

CS2810 Homework 8 – 80 points - Due by midnight November 9. Submit this homework through Eagle.

For this homework assignment you may work individually or in a group of up to 3 students. If working in a group, be sure that when you turn in your homework through Eagle you include the names of everyone in the group.

Questions

1. Assume that the instruction labeled LMN is at location 300; that the instruction labeled ABC is at location 200, and that the instruction labeled XYZ is at location 404. **ALL** of these values are octal.

(5) a.) Give the hexadecimal for the low order 16-bits in the machine instruction for:

```
LMN: beq $t1,$t2, ABC
```

(5) b.) Give the hexadecimal for the low order 16-bits in the machine instruction for:

```
LMN: beq $t1,$t2, XYZ
```

(5) c.) Give the hexadecimal for the low order 26-bits in the machine language instruction for:

```
LMN: j ABC
```

2. (15 points) Write a MIPS assembly language version of the following C code segment:

```
int A[100], B[100];
    for (i=1; i < 100; i++) {
        A[i] = A[i-1] + B[i];
    }
```

At the beginning of this code segment, the only values in registers are the base address of arrays A and B in registers \$a0 and \$a1 respectively. Avoid the use of multiplication instructions—they are unnecessary. Remember, in C, the first element of the array A is at A[0].

3. (20 points) Convert the C function below to MIPS assembly language. Make sure that your assembly language code could be called from a standard C program (that is to say, make sure you follow the MIPS calling conventions).

```
unsigned int sum(unsigned int n)
{
```

```

    if (n == 0) return 0;
    else return n + sum(n-1);
}

```

The stack grows downward (toward lower memory addresses).
The following registers are used in the calling convention:

Register Name	Register Number Usage
\$zero	0
\$at	1
\$v0, \$v1	2, 3
\$a0 - \$a3	4 - 7
\$t0 - \$t7	8 - 15
\$s0 - \$s7	16 - 23
\$t8, \$t9	24, 25
\$k0, \$k1	26, 27
\$gp	28
\$sp	29
\$fp	30
\$ra	31

4.) (10 points) In the snippet of MIPS assembler code below, how many times is instruction memory accessed? How many times is data memory accessed? (Count only accesses to memory, not registers.)

```

lw $v1, 0($a0)
addi $v0, $v0, 1
sw $v1, 0($a1)
addi $a0, $a0, 1

```

5.) (5 points) MIPS has the following pseudo-instruction

```
abs $t1    # $t1 = absolute value of $t1, overflow ignored
```

Give the minimal MIPS instruction sequence to perform this operation.

6.) (15 points) Stored, starting in memory location A, is a null terminated string of ASCII characters, each of which is a decimal digit. Write the MIPS code to store in register \$t0 the decimal equivalent of that string. For example, if the string is "174\0", then after executing your code, \$t0 should hold the value 174. You do not need to be concerned about negatives, overflow, or illegal characters.