CS 351 Computer Architecture Fall 2007

Home Work # 4 solutions

1. Implement Insertion Sorting in MIPS assembly language. Test the program on the input array of numbers 9, 4, 12, -6, 11, 27, 314, 0, 0, 41, -245, 409. You are to write a procedure called ***insert*** that executes the j-th pass of insertion sorting and a recursive procedure ***sort*** for which the pseudo-code is provided below. Also write a *print* procedure that prints the sorted array.

```
procedure sort( int[] A, int n)
{
 if (n == 1) return;
 sort(A, n - 1);
 insert(A, n - 1, A[n]);
}
```

Answer:

```
# data segment

.data
array:.word 9,4,12,-6,11,27,314,0,0,41,-245,409
length: .word 12
space:  .asciiz " "
newline: .asciiz "\n"
ans1:   .asciiz "The input array is "
ans2: .asciiz "The sorted array is "

# text segment

.text
.globl main

main:

       la $a0,ans1
       li $v0,4
       syscall                         # print

       la      $a0, array             #$a0 contains the base address
       la      $t0, length
       lw      $t0, ($t0)
       sll     $t0, $t0, 2
       addi    $t0, $t0, -4
       add     $a1, $a0, $t0          # $a1 contains the address of the
last memory location of the array
       jal     print

       la $a0, newline                 # print newline
       li $v0, 4
       syscall

       la      $a0, array
       jal     isort                    #insert
       la $a0,ans2
       li $v0,4
       syscall                          # print

       la      $a0, array               #$a0 contains the base address
       jal     print


       li $v0, 10                  # done
       syscall

# procedure print

print:move $t0, $a0
loop: blt $a1,$t0,exit                 # if $t0 > $a1 exit

       lw $a0,($t0)
       li $v0,1
       syscall

       la $a0, space                     # print comma
       li $v0,4
       syscall

       addi $t0, $t0, 4                   # move to the next array
element
       j loop

exit:  jr $ra

# procedure insert will insert the key ($a1) in the sorted list ($a0),
#($a0+4), ..., ($a1-4)

insert: lw $t1, ($a1)
        addi $t2, $a1, -4
  loop1: blt $t2, $a0, done
        lw $t3, ($t2)
        blt $t3, $t1, done
        sw $t3, 4 ($t2)
        addi $t2, $t2, -4
        j loop1
  done: sw $t1, 4 ($t2)
        jr $ra

isort: beq $a0, $a1, done1
       addi $sp, $sp, -8
       sw   $ra, 4 ($sp)
       sw   $a1, ($sp)
       addi $a1, $a1, -4
       jal isort
       lw  $a1, ($sp)
       jal insert
       lw $ra, 4 ($sp)
       lw $a1, ($sp)
       addi $sp, $sp, 8
 done1:jr $ra
```

2. Problem 7.8. The input for which you should submit the output is 2415919104.

Solution:

```
.text
        .globl  main
main:
        li      $v0,4           # mesg1 asking for a number
        la      $a0, msg1
        syscall
        li      $v0,5           # system call that reads an integer
        syscall
        move    $a0,$v0
        jal    sqrt
        move   $t0, $v0

exit:   li      $v0, 4          # print mesg2
        la      $a0, msg2
        syscall

li      $v0,1                   # print sum
        move    $a0, $t0
        syscall
        li      $v0,4           # print an end of line
        la      $a0, cr
        syscall
        li      $v0,10          # exit
        syscall

sqrt: move $t0, $zero
      la  $t1, max
      lw  $t1, ($t1)
      addi   $t6, $zero, 1
loop: sub $t2, $t1 $t0
      sle $t3, $t2, $t6
      bne $t3 $zero, done
      add $t4, $t0, $t1
      srl $t4, $t4, 1
      mul $t5, $t4, $t4
      beq $t5, $a0, done1
      blt $t5, $a0, right
      move $t1, $t4
      j loop
right: move $t0, $t4
       j loop
done1: move $v0, $t4
       j final
done:  move $v0, $t0
final: jr $ra

.data
max: .word 0x0000a000

msg1:   .asciiz "Enter a integer number:  "
msg2:   .asciiz "The square root is =    "
cr:     .asciiz "\n"
```

3. Implement Selection Sorting and test it on the same input presented for Problem 1 above. Most of the code for this problem can be found in the class notes.

Solution:

```
           # data segment

.data
array:.word 12 11 10 9 8 7 6 5 4 3 2 1
length: .word 12
space:  .asciiz " "
newline: .asciiz "\n"
ans1:   .asciiz "The input array is "
ans2: .asciiz "The sorted array is "

# text segment

.text
.globl main

main:

       la $a0,ans1
       li $v0,4
       syscall                         # print

       la      $a0, array             #$a0 contains the base address
       la      $t0, length
       lw      $t0, ($t0)
       sll     $t0, $t0, 2
       addi    $t0, $t0, -4
       add     $a1, $a0, $t0          # $a1 contains the address of the
       # last memory location of the array
       jal     print

       la $a0, newline                 # print newline
       li $v0, 4
       syscall

       la      $a0, array
       jal     ssort                    #call ssort

       la $a0,ans2
       li $v0,4
       syscall                         # print

       la      $a0, array              #$a0 contains the base address
       jal     print


       li $v0, 10                 # done
       syscall

# procedure print

print:move $t0, $a0
loop2: blt $a1,$t0,exit                 # if $t0 > $a1 exit
```

```
       lw $a0,($t0)
       li $v0,1
       syscall

       la $a0, space                    # print comma
       li $v0,4
       syscall

       addi $t0, $t0, 4                  # move to the next array element
       j loop2

exit:  jr $ra

# procedure select finds the smallest key among ($a0) ... ($a1) and swaps
# it with $a0

select:move   $t0, $a0
       move   $t1, $a0
  loop:addi   $t1, $t1, 4
       blt    $a1, $t1, done
       lw     $t2, ($t0)
       lw     $t3, ($t1)
       slt    $t4, $t2, $t3
       bne    $t4, $zero, loop
       move   $t0, $t1
       j loop
 done: lw     $t2, ($a0)
       lw     $t3, ($t0)
       sw     $t2, ($t0)
       sw     $t3, ($a0)
       jr     $ra

# selection sorting procedure

ssort: move  $s0, $a0
loop1:
       beq   $a1, $s0, done1  # finished sorting!
       addi  $sp, $sp, -8
       sw    $ra, ($sp)       # save $ra
       sw    $a0, 4 ($sp)     # save $a1
       move  $a0, $s0         # move $s0 into $a0
       jal   select
       lw    $ra, ($sp)       # restore the stack
       lw    $a0, 4 ($sp)
       addi  $sp, $sp, 8
       addi  $s0, $s0, 4
       j     loop1
done1: jr $ra
```