Relatório Batalha Naval

Disciplina: Organização de Computadores

Professor: Luciano L. Caimi

Alunos: Eduardo Barcaroli e Pedro Manfio Lill

1.Problema

O principal problema enfrentado, foi a de trabalhar em cima de uma linguagem que não é utilizada no nosso dia a dia e que trabalha com funções e variáveis desconhecidas até 1 ou 2 meses atrás.

Mas entre alguns outros problemas não resolvidos, os maiores foram a parte de atirar no navio, de resetar a matriz e de placar de recordes antigos. E dentre os problemas resolvidos, tivemos a função insere\_embarcacoes, implementar a matriz visual para o jogador, implementação das mensagens e validações das matrizes, dentre outros problemas que foram resolvidos ao longo do desenvolvimento do projeto.

Na parte do tiro foi mais complicado a parte de salvar os recordes de tiro e a contabilização de tiros dados, também não conseguimos fazer a matriz resetar após o jogo terminar e por fim, não conseguimos contabilizar os melhores resultados e fazer uma tabela de recordes do jogador.

2.Solução

Muitos dos problemas foram resolvidos de maneira muito simples com ajuda de funções da própria linguagem e um pouco de noção de programação, mas o problema da função insere\_embarcacoes por exemplo, foi resolvido com um simples for que percorria os valores iniciais de todos os barcos e ia interpretando se as posições estavam corretas após passarem por um if para fazer toda a validação de posicionamento e adicionar esse valor na matriz.

O segundo problema a ser resolvido foi o da matriz a ser mostrada para o jogador, que apos a gente tirar algumas duvidas com o professor nas monitorias foi resolvida parcialmente, mostrando uma matriz completa de 10x10 feita de @, porem a cada jogada ele nao salva a posição anterior deixando a matriz resetada a cada jogada (visualmente, mas internamente funcionando corretamente).

O terceiro problema a ser resolvido foi a questão das validações, onde tivemos que recorrer ao uso de muitos ifs para solucionar o problema de forma rápida, utilizamos ifs e elses recorrentes para fazer a mesmo validação na maioria das vezes, sendo de grande importância para o nosso código como um todo.

Os outros problemas foram resolvidos de uma maneira mais geral utilizando muito o guia prático e as monitorias disponibilizadas pelo professor Caimi e pelo monitor da disciplina.

3.Conclusão

Após muitas horas focadas no desenvolvimento desse trabalho, muitos problemas e soluções encontradas, concluímos que desenvolver em assembly pode ser um desafio tanto por ser uma linguagem muito recente no nosso repertório de desenvolvimento, quanto por ter que procurar solucoes em um ambiente desconhecido, porem a experiencia eh muito gratificante para ambos e desenvolve uma percepção de desenvolvimento que nós ainda não tínhamos atingido. Notamos que muitos problemas grandes podem ser resolvidos de maneira bem simples se usarmos a lógica de programação que nos foi ensinado anteriormente no curso e que muitos problemas que aparentavam ser pequenos demandaram muito mais tempo e dedicação que o esperado.

4.Programa

# Os navios do inimigo est?o colocados em uma string chamada ?navios?? presente na ?rea

# de dados (.data), que deve ser lida pela fun??o insere\_embarcacoes no in?cio do jogo. A

# string navios possui o seguinte padr?o. Na primeira linha ? informado o n?mero de navios

# inseridos. Cada uma das linhas seguintes possui um navio. As linhas que especificam navios

# possuem 4 valores, separados por um espa?o, sendo: o primeiro valor ? a disposi??o do navio

# sendo, 0 para navio na horizontal e 1 para navio na vertical; o segundo valor ? o comprimento do

# navio; o terceiro valor ? a linha inicial do navio e; o quarto valor ? a coluna inicial do navio.

# Observe o Exemplo:

# 3 -> numero de navios inseridos

# 1[vertical] 5[comprimento] 1[linha inicial] 1[coluna inicial] ->navio

# 0[horizontal] 5[comprimento] 2[linha inicial] 2[coluna inicial] ->navio

# 0[horizontal] 1[comprimento] 6[linha inicial] 4[coluna inicial] ->navio

# O posicionamento de navios mostrado acima resulta no seguinte posicionamento:

# A fun??o insere\_embarcacoes deve verificar a validade do posicionamento dos navios,

# gerando uma mensagem de erro para as seguintes situa??es:

#

.data

matriz: .word 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

matriz\_jogo: .word 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

navios: .string "3 1 5 0 0 0 5 2 2 0 1 6 4 "

