Printout

Wednesday, February 20, 2019 3:10 AM

```
# Drew Pesall - 1129022
# CS 370
# 2/11/2019


.data
gameBoard:   .ascii   "\n\n\n\n| . . . . | . . . . | . . . . | . . . . |   a b c d\t\t"
             .ascii         "\n| . . . . | . . . . | . . . . | . . . . |   e f g h\t\t"
             .ascii         "\n| . . . . | . . . . | . . . . | . . . . |   i j k l\t\t"
             .ascii         "\n| . . . . | . . . . | . . . . | . . . . |   m n o p\t\t"
             .asciiz        "\n|   (0)   |   (1)   |   (2)   |   (3)   |   (index)\n"

introText:       .asciiz   "Start a One-Player, 4×4×4x4, 3D Tic-Tac-Toe Game.\n"

offset:    .half   6,   8,  10,  12,  16,  18,  20,  22,  26,  28,  30,  32,  36,  38,  40,  42
           .half  60,  62,  64,  66,  70,  72,  74,  76,  80,  82,  84,  86,  90,  92,  94,  96
           .half 114, 116, 118, 120, 124, 126, 128, 130, 134, 136, 138, 140, 144, 146, 148, 150
           .half 168, 170, 172, 174, 178, 180, 182, 184, 188, 190, 192, 194, 198, 200, 202, 204

gridMessage:        .asciiz     "\nSelect a grid(0-3): "
gridError:          .asciiz     "\nPlease select a valid grid."

indexMessage:       .asciiz     "\nSelect an index(a-p): "
indexError:         .asciiz     "\nPlease select a valid index."

continueMessage:    .asciiz     "\nContinue playing?(y/n): "
newGameMessage:     .asciiz     "\nStart a new game?(y/n): "
validMessage:       .asciiz     "\nPlease select a valid option."

selectPieceMessage: .asciiz     "\nWhat would you like to play as?(x/o): "
selectPieceError:   .asciiz     "\nPlease select a valid piece."

occupiedPosition:   .asciiz     "\nThere is already a piece there."

pieceX:             .byte 'x'
pieceO:             .byte 'o'
grid0:              .byte '0'
grid1:              .byte '1'
grid2:              .byte '2'
grid3:              .byte '3'
yes:                .byte 'y'
no:                 .byte 'n'
zero:               .half 0
occupiedPositionMessage: .asciiz "\nThere is already a piece there."

```

```
userInput:          .space  4
userInputGrid:      .space  4
userInputCell:      .space  4

index1:             .byte   'a'
index2:             .byte   'p'

.text

main:

jal initialDisplay
jal selectPiece

li $v0, 10
    syscall

initialDisplay: # Displays Initial Game Board and Greeting

# Prints out greeting
li      $v0, 4
la      $a0, introText
syscall

jr      $ra

resetGameBoard:
lh      $t0, zero
add     $t0, $t4, $t0
mul     $t0, $t0, 2
lh      $t1, offset($t0)
li      $t2, '.'
sb      $t2, gameBoard($t1)
add     $t4, $t4, 1
ble     $t4, 64, resetGameBoard

j main

selectPiece:
#Display prompt
li      $v0, 4
la      $a0, selectPieceMessage
syscall

#Enter your desired piece
move    $a0,$t2
li      $v0, 8
la      $a0, userInput
li      $a1, 10
syscall

