Homework 3

Computer Architecture
CSCI 361, Fall 2014

Due: October 20, 2014 in class

Instructions: Complete the problems enumerated below. No collaboration is permitted on this or any assignment. Correct answers without work shown will not receive full credit. Answers must typed and neatly formatted. You will submit a printed hardcopy of your work at the beginning class on the specified due date. Include your name and email address on each page of your submission. Please include a header containing your name on every page. Late assignments will not be accepted.

Problem 1 [10 pts]

Consider the following MIPS loop:

```
LOOP: slt $t2, $0, $t1
      beq $t2, $0, DONE
      subi $t1, $t1, 1
      addi $s2, $s2, 2
      j LOOP
DONE:
```

(a) Assume that the register `$t1` is initialized to the value 10. What is the value in register `$s2` assuming the `$s2` is initially zero.

(b) For the loop above and using the values given in part (a), write the equivalent C code routine. Assume that the registers `$s1, $s2, $t1,` and `$t2` are integers `A, B, i`, and `temp`, respectively.

Problem 2 [10 pts]

Suppose the program counter (PC) is set to 0x2000 0000.

(a) Is it possible to use the branch-on-equal (beq) MIPS assembly instruction to set the PC to this same address?

(b) Is it possible to use the jump (j) MIPS assembly instruction to set the PC to the address as 0x4000 0000?

Problem 3 [10 pts]

Write MIPS assembly code that creates the 32-bit constant

$$0010\ 0000\ 0000\ 0001\ 0100\ 1001\ 0010\ 0100_2$$

and stores that value to register `$t1`.

Problem 4 [10 pts]

Provide a minimal set of MIPS instructions that may be used to implement the following pseudoinstructions:

(a)
```
not $t1, $t2  # bit-wise invert
```

(b)
```
move $t1, $t2
```

(c)
```
blt $t6, $t7, label  # branch if less than
```

Problem 5 [10 pts]

The follow instruction is not included in the MIPS instruction set:

```
rpt $t2, LOOP  # if(R[rs]>0) R[rs]=R[rs]-1, PC=PC+4+BranchAddr
```

where `LOOP` is some label.

(a) If this instruction were to be implemented in the MIPS instruction set, what is the most appropriate instruction format?

(b) What is the shortest sequence of MIPS instructions that performs the same operation?

Problem 6 [20 pts]

Translate the following C code to MIPS assembly code. Use a minimum number of instructions.

```
for( i= 0; i < a; i++)
  for( j= 0; j < b; j++)
      D[4 * j] = i + j;
```

Assume that the values of `a`, `b`, `i`, and `j` are in registers `$s0`, `$s1`, `$t0`, and `$t1`, respectively. Also, assume that register `$s2` holds the base address of the array `D`.

Problem 7 [20 pts]

Translate function `f` into MIPS assembly language:

```
int f (int a, int b, int c, int d ){
    return func(func(a, b), c + d);
}
```

As needed, use registers `$t0` through `$t7`, beginning with the lower-numbered registers first. Assume there exists a function `func` with declaration:

```
int func(int a, int b);
```

You do not need to write function `func` but should call it, using label `func`.

Problem 8 [10 pts]

(a) Assume 185 and 122 are unsigned 8-bit decimal integers. Calculate 185+122 and show your work. Is there overflow, underflow, or neither?

(b) Assume 185 and 122 are unsigned 8-bit decimal integers. Calculate 185−122 and show your work. Is there overflow, underflow, or neither?

(c) Assume 185 and 122 are signed 8-bit decimal integers stored in sign-magnitude format. Calculate 185+122 and show your work. Is there overflow, underflow, or neither?

(d) Assume 185 and 122 are signed 8-bit decimal integers stored in sign-magnitude format. Calculate 185−122 and show your work. Is there overflow, underflow, or neither?