br\_n: .string "\n"

digLinha: .string "\n Digite a linha: \n"

digColuna: .string "Digite a coluna: \n"

msgCpm: .string "O navio extrapola as dimensoes da matriz"

msgPosCol: .string "A posicao do navio eh invalida (coluna)"

msgPosLin: .string "A posicao do navio eh invalida (linha)"

msgSobrepos: .string "Ocorre sobreposicao nos navios"

interativo: .string "\n\n Escolha uma das opcoes: \n 0-reiniciar o jogo \n 1-estado atual da matriz de navios \n 2-fazer uma nova jogada \n "

# atirando: .string "\n Escolha em que posicao ira atirar:\n"

recorde: .string "\n --------- Recorde --------- \n"

recTiro: .string "Tiros : \n"

recAcertos: .string "Acertos : \n"

recAfundados: .string "Afundados : \n"

voce: .string "\n --------- Voce --------- \n"

vcTiros: .string "\n Tiros : \n"

vcAcertos: .string "\n Voce acertou!! \n"

vcErros: .string "Voce errou!! \n"

vcAfundados: .string "Afundados : \n"

vcUltimo: .string "Ultimo tiro : \n"

# matrizJogo: .word O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, O

space: .string " "

.text

main:

# jal identifica\_qtd # jal para identifica\_qtd

# add s0, zero, s0

# O = 79 (ascii)

jal insere\_embarcacoes

jal tela\_inicial

jal printa\_matriz

jal fim

insere\_embarcacoes:

# a insere\_embarcacoes faz a leitura da string navios. de forma semantica ela envia os numeros para o devido lugar

# e insere as embarcacoes de forma correta na matriz

# sao feitos testes de sobreposicao, e de extrapolar a matriz

la a0, navios # carrega navio em a0 -> navios: .string "3 (1 5 1 1) 0 5 2 2 0 1 6 4"

lb t4, 0(a0) # carrega navios em t4

addi t4, t4, -48 # a0/t4 => 3 cod ascci 0/ trasnformar a string 3 em int 3

li t3, 0 # carrega 0 em t3

li t5, 4 # carrega 4 em t5

li t6, 10 # carrega 10 em t6

li t0, 11 # carrega 11 em t0

li s7, 32 # carrega espaco em s7

addi s9, zero, 1 # contador de navios

teste\_1:

# lb t4, 0(a0) # carrega navios em t4

# addi t4, t4, -48 # a0/t4 => 3 cod ascci 0/ trasnformar a string 3 em int 3

beq t4, zero, end\_1 # identifica\_qtd

corpo\_1:

la s10, matriz # carrega matriz em s10

# horizontal ou vertical

addi a0, a0, 2 # horizontal[0] ou vertical[1]

lb s0, (a0) # direcao em s0

addi s0, s0, -48 # transforma a string em inteiro(ascii)

# comprimento

addi a0, a0, 3 # numero 3(4) da string

# addi a0, a0, 1 # ando uma casa em a0 e guardo em s8

lb s4, (a0)

bne s4, s7, msg\_cpm # se a pos em a0(s8) for diferente de space -> msgCpm

addi a0, a0, -1 # ando uma casa em a0 e guardo em s8

lb s1, (a0) # salvo a pos de a0 em s1

addi s1, s1, -48

# linha inicial

addi a0, a0, 3 # eu ando ate a posicao depois do numero (que deveria ser um espaco)

lb s5, (a0) # salvo em um reg temp

bne s5, s7, msg\_posLin # teste se ela ? diferente de espa?o

addi a0, a0, -1 # volto para a posicao que tem o numero

lb s2, (a0) # salvo em s2

addi s2, s2, -48 # transformo em int

# teste sobrepos coluna

add a4, s2, s1 # comprimento+linha inicial

bge a4, t0, msg\_posCol # verifica se eh >= 11

# coluna inicial

addi a0, a0, 3 # eu ando at? a posicao depois do numero (que deveria ser um espaco)

lb s6, (a0) # salvo em um reg temp

bne s6, s7, msg\_posCol # teste se ela ? diferente de espa?o

addi a0, a0, -1 # volto para a posicao que tem o numero

lb s3, (a0) # salvo em s3

addi s3, s3, -48 # transformo em int

# teste sobrepos linha

add a3, s3, s1 # comprimento+coluna inicial

bge a3, t0, msg\_posLin # verifica seeh >= 11

# (L \* QTD\_colunas + C) \* 4

mul s11, s2, t6 # l \*qtd colunas

add s11, s11, s3 # l \*qtd colunas + C

mul s11, s11, t5 # (l \*qtd colunas + C) \* 4

add s10, s10, s11 # s10 esta com a pos atual na matriz

teste\_2:

