

15.2.1

```

1 .cpu cortex-m3
2 .fpu neon-fp-armv8
3 .syntax unified @ modern syntax
4
5 @ Constants for assembler
6 elements16 @ number of elements in array
7 .equ intArray-52 @ int32 beginning
8 .equ locals-48 @ space for local vars
9
10 @ Constant program data
11 .section .rodata
12 .align 2
13 .text
14 .type main, %function
15
16 main:
17 sub sp, sp, 16 @ space for saving regs
18 str r4, [sp, 4] @ save r4
19 str fp, [sp, 8] @ fp
20 str lr, [sp, 12] @ lr
21 add fp, fp, 12 @ set our frame pointer
22 sub sp, sp, locals @ for the structs

```

```

23 str r7, [sp, 12] @ lr
24 add r4, fp, 12 @ set our frame pointer
25 sub sp, sp, locals @ for the array
26
27 add r4, fp, 0 @ address of array beginning
28 mov r5, r4 @ r5 = r4
29 fillloop:
30 r5, elements @ fill filled?
31 bne fillloop @ yes
32 ldr r4, promptAddr @ r4, prompt user
33 bl writeStr @ yes
34 bl getInteger @ get integer
35 ldr r5, r5, 2 @ offset is 4 + index
36 str r4, [r4, r5] @ at index-th element
37 add r5, r5, 4 @ index++
38 b fillloop
39
40 allFills:
41 ldr r4, displayAddr @ nice message
42 bl writeStr
43
44 add r4, fp, 1 @ index = 1
45 mov r5, r4 @ r5 = r4
46 type promptLine
47
48 promptLine:
49 r5, r5, 2 @ r5, offset is 4 + index
50 ldr r4, [r4, r5] @ at index-th element
51 bl printInteger @ print integer
52 bl newline
53
54 r5, r5, 1 @ index++
55 type promptLine

```

```

57 allDone:
58 bne printloop
59
60 mov r0, 0 @ return 0;
61 add sp, sp, locals @ deallocate local var
62 ldr r4, [sp, 0] @ restore r4
63 ldr r5, [sp, 4] @ r5
64 ldr fp, [sp, 8] @ fp
65 ldr lr, [sp, 12] @ lr
66 add sp, sp, 16 @ restore sp
67 bx lr @ return
68
69 promptAddr:
70 .word prompt
71 displayAddr:
72 .word display

```

15.5.1 structPass.s

```

1 .cpu cortex-m3
2 .fpu neon-fp-armv8
3 .syntax unified @ modern syntax
4
5 @ Constants for assembler
6 include "theTagStruct.s" @ theTag struct defs.
7 .equ y-36 @ y struct
8 .equ x-24 @ x struct
9 .equ locals24 @ space for the structs
10
11 @ Constant program data
12 .section .rodata
13 .align 2
14 display:
15 .ascii "x fields:\n"
16 display:
17 .ascii "y fields:\n"
18
19 @ The program
20 .text
21 .align 2
22 .global main
23 .type main, %function
24
25 main:
26 sub sp, sp, 16 @ space for saving regs
27 @ (keeping 8-byte sp align)
28 str r4, [sp, 4] @ save r4
29 str fp, [sp, 8] @ fp
30 str lr, [sp, 12] @ lr
31 add fp, fp, 12 @ set our frame pointer
32 sub sp, sp, locals @ for the structs

```

```

33 sub sp, sp, locals @ for the structs
34
35 @ fill the x struct
36 add r0, fp, x @ address of x struct
37 bl getStruct
38
39 @ fill the y struct
40 add r0, fp, y @ address of y struct
41 bl getStruct
42
43 @ display x struct
44 ldr r0, displayXAddr
45 bl writeStr
46 add r0, fp, x @ address of x struct
47 bl putStruct
48 bl newline
49
50 @ display y struct
51 ldr r0, displayYAddr
52 bl writeStr
53 add r0, fp, y @ address of y struct
54 bl putStruct
55 bl newline

```

```

56 bl putStruct
57 bl newline
58
59 mov r0, 0 @ return 0;
60 add sp, sp, locals @ deallocate local var
61 ldr r4, [sp, 4] @ restore r4
62 ldr fp, [sp, 8] @ fp
63 ldr lr, [sp, 12] @ lr
64 add sp, sp, 16 @ restore sp
65 bx lr @ return
66
67 .align 2
68 @ addresses of messages
69 displayXAddr:
70 .word displayX
71 displayYAddr:
72 .word displayY

