

Universidade Federal da Fronteira Sul
Componente Curricular: GEX612 - Organização de computadores
Ano/semestre: 2021.1
Docente: Luciano L. Caimi
Discentes: Leonardo Hideaki Nakamichi De Lima;
Nícolas Ruwer Hackenhar.

Trabalho de Programação Assembly - RISC-V: Jogo de batalha naval

01 DE OUTUBRO DE 2021

1. Problema:

O trabalho a ser implementado é o controle do jogo de batalha naval em uma matriz 10x10 utilizando o conjunto de instruções do processador RISC-V RV32IM. O jogo de batalha naval consiste em um jogo cujo o objetivo é afundar os navios do adversário, nos quais estarão em uma matriz. Com cada linha e coluna da matriz sendo representadas por números de 0 à 9, nas suas respectivas ordens.

Um dos problemas encontrados durante o desenvolvimento foi na atribuição do vetor de navios, e em seguida no vetor de tiros, onde ao inicializar o vetor de navios sobre escrevia o endereço do vetor de tiros.

Foi enfrentado outro problema na leitura dos métodos da jogabilidade, ao verificar se na posição selecionada pelo jogador havia um navio ou não. Pois, o endereço era lido da matriz de navios, mas era necessário alterar na matriz de tiros.

Outra dificuldade foi o fato de encerrar o jogo no momento que o jogador acertasse a última posição do último navio restante.

2. Solução:

Para realizar o trabalho, os métodos foram codificados na linguagem assembly, utilizando o conjunto de instruções RISC-V e com auxílio do software RARS na versão 1.5.

Trabalhamos no formato de pair programming e utilizando o github para o controle de versionamento das alterações.

Para solucionar a inicialização das matrizes, sem que fosse sobre posto o endereço inicial de um dos vetores, foi adicionado ao endereço armazenado no registrador (s7), que contém a matriz de tiros, o valor do contador contido em (a3) que no final da função inicializa a matriz, contém o último endereço pertencente a matriz de navios, gerando um deslocamento da posição inicial da matriz de tiros para 4 bytes a frente da matriz de navios.

ANEXAR IMAGEM DA FUNÇÃO

Na solução do segundo problema, na função `get_jogada`, foi adicionado um deslocamento de 400 bytes na posição calculada com base na linha e coluna inicial, informadas pelo jogador ao efetuar uma nova jogada, armazenado no registrador (a4) possibilitando assim, validar se nessa posição havia uma embarcação na matriz de navios e alterar a posição referente na matriz de tiros, conforme mostra na imagem abaixo.

```

get_jogada:
    la a0, msg_linha_jogada          # carrega em a0 a string da linha da jogada
    li a7, 4                          # informa impressão de string
    ecall                             # chama impressão

    addi a7, zero, 5                  # informa leitura de inteiro
    ecall                             # chama leitura
    add a1, zero, a0                  # carrega valor lido em a1

    blt a1, zero, jogada_invalida     # valida se a1 é menor que 0, então informa que a jogada é inválida
    addi t5, zero, 9                  # adiciona 9 ao registrador t5
    bgt a1, t5, jogada_invalida       # valida se a1 é maior que 9, então informa que a jogada é inválida

    la a0, msg_coluna_jogada          # carrega em a0 a string da coluna da jogada
    li a7, 4                          # informa impressão de string
    ecall                             # chama impressão

    addi a7, zero, 5                  # informa leitura de inteiro
    ecall                             # chama leitura
    add a2, zero, a0                  # carrega valor lido em a2

    blt a2, zero, jogada_invalida     # valida se a2 é menor que 0, então informa que a jogada é inválida
    addi t5, zero, 9                  # adiciona 9 ao registrador t5
    bgt a2, t5, jogada_invalida       # valida se a2 é maior que 9, então informa que a jogada é inválida

    addi t5, zero, 10                 # adiciona a t5 o número de colunas
    mul a3, a1, t5                     # multiplica a linha inicial pelo número de colunar
    add a3, a3, a2                     # adiciona a coluna inicial pela multiplicação anterior
    addi a4, zero, 4                  # carrega 4 ao registrador a4
    mul a3, a3, a4                     # multiplica por 4 o valor do cálculo da posição
    add a4, s0, a3                     # adiciona ao registrador a4 o endereço da matriz de navios mais o deslocamento calculado

    addi a6, zero, 42                  # carrega o valor do caractere asterisco ao registrado a6
    lbu a5, (a4)                       # carrega em a5 o valor da posição da jogada na matriz de navios
    addi a4, a4, 400                    # desloca a posição da jogada para a matriz de tiros
    lbu t6, (a4)                       # carrega em t6 o valor da posição da jogada na matriz de tiros
    addi a7, zero, 111                  # carrega em a7 o valor do caractere o
    beq t6, a7, jogada_invalida        # verifica se o valor carregado da posição da matriz de tiros já foi alterado para o
    bne a5, a6, tiro_certo              # verifica se o valor carregado da matriz de navios é diferente do caractere asterisco
    sb a7, (a4)                         # armazena o caractere o na posição da matriz de tiros
    add a4, zero, s7                     # adiciona em a4 a posição inicial da matriz de tiros
    jal imprime_matriz                  # chama a impressão da matriz de tiros
    j menu                             # volta para o menu

