##############################################################

# Homework #3

# name: Niral Patel

# sbuid: 110626877

##############################################################

.text

##############################

# PART 1 FUNCTIONS

##############################

smiley:

li $t0, 0xffff0000 #base address of 0,0

li $t1, '\0' #ascii for null

li $t2, 15 # foreground color

li $t3, 0 # background color

li $t4,0 #temp counter

sll $t3,$t3,4

add $t3,$t3,$t2 #t3 has background and foreground color in 8 bits

loop\_again:

sb $t1,($t0)

addi $t0,$t0,1

sb $t3,($t0)

addi $t0,$t0,1

addi $t4,$t4,1

blt $t4,100,loop\_again #stops after making all 100 cells black

eyes:

#(2,3),

li $t1, 'b' #ascii for null

li $t2, 7 # foreground color

li $t3, 11 # background color

sll $t3,$t3,4

add $t3,$t3,$t2 #t3 has background and foreground color in 8 bits

li $t0, 0xffff002E #base address of 0,0

sb $t1, ($t0)

addi $t0,$t0,1

sb $t3,($t0)

#(3,3),

li $t1, 'b' #ascii for null

li $t2, 7 # foreground color

li $t3, 11 # background color

sll $t3,$t3,4

add $t3,$t3,$t2 #t3 has background and foreground color in 8 bits

li $t0, 0xffff0042 #base address of 0,0

sb $t1, ($t0)

addi $t0,$t0,1

sb $t3,($t0)

#(2,6)

li $t1, 'b' #ascii for null

li $t2, 7 # foreground color

li $t3, 11 # background color

sll $t3,$t3,4

add $t3,$t3,$t2 #t3 has background and foreground color in 8 bits

li $t0, 0xffff0034 #base address of 0,0

sb $t1, ($t0)

addi $t0,$t0,1

sb $t3,($t0)

#,(3,6).

li $t1, 'b' #ascii for null

li $t2, 7 # foreground color

li $t3, 11 # background color

sll $t3,$t3,4

add $t3,$t3,$t2 #t3 has background and foreground color in 8 bits

li $t0, 0xffff002E #base address of 0,0

sb $t1, ($t0)

addi $t0,$t0,1

sb $t3,($t0)

li $t1, 'b' #ascii for null

li $t2, 7 # foreground color

li $t3, 11 # background color

sll $t3,$t3,4

add $t3,$t3,$t2 #t3 has background and foreground color in 8 bits

li $t0, 0xffff002E #base address of 0,0

sb $t1, ($t0)

addi $t0,$t0,1

sb $t3,($t0)

li $t1, 'b' #ascii for null

li $t2, 7 # foreground color

li $t3, 11 # background color

sll $t3,$t3,4

add $t3,$t3,$t2 #t3 has background and foreground color in 8 bits

li $t0, 0xffff0048 #base address of 0,0

sb $t1, ($t0)

addi $t0,$t0,1

sb $t3,($t0)

smile:

li $t1, 'e' #ascii for null

li $t2, 7 # foreground color

li $t3, 1 # background color

sll $t3,$t3,4

add $t3,$t3,$t2 #t3 has background and foreground color in 8 bits

li $t0, 0xffff007c #base address of 0,0

sb $t1, ($t0)

addi $t0,$t0,1

sb $t3,($t0)

li $t1, 'e' #ascii for null

li $t2, 7 # foreground color

li $t3, 1 # background color

sll $t3,$t3,4

add $t3,$t3,$t2 #t3 has background and foreground color in 8 bits

li $t0, 0xffff0092 #base address of 0,0

sb $t1, ($t0)

addi $t0,$t0,1

sb $t3,($t0)

li $t1, 'e' #ascii for null

li $t2, 7 # foreground color

li $t3, 1 # background color

sll $t3,$t3,4

add $t3,$t3,$t2 #t3 has background and foreground color in 8 bits

li $t0, 0xffff00A8 #base address of 0,0

sb $t1, ($t0)

addi $t0,$t0,1

sb $t3,($t0)

li $t1, 'e' #ascii for null

li $t2, 7 # foreground color

li $t3, 1 # background color

sll $t3,$t3,4

add $t3,$t3,$t2 #t3 has background and foreground color in 8 bits

li $t0, 0xffff00AA #base address of 0,0

sb $t1, ($t0)

addi $t0,$t0,1

sb $t3,($t0)

li $t1, 'e' #ascii for null

li $t2, 7 # foreground color

li $t3, 1 # background color

sll $t3,$t3,4

add $t3,$t3,$t2 #t3 has background and foreground color in 8 bits

li $t0, 0xffff0098 #base address of 0,0

sb $t1, ($t0)

addi $t0,$t0,1

sb $t3,($t0)

li $t1, 'e' #ascii for null

li $t2, 7 # foreground color

li $t3, 1 # background color

sll $t3,$t3,4

add $t3,$t3,$t2 #t3 has background and foreground color in 8 bits

li $t0, 0xffff0086 #base address of 0,0

sb $t1, ($t0)

addi $t0,$t0,1

sb $t3,($t0)

jr $ra

##############################

# PART 2 FUNCTIONS

##############################

open\_file:

li $a1, 0 # Open for reading

li $a2, 0

li $v0, 13

syscall # open a file (file descriptor returned in $v0)

jr $ra

close\_file:

li $v0,16

syscall

jr $ra

load\_map:

move $t8, $a1 #base address of cell array

# li $v0, 14 #syscall to read file

# la $a1,buffer #buffer to read

# li $a2, 1 #max amount to read

# syscall

li $t9,10 #number of columns/row

li $t3, 0 # used to count down rows

li $t4, 0 #used to count down columns

li $t5, 0 #address of cell

li $t6, 0 #value to store in cell

li $t1, 0 # to count coordiantes

############## loop to set all cells contents to 0 ##################

loop\_for\_all\_cells:

bge $t3,$t9,beg\_loop

mul $t5, $t3, $t9 # first number \* total columns

add $t5, $t5, $t4 # first number \* total columns + column

add $t5, $t5, $t8 # base\_addr + (i \* num\_columns + j)

sb $t6, ($t5) #t4 has final cell address to place bomb attributes

addi $t4,$t4,1

blt $t4,$t9,loop\_for\_all\_cells

addi $t3,$t3,1

li $t4,0

j loop\_for\_all\_cells

beg\_loop:

li $v0, 14 #syscall to read file

la $a1,buffer #buffer to read

li $a2, 1 #max amount to read

syscall

beqz $v0,check

loop:

#loop to read each coordinate

li $t0,0 #initialize to 0

li $t3,0 #initialize to 0

#lb $t0,($a1) #t0 has individual number of coordinate

#check final conditions when EOF reached

x\_co:

lb $t0,($a1) #t0 has individual number of coordinate

#beqz $t0,check

beq $t0,' ',move\_forward #move forward if space

beq $t0,'\n',move\_forward #move forward if new line

beq $t0,'\t',move\_forward #move forward if tab

beq $t0,'\r',move\_forward #move forward if r

blt $t0,48,invalid\_file #if number is less than 0, file invalid

bgt $t0,57,invalid\_file #if number is greater than 9, file invalid

j check\_single\_digit #check if number is not a single digit

check\_single\_digit: #if next byte is not any sort of space, file invalid

li $v0, 14 #syscall to read file

la $a1,buffer #buffer to read

li $a2, 1 #max amount to read

syscall

beqz $v0,check

lb $t2,($a1)

beq $t2,' ',valid\_sofar #move forward if space

beq $t2,'\n',valid\_sofar #move forward if new line

beq $t2,'\t',valid\_sofar #move forward if tab

beq $t2,'\r',valid\_sofar

beq $t0,'0',x\_co

j invalid\_file

valid\_sofar:

addi $t1,$t1,1 #add to count of coordinate units

li $v0, 14 #syscall to read file

la $a1,buffer #buffer to read

li $a2, 1 #max amount to read

syscall

beqz $v0,check

check\_again: # see if t3 is second part of coordinate

lb $t3, ($a1)