```

getStruct.s

```

1 .cpu cortex-m3
2 .fpu neon-fp-armv8
3 .syntax unified @ modern syntax
4
5 @ Constants for assembler
6 include "theTagStruct.s" @ theTag struct defs.
7
8 @ Constant program data
9 .section .rodata
10 .align 2
11 charPrompt:
12 .ascii "Enter a character: "
13 intPrompt:
14 .ascii "Enter an integer: "
15
16 @ The program
17 .text
18 .align 2
19 .global getStruct
20 .type getStruct, %function
21
22 getStruct:
23 sub sp, sp, 16 @ space for saving regs
24 @ (keeping 8-byte sp align)
25 str r4, [sp, 4] @ save r4
26 str fp, [sp, 8] @ fp
27 str lr, [sp, 12] @ lr
28 add fp, fp, 12 @ set our frame pointer
29 mov r4, r0 @ pointer to the struct

```

```

30 ldr r0, charPromptAddr
31 bl writeStr @ ask for a char
32 bl getChar
33 str r0, [r4, #char] @ struct->char = firstChar
34
35 ldr r0, intPromptAddr
36 bl writeStr @ ask for a char
37 bl getInteger
38 str r0, [r4, #int] @ struct->integer = secondChar
39
40 mov r4, 0 @ return 0;
41 ldr r4, [sp, 4] @ restore r4
42 ldr fp, [sp, 8] @ fp
43 ldr lr, [sp, 12] @ lr
44 add sp, sp, 16 @ restore sp
45 bx lr @ return
46
47 .align 2
48 @ addresses of messages
49 charPromptAddr:
50 .word charPrompt
51 intPromptAddr:
52 .word intPrompt

```

putStruct.s

```

1 .cpu cortex-m3
2 .fpu neon-fp-armv8
3 .syntax unified @ modern syntax
4
5 @ Constants for assembler
6 include "theTagStruct.s" @ theTag struct defs.
7
8 @ Constant program data
9 .section .rodata
10 .align 2
11 displayChar:
12 .ascii " "
13 displayInt:
14 .ascii " = anInt = "
15 displayOtherChar:
16 .ascii " = anotherChar = "
17
18 @ The program
19 .text
20 .align 2
21 .global putStruct
22 .type putStruct, %function
23
24 putStruct:
25 sub sp, sp, 16 @ space for saving regs
26 @ (keeping 8-byte sp align)
27 str r4, [sp, 4] @ save r4
28 str fp, [sp, 8] @ fp
29 str lr, [sp, 12] @ lr
30 add fp, fp, 12 @ set our frame pointer

```

```

31 mov r4, r0 @ pointer to the struct
32
33 ldr r0, displayCharAddr @ display aChar
34 bl writeStr
35 ldr r0, [r4, #char] @ struct->char
36 bl putChar
37 bl newline
38
39 ldr r0, displayIntAddr @ display anInt
40 bl writeStr
41 ldr r0, [r4, #int] @ struct->integer
42 bl putInteger
43 bl newline
44
45 ldr r0, displayOtherCharAddr @ display anotherChar
46 bl writeStr
47 ldr r0, [r4, #anotherChar] @ struct->anotherChar
48 bl putChar
49 bl newline
50
51 mov r0, 0 @ return 0;
52 ldr r4, [sp, 4] @ restore r4
53 ldr fp, [sp, 8] @ fp
54 ldr lr, [sp, 12] @ lr
55 add sp, sp, 16 @ restore sp
56 bx lr @ return
57
58 .align 2
59 @ addresses of messages
60 displayCharAddr:
61 .word displayChar
62 displayIntAddr:
63 .word displayInt
64 displayOtherCharAddr:
65 .word displayOtherChar

