CS/COE 0447 Fall 2009
Lab 7: Bit Manipulation
Solution

```
#Test Code:

.data
str:  .space 33
buf:  .space 32
newline:    .asciiz "\n"

.text
      #s0 holds src
      #s1 holds dest
      #s2 holds start
      #s3 holds end

      #Prompt for string src
      la $a0, str
      li $a1, 32
      li $v0, 8
      syscall

      #Convert string src to number
      la $a0, str
      jal strtobin
      move $s0, $v0

      #Prompt for string dest
      la $a0, str
      li $a1, 32
      li $v0, 8
      syscall

      #Convert string dest to number
      la $a0, str
      jal strtobin
      move $s1, $v0

      #Prompt for start
      li $v0, 5
      syscall
      move $s2, $v0

      #Prompt for end
      li $v0, 5
      syscall
      move $s3, $v0

      #Make call to setbits, convert to string and print string
      move $a0, $s0
      move $a1, $s2
      move $a2, $s3
      jal setbits
```

```
move $a0, $v0
la $a1, str
jal bintostr
la $a0, str
li $v0, 4
syscall
la $a0, newline
li $v0, 4
syscall

#Make call to clearbits, convert to string and print string
move $a0, $s0
move $a1, $s2
move $a2, $s3
jal clearbits
move $a0, $v0
la $a1, str
jal bintostr
la $a0, str
li $v0, 4
syscall
la $a0, newline
li $v0, 4
syscall

#Make call to copybits, convert to string and print string
move $a0, $s0
move $a1, $s2
move $a2, $s3
jal copybits
move $a0, $v0
la $a1, str
jal bintostr
la $a0, str
li $v0, 4
syscall
la $a0, newline
li $v0, 4
syscall

#Make call to insertbits, convert to string and print string
move $a0, $s0
move $a1, $s1
move $a2, $s2
move $a3, $s3
jal insertbits
move $a0, $v0
la $a1, str
jal bintostr
la $a0, str
li $v0, 4
syscall
la $a0, newline
li $v0, 4
syscall

#End
li $v0, 10
syscall
```

```
#Implement your functions here:

#Problem 1:
setbits:
      li $t0, 1               #t1 is always 1
      addi $a2, $a2, 1        #One pass the end position
      move $v0, $a0
sb_st1:
      beq $a1, $a2, sb_end1
      sllv $t1, $t0, $a1
      or $v0, $v0, $t1
      addi $a1, $a1, 1
      j sb_st1
sb_end1:
      jr $ra

#Problem 2:
clearbits:
      li $t0, 1               #t1 is always 1
      addi $a2, $a2, 1        #One pass the end position
      move $v0, $a0
cb_st1:
beq $a1, $a2, cb_end1
      sllv $t1, $t0, $a1
      not $t1, $t1
      and $v0, $v0, $t1
      addi $a1, $a1, 1
      j cb_st1
cb_end1:
      jr $ra

#Problem 3:
copybits:
      addi $sp, $sp, -4       #Adjust stack pointer
      sw $ra, 0($sp)          #Save return address

      srlv $a0, $a0, $a1      #Shift so that range is in the rightmost position
      sub $a1, $a2, $a1       #Clear bits starting one past the new end of the
                              #range
      addi $a1, $a1, 1
      li $a2, 31              #and ending at bit 31
      jal clearbits           #Return value is whatever clearbits returns

      lw $ra, 0($sp)          #Restore return address
      addi $sp, $sp, 4        #Adjust stack pointer
      jr $ra

#Problem 4:
insertbits:
      addi $sp, $sp, -20      #Adjust stack pointer
      sw $s1, 0($sp)          #Save registers
      sw $s2, 4($sp)          #Save registers
      sw $s3, 8($sp)          #Save registers
      sw $s4, 12($sp)         #Save registers
      sw $ra, 16($sp)         #Save return address

      move $s1, $a1           #s1 holds dest
      move $s2, $a2           #s2 holds start
```

```
      move $s3, $a3           #s3 holds end

      sllv $a0, $a0, $s2      #Shift src so that range is in the correct position
