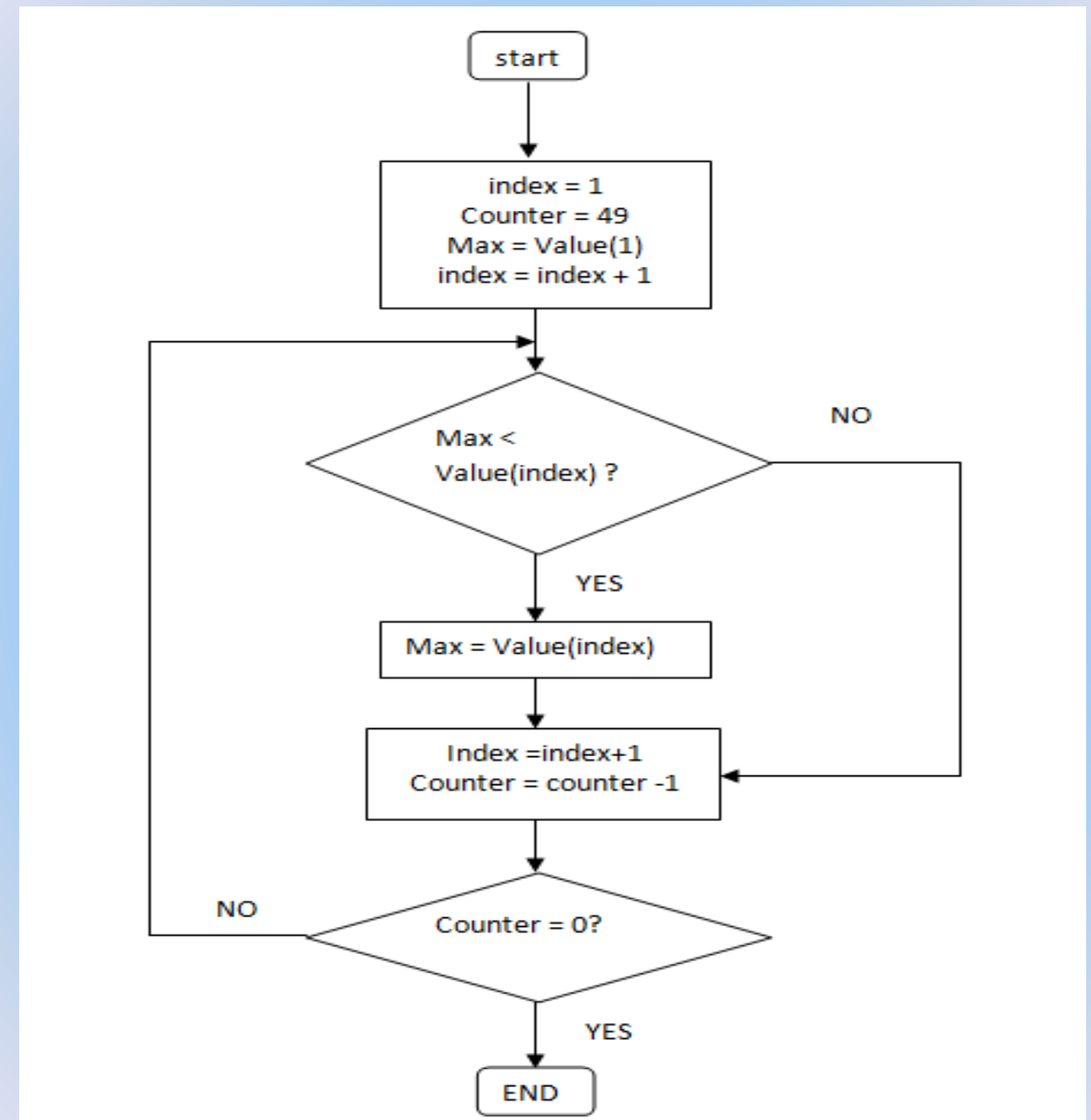
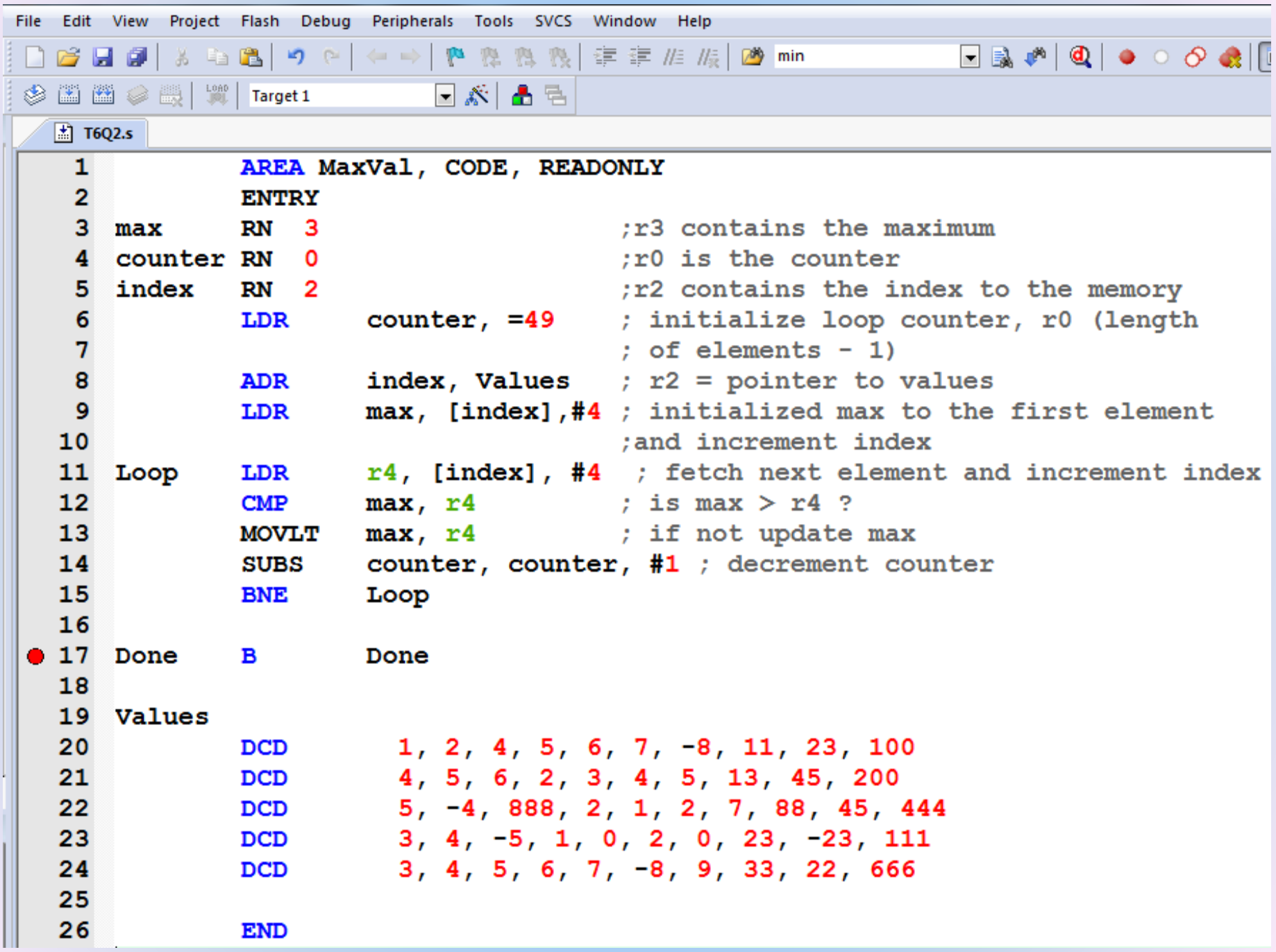
Tutorial 6