#beqz $t3,check

beq $t3,' ',move\_forward\_get\_co #move forward if space to get co

beq $t3,'\n',move\_forward\_get\_co #move forward if new line to get co

beq $t3,'\t',move\_forward\_get\_co #move forward if tab to get co

beq $t3,'\r',move\_forward\_get\_co #move forward if r to get co

blt $t3,48,invalid\_file #if number is less than 0, file invalid

bgt $t3,57,invalid\_file #if number is greater than 9, file invalid

j check\_single\_digit1

check\_single\_digit1: #if next byte is not any sort of space, file invalid

addi $t1,$t1,1

li $v0, 14 #syscall to read file

la $a1,buffer #buffer to read

li $a2, 1 #max amount to read

syscall

#beqz $v0,check

lb $t2,($a1)

beq $t2,' ',map\_out #move forward if space

beq $t2,'\n',map\_out #move forward if new line

beq $t2,'\t',map\_out #move forward if tab

beq $t2,'\r',map\_out

beq $t2,'\0',map\_out

beqz $v0, map\_out

addi $t1,$t1,-1

beq $t3,'0',check\_again

j invalid\_file

map\_out:

li $t2,1 # used to increase adjacent cells by this amount for having bomb next to it

li $t4,0 #will have final address

li $t5,32 ################# this should be 32, made more to set cells as revealed ################

li $t6,'0' #to convert

sub $t0,$t0,$t6 #has integer value of X coordiate

sub $t3,$t3,$t6 #has integer value of Y coordinate

############################ change bomb locations to 32 ##############################

mul $t4, $t0, $t9 # first number \* total columns

add $t4, $t4, $t3 # first number \* total columns + column

add $t4, $t4, $t8 # base\_addr + 2\*(i \* num\_columns + j)

lb $t5, ($t4)

beq $t5, 32, bomb\_exists

li $t5, 32

sb $t5, ($t4) #t4 has final cell address to place bomb attributes

### ## ## ## ADD 1 TO ALL CELLS AROUND BOMB CELL ## ## ###

# x and y coordainte cant be less than 0 but

# t0 has x coordiante, t3 has y coordiante

li $t6,0xffff0000 #to check a coorindate is in range above 0

li $t4, 0

mul $t4, $t0, $t9 # first number \* total columns

add $t4, $t4, $t3 # first number \* total columns + column

sll $t4,$t4,1

add $t4, $t4, $t6 # base\_addr + 2\*(i \* num\_columns + j)

li $t5, '\0'

sb $t5, ($t4) #t4 has final cell address to place bomb attributes

li $t7,9 #to check a coordinate is in range below 10

li $t4,0 #will have final address again

# when x = x-1

x\_1:

addi $t0,$t0,-1

addi $t3,$t3,1 #y+1

blt $t0,0,x0 #if x-1 is below 0 go to x

addi $t3,$t3,-1

# here x = x - 1 , y = y

mul $t4, $t0, $t9 # first number \* total columns

add $t4, $t4, $t3 # first number \* total columns + column

add $t4, $t4, $t8 # base\_addr + (i \* num\_columns + j)

lb $t5, ($t4) # get the value at add x-1,y

li $t2,32

sb $t2, ($t4)

bgt $t5,8,x\_1y\_1

addi $t5,$t5,1 # add 1 to the value

sb $t5, ($t4) # add the value back to that cell

addi $t5,$t5,'0'

li $t4,0

mul $t4, $t0, $t9 # first number \* total columns

add $t4, $t4, $t3 # first number \* total columns + column

sll $t4,$t4,1

add $t4, $t4, $t6 # base\_addr + (i \* num\_columns + j)

sb $t5, ($t4) # add the value back to that cell

addi $t4,$t4,1 # go to color byte

li $t5,0

addi $t5,$t5,13 # add 12 for fore ground color

sb $t5, ($t4) # store the color back

li $t4,0 #will have final address again

x\_1y\_1:

#here x = x - 1 , y = y-1

addi $t3,$t3,-1

blt $t3,0,x\_1y1 #if y-1 is below 0 go to y+1

mul $t4, $t0, $t9 # first number \* total columns

add $t4, $t4, $t3 # first number \* total columns + column

add $t4, $t4, $t8 # base\_addr + (i \* num\_columns + j)

lb $t5, ($t4) # get the value at add x-1,y

li $t2,32

sb $t2, ($t4)

bgt $t5,8,x\_1y1

addi $t5,$t5,1 #add 1 to the value

sb $t5, ($t4) #add the value back to that cell

addi $t5,$t5,'0'

li $t4,0

mul $t4, $t0, $t9 # first number \* total columns

add $t4, $t4, $t3 # first number \* total columns + column

sll $t4,$t4,1

add $t4, $t4, $t6 # base\_addr + (i \* num\_columns + j)

sb $t5, ($t4) #add the value back to that cell

addi $t4,$t4,1 # go to color byte

li $t5,0 # get color byte

addi $t5,$t5,13 # add 12 for fore ground color

sb $t5, ($t4) # store the color back

#here x = x - 1, y = (y-1)+2 = y+1

x\_1y1:

addi $t3,$t3,2

bgt $t3,9,x0 #if y+1 is greater than 9 go to x = x

mul $t4, $t0, $t9 # first number \* total columns

add $t4, $t4, $t3 # first number \* total columns + column

add $t4, $t4, $t8 # base\_addr + (i \* num\_columns + j)

lb $t5, ($t4) # get the value at add x-1,y

li $t2,32

sb $t2, ($t4)

bgt $t5,8,x0

addi $t5,$t5,1 #add 1 to the value

sb $t5, ($t4) #add the value back to that cell

addi $t5,$t5,'0'

mul $t4, $t0, $t9 # first number \* total columns

add $t4, $t4, $t3 # first number \* total columns + column

sll $t4,$t4,1

add $t4, $t4, $t6 # base\_addr + (i \* num\_columns + j)

sb $t5, ($t4) #add the value back to that cell

addi $t4,$t4,1 # go to color byte

li $t5,0 # get color byte

addi $t5,$t5,13 # add 12 for fore ground color

sb $t5, ($t4) # store the color back

li $t4,0 #will have final address again

# when x = x

x0:

addi $t0,$t0,1

#here x = x, y = y-1

addi $t3,$t3,-2

blt $t3,0,x0y1 #if y - 1 is below 0 go to y = y + 1

mul $t4, $t0, $t9 # first number \* total columns

add $t4, $t4, $t3 # first number \* total columns + column

add $t4, $t4, $t8 # base\_addr + (i \* num\_columns + j)

lb $t5, ($t4) # get the value at add x-1,y

li $t2,32

sb $t2, ($t4)