```

Com relação a finalização do jogo, foi adicionado um contador de tiros certos, baseado no comprimento dos navios inseridos, o mesmo era atribuído e incrementado na função `loop_navio_especs`, pelo registrador (t2) e retornado no registrador (s3), na função `fim_instrucoes`. Dessa forma foi possível comparar este contador com o contador de tiros certos atribuído e incrementado na função `tiro_certo`, para quando fossem iguais, direcionassem a execução do programa à função `winner`.

ANEXAR IMAGEM DAS FUNÇÕES

3. Conclusão:

Uma das dificuldades encontradas, foi no gerenciamento dos registradores, uma vez que os mesmo possuem uma quantia limitada para seu uso no armazenamento de valores, tornando necessário o cuidado para não substituir um valor armazenado em um registrador que fosse necessário o uso em outra função.

Por ser uma linguagem de baixo nível, a lógica de implementação da solução para o problema proposto precisa ser explicitada no conjunto de instruções disponível, onde não encontramos funções básicas que em outras linguagens de programação facilitam o desenvolvimento de uma solução.

```

.data
navios:                                     .asciz "3\n1 5 1 1\n0 5 2 2\n0 1 6 4"
msg_qtde_invalida:                         .asciz "Quantidade não pode ser 0 ou menor.\n"
msg_instrucao_invalida:                    .asciz "ERRO: Instrução fora do formato
esperado.\n"
msg_posicao_do_navio_invalida: .asciz      "ERRO:      Sobreposição      de
embarcações ou Tamanho da embarcação inválido para as coordenadas de
inserção!\n"
msg_jogada_invalida:                      .asciz "ERRO: Disparo já efetuado na
posição informada ou a mesma está fora do tabuleiro\n\n\n"
msg_inicio:                               .asciz "\n
      ..::!                               \n
      .###P                               \n
      .":   :####L   ...               \n
      .##   P'   :   \n
      'P':.. .#####P'..P             \n
      #####'.#####P'.####'          \n
      :##### :#####: :####:         \n
      ^_ZP"P## :#####L' "   !   ..   \n
      'q_#_   P#####_   _###         \n
      '#####_ "   ||   "   #####     \n
      '#####_.._||_..#####          \n
      #####:                           \n
      '#####P"P"                     \n
      .....                               \n
      \n"
msg_titulo:                               .asciz "\n#      #          #####
\n## # ## # # ## #      # # ## ##### ##### #      ##### \n# # # #
# # # # #      # # # # # # # # # \n# # ## ## ## ## ##