```

getChar.s

```

1 .cpu cortex-m3
2 .fpu neon-fp-armv8
3 .syntax unified @ modern syntax
4
5 @ Constants for assembler
6 .equ maxChars, 2 @ max input chars
7 .equ inputChar, -12 @ for input chars
8 .equ locals, 8 @ space for local vars
9
10 @ The program
11 .text
12 .align 2
13 .global getChar
14 .type getChar, %function
15
16 getChar:
17 sub sp, sp, 8 @ space for fp, [1]
18 str fp, [sp, 0] @ save fp
19 str [1], [sp, 4] @ and [1]
20 add fp, fp, 4 @ set our frame pointer
21 sub sp, sp, locals @ for the string
22
23 add r0, fp, inputChar @ place to store input
24 mov r1, maxChars @ limit input length
25 bl readLn
26
27 ldrb r0, [fp, inputChar] @ return inputChar
28
29 add sp, sp, locals @ deallocate local var
30 ldr fp, [sp, 0] @ restore caller fp
31 ldr [1], [sp, 4] @ and [1]
32 add sp, sp, 8 @ and sp
33 bx [1] @ return

```

15.7.1

incFraction.s

```

1 .cpu cortex-a53
2 .fpu neon-fp-armv8
3 .syntax unified @ modern syntax
4
5 @ Constants for assembler
6 .equ x,-12 @ x fraction object
7 .equ locals,8 @ space for fraction
8
9 @ The program
10 .text
11 .align 2
12 .global main
13 .type main,%function
14
15 main:
16     sub    sp, sp, 8 @ space for fp, lr
17     str    fp, [sp, 0] @ save fp
18     str    lr, [sp, 4] @ and lr
19     add    fp, sp, 4 @ set our frame pointer
20     sub    sp, sp, locals @ for the structs
21
22 @ construct the fraction
23     add    r0, fp, x @ address of x struct
24     bl     fractionConstr
25
26 @ get user input
27     add    r0, fp, x @ get user values
28     bl     fractionGet
29
30 @ add 1

```

```

28
29 v @ add 1
30     add    r0, fp, x @ add 1 to it
31     mov    r1, 1
32     bl     fractionAddInt
33
34 v @ display result
35     add    r0, fp, x @ get user values
36     bl     fractionDisplay
37
38     mov    r0, 0 @ return 0;
39     add    sp, sp, locals @ deallocate local var
40     ldr    fp, [sp, 0] @ restore caller fp
41     ldr    lr, [sp, 4] @ lr
42     add    sp, sp, 8 @ and sp
43     bx     lr @ return

```

fractionObject.s

```

1 v @ fraction object definition
2     .equ    num,0 @ numerator
3     .equ    den,4 @ denominator

```

fractionConstr.s

```

1 .cpu cortex-a53
2 .fpu neon-fp-armv8
3 .syntax unified @ modern syntax
4
5 @ Constants for assembler
6 .include "fractionObject.s" @ fraction object defs.
7
8 @ The code
9 .text
10 .align 2
11 .global fractionConstr
12 .type fractionConstr,%function
13
14 fractionConstr:
15     sub    sp, sp, 8 @ space for fp, lr
16     str    fp, [sp, 0] @ save fp
17     str    lr, [sp, 4] @ and lr
18     add    fp, sp, 4 @ set our frame pointer
19
20     mov    r1, 1 @ reasonable numerator
21     str    r1, [r0, num]
22
23     mov    r1, 2 @ reasonable denominator
24     str    r1, [r0, den]
25
26     mov    r0, 0 @ return 0;
27     ldr    fp, [sp, 0] @ restore caller fp
28     ldr    lr, [sp, 4] @ lr
29     add    sp, sp, 8 @ and sp
30     bx     lr @ return

```

fractionGet.s

```

1 .cpu cortex-a53
2 .fpu neon-fp-armv8
3 .syntax unified @ modern syntax
4
5 @ Constants for assembler
6 .include "fractionObject.s" @ fraction object defs.
7
8 @ Constant program data
9 .section .rodata
10 .align 2
11 numPrompt:
12     .ascii " Enter numerator: "
13 denPrompt:
14     .ascii "Enter denominator: "
15
16 @ The code
17 .text
18 .align 2
19 .global fractionGet
20 .type fractionGet,%function
21
22 fractionGet:
23     sub    sp, sp, 16 @ space for saving regs
24                     @ (keeping 8-byte sp align)
25     str    r4, [sp, 4] @ save r4
26     str    fp, [sp, 8] @ fp
27     str    lr, [sp, 12] @ lr
28     add    fp, sp, 12 @ set our frame pointer
29
30     mov    r4, r0 @ pointer to the object

