CS 3340 Computer Architecture

Homework 4

1. Consider the following code used to implement the instruction
   .   sllv $s0, $s1, $s2
   which uses the least significant 5 bits of the value in register $s2 to specify the amount register $s1 should be shifted left:

   ```
   .data
           mask: .word 0xfffff83f
   .text
   start:   lw    $t0, mask      # load mask into $t0
            lw    $s0, shifter   # load the instruction at label ‘‘shifter’’ into $s0
            and   $s0, $s0, $t0  # mask out the shift amount (making it zero)
            andi  $s2, $s2, 0x1f # mask out the least significant 5 bits of $s2
            sll   $s2, $s2, 6    # shift those bits left 6 places to align with shamt
            or    $s0, $s0, $s2  # or the value from $s2 into the shift inst’n as shamt
            sw    $s0, shifter   # store the modified inst’n back where we got it
   shifter: sll   $s0, $s1, 0    # execute the modified inst’n
   ```

   Add comments to the code and write a paragraph describing how it works. Note that the two `lw` instructions are pseudoinstructions that use a label to specify a memory address that contains the word of data to be loaded. Why do you suppose that writing `self-modifying code` such as this is a bad idea (and oftentimes not actually allowed)?

2. The following MIPS instruction sequence could be used to implement a new instruction that has two register operands. Give the instruction a name and describe what it does. Note that register $t0 is being used as a temporary:

   ```
            srl   $s1, $s1, 1   #
            sll   $t0, $s0, 31  # These 4 instructions accomplish
            srl   $s0, $s0, 1   # new ‘‘$s0 $s1’’
            or    $s1, $s1, $t0 #
   ```

3. Write a program in MIPS assembly language to convert an ASCII decimal string to an integer. Your program should expect register $a0 to hold the address of a null-terminated string containing some combination of the digits 0 through 9. Your program should compute the integer value equivalent to this string of digits, then place the number in register $v0. Your program need not handle negative numbers and need not be concerned about values larger than $2^{31} - 1$.

   If a nondigit character appears anywhere in the string, your program should stop with the value -1 in register $v0. For example, if register $a0 points to a sequence of three bytes $50_{10}, 52_{10}, 0_{10}$ (the null-terminated ASCII string 24), then when the program stops, register $v0 should contain the value 24_{10}, (or 11000 in binary). (The subscript `ten` means base 10.)
   Submit your solution to question 3 on WebCT.