```

```
##### # # # # # ##### \n# # # ##### # # ##### # # #
##### # # # # # \n# ### # # # # # # # # # # # #
# \n# # # # # # # ##### ##### # # # # # ##### #####
\n\n\n By Leonardo Nakamichi & Nicolas Hackenhar\n\n\n\n"
```

```
msg_linha_jogada: .asciz "\nInforme a posição relativa a linha da
jogada\n"
```

```
msg_coluna_jogada: .asciz "\nInforme a posição relativa a coluna
da jogada\n"
```

```
string_colunas: .asciz "\n 0 1 2 3 4 5 6 7 8 9\n"
```

```
string_menu: .asciz "0 - Novo Jogo\n1 - Nova jogada\n2 -
Sair\n\nEscolha uma opção: \n\n"
```

```
string_winner: .asciz "          Você
venceu!\n\n\n0 - Novo Jogo\n1 - Sair\n\nEscolha uma opção: \n\n"
```

```
string_init: .asciz "0 - Novo Jogo\n1 - Sair\n\nEscolha uma
opção: \n\n"
```

```
matriz: .word 100
```

```
matriz_player: .word 100
```

```
.text
la a0, msg_inicio # carrega desenho ASCII
embarcação
```

```
li a7, 4 # informa impressão de string
ecall # chama a impressão
```

```
la a0, msg_titulo # carrega desenho ASCII do
título
```

```
li a7, 4 # informa impressão de string
ecall # chama a impressão
```

```
la a0, string_init # carrega string do menu de
inicialização
```

```
li a7, 4 # informa impressão de string
ecall # chama impressão
```

```
addi a7, zero, 5 # informa leitura de inteiro
ecall # chama leitura
```

```
add a1, zero, a0 # carrega valor lido em a1
```

```
addi a2, zero, 0 # carrega 0 em a2
```

```
beq a1, a2, start # valida se a1 for 0, então
direciona para start
```

```
addi a2, zero, 1 # carrega 1 em a2
```

```
                beq      a1, a2, fim                # valida se a1 for 1, então
direciona para o fim
```

```
                j        fim                        # chama fim da execução
```

```
#####
#####
```

```
    # função: start
    #      função responsável pela inicialização do controle de início do jogo,
carregando os valores
    #      utilizados na inicialiação das matrizes e inserção das embarcações.
    # entrada: não recebe entradas
    # saída: não retorna nenhum valor
#####
#####
```

```
    start:
                la      a1, navios                # carrega em s1 a string
navios
                la      s0, matriz                # carrega em s0 a matriz
do jogo
                la      s7, matriz_player          # carrega em s7 a matriz
do jogador
```

```
                add     a4, zero, s0              # atribui a a4 a posição
inicial da matriz do jogo
                jal      inicializa_matriz         # chama a função de
inicialização da matriz
                add     s7, zero, a3              # move s7 para a posição
final da matriz já inicializada
```

```
                add     a4, zero, s7              # atribui a a4 o endereço
inicial da matriz de tiros
                jal      inicializa_matriz
```

```
                jal      insere_embarcacoes        # chama a função
para inserir as embarcações
```

```
                add     a4, zero, s7              # atribui a s4 o endereço
inicial da matriz de tiros
                jal      imprime_matriz            # chama a função
para imprimir a matriz
```

```
        jal    loop_jogo                # chama a função para
iterar o jogo
```

```
        ret
```

```
#####
#####
```

```
    # função: loop_jogo
    #    função responsável por controlar o jogo em execução após o mesmo
ser iniciado
```

```
    # entrada: não recebe entradas
```

```
    # saída: não retorna nenhum valor
```

```
#####
#####
```

```
    loop_jogo:
```

```
        add    t5, zero, zero          # zera o valor do
registrador t5
```

```
        addi   t6, zero, 100           # atribui a t6 o valor 100
```

```
        add    s4, zero, zero          # zera o valor do
registrador s4
```

```
        jal    menu                    # chama a função menu
```

```
        ret
```

```
#####
#####
```

```
    # função: menu
    #    função responsável por exibir as opções ao usuário e realizar a
chamada das funções
```

```
    #    referentes a opção selecionada.
```

```
    # entrada: não recebe entradas
```

```
    # saída: não retorna nenhum valor
```

```
#####
#####
```

```
    menu:
```

```
        la     a0, string_menu         # carrega em a0 a
string menu
```

```
        li     a7, 4                   # informa impressão de
string
```

```
        ecall                          # chama impressão
```

```

        addi    a7, zero, 5                # informa leitura de inteiro
        ecall                                     # chama leitura
        add     a1, zero, a0                # carrega valor lido em a1

        addi    a2, zero, 0                # atribui ao registrador a2 o
valor zero
        beq     a1, a2, start                # verifica se a opção
selecionada foi 0, então inicia um novo jogo
        addi    a2, zero, 1                # atribui ao registrador a2 o
valor 1
        beq     a1, a2, get_jogada          # verifica se a opção
selecionada foi 1, então faz uma nova jogada
        addi    a2, zero, 2                # atribui ao registrador a2 o
valor 2
        beq     a1, a2, fim                # verifica se a opção
selecionada foi 1, então encerra o programa
        ret

#####
#####
        # função: get_jogada
        #      função responsável por obter as entradas do usuário referente a
posição do disparo,
        #      processar a mesma e fazer a chamada das funções que alteram a
matriz com base no acerto ou
        #      erro do disparo.
        # entrada: s0 - posicao inicial do vetor da matriz de navios na memória
        #      s7 - posicao inicial do vetor de matriz das tiros na memória
        # saída: não retorna nenhum valor
        #####
        #####
        get_jogada:
                la      a0, msg_linha_jogada        # carrega em a0 a
string da linha da jogada
                li      a7, 4                        # informa impressão de
string
                ecall                                     # chama impressão

        addi    a7, zero, 5                # informa leitura de inteiro
        ecall                                     # chama leitura
        add     a1, zero, a0                # carrega valor lido em a1

        blt     a1, zero, jogada_invalida          # valida se a1 é menor que

