.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"

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

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 "\nEnter the next shot [0-9a-z]: "

shot\_input: .space 2

new: .asciiz "\nNew game? (y/n):"

buf: .space 200

var1: .word 3

new\_resp: .space 2

thanks: .asciiz "\nThanks for playing!"

.text

.globl main

#main procedure/function in program

main:

clearboard:

li $t3, 36

li $t7, 0

#clears first board

clearing:

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

#print board

li $v0, 4

la $a0, board

syscall

#get ships and add to board

#cruiser

li $v0, 4 #loads space

la $a0, cruiser #loads cruiser statement

syscall

#get input, store in word

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

#gets first byte from cruiser

la $t0, cruiser\_in

lb $a0, ($t0)

jal find\_spot

#gets second byte from cruiser

add $t0, $t0, 1

lb $a0, ($t0)

jal find\_spot

#gets third byte from cruiser

add $t0, $t0, 1

lb $a0, ($t0)

jal find\_spot

#destroyer

li $v0, 4

la $a0, destroyer

syscall

#get input, store in word

la $a0, destroy\_in

li $a1, 3

li $v0, 8

syscall

#gets first byte from cruiser

la $t0, destroy\_in

lb $a0, ($t0)

jal find\_spot

#gets second byte from cruiser

add $t0, $t0, 1

lb $a0, ($t0)

jal find\_spot

#submarine

li $v0, 4

la $a0, submarine

syscall

la $a0, sub\_in

li $a1, 2

li $v0, 8

syscall

#gets first byte from sub

la $t0, sub\_in

lb $a0, ($t0)

jal find\_spot

li $s7, 6

loop:

beq $s7, 0, newgame #branch if user has sunk all ships

#shot

li $v0, 4

la $a0, shot

syscall

#gets shot input

la $a0, shot\_input

li $a1, 2

li $v0, 8

syscall

#gets shot byte

la $t0, shot\_input

lb $a0, ($t0)

jal mark\_hit

j loop #jumps back to ask for another shot again

# Exit the program.

li $v0, 10

syscall

#place spot for shot on board 2

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 $t1, offset1($s0) # Load $t1 with the offset value (of the translated index).

lh $t2, offset2($s1) # '' for board 2

lb $t4, board($t1) # load board piece into $t4

li $t3, '+' # Put the marker in $t2.

beq $t4, '0', ship\_hit # if value is a ship, replace with X instead

sb $t3, board($t1) # Put the marker in the offset index on the board

sb $t3, board($t2) # Put the marker in board 2

la $a0, board

li $v0, 4

syscall

jr $ra

ship\_hit:

li $t5, 'X'

sb $t5, board($t1)

sb $t5, board($t2)

sub $s7, $s7, 1

la $a0, board

li $v0, 4

syscall

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, 36

mul $s0, $s0, 2 # Each offset is two-byte long.

mul $s1, $s1, 2

lh $t1, offset1($s0) # Load $t1 with the offset at the index $s0.

lh $t2, offset1($s1)

lb $t4, board($t1)

li $t3, '+' # Put the marker in $t2.

beq $t4, '0', ship\_hit

sb $t3, board($t1)

sb $t3, board($t2)

la $a0, board

li $v0, 4

syscall

jr $ra

invalid\_input:

la $a0, board #prints board again

li $v0, 4

syscall

jr $ra

la $a0, board #prints board again

li $v0, 4

li $v0, 10 #exits

syscall

############### places ship on board 1, do not need to adjust ##################

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

#add to board

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)

la $a0, board

li $v0, 4

syscall

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)

la $a0, board

li $v0, 4

syscall

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

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