bgt $t5,8,x0y1

addi $t5,$t5,1 #add 1 to the value

sb $t5, ($t4) #add the value back to that cell

addi $t5,$t5,'0'

mul $t4, $t0, $t9 # first number \* total columns

add $t4, $t4, $t3 # first number \* total columns + column

sll $t4,$t4,1

add $t4, $t4, $t6 # base\_addr + (i \* num\_columns + j)

sb $t5, ($t4) #add the value back to that cell

addi $t4,$t4,1 # go to color byte

li $t5,0 # get color byte

addi $t5,$t5,13 # add 12 for fore ground color

sb $t5, ($t4) # store the color back

#here x = x y = (y-1)+2 = y+1

x0y1:

addi $t3,$t3,2

bgt $t3,9,x1 # if y + 1 is greater than 9 go to x = x + 1

mul $t4, $t0, $t9 # first number \* total columns

add $t4, $t4, $t3 # first number \* total columns + column

add $t4, $t4, $t8 # base\_addr + (i \* num\_columns + j)

lb $t5, ($t4) # get the value at add x-1,y

li $t2,32

sb $t2, ($t4)

bgt $t5,8,x1

addi $t5,$t5,1 #add 1 to the value

sb $t5, ($t4) #add the value back to that cell

addi $t5,$t5,'0'

mul $t4, $t0, $t9 # first number \* total columns

add $t4, $t4, $t3 # first number \* total columns + column

sll $t4,$t4,1

add $t4, $t4, $t6 # base\_addr + (i \* num\_columns + j)

sb $t5, ($t4) #add the value back to that cell

addi $t4,$t4,1 # go to color byte

li $t5,0 # get color byte

addi $t5,$t5,13 # add 12 for fore ground color

sb $t5, ($t4) # store the color back

li $t4,0 #will have final address again

# when x = x + 1

x1:

addi $t0,$t0,1

bgt $t0,9,done\_adding #if x = X + 1 is greater than 9 go to end of loop

# here x = x + 1, y = y

addi $t3,$t3,-1

mul $t4, $t0, $t9 # first number \* total columns

add $t4, $t4, $t3 # first number \* total columns + column

add $t4, $t4, $t8 # base\_addr + (i \* num\_columns + j)

lb $t5, ($t4) # get the value at add x-1,y

li $t2,32

sb $t2, ($t4)

bgt $t5,8,x1y\_1

addi $t5,$t5,1 #add 1 to the value

sb $t5, ($t4) #add the value back to that cell

addi $t5,$t5,'0'

mul $t4, $t0, $t9 # first number \* total columns

add $t4, $t4, $t3 # first number \* total columns + column

sll $t4,$t4,1

add $t4, $t4, $t6 # base\_addr + (i \* num\_columns + j)

sb $t5, ($t4) #add the value back to that cell

addi $t4,$t4,1 # go to color byte

li $t5,0 # get color byte

addi $t5,$t5,13 # add 12 for fore ground color

sb $t5, ($t4) # store the color back

li $t4,0 #will have final address again

x1y\_1:

#here x = x + 1, y = y-1

addi $t3,$t3,-1

blt $t3,0,x1y1 #if y = y - 1 is less than 0 go to y = y + 1

mul $t4, $t0, $t9 # first number \* total columns

add $t4, $t4, $t3 # first number \* total columns + column

add $t4, $t4, $t8 # base\_addr + (i \* num\_columns + j)

lb $t5, ($t4) # get the value at add x-1,y

li $t2,32

sb $t2, ($t4)

bgt $t5,8,x1y1

addi $t5,$t5,1 #add 1 to the value

sb $t5, ($t4) #add the value back to that cell

addi $t5,$t5,'0'

mul $t4, $t0, $t9 # first number \* total columns

add $t4, $t4, $t3 # first number \* total columns + column

sll $t4,$t4,1

add $t4, $t4, $t6 # base\_addr + (i \* num\_columns + j)

sb $t5, ($t4) #add the value back to that cell

addi $t4,$t4,1 # go to color byte

li $t5,0 # get color byte

addi $t5,$t5,13 # add 12 for fore ground color

sb $t5, ($t4) # store the color back

#here x = x + 1, y = (y-1)+2 = y+1

x1y1:

addi $t3,$t3,2

bgt $t3,9,done\_adding #if y = y + 1 is greater than 9 go to end of loop

mul $t4, $t0, $t9 # first number \* total columns

add $t4, $t4, $t3 # first number \* total columns + column

add $t4, $t4, $t8 # base\_addr + (i \* num\_columns + j)

lb $t5, ($t4) # get the value at add x-1,y

li $t2,32

sb $t2, ($t4)

bgt $t5,8,done\_adding

addi $t5,$t5,1 #add 1 to the value

sb $t5, ($t4) #add the value back to that cell

addi $t5,$t5,'0'

mul $t4, $t0, $t9 # first number \* total columns

add $t4, $t4, $t3 # first number \* total columns + column

sll $t4,$t4,1

add $t4, $t4, $t6 # base\_addr + (i \* num\_columns + j)

sb $t5, ($t4) #add the value back to that cell

addi $t4,$t4,1 # go to color byte

li $t5,0 # get color byte

addi $t5,$t5,13 # add 12 for fore ground color

sb $t5, ($t4) # store the color back

done\_adding:

#addi $t1,$t1,-1

j move\_forward

move\_forward:

li $v0, 14 #syscall to read file

la $a1,buffer #buffer to read

li $a2, 1 #max amount to read

syscall

beqz $v0,check

j loop

move\_forward\_get\_co:

li $v0, 14 #syscall to read file

la $a1,buffer #buffer to read

li $a2, 1 #max amount to read

syscall

beqz $v0,check

j check\_again

bomb\_exists:

addi $t1,$t1,-1

j done\_adding

check:

blt $t1,2,invalid\_file #if file has less than 1 coordiante

bgt $t1,198,invalid\_file #if file has greater than 99 coordiates

li $t2,2 #used for division further down

div $t1,$t2

mfhi $t2 #t1 has the remainder which should be 0

bnez $t2,invalid\_file #invalid file is not even number of coordinate points

li $v0,1

li $t0, 0

sw $t0, cursor\_row

sw $t0, cursor\_col

jr $ra

invalid\_file:

li $v0,-1

jr $ra

##############################

# PART 3 FUNCTIONS

##############################

init\_display:

li $t0, 0xffff0000 #base address of 0,0

li $t1, '\0' #ascii for null

li $t2, 7 # foreground color

li $t3, 7 # background color

li $t4,0 #temp counter

li $t5,0

sll $t3,$t3,4

add $t3,$t3,$t2 #t3 has background and foreground color in 8 bits

loop\_again\_p3:

##########################################################################################changes

li $t6,'0'

li $t5,0

li $t1, '\0'

lb $t5, ($t0)

sub $t5,$t5,$t6

beqz $t5,add\_0

addi $t5,$t5,'0'

move $t1, $t5

add\_0:

sb $t1,($t0)

addi $t0,$t0,1

sb $t3,($t0)

addi $t0,$t0,1

addi $t4,$t4,1

blt $t4,100,loop\_again\_p3 #stops after making all 100 cells start state

#calculate address of cursor (originally 0,0)

cursor\_display:

li $t4, 0

lw $t5,cursor\_row #x coordinate of cursor

lw $t6,cursor\_col #y coordinate of cursor

li $t9, 10

mul $t4, $t5, $t9 # first number \* total columns

add $t4, $t4, $t6 # first number \* total columns + column

sll $t4,$t4,1

add $t4, $t4, $t0

lb $t9, ($t4) # t9 has the ASCII of cursor

li $t2, 11 # foreground color cover up by making the yellow initially

li $t3, 11 # background color

addi $sp,$sp,-16

sw $t3,0($sp)

sw $a0,4($sp)

sw $a1,8($sp)

sw $ra,12($sp)

move $a0, $t5

move $a1, $t6

move $a2, $t9

move $a3, $t2

jal set\_cell

lw $t8, 0($sp)

lw $a0, 4($sp)

lw $a1, 8($sp)

lw $ra, 12($sp)

addi $sp,$sp,16

jr $ra

set\_cell:

li $v0,0 # initial return value

move $t0,$a0 # row index

move $t1,$a1 # col index

move $t2,$a2 # ASCII

move $t3,$a3 # foreground color

lw $t4,0($sp) #background color

li $t5,0 #address of cell to modify

li $t8, 0xffff0000 #base address of 0,0

li $t9,10 #total columns and row for address calculation

blt $t0,0,invalid\_input # row cannot be less than 0

bge $t0,10,invalid\_input # row cannot be greater or equal to 10

blt $t1,0,invalid\_input # col cannot be less than 0

bge $t1,10,invalid\_input # col cannot be greater or equal to 10

blt $t3,0,invalid\_input # foreground cannot be less than 0

bge $t3,16,invalid\_input # foreground cannot be greater or equal to 16

blt $t4,0,invalid\_input # background cannot be less than 0

bge $t4,16,invalid\_input # background cannot be greater or equal to 16

mul $t5, $t0, $t9 # first number \* total columns

add $t5, $t5, $t1 # first number \* total columns + column

sll $t5, $t5, 1

add $t5, $t5, $t8 # base\_addr + (i \* num\_columns + j)

sll $t4,$t4,4 #move background over 4 bits

add $t4,$t4,$t3 #t4 has background and foreground color in 8 bits

sb $t2,($t5) #cell has ASCII

addi $t5,$t5,1 #cell moved over 1 byte

sb $t4,($t5) #cell has ASCII and COLOR

jr $ra

invalid\_input:

li $v0, -1 #invalid return val

jr $ra

reveal\_map:

move $t0, $a0 # t0 has the state of the game

#la $a1,

li $t6, 0 # to count down rows

li $t7, 0 # to count down cols

beq $t0,1,won

beq $t0,-1,lose

won:

addi $sp,$sp,-4

sw $ra,0($sp)

jal smiley

lw $ra,0($sp)

addi $sp,$sp,4

jr $ra

lose:

move $t1, $a1 # t1 has the cells array

lb $t2, ($t1) # t2 has the byte of the cells array

li $t4, 0 # t4 has black color for background, pushed on to stack

checker:

#check if cell is revealed

move $t1, $a1 # t1 has the cells array

lb $t2, ($t1) # t2 has the byte of the cells array

j revealed

increase\_coordinates:

addi $t7,$t7,1 # increase col

bge $t7,10,row\_increase # check if col still less than 10

addi $a1,$a1,1

j checker

row\_increase:

li $t7,0

addi $t6,$t6,1 # increase row by 1

bge $t6,10,reveal\_done

addi $a1,$a1,1 # go to next byte in cells array

j lose

revealed:

#check if flag

li $t3, 16 # to get the 32nd bit for bomb 0:no or 1:yes

and $t3,$t2,$t3 # get only the 32nd bit in t3, t2 still has entire byte value

beq $t3,16,flag # if the cell has bomb display bomb

#since not flag check if bomb

li $t3, 32 # to get the 32nd bit for bomb 0:no or 1:yes

and $t3,$t2,$t3 # get only the 32nd bit in t3, t2 still has entire byte value

beq $t3,32,bomb # if the cell has bomb display bomb

# not a bomb or a flag check number

li $t3,15 # to get first 4 bit value

and $t3,$t2,$t3

bgt $t3,0, number #reveal number

# not a bomb, not a flag, not a number

j empty\_cell

bomb:

li $t4,0 #temp set background to gray

addi $sp, $sp, -16 # move the back ground color on to stack

sw $t4, 0($sp) # background color on stack

sw $a0,4($sp) # state of game cannot be changed from this function

sw $a1,8($sp) # cells array cannot be modified from here

sw $ra, 12($sp)

move $a0, $t6 # a0 has row index

move $a1, $t7 # a1 has col index

li $a2,'B' # a2 has the ASCII

li $a3, 7 # a3 has white color for foreground

jal set\_cell # set that cell with correct attributes

lw $t4,0($sp)

lw $a0, 4($sp)

lw $a1, 8($sp)

lw $ra, 12($sp)

addi $sp,$sp, 16

j increase\_coordinates

flag:

li $t3, 32 # to get the 32nd bit for bomb 0:no or 1:yes

and $t3,$t2,$t3 # get only the 32nd bit in t3, t2 still has entire byte value

beq $t3,32,correct\_flag # if the cell has bomb display bomb

li $t4, 9 #background color bright red

j incorrect\_flag

correct\_flag:

li $t4, 10 #flag background is bright green

incorrect\_flag:

addi $sp, $sp, -16 # move the back ground color on to stack

sw $t4, 0($sp) # background color on stack

sw $a0,4($sp) # state of game cannot be changed from this function

sw $a1,8($sp) # cells array cannot be modified from here

sw $ra, 12($sp)

move $a0, $t6 # a0 has row index

move $a1, $t7 # a1 has col index

li $a2,'f' # a2 has the ASCII

li $a3, 12 # a3 has bright color for foreground

jal set\_cell # set that cell with correct attributes

lw $t4,0($sp)

lw $a0, 4($sp)

lw $a1, 8($sp)

lw $ra, 12($sp)

addi $sp,$sp, 16

j increase\_coordinates

number:

li $t4, 0 # flag background is black

addi $t3,$t3,'0'

addi $sp, $sp, -16 # move the back ground color on to stack

sw $t4, 0($sp) # background color on stack

sw $a0,4($sp) # state of game cannot be changed from this function

sw $a1,8($sp) # cells array cannot be modified from here

sw $ra, 12($sp)

move $a0, $t6 # a0 has row index

move $a1, $t7 # a1 has col index

move $a2, $t3 # a2 has the ASCII

li $a3, 13 # a3 has bright color for foreground

jal set\_cell # set that cell with correct attributes

lw $t4,0($sp)

lw $a0, 4($sp)

lw $a1, 8($sp)

lw $ra, 12($sp)

addi $sp,$sp, 16

j increase\_coordinates

empty\_cell:

li $t4, 0 # flag background is black

addi $t3,$t3,'\0'

addi $sp, $sp, -16 # move the back ground color on to stack

sw $t4, 0($sp) # background color on stack

sw $a0,4($sp) # state of game cannot be changed from this function

sw $a1,8($sp) # cells array cannot be modified from here

sw $ra, 12($sp)

move $a0, $t6 # a0 has row index

move $a1, $t7 # a1 has col index

move $a2, $t3 # a2 has the ASCII

li $a3, 15 # a3 has bright color for foreground

jal set\_cell # set that cell with correct attributes

lw $t4,0($sp)

lw $a0, 4($sp)

lw $a1, 8($sp)

lw $ra, 12($sp)

