```
# Name: Sam Zandiasadabadi
# Date: 05/26/2023
# Description: An ancient robot game. Translating final C++ robot program
into assembly
                        .data
x:                      .word       0:4         # x-coordinates of 4 robots
y:                      .word       0:4         # y-coordinates of 4 robots

initialCoord:           .asciiz     "Your coordinates: 25 25\n"
enterMove:              .asciiz     "Enter move (1 for +x, -1 for -x, 2
for + y, -2 for -y):"
finalCoord:             .asciiz     "Your coordinates: "
space:                  .asciiz     " "
endl:                   .asciiz     "\n"
robotCoord:             .asciiz     "Robot at "
gameOver:         .asciiz      "AAAARRRRGHHHHH... Game over\n"

#i          $s0
#myX              $s1
#myY              $s2
#move             $s3
#status           $s4
#temp,pointers  $s5,$s6
                        .text
#.globl         inc
#.globl         getNew

main:
        li          $s1,25                  #  myX = 25
        li          $s2,25                  #  myY = 25
        li          $s4,1                   #  status = 1

        la          $s5,x
        la          $s6,y

        sw          $0,($s5)          #  x[0] = 0; y[0] = 0;
        sw          $0,($s6)
        sw          $0,4($s5)               #  x[1] = 0; y[1] = 50;
        li          $s7,50
        sw          $s7,4($s6)
        sw          $s7,8($s5)              #  x[2] = 50; y[2] = 0;
        sw          $0,8($s6)
        sw          $s7,12($s5)             #  x[3] = 50; y[3] = 50;
        sw          $s7,12($s6)

        la          $a0,initialCoord        #  cout << "Your coordinates: 25
25\n";
        li          $v0,4
        syscall

        bne         $s4,1,main_exitw        #  while (status == 1) {

main_while:
```

```
        la      $a0,enterMove           #    cout << "Enter move (1 for
+x,
        li      $v0,4                   #    -1 for -x, 2 for + y, -2 for
-y):";
        syscall

        li      $v0,5                   #    cin >> move;
        syscall
        move    $s3,$v0

        bne     $s3,1,main_else1 #    if (move == 1)
        add     $s1,$s1,1               #    myX++;
        b       main_exitif

main_else1:
        bne     $s3,-1,main_else2       #    else if (move == -1)
        add     $s1,$s1,-1              #    myX--;
        b       main_exitif

main_else2:
        bne     $s3,2,main_else3        #    else if (move == 2)
        add     $s2,$s2,1       #    myY++;
        b       main_exitif

main_else3:
        bne     $s3,-2,main_exitif      #    else if (move == -2)
        add     $s2,$s2,-1              #    myY--;

main_exitif:
        la      $a0,x                   #    status =
moveRobots(&x[0],&y[0],myX,myY);
        la      $a1,y
        move    $a2,$s1
        move    $a3,$s2
        jal     moveRobots
        move    $s4,$v0


        la      $a0,finalCoord          #    cout << "Your coordinates: "
<< myX
        li      $v0,4                   #       << " " << myY << endl;
        syscall
        move    $a0,$s1
        li      $v0,1
        syscall
        la      $a0,space
        li      $v0,4
        syscall
        move    $a0,$s2
        li      $v0,1
        syscall
        la      $a0,endl
        li      $v0,4
        syscall
```

```
        la      $s5,x
        la      $s6,y
        li      $s0,0               #    for (i=0;i<4;i++)

main_for:
        la      $a0,robotCoord      #      cout << "Robot at " << x[i]
<< " "
        li      $v0,4               #           << y[i] << endl;
        syscall
        lw      $a0,($s5)
        li      $v0,1
        syscall
        la      $a0,space
        li      $v0,4
        syscall
        lw      $a0,($s6)
        li      $v0,1
        syscall
        la      $a0,endl
        li      $v0,4
        syscall
        add     $s5,$s5,4
        add     $s6,$s6,4
        add     $s0,$s0,1
        blt     $s0,4,main_for

        beq     $s4,1,main_while#  }
main_exitw:
        la      $a0,gameOver          #  cout << "AAAARRRRGHHHHH... Game
over\n";
        li      $v0,4
        syscall
        li      $v0,10              # }
        syscall

#i          $s0
#myX              $s1
#myY              $s2
#move             $s3
#status           $s4
#temp,pointers  $s5,$s6
moveRobots:
                                            # allocating enough
space/memory for the values to be stored
        addi      $sp,$sp,-20
        sw        $s1,($sp)           #
        sw        $s2,4($sp)          #
        sw        $s5,8($sp)          # creating and initializing int i,
*ptrX...