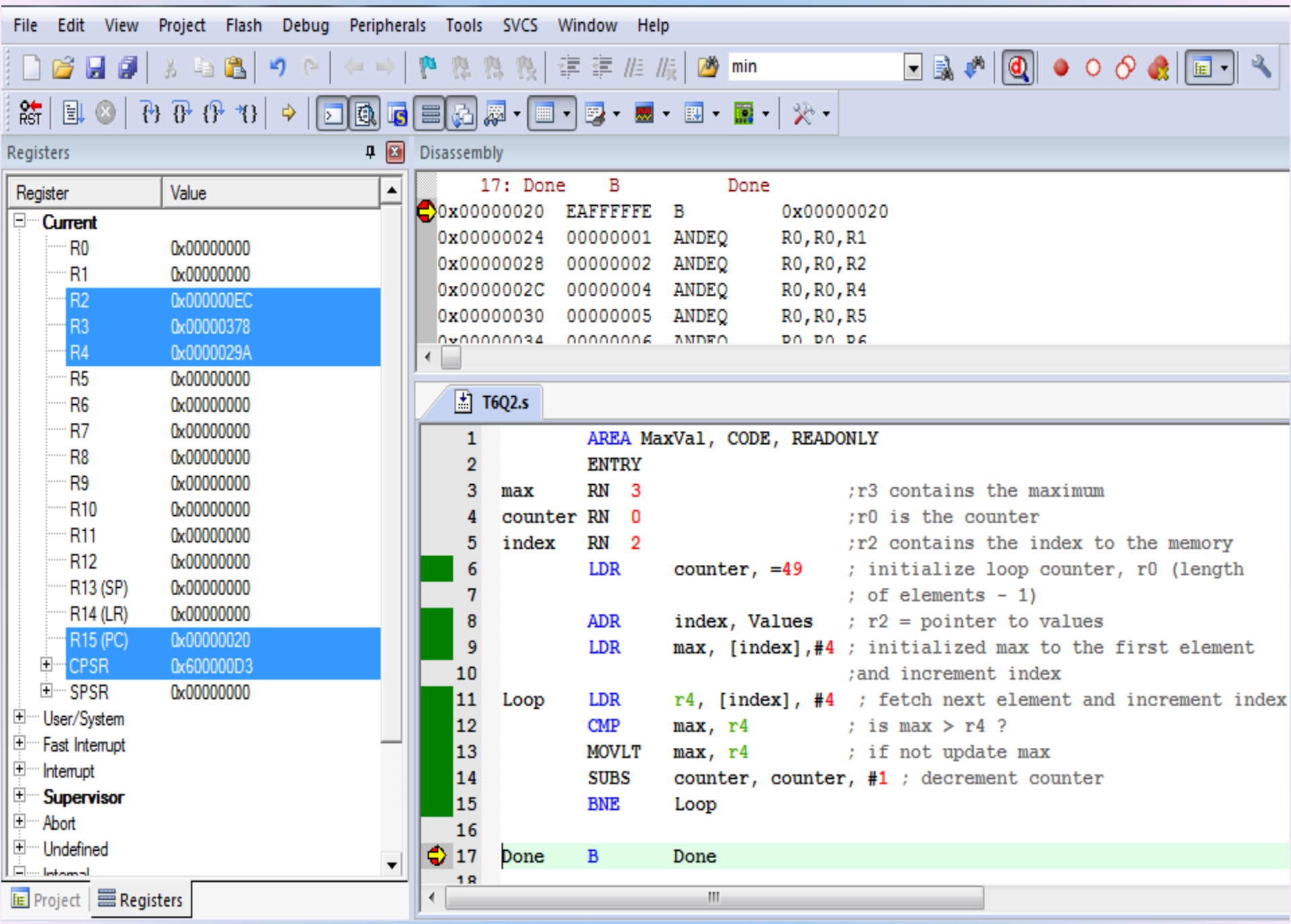
Q1.

