CSCI 2321 (Principles of Computer Design), Spring 2013

Homework 2

Credit: 30 points.

1 Reading

Be sure you have read chapter 2, sections 2.1 through 2.7.

2 Problems

Do the following problems. You may write out your answers by hand or using a word processor or other program, but please submit hard copy, either in class or in my mailbox in the department office.

1. (5 points) Do problems 2.6.1 and 2.6.4 from the textbook.

   For problem 2.6.1, use the following line of C code:

   ```
   B[8] = A[i] + A[j]
   ```

   For problem 2.6.4, use the following sequence of MIPS instructions:

   ```
   addi    $t0, $s6, 4
   add     $t1, $s6, $0
   sw      $t1, 0($t0)
   lw      $t0, 0($t0)
   add     $s0, $t1, $t0
   ```

2. (5 points) Do problems 2.10.1, 2.10.3, and 2.10.4 from the textbook.

   For problems 2.10.1 and 2.10.3, use the following binary value:

   0000 0001 0100 1011 0100 1000 0010 0010

   Do problem 2.10.4 twice, once for each of the following MIPS instructions:

   ```
   addi    $t0, $t0, 0
   ```

   and

   ```
   sw      $t1, 32($t2)
   ```

3. (5 points) Do problems 2.11.1, 2.11.2, and 2.11.3 from the textbook.

   For all three problems use the following base-16 number as input:

   0x01084020

4. (5 points) Do problems 2.13.1, 2.13.2, and 2.13.4 from the textbook.

   For problem 2.13.1, use the following initial values:

   `$t0` = 0xF00DD00D, `$t1` = 0x11111111

   and the following sequence of MIPS assembly language statements:

```
sll     $t2, $t0, 44
or      $t2, $t2, $t1
```

   Correction: In the `sll` instruction, the 44 is not a legitimate value (shift amount can be at most 31). Since I did not notice this until preparing a sample solution, I'll use this not-really-legit value anyway.

   For problem 2.13.2, use the same initial values as for 2.13.1 and the following sequence of MIPS assembly language statements:

```
sll     $t2, $t0, 4
andi    $t2, $t2, -1
```

   Correction: In the `andi` instruction, the -1 is — well, it *could* be a legitimate value if expressed as an unsigned constant (i.e., as 0xffff).

   For problem 2.13.4, use the following as initial values:

   `$t0` = 0x0000A5A5, `$t1` = 0x00005A5A

   and give results for the following two sequences of MIPS instructions (starting with the initial values in both cases):

```
sll     $t2, $t0, 1
andi    $t2, $t2, -1
```

   and

```
andi    $t2, $t1, 0x00F0
srl     $t2, $t2, 2
```

5. (10 points) Do problems 2.18.2 and 2.18.5 from the textbook.

   For problem 2.18.2, use the following C code:

```
for (i=0; i<a; i++)
        a += b;
```

   Correction: It seems unlikely that the author(s) of this problem really meant to write a loop in which the value of the variable used in the loop test (`a`) is changed in the loop body. However, it is legitimate C, so it can be compiled ...

   For problem 2.18.5, use the following sequence of MIPS instructions:

```
        addi    $t1, $0, 50
LOOP:   lw      $s1, 0($s0)
        add     $s2, $s2, $s1
        lw      $s1, 4($s0)
        add     $s2, $s2, $s1
        addi    $s0, $s0, 8
        subi    $t1, $t1, 1
        bne     $t1, $0, LOOP
```

Correction: There is actually no such instruction as `subi` in the MIPS instruction set. Presumably the author(s) of the problem meant to write

```
        addi    $t1, $t1, -1
```