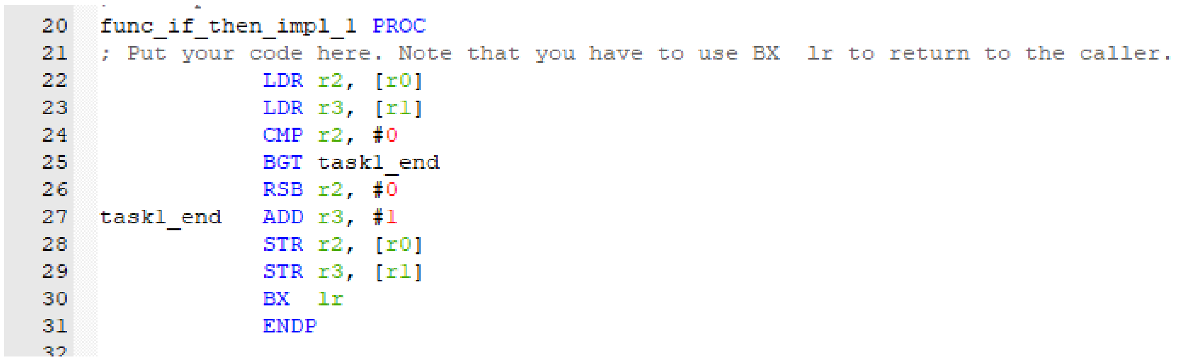
Jeremiah Webb CEC 320

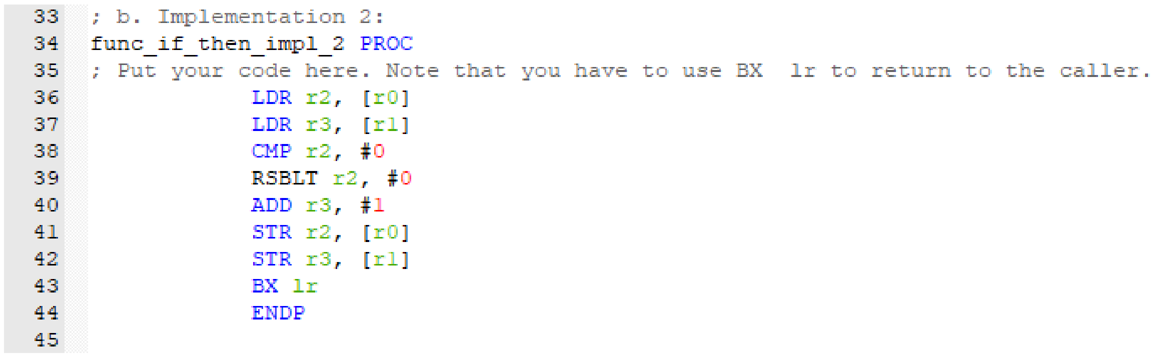
Workshop #9

Function 1



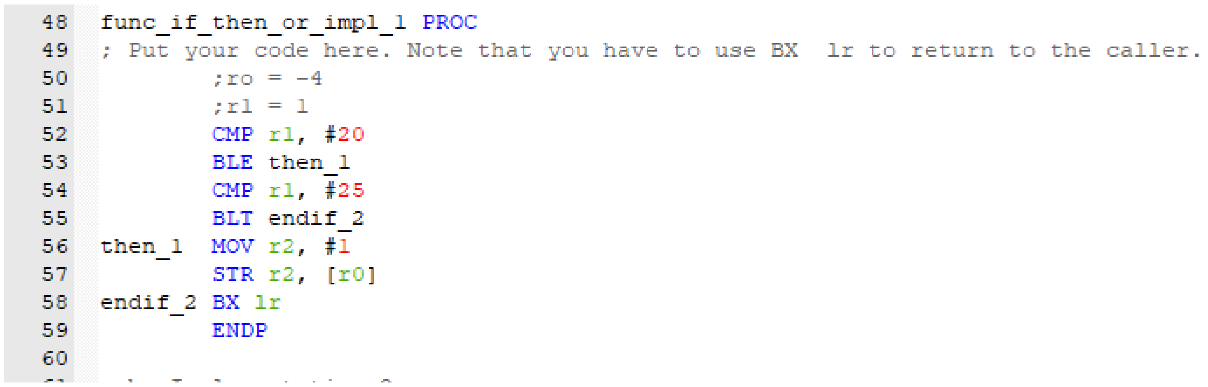
```
func_if_then_impl_1 PROC
; Put your code here. Note that you have to use BX  lr to return to the caller.
            LDR r2, [r0]
            LDR r3, [r1]
            CMP r2, #0
            BGT task1_end
            RSB r2, #0
task1_end   ADD r3, #1
            STR r2, [r0]
            STR r3, [r1]
            BX  lr
            ENDP
```