```

```
#Compare
la      $s2, pieceX
lb      $t2, ($s2)
la         $s3, userInput
lb         $t3, ($s3)
beq        $t2,$t3,gameLoopX
la     $s4, pieceO
lb     $t2, ($s4)
beq        $t2,$t3,gameLoopO

li     $v0, 4
la     $a0, selectPieceError
syscall

j      selectPiece

jr         $ra

selectGrid:

#Display prompt
li      $v0, 4
la      $a0, gridMessage
syscall

#Enter your desired grid
move    $a0,$t2
li     $v0, 8
la     $a0, userInputGrid
li     $a1, 10
syscall

#Compare
la         $s2, grid0
lb         $t2, ($s2)
la         $s6, userInputGrid
lb         $t3, ($s6)
beq        $t2,$t3,test
la     $s4, grid1
lb     $t2, ($s4)
beq        $t2,$t3,test
la     $s4, grid2
lb     $t2, ($s4)
beq        $t2,$t3,test
la     $s4, grid3
lb     $t2, ($s4)
beq        $t2,$t3,test

li     $v0, 4
la     $a0, gridError
syscall
```

```

j      selectGrid

jr          $ra

test:

jr     $ra

continueGameX:
# Continue?
li       $v0, 4
la       $a0, continueMessage
syscall

#Enter your desired choice
move    $a0,$t2
li     $v0, 8
la     $a0, userInput
li     $a1, 10
syscall

#Compare
la          $s2, yes
lb          $t2, ($s2)
la          $s3, userInput
lb          $t3, ($s3)
beq         $t2,$t3,gameLoopX
la     $s4, no
lb     $t2, ($s4)
beq         $t2,$t3,newGame

li     $v0, 4
la     $a0, validMessage
syscall

j      continueGameX

jr          $ra

continueGameO:
# Continue?
li       $v0, 4
la       $a0, continueMessage
syscall

#Enter your desired choice
move    $a0,$t2
li     $v0, 8
la     $a0, userInput
li     $a1, 10
```

```
syscall

#Compare
la         $s2, yes
lb         $t2, ($s2)
la         $s3, userInput
lb         $t3, ($s3)
beq        $t2,$t3,gameLoopO
la     $s4, no
lb     $t2, ($s4)
beq        $t2,$t3,newGame

li     $v0, 4
la     $a0, validMessage
syscall

j      continueGameO

jr         $ra

exit:
li     $v0, 10
syscall

newGame:
# New Game?
li      $v0, 4
la      $a0, newGameMessage
syscall

#Enter your desired choice
move    $a0,$t2
li     $v0, 8
la     $a0, userInput
li     $a1, 10
syscall

#Compare
la         $s2, yes
lb         $t2, ($s2)
la         $s3, userInput
lb         $t3, ($s3)
lh     $t4, zero
beq        $t2,$t3,resetGameBoard
la     $s4, no
lb     $t2, ($s4)
beq        $t2,$t3,exit

li     $v0, 4
la     $a0, validMessage
syscall
```

```

j      newGame

jr         $ra

selectIndex:
# Select Index
li      $v0, 4
la      $a0, indexMessage
syscall

#Enter your desired choice
move    $a0,$t2
li     $v0, 8
la     $a0, userInput
li     $a1, 10
syscall

#Compare
la         $s2, index1
lb         $t2, ($s2)
la         $s7, userInput
lb         $t3, ($s7)
blt        $t3,$t2,indexRetry
la     $s4, index2
lb     $t2, ($s4)
bgt        $t3,$t2,indexRetry

jr         $ra

indexRetry:

li     $v0, 4
la     $a0, indexError
syscall

j      selectIndex

jr     $ra

gameLoopX:
# Printing Board
li         $v0, 4
la         $a0, gameBoard
syscall

# Selecting Grid
jal    selectGrid


# Selecting Index
```

```
jal        selectIndex

# Printing Board
lh         $t0, zero
lh         $t1, zero
lh     $t2, zero
lh         $t3, zero
lh     $t4, zero

lb     $t1, ($s6)
lb         $t2, ($s7)
sub        $t1, $t1, 48  # Grid
sub        $t2, $t2, 'a' # Index

div        $t0, $t2, 4
mul        $t0, $t0, 16

mul        $t1, $t1, 4


add        $t0, $t0, $t1
div        $t2, $t2, 4
mfhi   $t2
add        $t0, $t2, $t0

mul        $t0, $t0, 2
lh         $t1, offset($t0)
li     $t2, 'X'

# Checks to see if there is already a piece there
lb     $t9, gameBoard($t1)
bne        $t9, '.', occupiedX

sb     $t2, gameBoard($t1)

li     $v0, 4
la     $a0, gameBoard
syscall

# Continue?
jal        continueGameX

jr         $ra

occupiedX:
li     $v0, 4
la     $a0, occupiedPosition
syscall
j      gameLoopX

jr     $ra
```

```

occupiedO:
li     $v0, 4
la     $a0, occupiedPosition
syscall
j      gameLoopO

jr     $ra

gameLoopO:
# Printing Board
li          $v0, 4
la          $a0, gameBoard
syscall

# Selecting Grid
jal    selectGrid


# Selecting Index
jal         selectIndex

# Printing Board
lh          $t0, zero
lh          $t1, zero
lh     $t2, zero
lh          $t3, zero
lh     $t4, zero

lb     $t1, ($s6)
lb          $t2, ($s7)
sub         $t1, $t1, 48  # Grid
sub         $t2, $t2, 'a' # Index

div         $t0, $t2, 4
mul         $t0, $t0, 16

mul         $t1, $t1, 4


add         $t0, $t0, $t1
div         $t2, $t2, 4
mfhi   $t2
add         $t0, $t2, $t0

mul         $t0, $t0, 2
lh          $t1, offset($t0)
li     $t2, 'O'

# Checks to see if there is already a piece there.
lb     $t9, gameBoard($t1)
```

```
bne        $t9, '.', occupiedO

sb     $t2, gameBoard($t1)

li     $v0, 4
la     $a0, gameBoard
syscall

# Continue?
jal        continueGameO

jr         $ra
```