# EEL4768C.04 Homework 2 Due 10/08/19

Brandon Thompson 5517

October 8, 2019

1. Consider memory storage of a 32-bit word stored at memory word 15 in in a byte-addressable memory.

   (a) What is the byte address of memory word 15?
   $15 \times 4 = 60$ or 0x3C in hexadecimal.

   (b) What are the byte addresses that memory word 15 span?
   Words are 32 bits, if the word starts at byte address 60 it will continue until byte address 63. 0x3C → 0x3F

   (c) Draw the number `0xFF223344` stored at word 15 in both big-endian and little-endian machines.

| Address: | 0x3C | 0x3D | 0x3E | 0x3F |
|---|---|---|---|---|
| Big Endian: | FF | 22 | 33 | 44 |
| Little Endian: | 44 | 33 | 22 | FF |

Figure 1: Big and little endian representation

2. Convert the following MIPS assembly code into machine language. Write the instructions in hexadecimal.
   `add $t0, $s0, $s1`
   opcode: 0, rs: 16, rt: 17, rd: 8, shamt: 0, funct: 32
   000000 10000 10001 01000 00000 100000
   0000|0010|0001|0001|0100|0000|0010|0000 → **0x02114020**
   `lw $t0, 0x20($t2)`
   opcode: 35, rs: 10, rt: 8, imm: 32
   100011 01010 01000 0000000000010000
   1000|1101|0100|1000|0000|0000|0001|0000 → **0x8D480010**
   `addi $s0, $0, -10`
   opcode: 8, rs: 0, rt: 16, imm: -10
   001000 00000 10000 1111111111110110
   0010|0000|0001|0000|1111|1111|1111|0110 → **0x2010FFF6**

3. The nori instruction is not part of the MIPS instruction set, because the same functionality can be implemented using existing instructions. write a short assembly code snippet that has the following functionality: $t0 = $t1 NOR 0xF234. Use as few instructions as possible.

```
ori $t0, $t1, 0xF234
nor $t0, $t0, $0
```

4. Implement the following high-level code segments using the slt instruction. Assume the integer variables g and h are in registers $s0 and $s1 respectively.

```
if(g ≤ h) {g = 0;}
else {h = 0;}

slt $t0, $s1, $s0    # if(g ≤ h)
beqz $t0, label1     # g is ≤ h
addi $s1, $0, 0      # h = 0
j end                # finish program
label1:
addi $s0, $0, 0      # g = 0
j end                # skip else statement
```