```
.data
board:    .ascii   "\n\n    . . . . . .     . . . . . .      0 1 2 3 4 5"
          .ascii    "\n    . . . . . .     . . . . . .      6 7 8 9 a b"
          .ascii    "\n    . . . . . .     . . . . . .      c d e f g h"
          .ascii    "\n    . . . . . .     . . . . . .      i j k l m n"
          .ascii    "\n    . . . . . .     . . . . . .      o p q r s t"
          .asciiz   "\n    . . . . . .     . . . . . .      u v w x y z\n"
sys_board: .ascii  "\n\n    . . . . . .     . . . . . .      0 1 2 3 4 5"
          .ascii    "\n    . . . . . .     . . . . . .      6 7 8 9 a b"
          .ascii    "\n    . . . . . .     . . . . . .      c d e f g h"
          .ascii    "\n    . . . . . .     . . . . . .      i j k l m n"
          .ascii    "\n    . . . . . .     . . . . . .      o p q r s t"
          .asciiz   "\n    . . . . . .     . . . . . .      u v w x y z\n"
offset1:  .half    6,   8,  10,  12,  14,  16
          .half   55,  57,  59,  61,  63,  65
          .half  104, 106, 108, 110, 112, 114
          .half  153, 155, 157, 159, 161, 163
          .half  202, 204, 206, 208, 210, 212
          .half  251, 253, 255, 257, 259, 261
offset2:  .half   22,  24,  26,  28,  30,  32
          .half   71,  73,  75,  77,  79,  81
          .half  120, 122, 124, 126, 128, 130
          .half  169, 171, 173, 175, 177, 179
          .half  218, 220, 222, 224, 226, 228
          .half  267, 269, 271, 273, 275, 277
offset3:  .half    6,   8,  10,  12,  14,  16        #These two offsets are for sys_board
          .half   55,  57,  59,  61,  63,  65
          .half  104, 106, 108, 110, 112, 114
          .half  153, 155, 157, 159, 161, 163
          .half  202, 204, 206, 208, 210, 212
          .half  251, 253, 255, 257, 259, 261
offset4:  .half   22,  24,  26,  28,  30,  32
          .half   71,  73,  75,  77,  79,  81
          .half  120, 122, 124, 126, 128, 130
          .half  169, 171, 173, 175, 177, 179
          .half  218, 220, 222, 224, 226, 228
          .half  267, 269, 271, 273, 275, 277
buf:            .space  200
var1:           .word   3
new_resp:       .space  2
cruiser_op:     .ascii "06c"
                .ascii "6ci"
                .ascii "cio"
                .ascii "iou"
                .ascii "17d"
                .ascii "7dj"
                .ascii "djp"
                .ascii "jpv"
                .ascii "28e"
                .ascii "8ek"
                .ascii "ekq"
                .ascii "kqw"
                .ascii "39f"
                .ascii "9fl"
                .ascii "flr"
                .ascii "lrx"
                .ascii "4ag"
                .ascii "agm"
                .ascii "gms"
                .ascii "msy"
                .ascii "5bh"
                .ascii "bhn"
                .ascii "hnt"
                .ascii "ntz"
                .ascii "012" #horizontal options
                .ascii "123"
                .ascii "234"
                .ascii "345"
                .ascii "678"
                .ascii "789"
                .ascii "cde"
                .ascii "def"
                .ascii "efg"
                .ascii "fgh"
                .ascii "ijk"
                .ascii "jkl"
                .ascii "klm"
                .ascii "lmn"
                .ascii "opq"
                .ascii "pqr"
                .ascii "qrs"
                .ascii "rst"
                .ascii "uvw"
                .ascii "vwx"
                .ascii "wxy"
                .asciiz "xyz"
destroyer_op:   .ascii "06"
                .ascii "6c"
```

```
.ascii "ci"
.ascii "io"
.ascii "ou"
.ascii "17"
.ascii "7d"
.ascii "dj"
.ascii "jp"
.ascii "pv"
.ascii "28"
.ascii "8e"
.ascii "ek"
.ascii "kq"
.ascii "qw"
.ascii "39"
.ascii "9f"
.ascii "fl"
.ascii "lr"
.ascii "rx"
.ascii "4a"
.ascii "ag"
.ascii "gm"
.ascii "ms"
.ascii "sy"
.ascii "5b"
.ascii "bh"
.ascii "hn"
.ascii "nt"
.ascii "tz"
.ascii "01" #horizontal options
.ascii "12"
.ascii "23"
.ascii "34"
.ascii "45"
.ascii "67"
.ascii "78"
.ascii "89"
.ascii "ab"
.ascii "cd"
.ascii "de"
.ascii "ef"
.ascii "fg"
.ascii "gh"
.ascii "ij"
.ascii "jk"
```

```
                .ascii "kl"