```



```

0, então informa que a jogada é inválida
    addi    t5, zero, 9                # adiciona 9 ao registrador
t5
    bgt     a1, t5, jogada_invalida    # valida se a1 é
maior que 9, então informa que a jogada é inválida

    la      a0, msg_coluna_jogada      # carrega em a0 a
string da coluna da jogada
    li      a7, 4                      # informa impressão de
string
    ecall                               # chama impressão

    addi    a7, zero, 5                # informa leitura de inteiro
    ecall   # chama leitura
    add     a2, zero, a0               # carrega valor lido em a2

    blt     a2, zero, jogada_invalida  # valida se a2 é menor que
0, então informa que a jogada é inválida
    addi    t5, zero, 9                # adiciona 9 ao registrador
t5
    bgt     a2, t5, jogada_invalida    # valida se a2 é
maior que 9, então informa que a jogada é inválida

    addi    t5, zero, 10               # adiciona a t5 o número de
colunas
    mul     a3, a1, t5                 # multiplica a linha inicial
pelo número de colunar
    add     a3, a3, a2                 # adiciona a coluna inicial
pela multiplicação anterior
    addi    a4, zero, 4                # carrega 4 ao registrador
a4
    mul     a3, a3, a4                 # multiplica por 4 o valor
do cálculo da posição
    add     a4, s0, a3                 # adiciona ao registrador
a4 o endereço da matriz de navios mais o deslocamento calculado

    addi    a6, zero, 42               # carrega o valor do
caracter asterisco ao registrado a6
    lbu     a5, (a4)                  # carrega em a5 o valor da
posição da jogada na matriz de navios
    addi    a4, a4, 400                # desloca a posição
da jogada para a matriz de tiros

```

```

        lbu    t6, (a4)                # carrega em t6 o valor da
posição da jogada na matriz de tiros
        addi   a7, zero, 111          # carrega em a7 o valor do
caractere o
        beq    t6, a7, jogada_invalida # verifica se o valor
carregado da posição da matriz de tiros já foi alterado para o
        bne    a5, a6, tiro_certo     # verifica se o valor
carregado da matriz de navios é diferente do caractere asterísco
        sb     a7, (a4)                # armazena o caractere o
na posição da matriz de tiros
        add    a4, zero, s7           # adiciona em a4 a posição
inicial da matriz de tiros
        jal    imprime_matriz         # chama a impressão da
matriz de tiros
        j      menu                   # volta para o menu

```

```

#####
#####

```

```

# função: jogada_invalida
#   função responsável por chamar a impressão da matriz seguida de uma
mensagem de jogada
#   inválida e retornar ao menu.
# entrada: s7 - posicao inicial do vetor de matriz das tiros na memória
# saída: não retorna nenhum valor

```

```

#####
#####

```

```

jogada_invalida:
        add    a4, zero, s7           # adiciona em a4 a posição
inicial da matriz de tiros
        jal    imprime_matriz         # chama a impressão da
matriz de tiros
        la     a0, msg_jogada_invalida # carrega em a0 a
mensagem de jogada inválida
        li     a7, 4                  # informa impressão de
string
        ecall                                # chama impressão
        j      menu                   # retorna para o menu

