EECS 373 Fall 2017 Homework #2

Due September 18th before the start of class. Late homework is not accepted.

Name: _____ unique name: _____

You are to turn in this assignment filling in the blanks as needed. ***Assignments that are unstapled, are not done on this worksheet, or are difficult to read will lose at least 50% of the possible points and we may not grade them at all.*** This is an individual assignment; all work should be your own. 50 points.

---

1. Using the ARMv7-M Architecture Reference Manual.
   a. In straightforward English, explain the difference between the 'B', 'BX' and 'BL' instructions. **[4 points]**

B: branches to an address given by an immediate value/label.
BX: branches to an address stored in a register
BL: branches to an address given by a label and sets the link register to PC+1.

   b. Write the hexadecimal for the machine code you would expect to get for the following instructions in Thumb mode.
   [6 points, 2 each)
      a. `LDR R1, [R3, #3]`

01101[00011][011][001] = 0x68D9

      b. `MOVW R4, #2000`

11110[0]100100[0000]0[111][0100][11010000] = 0xF24074D0

      c. `BX R14`

010001110[1110]000 = 0x4770

2. For each of the following program segments, assume you start with all memory locations in question equal to zero. Indicate the values found in *these* (*note that they aren't always sequential*!) memory locations when the programs end. Write all answers in hex. **[10 points, 5 for each part]**

a.
```
BASE_EMC = 0x64000000;
uint32_t *a = (uint32_t*)BASE_EMC;
*(a-1) = 0xabcdef89;
*a = 0x01234567;
```

Address	value
0x63ffffe	CD
0x63fffff	AB
0x64000000	67
0x64000001	45
0x64000002	23
0x64000003	01
0x64000004	00

b.
```
mov r2, #100
movw r1, #2000
movt r1, #100
strh r1, [r2], 2
str r1, [r2,1]!
strb r1,[r2,-1]
```

Address	Value
100	D0
101	07
102	D0
103	D0
104	07
105	64
106	00

Hint: Page A6-15 of the ARMv7-M Architecture Reference Manual may be useful here.

3. Write a program in UAL assembly which does the same thing as the following C code. Have the function return to whatever called it just as any function might. **[12 points]**

```
uint32_t avg_5(uint32_t* a){
    uint32_t sum = 0;
    int i = 0;
    for(i = 0; i < 5; i++){
        sum += a[i];
    }
    return sum / 5;
}
```

```
avg_5 :
    push{r4}
    mov r1, #0 //i =0
    mov r2, #0 // sum = 0
    mov r3, #5

loop:
    ldr r4, [r0, r1] @r4 = mem[r0 + r1] = a[i]
    add r2, r2, r4 @r2 = sum =  r2 + a[i]
    add r1, r1, #1 @i = i+1
    cmp r1, r3 @ check i= 5
    beq loop

    sdiv r0, r2, r3 @ r0 = sum / 5
    pop{r4}
    bx lr
```

4. Write a program in C that does the same thing as the following ARM assembly language code. Your C code must not exceed four lines and should compile without any warnings or errors. **[8 points]**

```
movw r1, #0x1248
movt r1, #0xB650
movw r2, #0xB700
movt r2, #0x1008
eor r0, r0, r2
str r0, [r1]
```

```
int main(uint32_t* ptr0) {
    uint32_t* ptr1 = 0xb6501248;
    uint32_t* ptr2 = 0xb1008700;
    *ptr1= ptr0 ^ ptr2;
}
```

5. Write an ABI compliant UAL function named “mulAcc” which takes 3 integer arguments a, b, and c; multiplies b and c together; and returns that value added to a. **[10 points]**

```
mulAcc:
        mul r1, r1, r2
        add r0, r0, r1
        bx lr
```