                .ascii "lm"
                .ascii "mn"
                .ascii "op"
                .ascii "pq"
                .ascii "qr"
                .ascii "rs"
                .ascii "st"
                .ascii "uv"
                .ascii "vw"
                .ascii "wx"
                .ascii "xy"
                .ascii "yz"

cruiser:        .asciiz         "\nEnter the cruiser 3x[0-9a-z]: "
cruiser_in:     .space  4
destroyer:      .asciiz         "\nEnter the destroyer 2x[0-9a-z]: "
destroy_in:     .space  3
submarine:      .asciiz         "\nEnter the submarine [0-9a-z]: "
sub_in:                 .space  2
shot:           .asciiz  "\nYour turn:\n Next shot [0-9a-z] or peek(/): "
shot_input:     .space  2
system:                 .asciiz         "\nSystem's turn:"
invalid:        .asciiz         "\nInvalid input"
dup_shot_message:.asciiz        "\nYou already went there. Try again [0-9a-z]: "
you_won:        .asciiz         "\nYou won!"
system_won:     .asciiz         "\nThe system won :("
new:            .asciiz         "\nNew game? (y/n):"
thanks:         .asciiz         "\nThanks for playing!"
.text
.globl main
#main procedure/function in program
main:
clearboard:
li   $t3, 36
li   $t7, 0
clearing:                       #clears first board
mul   $t0, $t7, 2               #offset = 2 bytes
lh    $t1, offset1($t0)         #$t1 with offset1 index
lh    $t5, offset2($t0)         #t5 with offset2 index
li    $t2, '.'                  #reset with .
sb    $t2, board($t1)           #replace with .
sb    $t2, board($t5)           #do the same thing on second board
addi  $t7, $t7, 1               #next
```

```
bne    $t3, $t7, clearing
clearsysboard:                          #clears the system board
li    $t3, 36
li    $t7, 0
clear2:                                 #clears second board
mul    $t0, $t7, 2                      #offset = 2 bytes
lh     $t1, offset3($t0)                #$t1 with offset1 index
lh     $t5, offset4($t0)                #t5 with offset2 index
li     $t2, '.'                         #reset with .
sb     $t2, sys_board($t1)              #replace with .
sb     $t2, sys_board($t5)              #do the same thing on second board
addi   $t7, $t7, 1                      #next
bne    $t3, $t7, clear2
############################         SYSTEM CRUISER          #####################
system_cruiser:
xor   $a0,$a0,$a0                               #seed number
li    $a1, 48
li    $v0, 42                                   #randomly choose number
syscall
mul   $a0, $a0, 3                               #index
li    $s5, 3                                    #counter for 3 pieces of cruiser ship
cruiser_loop:
beqz  $s5, system_destroyer                     #if counter has reached 0, all pieces have been placed
li    $t2, 'C'                                  #load piece to put on board
lb    $t0, cruiser_op($a0)                      #loads first bit of random ship choice
bgt   $t0, '9', put_letter                      #if greater than nine, it's a letter
sub   $t0, $t0, '0'                             #in num range so subtract 0 for proper index
mul   $t0, $t0, 2                               #each offset is 2 bits
lh    $t1, offset3($t0)                         #load offset # into $t1 of specified spot
sb    $t2, sys_board($t1)                       #replace . with O
add   $a0, $a0, 1                               #go through loop again for next bit of random ship
choice
sub   $s5, $s5, 1                               #count down
j     cruiser_loop
put_letter:
li    $t2, 'C'
lb    $t0, cruiser_op($a0)                      #loads first bit of random ship choice
sub   $t0, $t0, 'a'                             #in letter range, subtract 'a' for proper index
add   $t0, $t0, 10
mul   $t0, $t0, 2                               #each offset is 2 bits
lh    $t1, offset3($t0)                         #load offset # into $t1 of specified spot
sb    $t2, sys_board($t1)                       #replace . with C
add   $a0, $a0, 1
sub   $s5, $s5, 1
```

```
j     cruiser_loop
################# SYSTEM DESTROYER #################################