      addi $a1, $s3, 1        #Clear bits starting one past the current end of the
                              #range
      li $a2, 31              #and ending at bit 31
      jal clearbits
      move $s4, $v0           #Save return value

      move $a0, $s1           #Clear range of dest
      move $a1, $s2
      move $a2, $s3
      jal clearbits

      or $v0, $v0, $s4  #or both results together

      lw $s1, 0($sp)          #Restore registers
      lw $s2, 4($sp)          #Restore registers
      lw $s3, 8($sp)          #Restore registers
      lw $s4, 12($sp)         #Restore registers
      lw $ra, 16($sp)         #Restore return address
      addi $sp, $sp, 20       #Adjust stack pointer
      jr $ra


strtobin:
      addi $sp, $sp, -8       #Adjust stack pointer
      sw $s0, 0($sp)          #Save registers
      sw $ra, 4($sp)          #Save return address

      move $s0, $a0           #s0 holds the address of the string
      jal strlen              #Get length of string
      li $t0, 0               #Index into buffer
      addi $t1, $v0, -1       #Index into string
      la $t2, buf             #Address of buffer
      li $t4, 48              #If string empty, sign extend with zero
_start2:
      beq $t0, 32, _end2      #Loop 32 times
      bge $t0, $v0, _endif2   #If index is less than lenght of string, copy
                              #character to buffer, else copy the last character
                              #copied
      add $t3, $s0, $t1
      lb $t4, 0($t3)
      j _endif2
_endif2:
      add $t3, $t2, $t0
      sb $t4, 0($t3)
      addi $t0, $t0, 1        #Increment index into buffer
      addi $t1, $t1, -1       #Decrement index into string
      j _start2
_end2:      li $t0, 0         #Index into buffer
      li $v0, 0   #Sum
      li $t6, 1   #Always 1
_start3:
      beq $t0, 32, _end3      #Loop 32 times
      add $t3, $t2, $t0       #Load character
      lb $t4, 0($t3)
      beq $t4, 48, _endif3    #If loaded character is 48, skip
      sllv $t7, $t6, $t0      #Shift 1 index number of times
```

```
      add $v0, $v0, $t7       #Add to return value
_endif3:
      addi $t0, $t0, 1        #Increment index
      j _start3
_end3:

      lw $s0, 0($sp)          #Restore registers
      lw $ra, 4($sp)          #Restore return address
      addi $sp, $sp, 8        #Adjust stack pointer
      jr $ra                  #Jump to caller

bintostr:
      li $t0, 0               #Bit number
      li $t3, 31              #Length of the string - 1
_start4:
      beq $t0, 32, _end4      #Loop 32 times
      srlv $t1, $a0, $t0      #Shift right index number of times
      andi $t1, $t1, 1        #Get rightmost bit
      beq $t1, $zero, _else4  #If righmost bit is zero, jump to else
      li $t2, 49              #Store "1"
      j _endif4
_else4:
      li $t2, 48              #Store "0"
_endif4:
      sub $t4, $t3, $t0       #Get index into string
      add $t4, $t4, $a1       #Compute address to store
      sb $t2, 0($t4)          #Store "0" or "1"
      addi $t0, $t0, 1        #Increment index
      j _start4
_end4:
      sb $zero, 32($a1)       #Put null at end of string
      jr $ra                  #Jump to caller

#This function returns the length of a string that ends in either a null or a
newline
strlen:
      move $t0, $a0
_start1:
lb $t1, 0($t0)
      beq $t1, $zero, _end1
      beq $t1, 0x0A, _end1
      addi $t0, $t0, 1
      j _start1
_end1:
      sub $v0, $t0, $a0
      jr $ra
```