beq s1, zero, incremento\_1 # s8-> count\_comprimento. s1 comprimento

corpo\_2:

lb t1, 0(s10) # ponteiro em t1 para s10

bne t1, t3, msg\_sobrepos # se a posicao atual for diferente de 0 -> msg\_sobrepos

sw s9, (s10) # salva o numero da embarcacao atual na matriz

incremento\_2:

addi s1, s1, -1 # decremento um no comprimento

beq s0, t3, horizontal # se s0 = 0 entao eh horizontal

j vertical # se nao ? vertical

horizontal:

addi s10, s10, 4 # add o numero do barco de 4 em 4 pos

# add s5, zero, ra

# jal funcao

# add ra, zero, s5

j teste\_2

vertical:

addi s10, s10, 40 # add 40 para andar 10 pos na matriz

j teste\_2

incremento\_1:

addi s9, s9, 1 # incremento 1 em s9(numero da embarca??o)

addi t4, t4, -1 # decremento 1 em t4 (numero de navios)

j teste\_1

end\_1:

ret

# mensagens de erro do programa

msg\_cpm:

la a0, msgCpm

li a7, 4

ecall

j fim

msg\_posCol:

la a0, msgPosCol

li a7, 4

ecall

j fim

msg\_posLin:

la a0, msgPosLin

li a7, 4

ecall

j fim

msg\_sobrepos:

la a0, msgSobrepos

li a7, 4

ecall

j fim

printa\_matriz: # funcao do alex

# o printa matriz tem um loop de 10 em 10 e de 100 em 100

# e printa de uma em uma no formato de matriz 10x10

add t0, zero, zero # quando chegar em 100, termina

addi t1, zero, 100

add t2, zero, zero # a cada 10, um \n

addi t3, zero, 10

la a1, matriz

teste\_condicao\_prin:

beq t0, t1, fim\_prin

beq t2, t3, pula\_prin

j corpo\_laco\_prin

pula\_prin:

add t2, zero, zero

li a0, 10

li a7, 11

ecall

corpo\_laco\_prin:

# addi a0, a0, -12

lw a0, (a1)

li a7, 1

ecall

li a0, 32 # 32 = space (tabela ascii)

li a7, 11 # 11 = \n (tabela ascii)

ecall

incremento\_controle\_prin:

addi a1, a1, 4

addi t0, t0, 1

addi t2, t2, 1

j teste\_condicao\_prin

fim\_prin:

ret

printa\_matriz\_pronta:

# a printa matriz eh a matriz interativa do jogo, nela verificamos o valor na matriz e substituimos de acordo

# se s5 = 0 => @(arroba), se s5=10=>x, se s5>10=>valor da matriz

# ainda temos um bug(ou melhor, um erro), só printamos a matriz interativa com o tiro atual

# ou seja ela mostra muitos @(arrobas) e um x ou o valor inserido na matriz

# acredito que um dos testes nao esta feito corretamente, e tambem acredito que deveriamos fazer menos

# comparacoes dentro dessa funcao.Alguns dos testes e valores deveriam estar prontos e apenas serem processados aqui

# Infelizmente nao conseguimos resolver isso

# atualmente o teste feito eh somente em relação a posicao s5(o tiro), logo o print nao assimila de forma coesa

# as outras informacoes da matriz

add t0, zero, zero

addi t1, zero, 100

add t2, zero, zero # a cada 10, um \n

addi t3, zero, 10

add s6, zero, zero

addi s7, zero, 1

la a1, matriz

la a0, recorde

teste\_condicao\_prin\_pronta:

beq t0, t1, fim\_prin\_pronta

beq t2, t3, pula\_prin\_pronta

j corpo\_laco\_prin\_pronta

pula\_prin\_pronta:

add t2, zero, zero

li a0, 10

li a7, 11

ecall

corpo\_laco\_prin\_pronta:

beq s10, a1, loop\_pronta # se a pos s5 for igual a s10

printa\_trac:

li a0, 64

li a7, 11

ecall

li a0, 32 # 32 = space (tabela ascii)

li a7, 11 # 11 = \n (tabela ascii)

