2  .fpu    neon-fp-armv8
3  .syntax unified      @ modern syntax
4
5 @ Useful source code constants
6 .equ    STDIN,0
7 .equ    NUL,0
8 .equ    LF,10 @ '\n' under Linux
9 .text
10 .align  2
11 .global readLn
12 .type   readLn, %function
13
14 readLn:
15     sub    sp, sp, 16      @ space for saving regs
16     str    r4, [sp, 4]    @ save r4
17     str    r5, [sp, 8]    @   r5
18     str    fp, [sp, 12]   @   fp
19     str    lr, [sp, 16]   @   lr
20     add    fp, sp, 12     @ set our frame pointer
21     mov    r4, r0         @ r4 = string pointer
22     mov    r5, 0          @ r5 = count
23     mov    r0, STDIN      @ read from keyboard
24     mov    r1, r4         @ address of current storage
25     mov    r2, 1          @ read 1 byte
26     bl     read
27
28 whileLoop:
29     ldrb   r3, [r4]       @ get just read char
30     cmp    r3, LF         @ end of input?
31     beq    endOfString   @ yes, input done
32     add    r4, r4, 1      @ no, increment pointer var
33     add    r5, r5, 1      @ count++
34     mov    r0, STDIN      @ read from keyboard
35     mov    r1, r4         @ address of current storage
36     mov    r2, #1        @ read 1 byte
37     bl     read
38     b      whileLoop     @ and check for end
39
40 endOfString:
41     mov    r0, NUL        @ string terminator
42     strb   r0, [r4]       @ write over '\n'
43     mov    r0, r5         @ return count;
44     ldr    r4, [sp, 4]    @ restore r4
45     ldr    r5, [sp, 8]    @   r5
46     ldr    fp, [sp, 12]   @   fp
47     ldr    lr, [sp, 16]   @   lr
48     add    sp, sp, 16     @ space for saving regs
49     bx     lr            @ return

```

```

1  .cpu    cortex-a53
2  .fpu    neon-fp-armv8
3  .syntax unified      @ modern syntax
4
5 @ Constant for assembler
6 .equ    nBytes,50 @ amount of memory for string
7
8 @ Constant program data
9 .section .rodata
10 .align  2
11
12 prompt:
13     .asciz "Enter some text: "
14
15 @ The program
16 .text
17 .align  2
18 .global main
19 .type   main, %function
20
21 main:
22     sub    sp, sp, 16      @ space for saving regs
23                             @ (keeping 8-byte sp align)
24     str    r4, [sp, 4]    @ save r4
25     str    fp, [sp, 8]    @   fp
26     str    lr, [sp, 12]   @   lr
27     add    fp, sp, 12     @ set our frame pointer
28     mov    r0, nBytes     @ get memory from heap
29     bl     malloc
30     mov    r4, r0         @ pointer to new memory
31     ldr    r0, promptAddr @ prompt user
32     bl     writeStr
33     mov    r0, r4         @ get user input
34     bl     readLn
35     mov    r0, r4         @ echo user input
36     bl     writeStr
37     mov    r0, r4         @ free heap memory
38     bl     free
39     mov    r0, 0          @ return 0;
40     ldr    r4, [sp, 4]    @ restore r4
41     ldr    fp, [sp, 8]    @   fp
42     ldr    lr, [sp, 12]   @   lr
43     add    sp, sp, 16     @   sp
44     bx     lr            @ return
45
46 promptAddr:
47     .word  prompt

```

13.3.4

