Q1. a. Shift a register value right (1pts)

Shift in 0's (1pt)

Shift by other register value (1pt)

* No points if text is copied (plagarized) from ARM ARM (unless they cite the source)

b.
```
    0000:0 : : : :: : mm:m ddd
 i: 0000:0110:111 0 :1 0 1 0      => 0x06EA   (2 pts)
ii: 0100:0000:10 mm:m ddd
    0100:0000:10 1 0 :1 0 1 0     => 0x40AA   (2pts)
```

c. R11 cannot be encoded (Rd is on 3 bits for encoding T1) (1 pt)

LSLS.W R11, R3, #13 (1 pt)

```
110:1010:0 1 0 S:1111:0 : : :  d d d d: : : 0 0:m m m m   => 0xEA5F3843   (1 pt)
              :0101:       :0011:1011:0100:0011
```

Q2. a.

BASE	00	01	02	03
....04	00	00	00	00
00	98	BA	DC	FE
FC	67	45	23	01
F8	00	00	00	00

2 pts endianess

3 pts correct location in memory

b.
```
mov   r1, #0x100        r1 = 0x100
movw  r2, #136          r2 = 0x88
movt  r2, #8            r2 = 0x00080088
strh  r2,[r1], #-3      [100] = 0x88, [101] = 0x00 ; r1 = 94
str   r2,[r1, #2]       r1 = 102, [102] = 88, [103] = 00, [104] = 08, [105 = 00]
```

0FE	0
0FF	0
100	0x88
101	0
102	0x88
103	0
104	0x08
105	0

5 pts if correct. 2pts if you can follow their work and they were close.

Q3. Two solutions... (examples) THIS QUESTION IS HARD TO GRADE! PLEASE TAKE YOUR TIME! -5 pts if no comments. -2 pts per mistake until 0.

callee-save:

```
main:   push  {r4, r5, r6, lr}    ; callee-save regs
        mov   r4, #1              ; a=1
        mov   r5, #0              ; i=0
        mov   r6, #2              ; stride=2
loop:   cmp   r5, #9              ; i<9?
        bge   done                ; break
        add   r4, r5              ; a += i
        mov   r0, r4              ; arg0 = a
        bl    print               ; call print
        add   r5, r6              ; i += 2
        b     loop                ; (loop)
done:   mov   r0, r4              ; set return = a
        pop   {r4, r5, r6, pc}    ; restore regs + return
```

must save any regs used >r3.

must have label main

set initial values?

inefficient is ok

correct branch

arg in r0?

must bl not just b

don't forget to increment i

return in r0?

must restore regs >r3

pop is PC, NOT LR

caller-save:

```
main:   push  {lr}        ; save LR
        mov   r0, #1      ; a=1
        mov   r1, #0      ; i=0
loop:   cmp   r1, #9      ; i<9?
        bge   done        ; break
        add   r0, r1      ; a=a+i
        push  {r0, r1}    ; save regs.
        bl    print       ; call print
        pop   {r0, r1}    ; restore regs
        add   r1, #2      ; i+=2
        b     loop        ; (loop)
done:   pop   {pc}        ; return
```

print call will overwrite LR, must save

must cmp before branch (unless careful use of subs + offset, UNLIKELY!)

print fn call corrupts r0-r3, must save/restore if they are used

Q4.

```
uint32_t* x = (uint32_t *) 0xA5001008;
*x = *x - 5;
```

create constant	2pts
assign as pointer	2pts
↳ minor syntax issue	-1pt
load pointer value	2pts
do subtraction	2pts
store result to pointer	2pts

Q5.

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

-2 pts if missing label "mean"

-2 pts if missing bx lr

-2 pts per instruction > 5 instructions

+5 pts extra credit if < 5 instructions (must be correct)

-5 pts if doesn't calculate mean correctly but is close

-10 pts if not close

(cannot go below 0).