EEE3002 Microprocessors Tutorial 5 (2014 Sem 1)

1) What constant would be loaded into register r7 by the following instructions?
   a) MOV r7, #0x8C, 4
   b) MOV r7, #0x42, 30
   c) MVN r7, #2
   d) MVN r7, #0x8C, 4

2) Whenever possible, use the byte rotation scheme described in the lecture notes for the MOV instruction to generate an instruction needed to load the following constants into register r2. If it is not possible to use a MOV instruction, provide an alternative instruction that do the job.
   a) 0xA400
   b) 0x7D8
   c) 0x17400
   d) 0x1980

3) Without using the MUL instruction, give instructions that multiply register r4 by :
   a) 135
   b) 255
   c) 18
   d) 16,384

   and place your result in register r0. Your solution for each part should only have two or less instructions.

4) Write a program that converts a Celsius value in register r0 in Q0 format into Fahrenheit and stored the result in register r3 in Q15 format. Use the following relationship

$$F = 1.8C + 32$$

   where C and F are Celsius and Fahrenheit degrees. Also provide the **nearest** Fahrenheit degree in Q0 format and store the result in register r2.

EE3002 Microprocessors Tutorial 5 Solutions

1) It may be easier to do the rotation in binary if the shift is not multiples of 4.
   a) MOV r7, #0x8C, 4
      0x8C = 1000 1100B (B : Binary)
      Rotate Right by 4 bits becomes
      1100 0000 0000 0000 0000 0000 0000 1000B
      r7 contains 1100 0000 0000 0000 0000 0000 0000 1000B
      or 0xC0000008

   b) MOV r7, #0x42, 30
      0x42 = 0100 0010B
      Rotate Right by 30 bits == Rotate left by 2 bits
      r7 contains 0100001000B or 0x108

   c) MVN r7, #2
      2 = 10B
      Negate 10B =
      1111 1111 1111 1111 1111 1111 1111 1101B
      r7 contains 0xFFFFFFFD

   d) MVN r7, #0x8C, 4
      0x8C = 1000 1100B (B : Binary)
      Negate it becomes 1111 1111 1111 1111 1111 1111 0111 0011B
      = 0xFFFFFF73
      Rotate Right by 4 => 0x3FFFFFF7

2) Answer may not be unique!!!
   a) 0xA400
      By inspection, it is clear that 0xA4 can fit into a byte and the number 0xA400 constructed by rotating 0xA4 8 bits to the left or 32-8 =24 bits to the right.
      MOV r2, #0xA4, 24

      Alternative solution
      0xA400 in Binary is 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0B
      The number can also be expressed as 1 0 1 0 0 1B rotating 10 bits to the left or 22 bits to the right.
      MOV r2, #0x29, 22

   b) 0x7D8

0x7D8 in binary is 1 1 1 1 1 0 1 1 0 0 0B
Note that the underlined portion 1 1 1 1 1 0 1 1 can fit into 8 bits but this requires a rotate left of 3 bits. The amount of rotate must be an even number. Hence it is not possible to represent it by the MOV instruction.

Instead must use LDR r2, =0x7D8

c) 0x17400 in binary is 1 0 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0B
The number can be expressed as 1 0 1 1 1 0 1 B (0x5D) rotate left by 10 bits or 22 bits to the right.
MOV r2, #0x5D, 22

d) 0x1980 in binary is 1 1 0 0 1 1 0 0 0 0 0 0 0B
The number can be expressed as 1 1 0 0 1 1 0 B (0x66) rotate left by 6 bits or 26 bits to the right.
MOV r2, #0x66, 26

3) Try to find the multipliers closest to the numbers that are to the powers of 2
a) Multiply by 135?
The closest is multiply by 128 or 2^7 then add 7 times of it
Multiply by 7 can be constructed from multiply by 8 then subtract 1.

```
RSB   r1, r4, r4, LSL #3; r1 = r4*8 – r4 = 7*r4
ADD   r0, r1, r4, LSL #7; r0 = r1 + r4*128
```

b) Multiply by 255 can be perform by multiplying by 256 then subtract 1
```
RSB   r0, r4, r4, LSL #8; r0 = r4*256 – r4 = 255*r4
```

c) Multiply by 18 can be perform by multiplying by 16 and add 2 times of it
```
ADD  r1, r4, r4 ; r1 = 2*r4
ADD  r0, r1, r4, LSL #4 ; r0 = r1+16*r4
```

d) Multiply by 16384 or 2^{14}
```
MOV  r0, r4, LSL #14 ; r0 must first be initialized to 0
```

4) First convert 1.8 into Q15 format.
Round(1.8×2^{15}) = 58982 = 0xE666

Hence 1.8 in Q15 can be represented as 0xE666

Next convert 32 into Q15.
Round(32×2^{15}) = 0x100000
Hence 32 in Q15 can be represented as 0x100000

```
        AREA Celsius_Fahrenheit, CODE, READONLY
        ENTRY
CELS EQU  0xF               ; Celsius value in Q0 = 15
        LDR  r0, =CELS      ; load Celsius value in Q0
        LDR  r1, =0xE666    ; load 1.8 in Q15
        MUL  r3, r0, r1     ; signed multiply, result in Q15
        LDR  r2, =0x100000 ; load 32 in Q15
        ADD  r3, r3, r2     ; add 32
        ADD  r2, r3, #0x4000 ; add 0.5 before truncation
        MOV  r2, r2, LSR #15 ; shift right 15 to convert to Q0

Done    B    Done
        END
```

Notes

The first multiplication is Q0 X Q15 or (32.0) X (17.15). Ideally the result should be (49.15). However since 32 bit multiplication is used instead of 64 bit multiplication. The result is truncated to (17.15) or Q15 format.

Q15 or (17.15) format can be converted to Q0 or (32.0) by a right shift of 15 bits