addi $sp,$sp, 16

j increase\_coordinates

reveal\_done:

li $t0, 0xffff0000

lw $t1, cursor\_row

lw $t2, cursor\_col

li $t5, 0

li $t9, 10

mul $t5, $t1, $t9 # first number \* total columns

add $t5, $t5, $t2 # first number \* total columns + column

sll $t5, $t5,1

add $t5, $t5, $t0 # base\_addr + (i \* num\_columns + j)

li $t7, 'e'

sb $t7, ($t5)

addi $t5,$t5, 1

li $t7, 9

sll $t7,$t7,4

addi $t7,$t7,15

sb $t7, ($t5)

jr $ra

##############################

# PART 4 FUNCTIONS

##############################

perform\_action:

move $t0, $a0 # t0 has the cell array

move $t1, $a1 # t1 has the action - f/F/r/R/w/W/a/A/s/S/d/D

lw $t2, cursor\_row # t2 has the row coordinate of cursor

lw $t3, cursor\_col # t3 has the col coordinate of cursor

li $v0, 0 # move is valid unless told

li $t5, 0 # will have address in t5 for cell to perfom action on

li $t6, 10 # total number of columns

mul $t5, $t2, $t6 # first number \* total columns

add $t5, $t5, $t3 # first number \* total columns + column

add $t5, $t5, $t0 # base\_addr + (i \* num\_columns + j)

beq $t1,'f',flag\_it

beq $t1,'F',flag\_it

beq $t1,'r',reveal\_it

beq $t1,'R',reveal\_it

beq $t1,'w',forward\_it

beq $t1,'W',forward\_it

beq $t1,'a',left\_it

beq $t1,'A',left\_it

beq $t1,'d',right\_it

beq $t1,'D',right\_it

beq $t1,'s',backward\_it

beq $t1,'S',backward\_it

j invalid\_action1

flag\_it:

#check if cell is already revealed or not

lb $t7, ($t5) # get byte of the cursor cell from the cells\_array

li $t4,16 # to check for a flag

and $t4,$t4,$t7 # get the 4th bit to see if flag or not

beq $t4,16,remove\_flag # flag there, remove it now

li $t4,64 #check if cell is not revealed, 6th bit

and $t4,$t4,$t7

beq $t4,64,invalid\_action1 # if revealed, cannot flag the cell

#addi $t4,$t4,64 #make cell revealed status before adding flag so flag can be seen

#add a flag since not there

addi $t7,$t7,16 # cell is flagged

sb $t7, ($t5)

li $t8,11 #yellow background

addi $sp,$sp,-16

sw $t8,0($sp) #background yellow

sw $a0,4($sp)

sw $a1,8($sp)

sw $ra,12($sp)

move $a0, $t2 #cursor\_row

move $a1, $t3 #cursor\_col

li $a2, 'f' #ascii flag

li $a3, 12 #blue foreground

jal set\_cell

lw $t8 , 0($sp)

lw $a0, 4($sp)

lw $a1, 8($sp)

lw $ra, 12($sp)

addi $sp,$sp,16

li $v0,0 # move was valid

jr $ra

remove\_flag:

addi $t7,$t7,-16 # remove flag

sb $t7, ($t5)

#addi $t4,$t4,-64

li $t5, 0 # will have address in t5 for cell to perfom action on

li $t6, 10 # total number of columns

mul $t5, $t2, $t6 # first number \* total columns

add $t5, $t5, $t3 # first number \* total columns + column

add $t5, $t5, $t0 # base\_addr + (i \* num\_columns + j)

lb $t4, ($t5)

li $t6, '\0'

li $t9, 13

andi $t5,$t4,64

beq $t5, 64, change\_color

li $t9, 11

change\_color:

andi $t4, $t4, 15

blez $t4, not\_a\_null

addi $t6, $t4, '0'

not\_a\_null:

li $t8,11 #yellow background

addi $sp,$sp,-16

sw $t8,0($sp)

sw $a0,4($sp)

sw $a1,8($sp)

sw $ra,12($sp)

move $a0, $t2

move $a1, $t3

move $a2, $t6

move $a3, $t9

jal set\_cell

lw $t8, 0($sp)

lw $a0, 4($sp)

lw $a1, 8($sp)

lw $ra, 12($sp)

addi $sp,$sp,16

li $v0,0 #move was valid

jr $ra

reveal\_it:

lb $t7, ($t5) # get byte of the cursor cell

li $t4,64 # to check if revealed

and $t4,$t4,$t7 # get the 4th bit to see if revealed or not

beq $t4,64,invalid\_action1

lb $t7, ($t5)

li $t4, 16

and $t4,$t4,$t7

bne $t4,16,had\_a\_flag

addi $t7, $t7, -16

sb $t7, ($t5)

had\_a\_flag:

lb $t7, ($t5)

addi $t7,$t7,64

sb $t7, ($t5) #made the cell revealed ####################################################

lb $t7, ($t5)

andi $t4, $t7, 32 #check bomb

li $t9, 'b'

beq $t4, 32, make\_null\_now

lb $t9, ($t5)

andi $t9,$t9,15

addi $t9,$t9,'0' #t7 has the ASCII to store

bgt $t9,'0',make\_null\_now

li $t9,'\0'

make\_null\_now:

li $t8,11 #yellow background

addi $sp,$sp,-20

sw $t8,0($sp)

sw $a0,4($sp)

sw $a1,8($sp)

sw $t3,12($sp)

sw $ra,16($sp)

move $a0, $t2

move $a1, $t3

move $a2, $t9

li $a3, 13

jal set\_cell

lw $t8 , 0($sp)

lw $a0, 4($sp)

lw $a1, 8($sp)

lw $t3,12($sp)

lw $ra, 16($sp)

addi $sp,$sp,20

addi $sp,$sp,-4

sw $ra,0($sp)

lw $a1, cursor\_row

lw $a2, cursor\_col

jal search\_cells

lw $ra,0($sp)

addi $sp,$sp,4

li $v0, 0

jr $ra

left\_it:

li $t7,1 # to see what cursor - 1 col means

sub $t7,$t3,$t7 # add 1 to the col and store in t7

blt $t7,0,invalid\_action1 # if col is greater or equal to 10 invalid action

#before moving cursor, reveal current cell back to original

# if cell was revealed, make is black background or else grey background

lw $t2, cursor\_row # t2 has the row coordinate of cursor

lw $t3, cursor\_col # t3 has the col coordinate of cursor

li $v0, 0 # move is valid unless told

li $t5, 0 # will have address in t5 for cell to perfom action on

li $t6, 10 # total number of columns

mul $t5, $t2, $t6 # first number \* total columns

add $t5, $t5, $t3 # first number \* total columns + column

add $t5, $t5, $t0 # base\_addr + (i \* num\_columns + j)

lb $t7, ($t5)

li $t8, 0 #background color b

li $t1,13

li $t4,64 #check if cell is not revealed, 6th bit

and $t4,$t4,$t7 #color bright blue

beq $t4,64,grey\_back\_left

li $t1,12

li $t4,16 #check if cell is not revealed, 6th bit

and $t4,$t4,$t7

li $t8, 7 #color bright blue

beq $t4,16,grey\_back\_left

li $t8, 7 #background color g

li $t1, 7 #color bright blue

grey\_back\_left:

#since cell is revealed, check if flag or just a number to set the ascii properly