        sw        $s6,12($sp)         # *ptrY, alive
        sw        $ra,16($sp)         #
```

```
        move        $s5,$a0             #
        move        $s6,$a1             # ptrX = arg0;
        move        $s1,$a2             # ptrY = arg1;
        move        $s2,$a3             #
        li          $s0,0               #

for:                                    # for() {
        lw          $a0,($s5)           # allocating enough space for the
values...
        move        $a1,$s1             # to be stored and be called upon
        jal         getNew              # *ptrX = getNew(*ptrX,arg2);
        sw          $v0,($s5)
        lw          $a0,($s6)           # allocating enough space for the
values...
        move        $a1,$s2             # to be stored and be called upon
        jal         getNew              # *ptrY = getNew(*ptrX,arg3);
        sw          $v0,($s6)

        li          $v0,1               # int alive = 1;
        lw          $a0,($s5)
        lw          $a1,($s6)
        bne         $a0,$s1,makeEqual   # if (*ptrX != arg2) { // call
'makeEqual'}
        bne         $a1,$s2,makeEqual   # if (*ptrY != arg3) { // call
'makeEqual')
        li          $v0,0               # int alive = 0;
        j           endLoop             # go to 'endLoop'


makeEqual:                          # making pointer values equal to
respective 'arg' variable
        addi        $s5,$s5,4           # ptrX++;
        addi        $s6,$s6,4           # ptrY++;
        addi        $s0,$s0,1           # i++;
        blt         $s0,4,for           # i < 4  // for the for loop;


endLoop:                            # to exit the loop
        lw          $ra,16($sp)         #
        lw          $s6,12($sp)         #
        lw          $s5,8($sp)          #
        lw          $s2,4($sp)          # loading the values inside each
array
        lw          $s1,($sp)           #
        addi        $sp,$sp,20          #
        jr          $ra                 # jumping to the $ra register to
return the data

getNew:                             # int getNew(int arg0, int arg1 {
                                            # allocating memory in order
to create/store more values
        addi        $sp,$sp,-8          #
        sw          $s5,($sp)           #
        sw          $ra,4($sp)          #
```

```
        sub         $s5,$a0,$a1          # int temp = arg0 - arg1;
        blt         $s5,10,elseIf1       # if (temp >= 10) // we continue
to next line
                                         # if not, we call upon 'elseIf1'
        sub         $v0,$a0,10           # result = arg0 - 10;
        j           endIf                # we jump to end of if statement

elseIf1:                              # first 'else if' statement
        ble         $s5,0,elseIf2        # else if (temp > 0) // we
continue to next line
                                         # if not, we call upon 'elseIf2'
        sub         $v0,$a0,1            # result = arg0 - 1;
        j           endIf                # we jump to end of if statement

elseIf2:                              # second 'else if' statement
        bne         $s5,0,elseIf3        # else if (temp == 0) // we
continue to next line
                                         # if not, we call upon 'elseIf3'
        addi        $v0,$a0,0            # result = arg0;
        j           endIf                # we jump to end of if statement

elseIf3:                              # third 'else if' statement
        ble         $s5,-10,elseIf4      # else if (temp > -10) // we
continue to next line
                                         # if not, we call upon 'elseIf4'
        addi        $v0,$a0,1            # result = arg0 + 1;
        j           endIf                # we jump to end of if statement

elseIf4:                              # fourth 'else if' statement
        bgt         $s5,-10,endIf        # else if (temp <= -10) // we
continue to next line
        addi        $v0,$a0,10           # result = arg0 + 10;

endIf:                                   # ending the if statement
        lw          $ra,4($sp)           #
        lw          $s5,($sp)            # storing and recalling the values
        addi        $sp,$sp,8            #
        jr          $ra                  # returning the values to the $ra
register
```