1 Review for Mid-term Exam 1

Most frequently asked question about any exam: *How many questions are there on the exam?*

3 (on this exam)

Other facts about the organization:

- Exam takes 50 minutes.
- No written references or calculators are allowed.
- To receive any partial credit, show your work and comment your code.

2 Things to know for the Exam

(*an incomplete list*)

1. Bitwise Logical Programming:
   - Shifting
   - Bitwise `and`, `or`, `not`
   - Applying these operations to extract and modify bitfields
2. MIPS assembly:
   - arithmetic, memory, control flow
   - translate C to assembly
   - translate assembly to C (*can you name this function? what does it do?*)
   - should recognize loops, if's, types, arrays, etc.
3. Functions:
   - register saving conventions
   - stack (*how does it work for function calls?*)
   - recursion
4. Basic understanding of assembly level concepts
   - pointers, pointer arithmetic, structures
   - machine language
   - how machines know the type of a variable
   - compiling, linking, loading, MIPS memory image
   - role of assembly programming, 90/10 rule
5. The following things will not be tested until Mid-term 2
   - Floating Point Representations (single-precision IEEE 754)
   - Endianness (but we *don't* expect you to memorize which is little and which is big)
   - I/O programming and interrupts

3 Problems

1. **Understanding MIPS programs**

```
flathead:
     addi $t0, $a2, 1
loop:
     bge  $t0, $a1, exit
     mul  $t1, $t0, 4
     add  $t1, $t1, $a0
     lw   $t2, 0($t1)
     sub  $t1, $t1, 4
     sw   $t2,  0($t1)
     addi $t0, $t0, 1
     j loop
exit:
     jr   $ra
```

   (a) Translate the `flathead` function above into a high-level language like C or Java. You should include a header that lists the types of any arguments and return values. Also, your code should be as concise as possible, without any gotosor explicit pointers. We will not deduct points for syntax errors unless they are significant enough to alter the meaning of your code.

   (b) Describe briefly, in English, what this function does.

2. **Bit-wise Logical and Shifting** Compute the result of the following expression and write it in hexadecimal notation:

```
0xdeadbeef | (0x81 << 22)
```

3. Here is a C function *count*. Its arguments A[ ] and n are an integer array and the number of items in the array. The code passes each element of A[ ] to another function `test`, and counts the number of times `test` returns 1.
   Translate this into a MIPS assembly language function. Argument registers `$a0` and `$a1` will correspond to A[ ] and n, and the return value should be placed in `$v0` as usual.

   - Assume that we already have the MIPS function test, which takes an integer argument in `$a0` and returns 0 or 1 in `$v0`.
   - You will not be graded on the efficiency of your code, but you must follow all MIPS conventions.

```
int count(int A[], int n)
{
     int i;
     int num = 0;

     for(i = 0; i < n; i++){
          if (test(A[i]) == 1) {
               num++;
          }
     }
     return num;
}
```