CS 2321 Solutions to Laboratory Problem Set 2

February 9, 2006

- Problem 2.2

  Using J (to do the arithmetic):

```
   16 #. 7 , (6 $ 15) , 10
2147483642
```

- Problem 2.3

  Using J (to do the arithmetic):

```
   12 10 15 14 15 10 12 14 { '0123456789ABCDEF'
CAFEFACE
```

- Problem 2.4

  Since the MIPS architecture includes add-immediate, and since the immediate field is signed, subtract-immediate would be redundant. In the spirit of RISC, the instruction is not included.

- Problem 2.6

```
     sll  $t0, $t3, 9   # shift $t3 left by 9, store in $t0
     srl  $t0, $t0, 15  # shift $t0 right by 15
```

- Problem 2.15

  The results from using *if* - *else* are better.

```
set_array: addi  $sp, $sp, -52      # move stack pointer
           sw    $fp, 48($sp)       # save frame pointer
           sw    $ra, 44($sp)       # save return address
           sw    $a0, 40($sp)       # save parameter (num)
           addi  $fp, $sp, 48       # establish frame pointer

           add   $s0, $zero, $zero  # i = 0
           addi  $t0, $zero, 10     # max iterations is 10
loop:      sll   $t1, $s0, 2        # $t1 = i * 4
           add   $t2, $sp, $t1      # $t2 = address of array[i]
           add   $a0, $a0, $zero    # pass num as parameter
           add   $a1, $s0, $zero    # pass i as parameter
           jal   compare            # call compare(num, i)
           sw    $v0, 0($t2)        # array[i] = compare(num, i);
           addi  $s0, $s0, 1
           bne   $s0, $t0, loop     # loop if i<10

           lw    $a0, 40($sp)       # restore parameter (num)
```

```
            lw    $ra, 44($sp)       # restore return address
            lw    $fp, 48($sp)       # restore frame pointer
            addi  $sp, $sp, 52       # restore stack pointer
            jr    $ra                # return

compare:    addi  $sp, $sp, -8       # move stack pointer
            sw    $fp, 4($sp)        # save frame pointer
            sw    $ra, 0($sp)        # save return address
            addi  $fp, $sp, 4        # establish frame pointer

            jal   sub                # can jump directly to sub
            slt   $v0, $v0, $zero    # if sub(a,b) >= 0, return 1
            slti  $v0, $v0, 1

            lw    $ra, 0($sp)        # restore return address
            lw    $fp, 4($sp)        # restore frame pointer
            addi  $sp, $sp, 8        # restore stack pointer
            jr    $ra                # return

sub:        sub   $v0, $a0, $a1      # return a-b
            jr    $ra                # return
```

- Problem 2.30

```
        sll  $a2, $a2, 2         # max i= 2500 * 4
        sll  $a3, $a3, 2         # max j= 2500 * 4
        add  $v0, $zero, $zero   # $v0 = 0
        add  $t0, $zero, $zero   # i = 0
outer:  add  $t4, $a0, $t0       # $t4 = address of array 1[i]
        lw   $t4, 0($t4)         # $t4 = array 1[i]
        add  $t1, $zero, $zero   # j = 0
inner:  add  $t3, $a1, $t1       # $t3 = address of array 2[j]
        lw   $t3, 0($t3)         # $t3 = array 2[j]
        bne  $t3, $t4, skip      # if (array 1[i] != array 2[j]) skip $v0++
        addi $v0, $v0, 1         # $v0++ skip
        addi $t1, $t1, 4         # j++
        bne  $t1, $a3, inner     # loop if j != 2500 * 4
        addi $t0, $t0, 4         # i++
        bne  $t0, $a2, outer     # loop if i != 2500 * 4
```

The code determines the number of matching elements between the two arrays and returns this number in register $v0.