EECS 373 F14 - HW2 - Solution Key
Prabal Dutta

Q1.a. LSL (immediate)

The LSL (immediate) instructon "shifts a register value left by an immediate number of bits, shifting in zeros, and writes the result to the destination register. It can optionally update the condition flags based on the result." Source: ARM ARM Section A7.7.67 of the document ARM DDI 0403D, ID021310.
web.eecs.umich.edu/~prabal/teaching/eecs373-f12/readings/ARMv7-M_ARM.pdf

Q1.b. Part (a)
LSR R2, R3, #3 -> use encoding T1 b/c R2, R3 < R8 & #3 < 32
|000|01|00011|011|010|
0000 1000 1101 1010
0 8 D A

Part (b)
LSR R2, R3, #23 -> use encoding T1 b/c R2, R3 < R8 & #23 < 32
|000|01|10111|011|010|
0000 1101 1101 1010
0 D D A

Part (c)
LSR R11, R3, #3 -> use encoding T2 b/c R11 > R8
|11101|01|0010|0|1111|0|000|1011|11|01|0011|
1110 1010 0100 1111 0000 1011 1101 0011
E A 4 F 0 B D 3

Q2.a. Assume little endian, and entries in hex

Base Addr	00	01	02	03
0x74000004	00	00	00	00
0x74000000	10	32	54	76
0x73FFFFFC	EF	CD	AB	89
0x73FFFFF8	00	00	00	00

Q2.b. Fill out memory

```
mov  r2, #100      // r2 <- 0x64
movw r1, #255      // r1 <- 0xFF
movt r1, #15       // r1 <- 0x000F00FF
strb r1, [r2, #2]! // mem(r2+2)<-byte(r1): [102]=0xFF; r2 <- 102
str  r1, [r2], #1  // mem(r2)<-r1: [102]=0x000F00FF; r2<-103
strh r2, [r2, #-3] // mem(r2-3)<-hword(r2): [100]=0x0067
```

Mem	Val
100	0x67
101	0x00
102	0xFF
103	0x00
104	0x0F
105	0x00
106	0x00
107	0x00

3. C program -> Assembly program

```
  void main() {
    int i, a=1;
    for (i=0;i<5;i++) {
      a = a + i;
      print(a);
    }
  }

  main:   push  {r4, r5, lr}  % callee save
          mov   r4, #1        % a=0
          mov   r5, #0        % i=0
  loop:   cmp   r5, #5        % i<5?
          bge   done          % break
          add   r4, r5        % a=a+i
          mov   r0, r4        % pass a in r0
          bl    print         % call
          b     loop          % loop
  done:   pop   {r4, r5, pc}  % callee restore
```

4. Assembly program -> C program

```
  movw r0, #0030
  movt r0, #2008
  ldr  r1, [r0]
  add  r1, r1
  str  r1, [r0]

  uint32_t* a = (uint32_t*)0x20080030;
  *a += *a;
```

5. ABI-compliant that returns average or four integer arguments.

```
  mean:  add  r0, r0, r1
         add  r0, r0, r2
         add  r0, r0  r3
         lsr  r0, r0, #2
         bx   lr
```