EEE3002 Microprocessors Tutorial 6 (2014 Sem 1)

1. Write a assembly language program that reverses the bits in a register, so that a register containing $d_{31}d_{30}d_{29}\ldots d_1d_0$ now contains $d_0d_1\ldots d_{29}d_{30}d_{31}$.

2. 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.

3. 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++) {
       ….}}}
```

EE3002 Microprocessors Tutorial 6 Solutions

1) The pseudocode is as follows;

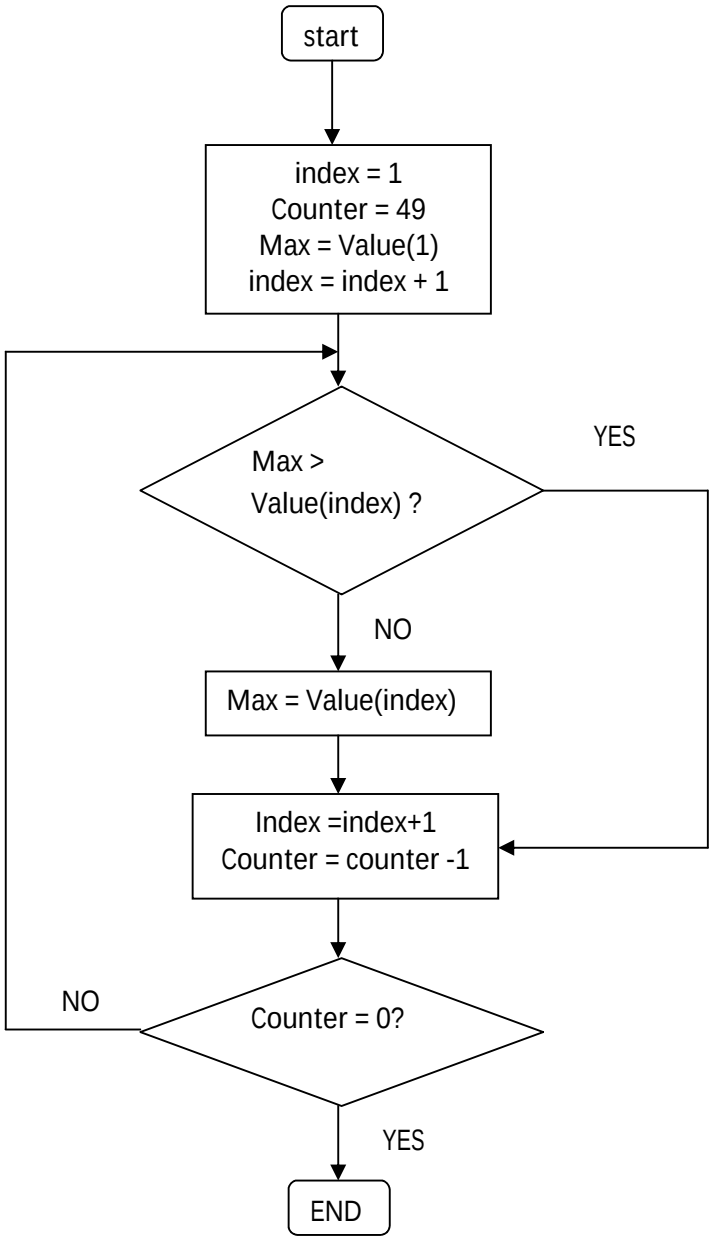
r0 contains number to be shifted
r2 the result register is initialized to 0
Repeat the following 32 times
    Extract the least significant bit from r0 into r3
    Add it to r2 after shifting it 1bit left (the first time is irrelevant as r2 contains all zeroes)
    Shift r0 right
END

Work through
r0 contains $d_{31}d_{30}d_{29}\ldots d_1\, d_0$ (Binary number to be shifted)
r2 = 0
Pass 1
d_0 is extracted
Add to r2 => r2 = 0….0 d_0
Shift r0 right by 1 bit => r0 = 0 $d_{31}d_{30}d_{29}\ldots d_1$
Pass 2
d_1 is extracted
r2 shifted left => r2 = 0…0 d_0 0
Add to r2 => 0…0 d_0 d_1
Shift r0 right by 1 bit => r0 = 0 0 $d_{31}d_{30}d_{29}\ldots d_2$
Pass 3
d_2 is extracted
r2 shifted left => r2 = 0…0 d_0 $d_1$0
Add to r2 => 0…0 d_0 d_1 d_2
Shift r0 right by 1 bit => r0 = 0 0 0 $d_{31}d_{30}d_{29}\ldots d_3$
….

```
      AREA ReverseReg, CODE, READONLY
      ENTRY

      LDR  r0, =0x1234568 ; r0 = register to be reversed
      LDR  r1, =32         ; 32-bit register bit counter
      MOV r2, #0           ; r2 = result (will be r0 reversed)
Loop  AND r3, r0, #1       ; r3 = current LSB of register to be
                           ;reversed
      ADD r2, r3, r2, LSL #1  ; shift result left 1, new
                              ; result LSB is r3
      MOV r0, r0, LSR #1      ; shift right to look at new LSB
                             ; of register to be reversed
      SUBS r1, r1, #1        ; loop 32 times
      BNE Loop
Done  B    Done
      END
```

2) Flowchart of Program



```
        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
```

3) The following C codes can be converted to assembly as follows:

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        ; i<10
                BLT      Loop0
```

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      ;i<11
                BLT      Loop0
```

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      ;k<30
                BLT     Loop2        ;
                ADD    r1, r1, #1    :j++
                CMP     r1, #40      ;j<40
                BLT      Loop1
                SUB      r0, r0, #1  ; i--
                CMP     r0, #3       ;i>=3
                BGE      Loop0
```