- Find the maximum value in a list of 32-bit values located in memory.
- Assume the values are in two's complement representations.
- Your program should have 50 values in the list.


start
index = 1
Counter = 49
Max = Value(1)
index = index + 1
Max <
Value(index) ?
NO
YES
Max = Value(index)
Index =index+1
Counter = counter -1
Counter = 0?
NO
YES
END

```
        AREA MaxVal, CODE, READONLY
        ENTRY
max     RN      3               ;r3 contains the maximum
counter RN      0               ;r0 is the counter
index   RN      2               ;r2 contains the index to the memory
        LDR  counter, =49       ; initialize loop counter, r0 (length
                                ; of elements - 1)
        ADR  index, Values       ; r2 = pointer to values
        LDR max, [index],#4     ; initialized max to the first element
                                 ; and increment index
Loop    LDR  r4, [index], #4            ; fetch next element and
                                        ; increment index
        CMP  max, r4                    ; is max > r4 ?
        MOVLT max, r4                   ; if not update max
        SUBS counter, counter, #1       ; decrement counter
        BNE  Loop
Done    B    Done
Values
        DCD       1, 2, 4, 5, 6, 7, -8, 11, 23, 100
        DCD       4, 5, 6, 2, 3, 4, 5, 13, 45, 200
        DCD       5, -4, 888, 2, 1, 2, 7, 88, 45, 444
        DCD       3, 4, -5, 1, 0, 2, 0, 23, -23, 111
        DCD       3, 4, 5, 6, 7, -8, 9, 33, 22, 666
        END
```