```

```

#####
#####

```

```

# função: tiro_certo
#   função responsável por inserir o caractere da embarcação no formato

```

minúsculo no endereço

```
#      da posição na matriz de tiros.
# entrada: s3 - contador de posicoes de navios inseridos
#      s4 - contador de disparos assertivos
#      s7 - posicao inicial do vetor de matriz das tiros na memória
#      a4 - posicao do tiro efetuado pelo jogador na matriz de tiros
#      a5 - valor presente no endereço referente a jogada na matriz de navios
# saida: s4 - contador de disparos assertivos
#####
#####
    tiro_certo:
        lb      a6, (a4)                # carrega em a6 o valor contido
na posição da matriz de tiros
        addi    a7, a5, 32              # adiciona 32 ao valor lido da
matriz de navios para transformar em minúsculo
        beq     a6, a7, jogada_invalida # valida se o valor
contido na posição da matriz de tiros já é o caractere minúsculo
        addi    s4, s4, 1              # incrementa um o contador de
tiros certos
        sb      a7, (a4)              # armazena na posição da matriz
de tiros o caractere minúsculo do navio acertado
        add     a4, zero, s7          # adiciona em a4 a posição
inicial da matriz de tiros
        jal     imprime_matriz        # chama a impressão da
matriz de tiros
        beq     s3, s4, winner        # verifica se o número de tiros
certos é igual ao de posições inseridas
        j       menu                 # retorna para o menu

#####
#####
# função: winner
#      função responsável por exibir a mensagem de partida encerrada e
ofertar as opções de
#      iniciar novo jogo ou sair, chamando a função responsável pela opção
escolhida
# entrada: não recebe entradas
# saida: não retorna nenhum valor
#####
#####
winner:
    la      a0, string_winner        # carrega em a0 a string de
```

```

vencedor
    li      a7, 4                # informa impressão de
string
    ecall                                # chama impressão

    addi    a7, zero, 5          # informa leitura de inteiro
    ecall                                # chama leitura
    add     a1, zero, a0         # carrega valor lido em a1

    addi    a2, zero, 0          # carrega 0 em a2
    beq     a1, a2, start        # valida se a1 for 0, então
direciona para start
    addi    a2, zero, 1          # carrega 1 em a2
    beq     a1, a2, fim          # valida se a1 for 1, então
direciona para o fim
    j       winner              # retorna a função winner

```

```

#####
#####
# função: insere_embarcacoes
#   função responsável por iniciar o loop de inserção das embarcações
# entrada: não recebe entradas
# saída: t0 - índice do navio inserido
#   t1 - recebe o número de colunas da matriz de navios
#####
#####
insere_embarcacoes:
    add     t0, zero, zero       # carrega em t0 o valor 0
    addi    t1, zero, 10         # carrega em t1 o número de
colunas
    j       get_n_navios        # chama get_n_navios

```

```

#####
#####
# função: get_n_navios
#   função responsável por ler a primeira instrução da string de navios,
que indica a quantidade
#   de navios a serem inseridos
# entrada: t1 - recebe o número de colunas da matriz de navios
#   a1 - posição inicial da string de navios
# saída: s1 - quantidade de navios a serem inseridos
#   a3 - valor ascii do caractere espaço