system_destroyer:
xor   $a0,$a0,$a0                    #seed number
li    $a1, 50
li    $v0, 42                        #randomly choose number
syscall
mul   $a0, $a0, 2                    #index
li    $s5, 2                         #counter for 3 pieces of cruiser ship
destroyer_loop:
beqz  $s5, system_submarine          #if counter has reached 0, all pieces have been placed
li    $t2, 'D'                       #load piece to put on board
lb    $t0, destroyer_op($a0)         #loads first bit of random ship choice
beq   $t0, 'C', system_destroyer     #if value is a ship, retry random #
add   $a1, $a0, 1                    #double checks next bit to make sure it doesn't
override the cruiser ship
lb    $t3, destroyer_op($a1)
beq   $t3, 'C', system_destroyer
bgt   $t0, '9', put_letter2          #if greater than nine, it's a letter
sub   $t0, $t0, '0'                  #in num :wrange so subtract 0 for proper index
mul   $t0, $t0, 2                    #each offset is 2 bits
lh    $t1, offset3($t0)              #load offset # into $t1 of specified spot
sb    $t2, sys_board($t1)            #replace . with O
add   $a0, $a0, 1                    #go through loop again for next bit of random ship
choice
sub   $s5, $s5, 1                    #count down
j     destroyer_loop
put_letter2:
li    $t2, 'D'
lb    $t0, destroyer_op($a0)         #loads first bit of random ship choice
sub   $t0, $t0, 'a'                  #in letter range, subtract 'a' for proper index
add   $t0, $t0, 10
mul   $t0, $t0, 2                    #each offset is 2 bits
lh    $t1, offset3($t0)              #load offset # into $t1 of specified spot
sb    $t2, sys_board($t1)            #replace . with C
add   $a0, $a0, 1
sub   $s5, $s5, 1
j     destroyer_loop
################# SYSTEM SUBMARINE #################################
system_submarine:                    #places submarine
xor   $a0,$a0,$a0                    #seed number
li    $a1, 36                        #set range 0-35
li    $v0, 42                        #randomly choose number
syscall
```

```
mul   $t0, $a0, 2                     # Each offset is two-byte long.
lh    $t1, offset3($t0)               # Load $t1 with the offset value (of the translated
index).
lb    $t4, sys_board($t1)             # load board piece into $t4
beq   $t4, 'D', system_submarine      #if already contains 0, retry random #
beq   $t4, 'C', system_submarine      #if already contains 0, retry random #
li    $t3, 'S'                        # Put the marker in $t3.
sb    $t3, sys_board($t1)
################ get ships from player and adds to board ################
li    $v0, 4                  #print board to start game
la    $a0, board
syscall
player_ships:                 #cruiser
li    $v0,    4               #loads space
la    $a0,    cruiser         #loads cruiser statement
syscall
la    $a0,    cruiser_in      #sets $a0 to space allocated
li    $a1,    4               #gets length of space
li    $v0,    8               #load opcode (8)
syscall                       #sees 8, asks for input, puts string in $a0
la   $t0, cruiser_in          #gets first byte from cruiser
lb   $a0, ($t0)
jal        find_spot
add  $t0, $t0, 1              #gets second byte from cruiser
lb   $a0, ($t0)
jal   find_spot
add  $t0, $t0, 1              #gets third byte from cruiser
lb   $a0, ($t0)
jal   find_spot
la   $a0, board
li   $v0, 4
syscall
li    $v0,    4               #destroyer
la    $a0,    destroyer
syscall
la    $a0,    destroy_in      #get input, store in word
li    $a1,    3
li    $v0,    8
syscall
la    $t0,    destroy_in      #gets first byte from cruiser
lb    $a0,    ($t0)
jal find_spot
add   $t0, $t0, 1             #gets second byte from cruiser
lb    $a0,    ($t0)
```

```
jal find_spot
la   $a0, board
li   $v0, 4
syscall
li    $v0,    4                        #submarine
la    $a0,    submarine
syscall
la    $a0,    sub_in
li    $a1,    2
li    $v0,    8
syscall
la    $t0,    sub_in                   #gets first byte from sub
lb    $a0,    ($t0)
jal find_spot
li    $s7,   6                         #if this counter gets down to 0, the player won!