ecall

j incremento\_controle\_prin\_pronta

loop\_pronta: # teste entre a word s5 e 10

blt t3, s5 printa\_navio # maior i=ou igual 10. 193

beq s5, t3, printa\_x # se s5 = 10 printo o x

printa\_navio:

lw a0, (a1) # grava/printa na matriz

li a7, 1

ecall

li a0, 32 # 32 = space (tabela ascii)

li a7, 11 # 11 = \n (tabela ascii)

ecall

j incremento\_controle\_prin\_pronta

printa\_x:

li a0, 88 # print @ (Arroba)

li a7, 11

ecall

li a0, 32 # 32 = space (tabela ascii)

li a7, 11 # 11 = \n (tabela ascii)

ecall

j incremento\_controle\_prin\_pronta

incremento\_controle\_prin\_pronta:

# incrementos para o loop

addi a1, a1, 4

addi t0, t0, 1

addi t2, t2, 1

addi s6, s6, 1

j teste\_condicao\_prin\_pronta

fim\_prin\_pronta:

ret

# essa funcao eh a tela inicial do nosso jogo

# a partir dela temos 3 opcoes

# 0- reiniciar o jogo. A ideia era reiniciar/zerar a matriz, mas a funcao esta incompleta

# 1- mostrar estado atual da matriz. Escolhendo a opcao 1 podemos ter a visao do estado da matriz. a funcao esta correta

# porem com um pequeno bug, nao criamos uma condicao que impedisse somar duas vezes o valor na mesma posicao

# 2- aqui eh onde realizamos os tiros. eh feito uma comparaca entre a soma das posicoes do tiro (L \* QTD\_colunas + C) \* 4)

# e a mesma posicao da matriz. Se o valor for maior que 0, significa que acertamos. caso o valor seja igual a 0 erramos o tiro

# acreditamos que no rotulo tela\_inicial poderiamos ter feito mais testes, e criado outras condicoes para nao ter que

# fazer tantos testes no printa\_matriz\_pronta

tela\_inicial:

la s10, matriz # carrego a matriz em s10

# load immediate em valores usados posteriormente

li t0, 0

li t1, 1

li t2, 2

li t3, 0

li s4, 3

li t5, 4

li t6, 10

li s7, 0

add s1, zero, zero # zero o s1, e uso para linha

add s2, zero, zero # zero o s2, e uso para coluna

display\_interativo:

la a0, interativo # chamada do display

li a7, 4 # 4 -> printstring

ecall

li a7, 5 # 5-> readInt

ecall

teste\_tela1:

# testes do valor escrito pelo usuario

beq a0, t0, reiniciar

beq a0, t1, estAtual

beq a0, t2, atirar

bge a0, s4, display\_interativo # teste

reiniciar:

jal tela\_inicial

estAtual:

jal printa\_matriz

j tela\_inicial

atirar:

la s10, matriz

la a0, digLinha

li a7, 4

ecall

li a7, 5 # 5-> readInt

ecall

add s1, a0, zero

# li a0, 10 # usado para dar \n

# li a7, 11 # \n ascii

# ecall

la a0, digColuna

li a7, 4

ecall

li a7, 5 # 5-> readInt

ecall

add s2, a0, zero

# (L \* QTD\_colunas + C) \* 4

mul s11, s1, t6 # l \*qtd colunas

add s11, s11, s2 # l \*qtd colunas + C

mul s11, s11, t5 # (l \*qtd colunas + C) \* 4

add s10, s10, s11 # s10 esta com a pos atual na matriz

lb s5, 0(s10) # ponteiro s5 para a posicao atual da matriz carregando o byte

testeTiro:

bne s5, t0, acertou # comparo o byte da matriz com 0(t0), se for diferente = acertei

beq s5, t0, errou # se s5 = t0, errei

acertou:

la a0, vcAcertos # carrega a string

li a7, 4

ecall

addi s5, s5, 10 # adiciona 10, entao nosso valor sera maior que 10

sw s5, (s10) # armazeno isso

addi s7, s7, 1 # esse era pra ser um contador de tiro

jal printa\_matriz\_pronta # print da matriz

j tela\_inicial # volta pra tela inicial

errou:

la a0, vcErros # carrega a string

li a7, 4

ecall

addi s5, s5, 10 # valor sera igual a 10

sw s5, (s10) # salvo

jal printa\_matriz\_pronta # print com um x no meio dos arrobas

j tela\_inicial

endII:

ret

fim:

nop