```

fractionGet.s

```

1     ldr    r0, numPromptAddr
2     bl     writeStr @ ask for numerator
3     bl     getDecInt @ get it
4     str    r0, [r4, num] @ store at this->num
5
6     ldr    r0, denPromptAddr
7     bl     writeStr @ ask for denominator
8     bl     getDecInt @ get it
9     str    r0, [r4, den] @ store at this->den
10
11     mov    r0, 0 @ return 0;
12     ldr    r4, [sp, 4] @ restore r4
13     ldr    fp, [sp, 8] @ fp
14     ldr    lr, [sp, 12] @ lr
15     add    sp, sp, 16 @ sp
16     bx     lr @ return
17
18 .align 2
19 @ addresses of messages
20 numPromptAddr:
21     .word    numPrompt
22 denPromptAddr:
23     .word    denPrompt

```

fractionAddInt.s

```

1      .cpu    cortex-a53
2      .fpu    neon-fp-armv8
3      .syntax unified      @ modern syntax
4
5  @ Constants for assembler
6      .include "fractionObject.s" @ fraction object defs.
7
8  @ The code
9      .text
10     .align 2
11     .global fractionAddInt
12     .type   fractionAddInt, %function
13
14 fractionAddInt:
15     sub     sp, sp, 16      @ space for saving regs
16                                     @ (keeping 8-byte sp align)
17     str     r4, [sp, 4]    @ save r4
18     str     fp, [sp, 8]    @ fp
19     str     lr, [sp, 12]   @ lr
20     add     fp, sp, 12     @ set our frame pointer
21
22     mov     r4, r0         @ this pointer
23
24     ldr     r0, [r4, den]  @ get denominator
25     mul     r2, r1, r0     @ integer X denominator
26     ldr     r0, [r4, num]  @ get numerator
27     add     r2, r2, r0     @ numerator + (int X den)
28     str     r2, [r4, num]  @ save new numerator
29
30     mov     r0, 0          @ return 0;
31     ldr     r4, [sp, 4]    @ restore r4
32     ldr     fp, [sp, 8]    @ fp
33     ldr     lr, [sp, 12]   @ lr
34     add     sp, sp, 16     @ and sp
35     bx     lr             @ return

```

fractionDisplay.s

```

1      .cpu    cortex-a53
2      .fpu    neon-fp-armv8
3      .syntax unified      @ modern syntax
4
5  @ Constants for assembler
6      .include "fractionObject.s" @ fraction object defs.
7
8  @ The code
9      .text
10     .align 2
11     .global fractionDisplay
12     .type   fractionDisplay, %function
13
14 fractionDisplay:
15     sub     sp, sp, 16      @ space for saving regs
16                                     @ (keeping 8-byte sp align)
17     str     r4, [sp, 4]    @ save r4
18     str     fp, [sp, 8]    @ fp
19     str     lr, [sp, 12]   @ lr
20     add     fp, sp, 12     @ set our frame pointer
21
22     mov     r4, r0         @ this pointer
23
24     ldr     r0, [r4, num]  @ display numerator
25     bl     putDecInt       @ r0, r/
26     bl     putchar         @ slash for fraction
27     ldr     r0, [r4, den]  @ display denominator
28     bl     putDecInt       @ r0, r/
29     bl     newLine
30
31     mov     r0, 0          @ return 0;
32     ldr     r4, [sp, 4]    @ restore r4
33     ldr     fp, [sp, 8]    @ fp
34     ldr     lr, [sp, 12]   @ lr
35     add     sp, sp, 16     @ and sp
36     bx     lr             @ return

```

putChar.s

```

1      .cpu    cortex-a53
2      .fpu    neon-fp-armv8
3      .syntax unified      @ modern syntax
4
5  @ Constants for assembler
6      .equ    STDOUT, 1
7      .equ    theChar, -5   @ for string
8      .equ    locals, 8    @ space for local var
9
10 @ The code
11     .text
12     .align 2
13     .global putChar
14     .type   putChar, %function
15
16 putChar:
17     sub     sp, sp, 8      @ space for fp, lr
18     str     fp, [sp, 0]    @ save fp
19     str     lr, [sp, 4]    @ and lr
20     add     fp, sp, 4      @ set our frame pointer
21     sub     sp, sp, locals
22
23     strb    r0, [fp, theChar] @ write needs address
24     mov     r0, STDOUT     @ write to screen
25     add     r1, fp, theChar @ address of theChar
26     mov     r2, 1          @ write 1 byte
27     bl     write
28
29     mov     r0, 0          @ return 0;
30     add     sp, sp, locals @ deallocate local var
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

```