li    $s6,   6                         #if this counter gets down to 0, the system won :(
la   $a0, board                        #prints board to begins
li   $v0, 4
syscall
################################### players turn ################
loop:
beq   $s7, 0, player_won        #branch if user has sunk all ships
beq   $s6, 0, sys_won
li    $v0,    4
la    $a0,    shot
syscall
players_turn:
la    $a0,    shot_input #gets shot input
li    $a1,    2
li    $v0,    8
syscall
la    $t0,    shot_input #gets shot byte
lb    $a0,    ($t0)
jal mark_hit
###########################system's turn ###################
system_turn:
li    $v0,4                                  # print a string
la    $a0,system                             # print statement 2
syscall
system_turn_noprint:
xor   $a0,$a0,$a0                            #seed number
li    $a1, 36                                #set range 0-35
li    $v0, 42                                #randomly choose number
syscall
```

```
mul   $s0, $a0, 2                      # Each offset is two-byte long.
mul   $t0, $a0, 2
lh    $t1, offset1($s0)                # Load $t1 with the offset value (of the translated
index).
lh    $t2, offset4($t0)                # Load $t2 with offset4 value
lb    $t4, board($t1)                  # load board piece into $t4
li    $t3, '+'                         # Put the marker in $t3.
beq   $t4, 'X', system_turn_noprint   #eliminate duplicates
beq   $t4, '+', system_turn_noprint   #eliminate duplicates
beq   $t4, '0', player_ship_hit        # if value is a ship, replace with X instead
sb    $t3, board($t1)                  # Put the marker in the offset index player left board
sb    $t3, sys_board($t2)              # Put the marker in system right board
la    $a0, board
li    $v0, 4
syscall
j     loop                             #jumps back to ask for another shot again
player_ship_hit:
li   $t5, 'X'
sb   $t5, board($t1)
sb   $t5, sys_board($t2)
sub  $s6, $s6, 1
la   $a0, board
li   $v0, 4
syscall
j    system_turn
################### Places shots when player takes turn #######
mark_hit:
blt  $a0, '0', mark_letter        #if less than 0, not a number. tests to see if letter
bgt  $a0, '9', mark_letter        #if greater than 9, not a number. tests to see if letter
sub  $s0, $a0, '0'                #otherwise, subtracts difference for ascii value
sub  $s1, $a0, '0'                #for board 2
mul  $s0, $s0, 2                  # Each offset is two-byte long.
mul  $s1, $s1, 2
lh   $t1, offset2($s0)            # Load $t1 with the offset value (of the translated index).
lh   $t2, offset3($s1)            # '' for board 2
lb   $t4, sys_board($t2)          # load board piece into $t4
li   $t3, '+'                     # Put the marker in $t3.
beq  $t4, 'D', ship_hit           # if value is a ship, replace with X instead
beq  $t4, 'C', ship_hit           # if value is a ship, replace with X instead
beq  $t4, 'S', ship_hit           # if value is a ship, replace with X instead
beq  $t4, '+', dup_shot           # if value is a ship, replace with X instead
sb   $t3, board($t1)              # Put the marker in the offset index player left board
sb   $t3, sys_board($t2)          # Put the marker in system right board
la   $a0, board
```

```
li   $v0, 4
syscall
jr   $ra
mark_letter:
blt   $a0, 'a', invalid_input      #if v0 is less than a, not a letter.
bgt   $a0, 'z', invalid_input     #if v0 is more than z, not a letter.
