# Van Slyke, Richard ID no. goes here
# SPIM Demonstration program
# Lecture Section goes here
# 02/01/2006
# This program takes as input sixteen digits in a
# 4 x 4 grid. Each digit must be a 0, 1, 2, or 3,
# in any order. The program checks to see if
# each digit appears exactly once in each row.
# If there are duplicates the first column with
# a duplicate is printed out for each row.
# For your first SPIM assignment, you are to modify
# this program to check to see it there are duplicates
# in the columns.
# If there are neither row duplicates or column duplicates
# you have entered a latin square!!

# Registers:
# $t0 Row index
# $t1 Column index
# $t2 Element value
# $t3 Scratch
# $t4 Base address for grid array
# $t5 Base address for check vector
# $t6 Scratch

.text          # Assembly directive indicating instructions
.globl main     # "main" is defined as a global

main:
    # Load base address for grid
    la $t4, $grid
    # Load base address for check vector
    la $t5, $check
    # ENTER GRID VALUES
    # Initialize
    add $t0, $0, $0 # Initialize row index to zero
    # LOOP FOR ROWS
    OuterLoop:
        la $a0, $msgr  # $a0 points to start of row message
        li $v0, 4     # Set the print string system call
        syscall
        add $a0, $t0, $0  # Print out row number
        li $v0, 1
        syscall
        add $t1, $0, $0 # Initialize column index to zero
        # LOOP FOR COLUMNS
        InnerLoop:
        # ASK FOR ELEMENT
        la $a0, $msgel # $a0 points to start of element request
        li $v0, 4     # Set the print string system call
        syscall
        add $a0, $t0, $0  # Prepare to print row index
        li $v0, 1
        syscall
        la $a0, $msge2 # $a0 points to middle of element request
li $v0, 4    # Set the print string system call
syscall
add $a0, $t1, $0    # Prepare to print column index
li $v0, 1    # Set print integer system call
syscall
la $a0, $msge3 # $a0 points to end of element request
li $v0, 4    # Set the print string system call
syscall
retry:    li $v0, 5    # Read element value
syscall
# VALIDATE ENTRY
add  $t2, $v0, $0
slt $t3, $t2, $0
bne $t3, $0 bad
slti $t3, $t2, 4
beq $t3, $0, bad
j good
# Entry out of range
bad: add $a0, $t2, $0
li $v0, 1
syscall
la $a0, $msgb
li $v0, 4
syscall
j retry
# SAVE ELEMENT IN GRID ARRAY
good: sll  $t3, $t0, 2
add  $t3, $t3, $t1
add  $t3, $t3, $t4
sb  $t2, 0($t3)
addi $t1, $t1, 1    # Increment column index
slt $t3, $t1, 4    # Check for end of row
bne $t3, $t0, InnerLoop
addi $t1, $t1, 1    # Increment row index
slt $t3, $t0, 4    # Check for end
bne $t3, $t0, OuterLoop
la $a0, $msgx # $a0 points to the signoff message
li $v0, 4    # Set the print string system call
syscall
# CHECK FOR DUPLICATES
COuter:
# Check for duplicates in row
# Zero out check vector
add $t1, $0, $0 # Initialize column index to zero
zcheck:    sll $t3, $t1, 2 # We are using words now!
add $t3, $t3, $t1
sw $0, 0($t3)
addi $t1, $t1, 1
slt $t3, $t1, 4
bne $t3, $t0, zcheck
# Check for duplicate numbers in rows
add $t1, $0, $0 # Initialize column index to zero
dcheck:    sll $t3, $t0, 2
add $t3, $t3, $t1
add $t3, $t3, $t4    # Effective address of entry
lb  $t2, 0($t3)  # Retrieve entry
sli $t6, $t2, 2  # Multiply by 4 for check words
add $t6, $t6, $t5  # Effective memory address for check
lw $t3, 0($t6)
beq $t3, $0, ok  # If 0, we haven't seen entry before

# Found duplicate; print message
la $a0, $msgd1  # $a0 points to start of element request
li $v0, 4  # Set the print string system call
 syscall
add $a0, $t0, $0  # Prepare to print row index
li $v0, 1  # Set print integer system call
 syscall
la $a0, $msgd2  # $a0 points to middle of element request
li $v0, 4  # Set the print string system call
 syscall
add $a0, $t1, $0  # Prepare to print column index
li $v0, 1  # Set print integer system call
 syscall
la $a0, $msgd3  # $a0 points to end of element request
li $v0, 4  # Set the print string system call
 syscall
add $a0, $t2, $0  # Prepare to print element
li $v0, 1  # Set print integer system call
 syscall
ok:  addi $t3, $t3, 1  # Increment count in check vector
sw $t3, 0($t6)
addi $t1, $t1, 1  # Increment column
slt $t3, $t1, 4
bne $t3, $0, dcheck
add $t0, $t0, 1  # Increase row
slt $t3, $t0, 4
bne $t3, $0, COuter

# EXIT
la $a0, $msgz  # $a0 points to the signoff message
li $v0, 4  # Set the print string system call
 syscall
li $v0, 10  # Exit
syscall
.data  # Assembly directive indicating data follows
$msgb: .asciiz " Entry outside of range, try again or ^C to bail out \n"
$msgd1: .asciiz "Duplicate in row "
$msgd2: .asciiz " column "
$msgd3: .asciiz " of element "
$msgel1: .asciiz " Enter a 0, 1, 2 or 3 for element in row "
$msgel2: .asciiz " Column "
$msgel3: .asciiz " Then hit ENTER\n"
$msgx: .asciiz "Successful Input!\n"
$msgz: .asciiz " Completed Scan! \n"
.align 2  # Assembly directive to word align (p. A-47)
$grid: .space 16  # 16 bytes for grid (1 byte per entry)
$check: .space 16  # 4 words for check vector.
# For speed we use one word per entry