Sai Kathika saiprem@ucsb.edu
Pete Makrygiannis pmakrygiannis@ucsb.edu
Rahul Varghese rvarghese@ucsb.edu

ECE 154A Lab 2

```
################################################################################
# File: div.s
# Skeleton for ECE 154a project
################################################################################

        .data
student:
        .asciiz "Sai Kathika, Pete Makrygiannis, Rahul Varghese"
        .globl  student
nl:     .asciiz "\n"
        .globl nl


op1:    .word 20                        # divisor for testing
op2:    .word 700           # dividend for testing


        .text

        .globl main
main:                                           # main has to be a global label
        addi    $sp, $sp, -4                    # Move the stack pointer
        sw      $ra, 0($sp)                     # save the return address

        move    $t0, $a0            # Store argc
        move    $t1, $a1            # Store argv

        li      $v0, 4                          # print_str (system call 4)
        la      $a0, student                    # takes the address of string as an argument
        syscall

        slti    $t2, $t0, 2         # check number of arguments
        bne   $t2, $zero, operands
        j       ready

operands:
        la      $t0, op1
        lw      $a0, 0($t0)
        la      $t0, op2
        lw      $a1, 0($t0)


ready:
        jal     divide                          # go to multiply code

        jal     print_result                    # print operands to the console

                                                # Usual stuff at the end of the main
        lw      $ra, 0($sp)                     # restore the return address
        addi    $sp, $sp, 4
        jr      $ra                             # return to the main program


divide:
################################################################################
# Your code goes here.
# Should have the same functionality as running
#       divu    $a1, $a0
#       mflo    $a2     quotient
#       mfhi    $a3     remainder
# divisor $a0 (smaller number) B
```

```
# dividend $a1 (larger number) A
# $a1/$a0 = $a2 + reminder $a3
################################################################################
# divu      $a1, $a0
# mflo      $a2
# mfhi      $a3

   move $a2, $zero

   move $a3, $zero #remainder

   move $t1, $zero

   move $t6, $a0
   move $t7, $a1

   loop:
            slt $t2, $a0, $a1
            beq $t2, $zero, loop1
            sll $a0, $a0, 1
            addi $t1, $t1, 1
            beq $t2, 1, loop

   loop1:
            slt $t4, $zero, $t1
            beq $t4 , $zero, done
            sll $a2, $a2, 1
            srl $a0, $a0, 1

            slt $t5, $a0, $a1
            addi $t1,$t1,-1
            beq $t5, 1, if
            beq $a0,$a1,if
            j loop1



   if:
            addi $a2,$a2,1
            beq $t5, $zero, done #### here if it enters since a == b
            sub $a1,$a1,$a0
            j loop1
   done:
            rem   $a3, $t7, $t6
            move $a0, $t6
            move $a1, $t7



################################################################################
# Do not edit below this line
################################################################################
            jr          $ra
```

```
###############################################################################
# File: mult.s
# Skeleton for ECE 154a project
###############################################################################

        .data
student:
        .asciiz "Sai Kathika, Pete Makrygiannis, Rahul Varghese"
        .globl  student
nl:     .asciiz "\n"
        .globl nl


op1:    .word 5                         # change the multiplication operands
op2:    .word 2                 # for testing.


        .text

        .globl main
main:                                   # main has to be a global label
        addi    $sp, $sp, -4            # Move the stack pointer
        sw      $ra, 0($sp)             # save the return address

        move    $t0, $a0        # Store argc
        move    $t1, $a1        # Store argv

        li      $v0, 4                  # print_str (system call 4)
        la      $a0, student            # takes the address of string as an argument
        syscall

        slti    $t2, $t0, 2     # check number of arguments
        bne   $t2, $zero, operands
        j       ready

operands:
        la      $t0, op1
        lw      $a0, 0($t0)
        la      $t1, op2
        lw      $a1, 0($t1)


ready:
        jal     multiply        # go to multiply code

        jal     print_result            # print operands to the console

                                        # Usual stuff at the end of the main
        lw      $ra, 0($sp)             # restore the return address
        addi    $sp, $sp, 4
        jr      $ra                     # return to the main program


multiply:
###############################################################################
# Your code goes here.
# Should have the same functionality as running
#       multu   $a1, $a0
#       mflo    $a2
###############################################################################
```

```
        move $t0,$zero
        move $t7, $a0
        move $t6, $a1
        multiplication_loop:
           andi $t2,$a1,1
           beq $t0,$zero,loop
           addu $t0,$t0,$a0
        loop:
           sll $a0,$a0,1
           srl $a1,$a1,1
           bne $a1,$zero,multiplication_loop

        srl $a0,$a0,1
        move $a2,$a0
        move $a0, $t7
        move $a1, $t6

################################################################################
# Do not edit below this line
################################################################################
        jr      $ra



print_result:
        move    $t0, $a0
        li      $v0, 4
        la      $a0, nl
        syscall

        move    $a0, $t0
        li      $v0, 1
        syscall
        li      $v0, 4
        la      $a0, nl
        syscall

        li      $v0, 1
        move    $a0, $a1
        syscall
        li      $v0, 4
        la      $a0, nl
        syscall

        li      $v0, 1
        move    $a0, $a2
        syscall
        li      $v0, 4
        la      $a0, nl
        syscall

        jr $ra
```