```

1  .cpu cortex-a53
2  .fpu neon-fp-armv8
3  .syntax unified @ modern syntax
4  .equ STDOUT,0
5  .equ NUL,0
6  .equ LF,10 @ '\n' under Linux
7  .text
8  .align 2
9  .global readLn
10 .type readLn, %function
11 readLn:
12     sub sp, sp, 24 @ space for saving regs
13     str r4, [sp, 4] @ save r4
14     str r5, [sp, 8] @ r5
15     str r6, [sp, 12] @ r6
16     str fp, [sp, 16] @ fp
17     str lr, [sp, 20] @ lr
18     add fp, sp, 20 @ set our frame pointer
19     mov r4, r0 @ r4 = string pointer
20     mov r5, 0 @ r5 = count
21     mov r6, r1 @ r6 = max chars
22     sub r6, r6, 1 @ for NUL
23     mov r0, STDIN @ read from keyboard
24     mov r1, r4 @ address of current storage
25     mov r2, 1 @ read 1 byte
26     bl read
27 whileLoop:
28     ldrb r3, [r4] @ get just read char
29     cmp r3, LF @ end of input?
30     beq endOfString @ yes, input done
31     cmp r5, r6 @ max chars?
32     bge ignore @ yes, ignore rest
33     add r4, r4, 1 @ no, increment pointer var
34     add r5, r5, 1 @ count++

```

```

35 ignore:
36     mov r0, STDIN @ read from keyboard
37     mov r1, r4 @ address of current storage
38     mov r2, 1 @ read 1 byte
39     bl read
40     b whileLoop @ and check for end
41 endOfString:
42     mov r0, NUL @ string terminator
43     strb r0, [r4] @ write over '\n'
44     mov r0, r5 @ return count;
45     ldr r4, [sp, 4] @ restore r4
46     ldr r5, [sp, 8] @ r5
47     ldr r6, [sp, 12] @ r6
48     ldr fp, [sp, 16] @ fp
49     ldr lr, [sp, 20] @ lr
50     add sp, sp, 24 @ sp
51     bx lr @ return

```

13.3.5

```

1  .cpu cortex-a53
2  .fpu neon-fp-armv8
3  .syntax unified @ modern syntax
4  .equ nBytes,5 @ amount of memory for string
5  .section .rodata
6  .align 2
7  prompt:
8  .asciz "Enter some text: "
9  .text
10 .align 2
11 .global main
12 .type main, %function
13 main:
14     sub sp, sp, 16 @ space for saving regs
15     @ (keeping 8-byte sp align)
16     str r4, [sp, 4] @ save r4
17     str fp, [sp, 8] @ fp
18     str lr, [sp, 12] @ lr
19     add fp, sp, 12 @ set our frame pointer
20     mov r0, nBytes @ get memory from heap
21     bl malloc
22     mov r4, r0 @ pointer to new memory
23     ldr r0, promptAddr @ prompt user
24     bl writeStr
25     mov r0, r4 @ get user input
26     mov r1, nBytes @ limit input size
27     bl readLn
28     mov r0, r4 @ echo user input
29     bl writeStr
30     mov r0, r4 @ free heap memory
31     bl free
32     mov r0, 0 @ return 0;
33     ldr r4, [sp, 4] @ restore r4
34     ldr fp, [sp, 8] @ fp
35     ldr lr, [sp, 12] @ lr
36     add sp, sp, 16 @ sp
37     bx lr @ return
38 promptAddr:
39     .word prompt

```

```

1  .cpu cortex-a53
2  .fpu neon-fp-armv8
3  .syntax unified @ modern syntax
4
5  @ Useful source code constants
6  .equ STDOUT,1
7
8  @ Constant program data
9  .section .rodata
10 .align 2
11 theChar:
12     .ascii "\n"
13
14 @ The code
15 .text
16 .align 2
17 .global newLine
18 .type newLine, %function
19 newLine:
20     sub sp, sp, 8 @ space for fp, lr
21     str fp, [sp, 0] @ save fp
22     str lr, [sp, 4] @ and lr
23     add fp, sp, 4 @ set our frame pointer
24
25     mov r0, STDOUT @ write to screen
26     ldr r1, theCharAddr @ address of newline char
27     mov r2, 1 @ write 1 byte
28     bl write
29
30     mov r0, 0 @ return 0;
31     ldr fp, [sp, 0] @ restore caller fp
32     ldr lr, [sp, 4] @ lr
33     add sp, sp, 8 @ and sp
34     bx lr @ return
35
36 theCharAddr:
37     .word theChar

```