Function 2



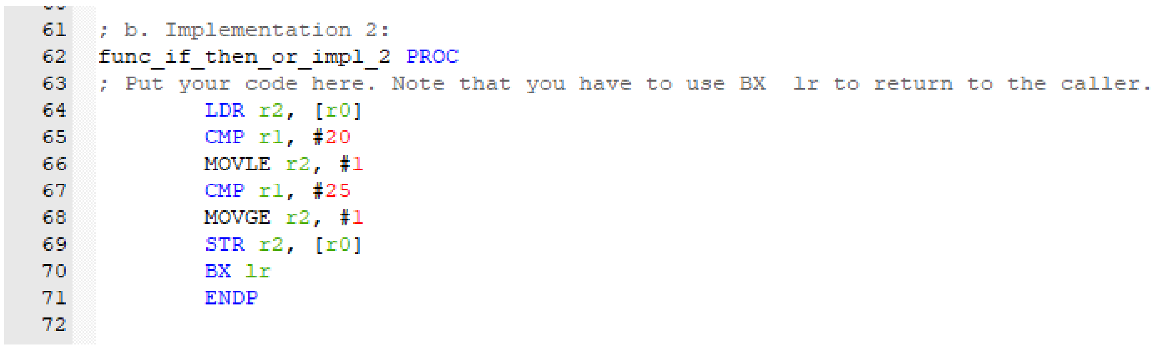
```
; b. Implementation 2:
func_if_then_impl_2 PROC
; Put your code here. Note that you have to use BX  lr to return to the caller.
            LDR r2, [r0]
            LDR r3, [r1]
            CMP r2, #0
            RSBLT r2, #0
            ADD r3, #1
            STR r2, [r0]
            STR r3, [r1]
            BX lr
            ENDP
```

Function 3



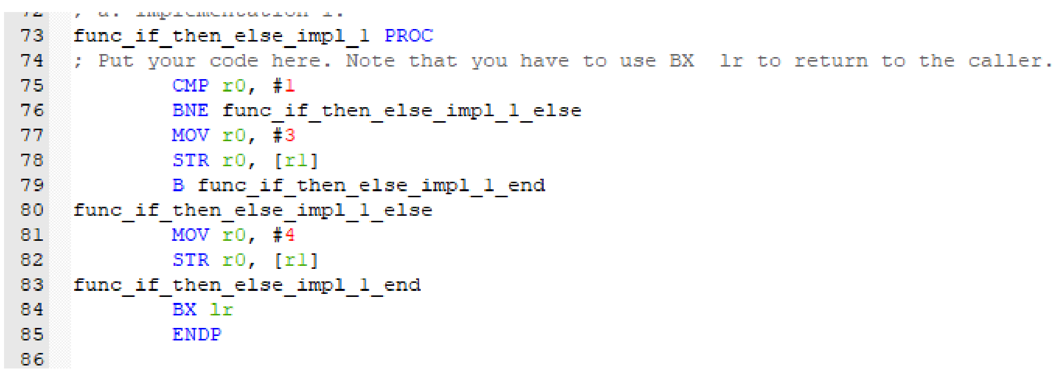
```
func_if_then_or_impl_1 PROC
; Put your code here. Note that you have to use BX  lr to return to the caller.
        ;ro = -4
        ;r1 = 1
        CMP r1, #20
        BLE then_1
        CMP r1, #25
        BLT endif_2
then_1  MOV r2, #1
        STR r2, [r0]
endif_2 BX lr
        ENDP
```