#first just check flag

#li $t9,'f' # assume its a flag

li $t0, 0xffff0000

mul $t5, $t2, $t6 # first number \* total columns

add $t5, $t5, $t3 # first number \* total columns + column

sll $t5,$t5,1

add $t5, $t5, $t0 # base\_addr + (i \* num\_columns + j)

lb $t7, ($t5)

move $t9,$t7

prev\_flag\_left:

addi $sp,$sp,-16

sw $t8,0($sp)

sw $a0,4($sp)

sw $a1,8($sp)

sw $ra,12($sp)

move $a0, $t2

move $a1, $t3

move $a2, $t9

move $a3, $t1

jal set\_cell

lw $t8, 0($sp)

lw $a0, 4($sp)

lw $a1, 8($sp)

lw $ra, 12($sp)

addi $sp,$sp,16

lw $t3, cursor\_col # t3 has the col coordinate of cursor

addi $t3,$t3,-1

sw $t3, cursor\_col

move $t0, $a0

addi $sp,$sp,-8

sw $t0,0($sp)

sw $ra,4($sp)

jal cursor\_moved

lw $t0,0($sp)

lw $ra,4($sp)

addi $sp,$sp,8

li $v0,0 # move was valid

jr $ra

#j game\_status

right\_it:

li $t7,1 # to see what cursor + 1 col means

add $t7,$t7,$t3 # add 1 to the col and store in t7

bge $t7,10,invalid\_action1 # if col is greater or equal to 10 invalid action

lw $t2, cursor\_row # t2 has the row coordinate of cursor

lw $t3, cursor\_col # t3 has the col coordinate of cursor

li $v0, 0 # move is valid unless told

li $t5, 0 # will have address in t5 for cell to perfom action on

li $t6, 10 # total number of columns

mul $t5, $t2, $t6 # first number \* total columns

add $t5, $t5, $t3 # first number \* total columns + column

add $t5, $t5, $t0 # base\_addr + (i \* num\_columns + j)

lb $t7, ($t5)

li $t8, 0 #background color b

li $t1,13

li $t4,64 #check if cell is not revealed, 6th bit

and $t4,$t4,$t7 #color bright blue

beq $t4,64,grey\_back\_right

li $t1,12

li $t4,16 #check if cell is not revealed, 6th bit

and $t4,$t4,$t7

li $t8, 7 #color bright blue

beq $t4,16,grey\_back\_right

li $t8, 7 #background color g

li $t1, 7

grey\_back\_right:

#since cell is revealed, check if flag or just a number to set the ascii properly

#first just check flag

li $t0, 0xffff0000

mul $t5, $t2, $t6 # first number \* total columns

add $t5, $t5, $t3 # first number \* total columns + column

sll $t5,$t5,1

add $t5, $t5, $t0 # base\_addr + (i \* num\_columns + j)

lb $t7, ($t5)

move $t9,$t7 # t9 has the ASCII

#li $t4,16 #check if cell is flagged, if so make ASCII 'f', or else make null temp................................

#and $t4,$t4,$t7

#beq $t4,16,prev\_flag\_right # if flagged, keep the ascii flag

#li $t9, '\0' # or else make it a null

#li $t6,7

prev\_flag\_right:

addi $sp,$sp,-16

sw $t8,0($sp)

sw $a0,4($sp)

sw $a1,8($sp)

sw $ra,12($sp)

move $a0, $t2

move $a1, $t3

move $a2, $t9

move $a3, $t1

jal set\_cell

lw $t8, 0($sp)

lw $a0, 4($sp)

lw $a1, 8($sp)

lw $ra, 12($sp)

addi $sp,$sp,16

lw $t3, cursor\_col # t3 has the col coordinate of cursor

addi $t3,$t3,1

sw $t3, cursor\_col

move $t0, $a0

addi $sp,$sp,-8

sw $t0,0($sp)

sw $ra,4($sp)

jal cursor\_moved

lw $t0,0($sp)

lw $ra,4($sp)

addi $sp,$sp,8

li $v0,0 # move was valid

jr $ra

forward\_it:

li $t7,1 # to see what cursor + 1 row means

sub $t7,$t2,$t7 # add 1 to the row and store in t7

blt $t7,0,invalid\_action1 # if row is greater or equal to 10 invalid action

lw $t2, cursor\_row # t2 has the row coordinate of cursor

lw $t3, cursor\_col # t3 has the col coordinate of cursor

li $v0, 0 # move is valid unless told

li $t5, 0 # will have address in t5 for cell to perfom action on

li $t6, 10 # total number of columns

mul $t5, $t2, $t6 # first number \* total columns

add $t5, $t5, $t3 # first number \* total columns + column

add $t5, $t5, $t0 # base\_addr + (i \* num\_columns + j)

lb $t7, ($t5)

li $t8, 0 #background color b

li $t1,13

li $t4,64 #check if cell is not revealed, 6th bit

and $t4,$t4,$t7 #color bright blue

beq $t4,64,grey\_back\_forward

li $t1,12

li $t4,16 #check if cell is not revealed, 6th bit

and $t4,$t4,$t7

li $t8, 7 #color bright blue

beq $t4,16,grey\_back\_forward

li $t8, 7 #background color g

li $t1, 7

grey\_back\_forward:

#since cell is revealed, check if flag or just a number to set the ascii properly

#first just check flag

li $t0, 0xffff0000

mul $t5, $t2, $t6 # first number \* total columns

add $t5, $t5, $t3 # first number \* total columns + column

sll $t5,$t5,1

add $t5, $t5, $t0 # base\_addr + (i \* num\_columns + j)

lb $t7, ($t5)

move $t9,$t7

previous\_flag\_forward:

addi $sp,$sp,-16

sw $t8,0($sp)

sw $a0,4($sp)

sw $a1,8($sp)

sw $ra,12($sp)

move $a0, $t2

move $a1, $t3

move $a2, $t9

move $a3, $t1

jal set\_cell

lw $t8, 0($sp)

lw $a0, 4($sp)

lw $a1, 8($sp)

lw $ra, 12($sp)

addi $sp,$sp,16

lw $t2, cursor\_row # t3 has the col coordinate of cursor

addi $t2,$t2,-1

sw $t2, cursor\_row

move $t0, $a0

addi $sp,$sp,-8

sw $t0,0($sp)

sw $ra,4($sp)

jal cursor\_moved

lw $t0,0($sp)

lw $ra,4($sp)

addi $sp,$sp,8

li $v0,0 # move was valid

jr $ra

#j game\_status

backward\_it:

li $t7,1 # to see what cursor + 1 row means

add $t7,$t7,$t2 # add 1 to the row and store in t7

bge $t7,10,invalid\_action1 # if row is greater or equal to 10 invalid action

lw $t2, cursor\_row # t2 has the row coordinate of cursor

lw $t3, cursor\_col # t3 has the col coordinate of cursor

li $v0, 0 # move is valid unless told

li $t5, 0 # will have address in t5 for cell to perfom action on

li $t6, 10 # total number of columns

mul $t5, $t2, $t6 # first number \* total columns

add $t5, $t5, $t3 # first number \* total columns + column

add $t5, $t5, $t0 # base\_addr + (i \* num\_columns + j)

lb $t7, ($t5)

li $t8, 0 #background color b

li $t1,13

li $t4,64 #check if cell is not revealed, 6th bit

and $t4,$t4,$t7 #color bright blue

beq $t4,64,grey\_back\_backward

li $t1,12

li $t4,16 #check if cell is not revealed, 6th bit

and $t4,$t4,$t7

li $t8, 7 #color bright blue

beq $t4,16,grey\_back\_backward

li $t8, 7 #background color g

li $t1, 7

grey\_back\_backward:

#since cell is revealed, check if flag or just a number to set the ascii properly

#first just check flag

li $t0, 0xffff0000

mul $t5, $t2, $t6 # first number \* total columns

add $t5, $t5, $t3 # first number \* total columns + column

sll $t5,$t5,1

add $t5, $t5, $t0 # base\_addr + (i \* num\_columns + j)

lb $t7, ($t5)

move $t9,$t7

prev\_flag\_b:

addi $sp,$sp,-16

sw $t8,0($sp)

sw $a0,4($sp)

sw $a1,8($sp)

sw $ra,12($sp)

move $a0, $t2

move $a1, $t3

move $a2, $t9

move $a3, $t1

jal set\_cell

lw $t8, 0($sp)

lw $a0, 4($sp)

lw $a1, 8($sp)

lw $ra, 12($sp)

addi $sp,$sp,16

lw $t2, cursor\_row # t3 has the col coordinate of cursor

addi $t2,$t2,1

sw $t2, cursor\_row

move $t0, $a0

addi $sp,$sp,-8

sw $t0,0($sp)

sw $ra,4($sp)

jal cursor\_moved

lw $t0,0($sp)

lw $ra,4($sp)

addi $sp,$sp,8

li $v0,0 # move was valid

jr $ra

#j game\_status

invalid\_action1:

li $v0,-1

jr $ra

#move $a0, $v0

#j game\_status

cursor\_moved:

lw $t5,cursor\_row #x coordinate of cursor

lw $t6,cursor\_col #y coordinate of cursor

li $t7, 0 # will have address in t5 for cell to perfom action on

li $t8, 10

mul $t7, $t5, $t8 # first number \* total columns

add $t7, $t7, $t6 # first number \* total columns + column

add $t7, $t7, $t0 # base\_addr + (i \* num\_columns + j)

lb $t8, ($t7)

li $t7, 64

and $t7, $t7,$t8

li $t1,12

beq $t7,64,revealed\_cursor

li $t7, 16

and $t7, $t7,$t8

beq $t7,16,revealed\_cursor

li $t1, 11

revealed\_cursor:

li $t9, 0xffff0000

li $t7, 0 # will have address in t5 for cell to perfom action on

li $t8, 10 # total number of columns

mul $t7, $t5, $t8 # first number \* total columns

add $t7, $t7, $t6 # first number \* total columns + column

sll $t7,$t7,1

add $t7, $t7, $t9 # base\_addr + (i \* num\_columns + j)

lb $t8, ($t7)

li $t3, 11 # background color

addi $sp,$sp,-16

sw $t3,0($sp)

sw $a0,4($sp)

sw $a1,8($sp)

sw $ra,12($sp)

move $a0, $t5

move $a1, $t6

move $a2, $t8

move $a3, $t1 #keep foreground and background color yellow

jal set\_cell

lw $t8, 0($sp)

lw $a0, 4($sp)

lw $a1, 8($sp)

lw $ra, 12($sp)

addi $sp,$sp,16

#li $v0, 0

jr $ra

game\_status:

li $v0,1 #initial game state is YOU WON

move $t0,$a0 #t0 has the cells\_array

li $t1, 0 # row pointer

li $t2, 0 # col pointer

li $t3,0 # will have address

li $t9,10 # number of cols

calc\_new\_address:

mul $t3, $t1, $t9 # first number \* total columns

add $t3, $t3, $t2 # first number \* total columns + column

add $t3, $t3, $t0 # base\_addr + (i \* num\_columns + j)

lb $t4, ($t3) # get the byte at that address

# check does it have a bomb

li $t5,32 # get the bit to check if a bomb

and $t5,$t5,$t4

beq $t5,32,check\_flag

li $t5,16 # get the bit to check if a bomb

and $t5,$t5,$t4

beq $t5,16,check\_bomb

after\_flag\_check:

addi $t2,$t2,1 # increase col by 1

blt $t2,10, calc\_new\_address # go to next column

li $t2,0 #change col to 0

addi $t1,$t1,1 # increase row by 1

blt $t1,10, calc\_new\_address

j all\_cells\_checked

check\_flag:

li $t5,16 # get the bit to check if a flag

and $t5,$t5,$t4

beq $t5,16,after\_flag\_check

li $t5,64 # check if bomb cell was revelealed

and $t5, $t5,$t4

beq $t5,64, bomb\_revealed

li $v0,0 # since spot had bomb and no flag found, still continue game

j after\_flag\_check

check\_bomb:

li $t5,32 # get the bit to check if a flag

and $t5,$t5,$t4

beq $t5,32,after\_flag\_check

li $t5,64 # get the bit to check if a flag

and $t5,$t5,$t4

beq $t5,64,bomb\_revealed

li $v0,0 # you lose state

j after\_flag\_check

bomb\_revealed:

li $v0,-1 # you lose state

jr $ra

all\_cells\_checked:

#li $v0,0

jr $ra

##############################

# PART 5 FUNCTIONS

##############################

search\_cells:

move $t0, $a0 # t0 has the cells array

move $t1, $a1 # t1 has the row

move $t2, $a2 # t2 has the col

li $t7,0 #to store final address

li $t8, 10 #number of rows and cols

move $fp, $sp # fp has start of sp value

addi $sp,$sp,-4

sw $t1,($sp)

addi $sp,$sp,-4

sw $t2,($sp)

p5loop:

beq $fp,$sp,end

lw $t2,($sp) #t2 has col

addi $sp,$sp,4

lw $t1,($sp) #t1 has row

addi $sp,$sp,4

li $t7,0

li $t8,10

mul $t7, $t1, $t8 # first number \* total columns

add $t7, $t7, $t2 # first number \* total columns + column

add $t7, $t7, $t0 #t7 will have address of cells array with current row and col

lb $t6, ($t7) #t6 has the byte of that particular cell

andi $t6,$t6,16 # check if flag

beq $t6,16,next\_search0 # if a flag skip next part and go to next search

####################################

lb $t6, ($t7) # get byte of the cursor cell

li $t4,64 # to check if revealed

and $t4,$t4,$t6 # get the 4th bit to see if revealed or not

beq $t4,64,next\_search0

lb $t6, ($t7)

li $t4, 16

and $t4,$t4,$t6

beq $t4,16,next\_search0

lb $t6, ($t7)

addi $t6,$t6,64

sb $t6, ($t7)

had\_a\_flag5:

#lb $t6, ($t7)

#addi $t6,$t6,64

#sb $t6, ($t7) #made the cell revealed ####################################################

#### #lb $t6, ($t7)

#li $t4, 32

#and $t4,$t4,$t6

#### #beq $t4,32,next\_search0

lb $t6, ($t7)

andi $t4, $t6, 32 #check bomb

li $t9, 'b'

beq $t4, 32, make\_null\_now5

lb $t9, ($t7)

andi $t9,$t9,15

addi $t9,$t9,'0' #t9 has the ASCII to store

bgt $t9,'0',make\_null\_now5

li $t9,'\0'

make\_null\_now5:

li $t8,0 #black background

addi $sp,$sp,-4

sw $a0,($sp)

addi $sp,$sp,-4

sw $a1,($sp)

addi $sp,$sp,-4

sw $t7,($sp)

addi $sp,$sp,-4

sw $t1,($sp)

addi $sp,$sp,-4

sw $t2,($sp)

addi $sp,$sp,-4

sw $t0,($sp)

addi $sp,$sp,-4

sw $ra,($sp)

addi $sp,$sp,-4

sw $t8,($sp)