File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
min
Target 1
T6Q2.s
1 AREA MaxVal, CODE, READONLY
2 ENTRY
3 max RN 3 ;r3 contains the maximum
4 counter RN 0 ;r0 is the counter
5 index RN 2 ;r2 contains the index to the memory
6 LDR counter, =49 ; initialize loop counter, r0 (length
7 ; of elements - 1)
8 ADR index, Values ; r2 = pointer to values
9 LDR max, [index],#4 ; initialized max to the first element
10 ;and increment index
11 Loop LDR r4, [index], #4 ; fetch next element and increment index
12 CMP max, r4 ; is max > r4 ?
13 MOVLT max, r4 ; if not update max
14 SUBS counter, counter, #1 ; decrement counter
15 BNE Loop
16
17 Done B Done
18
19 Values
20 DCD 1, 2, 4, 5, 6, 7, -8, 11, 23, 100
21 DCD 4, 5, 6, 2, 3, 4, 5, 13, 45, 200
22 DCD 5, -4, 888, 2, 1, 2, 7, 88, 45, 444
23 DCD 3, 4, -5, 1, 0, 2, 0, 23, -23, 111
24 DCD 3, 4, 5, 6, 7, -8, 9, 33, 22, 666
25
26 END


File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
min
Registers
Register
Value
Current
R0 0x00000000
R1 0x00000000
R2 0x000000EC
R3 0x00000378
R4 0x0000029A
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x00000000
R14 (LR) 0x00000000
R15 (PC) 0x00000020
CPSR 0x600000D3
SPSR 0x00000000
User/System
Fast Interrupt
Interrupt
Supervisor
Abort
Undefined
Project
Registers
Disassembly
17: Done B Done
0x00000020 EAFFFFFE B 0x00000020
0x00000024 00000001 ANDEQ R0,R0,R1
0x00000028 00000002 ANDEQ R0,R0,R2
0x0000002C 00000004 ANDEQ R0,R0,R4
0x00000030 00000005 ANDEQ R0,R0,R5
T6Q2.s
1 AREA MaxVal, CODE, READONLY
2 ENTRY
3 max RN 3 ;r3 contains the maximum
4 counter RN 0 ;r0 is the counter
5 index RN 2 ;r2 contains the index to the memory
6 LDR counter, =49 ; initialize loop counter, r0 (length
7 ; of elements - 1)
8 ADR index, Values ; r2 = pointer to values
9 LDR max, [index],#4 ; initialized max to the first element
10 ;and increment index
11 Loop LDR r4, [index], #4 ; fetch next element and increment index
12 CMP max, r4 ; is max > r4 ?
13 MOVLT max, r4 ; if not update max
14 SUBS counter, counter, #1 ; decrement counter
15 BNE Loop
16
17 Done B Done

2. Convert the following C iteration statements into assembly language. For example, in C language, the following *for* loop:

```
for (i = 10; i> 0; i--) {
.... // loop body
}
```

can be written in ARM assembly language as

```
        MOV     r0, #10     ; r0 is used to store i
loop            . . .       ; loop body
        SUBS    r0, r0, #1  ; loop 10 times
        BNE     loop
```

Write the assembly language equivalent for the following using r0 for i, r1 for j and r3 for k:

a) for (i=0; i<10; i++) { ...}

b) for (i = 1; i < 11; i++) { //nested loop
for (j = 5; j > 0; j--) {
.....} }

c) for (i = 18; i >= 3; i--) {
for (j= 16; j < 40; j++) {
for (k = 0; k < 30; k++) {
....}}}

2. The following C codes can be converted

a) C language: for (i=0; i<10; i++) { ...}

Assembly language:

```
        MOV     r0, #0          ; i=0
Loop0   …
        ADD     r0, r0, #1      ; i++
        CMP     r0, #10         ;
        BLT     Loop0           ; i < 10
```

b) for (i = 1; i < 11; i++) { //nested loop
for (j = 5; j > 0; j--) {
.....} }

Assembly language:

```
        MOV     r0, #1          ;i = 1
Loop0   MOV     r1, #5          ;j = 5
Loop1   …
        SUBS    r1, r1, #1      ; j--
        BNE     Loop1           ; j > 0
        ADD     r0, r0, #1      ; i++
        CMP     r0, #11         ;
        BLT     Loop0           ; i < 11
```

c) for (i = 18; i >= 3; i--) {
for (j= 16; j < 40; j++) {
for (k = 0; k < 30; k++) {
....}}}

Assembly language:

```
        MOV  r0, #18          ;i = 18
Loop0   MOV  r1, #16          ;j = 16
Loop1   MOV  r2, #0           ;k = 0
Loop2   …
        ADD  r2, r2, #1       ;k++
        CMP  r2, #30          ;
        BLT  Loop2            ;k < 30
        ADD  r1, r1, #1       ;j++
        CMP  r1, #40          ;
        BLT  Loop1            ;j < 40
        SUB  r0, r0, #1       ; i--
        CMP  r0, #3           ;
        BGE  Loop0            ;I >= 3
```

The End of Tutorial 6