Function 4



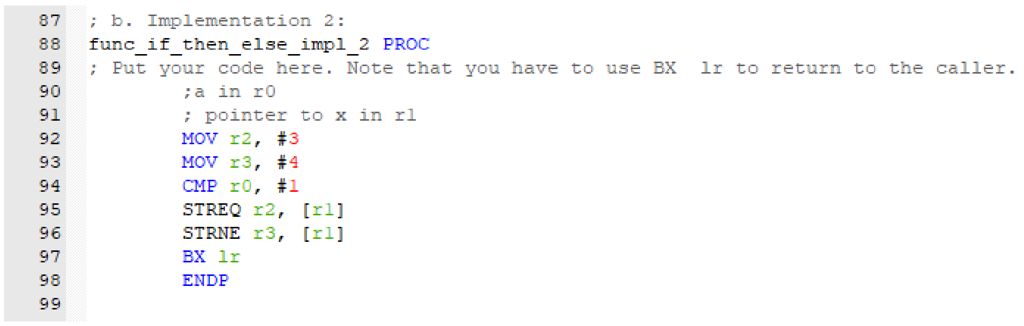
```
; b. Implementation 2:
func_if_then_or_impl_2 PROC
; Put your code here. Note that you have to use BX  lr to return to the caller.
        LDR r2, [r0]
        CMP r1, #20
        MOVLE r2, #1
        CMP r1, #25
        MOVGE r2, #1
        STR r2, [r0]
        BX lr
        ENDP
```

Function 5



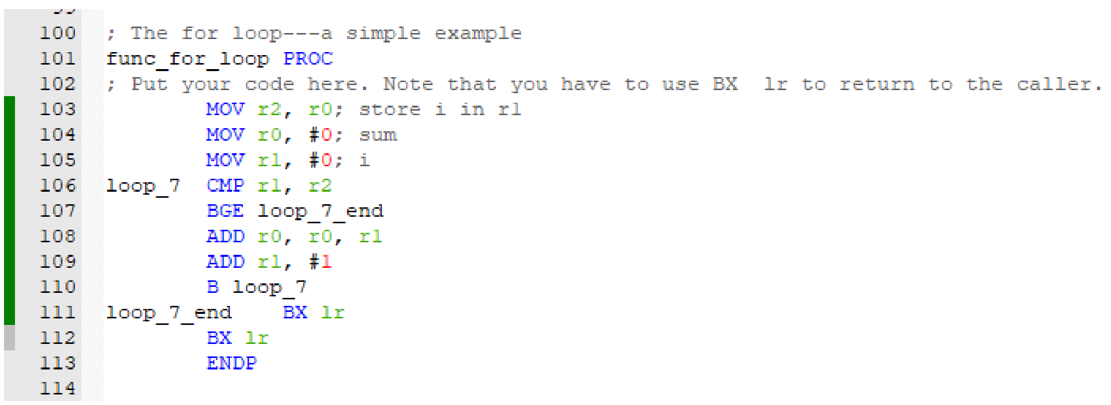
```
func_if_then_else_impl_1 PROC
; Put your code here. Note that you have to use BX  lr to return to the caller.
        CMP r0, #1
        BNE func_if_then_else_impl_1_else
        MOV r0, #3
        STR r0, [r1]
        B func_if_then_else_impl_1_end
func_if_then_else_impl_1_else
        MOV r0, #4
        STR r0, [r1]
func_if_then_else_impl_1_end
        BX lr
        ENDP
```