#addi $sp,$sp,-4

#sw $ra,($sp)

#addi $sp,$sp,-4

move $a0, $t1

move $a1, $t2

move $a2, $t9

li $a3, 13

jal set\_cell

lw $t8,($sp)

addi $sp,$sp,4

lw $ra,($sp)

addi $sp,$sp,4

lw $t0,($sp)

addi $sp,$sp,4

lw $t2,($sp)

addi $sp,$sp,4

lw $t1,($sp)

addi $sp,$sp,4

lw $t7,($sp)

addi $sp,$sp,4

lw $a1,($sp)

addi $sp,$sp,4

lw $a0,($sp)

addi $sp,$sp,4

#lb $t6, ($t7)

#addi $t6,$t6,64

#sb $t6, ($t7)

#########################################

next\_search0:

lb $t6, ($t7) #t6 has the byte of that particular cell

andi $t6,$t6,15 # check number

bne $t6,0,p5loop

addi $t1,$t1,1 # row = row + 1

bge $t1,10,next\_search0\_1

li $t8, 10

mul $t7, $t1, $t8 # first number \* total columns

add $t7, $t7, $t2 # first number \* total columns + column

add $t7, $t7, $t0 #t7 will have address of cells array with current row and col

lb $t6, ($t7)

andi $t6,$t6,64

bnez $t6,next\_search0\_1

lb $t6, ($t7)

andi $t6,$t6,16

beq $t6,16,next\_search0\_1

addi $sp,$sp,-4

sw $t1,($sp)

addi $sp,$sp,-4

sw $t2,($sp)

next\_search0\_1:

addi $t1,$t1,-1 #reduce row to row + 1 - 1 = row

addi $t2, $t2,1 #increase col by 1, col = col + 1

bge $t2,10,next\_search0\_2

li $t8,10

mul $t7, $t1, $t8 # first number \* total columns

add $t7, $t7, $t2 # first number \* total columns + column

add $t7, $t7, $t0 #t7 will have address of cells array with current row and col

lb $t6, ($t7)

andi $t6,$t6,64

bnez $t6,next\_search0\_2

lb $t6, ($t7)

andi $t6,$t6,16

beq $t6,16,next\_search0\_2

addi $sp,$sp,-4

sw $t1,($sp)

addi $sp,$sp,-4

sw $t2,($sp)

next\_search0\_2:

addi $t1,$t1,-1 #reduce row to row - 1

addi $t2, $t2,-1 #increase col by 1, col = col + 1 - 1 = col

blt $t1,0,next\_search0\_3

li $t8,10

mul $t7, $t1, $t8 # first number \* total columns

add $t7, $t7, $t2 # first number \* total columns + column

add $t7, $t7, $t0 #t7 will have address of cells array with current row and col

lb $t6, ($t7)

andi $t6,$t6,64

bnez $t6,next\_search0\_3

lb $t6, ($t7)

andi $t6,$t6,16

beq $t6,16,next\_search0\_3

addi $sp,$sp,-4

sw $t1,($sp)

addi $sp,$sp,-4

sw $t2,($sp)

next\_search0\_3:

addi $t1,$t1,1 #reduce row to row - 1 + 1 = row

addi $t2, $t2,-1 #reduce col by 1, col = col - 1

blt $t2,0,next\_search0\_4

li $t8,10

mul $t7, $t1, $t8 # first number \* total columns

add $t7, $t7, $t2 # first number \* total columns + column

add $t7, $t7, $t0 #t7 will have address of cells array with current row and col

lb $t6, ($t7)

andi $t6,$t6,64

bnez $t6,next\_search0\_4

lb $t6, ($t7)

andi $t6,$t6,16

beq $t6,16,next\_search0\_4

addi $sp,$sp,-4

sw $t1,($sp)

addi $sp,$sp,-4

sw $t2,($sp)

next\_search0\_4:

addi $t1,$t1,-1 #reduce row to row - 1 # col = col - 1

blt $t2,0,next\_search0\_5

blt $t1,0,next\_search0\_5

li $t8,10

mul $t7, $t1, $t8 # first number \* total columns

add $t7, $t7, $t2 # first number \* total columns + column

add $t7, $t7, $t0 #t7 will have address of cells array with current row and col

lb $t6, ($t7)

andi $t6,$t6,64

bnez $t6,next\_search0\_5

lb $t6, ($t7)

andi $t6,$t6,16

beq $t6,16,next\_search0\_5

addi $sp,$sp,-4

sw $t1,($sp)

addi $sp,$sp,-4

sw $t2,($sp)

next\_search0\_5:

addi $t2,$t2,2 # increase col to col - 1 + 2 = col + 1

bge $t2,10,next\_search0\_6

blt $t1,0,next\_search0\_6

li $t8,10

mul $t7, $t1, $t8 # first number \* total columns

add $t7, $t7, $t2 # first number \* total columns + column

add $t7, $t7, $t0 #t7 will have address of cells array with current row and col

lb $t6, ($t7)

andi $t6,$t6,64

bnez $t6,next\_search0\_6

lb $t6, ($t7)

andi $t6,$t6,16

beq $t6,16,next\_search0\_6

addi $sp,$sp,-4

sw $t1,($sp)

addi $sp,$sp,-4

sw $t2,($sp)

next\_search0\_6:

addi $t1,$t1,2 #reduce row to row - 1 + 2 = row + 1

addi $t2,$t2,-2 # increase col to col + 1 - 2 = col - 1

blt $t2,0,next\_search0\_7

bge $t1,10,next\_search0\_7

li $t8,10

mul $t7, $t1, $t8 # first number \* total columns

add $t7, $t7, $t2 # first number \* total columns + column

add $t7, $t7, $t0 #t7 will have address of cells array with current row and col

lb $t6, ($t7)

andi $t6,$t6,64

bnez $t6,next\_search0\_7

lb $t6, ($t7)

andi $t6,$t6,16

beq $t6,16,next\_search0\_7

addi $sp,$sp,-4

sw $t1,($sp)

addi $sp,$sp,-4

sw $t2,($sp)

next\_search0\_7:

addi $t2,$t2,2 # increase col to col - 1 + 2 = col + 1

bge $t2,10,p5loop

bge $t1,10,p5loop

li $t8,10

mul $t7, $t1, $t8 # first number \* total columns

add $t7, $t7, $t2 # first number \* total columns + column

add $t7, $t7, $t0 #t7 will have address of cells array with current row and col

lb $t6, ($t7)

andi $t6,$t6,64

bnez $t6,p5loop

lb $t6, ($t7)

andi $t6,$t6,16

beq $t6,16,p5loop

addi $sp,$sp,-4

sw $t1,($sp)

addi $sp,$sp,-4

sw $t2,($sp)

j p5loop

end:

jr $ra

#################################################################

# Student defined data section

#################################################################

.data

.align 2 # Align next items to word boundary

cursor\_row: .word -1

cursor\_col: .word -1

buffer: .space 1

#place any additional data declarations here