sub   $s0, $a0, 'a'                     #otherwise, subtract/add the difference of ascii value
sub   $s1, $a0, 'a'
add   $s0, $s0, 10
add   $s1, $s0, 0                       #don't need to add 10 for the location
mul   $s0, $s0, 2           # Each offset is two-byte long.
mul   $s1, $s1, 2
lh    $t1, offset2($s0)              # Load $t1 with the offset at the index $s0.
lh    $t2, offset3($s1)              # Load $t2 with offset of the left sys board at index $s1
lb    $t4, sys_board($t2)
li    $t3, '+'                  # Put the marker in $t2.
beq  $t4, 'D', ship_hit             # if value is a ship, replace with X instead
beq  $t4, 'C', ship_hit             # if value is a ship, replace with X instead
beq  $t4, 'S', ship_hit             # if value is a ship, replace with X instead
beq  $t4, '+', dup_shot            # if value is a ship, replace with X instead
sb    $t3, board($t1)
sb    $t3, sys_board($t2)
la    $a0, board
li    $v0, 4
syscall
jr    $ra
invalid_input:
beq   $a0, '/', peek
la    $a0, invalid   #prints board again
li    $v0, 4
syscall
jr    $ra
peek:
la    $a0, sys_board
li    $v0, 4
syscall
jr    $ra
la    $a0, board    #prints board again
li    $v0, 4
li    $v0, 10          #exits
syscall
ship_hit:
li   $t5, 'X'
sb   $t5, board($t1)
sb   $t5, sys_board($t2)
sub  $s7, $s7, 1
la   $a0, board
li   $v0, 4
syscall
j loop
############### places players ships ##########################################
find_spot:
blt  $a0, '0', not_number       #if less than 0, not a number. tests to see if letter
bgt  $a0, '9', not_number       #if greater than 9, not a number. tests to see if letter
sub  $s0, $a0, '0'              #otherwise, subtracts difference for ascii value
mul  $s0, $s0, 2                # Each offset is two-byte long.
lh   $t1, offset1($s0)          # Load $t1 with the offset of the index $t0.
li   $t2, '0'                   # Put the marker in $t2.
sb   $t2, board($t1)
jr   $ra
not_number:
blt   $a0, 'a', not_letter      #if v0 is less than a, not a letter.
bgt   $a0, 'z', not_letter      #if v0 is more than z, not a letter.
sub   $s0, $a0, 'a'             #otherwise, subtract/add the difference of ascii value
add   $s0, $s0, 10
mul   $s0, $s0, 2               # Each offset is two-byte long.
lh    $t1, offset1($s0)         # Load $t1 with the offset of the index $t0.
li    $t2, '0'                  # Put the marker in $t2.
sb    $t2, board($t1)
jr    $ra
not_letter:
la     $a0, board               #prints board again
li     $v0, 4
syscall
jr     $ra
la     $a0, board               #prints board again
li     $v0, 4
li     $v0, 10
syscall
######################## check for duplicates #####################
dup_shot:
la    $a0, dup_shot_message
li    $v0, 4
syscall
j      players_turn
check_dup:
li          $a0,0                          # set 'boolean' value
```

```
lh        $t1, offset4($t0)              # Load $t1 with offset of position we are checking
(offset4).
lb        $t2, board($t1)
beq       $t2, '.', not_dup              # if not duplicate, quit with $a0 = 0
li        $a0,1                          # else, $a0 = 1
not_dup:
jr $31 # done
######################## end of game stuff ########################
player_won:
la   $a0, you_won
li   $v0, 4
syscall
j newgame
sys_won:
la   $a0, system_won
li   $v0, 4
syscall
j newgame
######################### new game ################################
newgame:
la   $a0, new
li   $v0, 4
syscall
li   $v0, 8
la   $a0, new_resp
li   $a1, 2
syscall
lb   $t0, new_resp
li   $t1, 'y'
beq  $t0, $t1, clearboard                #if user enters y, new game. go back to top
bne  $t0, $t1, thankyou                  #else, print thank you and quit
thankyou:
la   $a0, thanks                         #prints thank you
li   $v0, 4
syscall
li   $v0, 10                             #quits game
syscall
```