Function 6



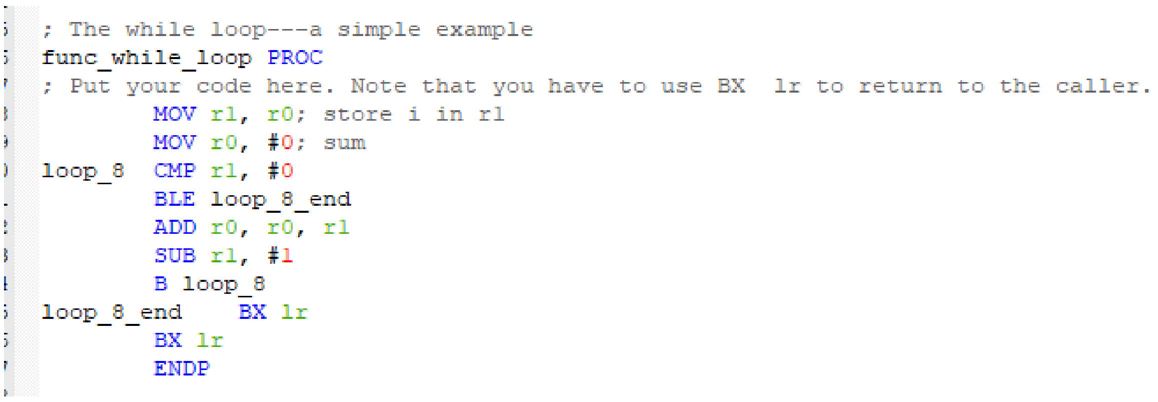
```
; b. Implementation 2:
func_if_then_else_impl_2 PROC
; Put your code here. Note that you have to use BX  lr to return to the caller.
        ;a in r0
        ; pointer to x in r1
        MOV r2, #3
        MOV r3, #4
        CMP r0, #1
        STREQ r2, [r1]
        STRNE r3, [r1]
        BX lr
        ENDP
```

Function 7



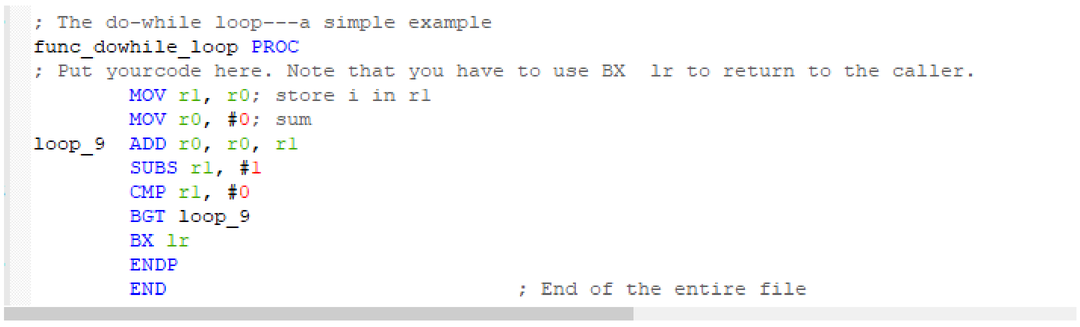
```
; The for loop---a simple example
func_for_loop PROC
; Put your code here. Note that you have to use BX  lr to return to the caller.
        MOV r2, r0; store i in r1
        MOV r0, #0; sum
        MOV r1, #0; i
loop_7  CMP r1, r2
        BGE loop_7_end
        ADD r0, r0, r1
        ADD r1, #1
        B loop_7
loop_7_end      BX lr
        BX lr
        ENDP
```

Function 8



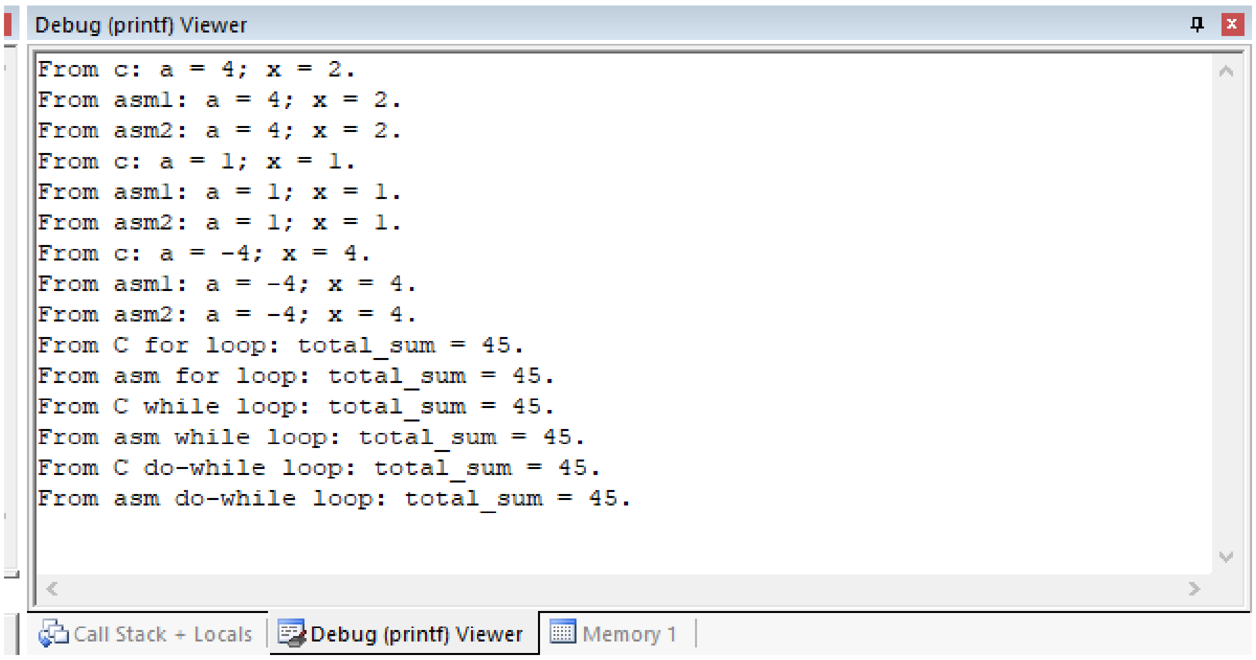
```
; The while loop---a simple example
func_while_loop PROC
; Put your code here. Note that you have to use BX  lr to return to the caller.
        MOV r1, r0; store i in r1
        MOV r0, #0; sum
loop_8  CMP r1, #0
        BLE loop_8_end
        ADD r0, r0, r1
        SUB r1, #1
        B loop_8
loop_8_end    BX lr
        BX lr
        ENDP
```

Function 9



```
; The do-while loop---a simple example
func_dowhile_loop PROC
; Put yourcode here. Note that you have to use BX  lr to return to the caller.
        MOV r1, r0; store i in r1
        MOV r0, #0; sum
loop_9  ADD r0, r0, r1
        SUBS r1, #1
        CMP r1, #0
        BGT loop_9
        BX lr
        ENDP
        END                             ; End of the entire file
```

Print Window Results:



```
From c: a = 4; x = 2.
From asm1: a = 4; x = 2.
From asm2: a = 4; x = 2.
From c: a = 1; x = 1.
From asm1: a = 1; x = 1.
From asm2: a = 1; x = 1.
From c: a = -4; x = 4.
From asm1: a = -4; x = 4.
From asm2: a = -4; x = 4.
From C for loop: total_sum = 45.
From asm for loop: total_sum = 45.
From C while loop: total_sum = 45.
From asm while loop: total_sum = 45.
From C do-while loop: total_sum = 45.
From asm do-while loop: total_sum = 45.
```