```

```
#####  
#####
```

```
    get_n_navios:  
        lbu    t3, (a1)                # carrega em t3 o endereço  
inicial da string de instruções  
        add    t2, zero, zero          # zera o registrador t2  
        beq    t3, t1, loop_navio_especs # verifica se o valor lido é  
igual ao ascii da quebra de linha  
        addi   t2, zero, 48            # carrega 48 em t2, valor ascii  
de 0  
        ble    t3, t2, qtde_invalida   # valida se t3 não é o valor  
ascii de 0  
        addi   t3, t3, -48              # transforma o valor ascii do  
caractere em valor decimal  
        addi   a3, zero, 32            # carrega em a3 o valor ascii  
do caractere espaço  
        add    s1, zero, t3            # armazena em s1 o valor  
referente a quantidade de navios a inserir  
        addi   a1, a1, 1               # passa para o próximo  
caractere da string  
        j      get_n_navios            # retorna a função  
get_n_navios
```

```
#####  
#####
```

```
    # função: loop_navio_especs  
    # função responsável por extrair as características do navio a ser  
inserido da string de navios  
    # entrada: t1 - recebe o número de colunas da matriz de navios  
    # t2 - somatório do comprimento dos navios inseridos  
    # a1 - posição da string de navios  
    # a3 - valor ascii do caractere espaço  
    # saída: s2 - posição do navio  
    # s3 - comprimento do navio  
    # s4 - linha inicial do navio  
    # s5 - coluna inicial do navio  
    # t2 - somatório do comprimento dos navios inseridos  
    # t3 - caractere final das instruções de características do navio
```

```
#####  
#####
```

```
    loop_navio_especs:  
        addi   a1, a1, 1                # passa para o próximo
```

```

caractere da string
    lbu    s2, (a1)                # armazena em s2 o valor
armazenado no endereço a1, que indica a posição do navio
    addi   s2, s2, -48             # transforma o valor ascii do
caractere em valor decimal
    addi   a1, a1, 1               # passa para o próximo
caractere da string
    lbu    s6, (a1)                # armazena em s6 o valor
armazenado no endereço a1
    bne    s6, a3, instrucao_invalida # valida se o valo de s6 é o
valor ascii do caractere espaço
    addi   a1, a1, 1               # passa para o próximo
caractere da string
    lbu    s3, (a1)                # armazena em s3 o valor
armazenado no endereço a1, que indica o comprimento do navio
    addi   s3, s3, -48             # transforma o valor ascii do
caractere em valor decimal
    add    t2, t2, s3              # incrementa o contador de
posições inseridas com o comprimento do navio
    addi   a1, a1, 1               # passa para o próximo
caractere da string
    lbu    s6, (a1)                # armazena em s6 o valor
armazenado no endereço a1
    bne    s6, a3, instrucao_invalida # valida se o valo de s6 é o
valor ascii do caractere espaço
    addi   a1, a1, 1               # passa para o próximo
caractere da string
    lbu    s4, (a1)                # armazena em s4 o valor
armazenado no endereço a1, que indica a linha inicial do navio
    addi   s4, s4, -48             # transforma o valor ASCII do
caractere em valor decimal
    addi   a1, a1, 1               # passa para o próximo
caractere da string
    lbu    s6, (a1)                # armazena em s6 o valor
armazenado no endereço a1
    bne    s6, a3, instrucao_invalida # valida se o valo de s6 é o
valor ascii do caractere espaço
    addi   a1, a1, 1               # passa para o próximo
caractere da string
    lbu    s5, (a1)                # armazena em s5 o valor
armazenado no endereço a1, que indica a coluna inicial do navio
    addi   s5, s5, -48             # transforma o valor ASCII do

```

```

caractere em valor decimal
    addi  a1, a1, 1                # passa para o próximo
caractere da string
    lbu   t3,(a1)                  # armazena em t3 o valor
armazenado no endereço a1
    addi  t0, t0, 1                # incrementa em 1 o contador
de navios inseridos
    j      insere_navio            # chama a função
insere_navio

```

```

#####
#####

```

```

# função: insere_navio
#   função responsável por calcular a posição de inserção do navio na
matriz de navios e
#   direcionar para a função de inserção com base na orientação do navio
# entrada: t1 - recebe o número de colunas da matriz de navios
#   s2 - posição do navio
#   s3 - comprimento do navio
#   s4 - linha inicial do navio
#   s5 - coluna inicial do navio
# saída: t1 - recebe o número de colunas da matriz de navios
#   t3 - caractere final das instruções de características do navio
#   t4 - posição de inserção do navio na matriz de navios
#   t5 - contador de posições inseridas
#   s2 - posição do navio
#   s3 - comprimento do navio
#   s4 - linha inicial do navio
#   s5 - coluna inicial do navio

```

```

#####
#####

```

```

insere_navio:
    mul   s6, s4, t1                # realiza a multiplicação da
linha inicial pela quantidade de colunas
    add   s6, s6, s5                # realiza a adição da
multiplicação anterior com a coluna inicial
    addi  t4, zero, 4                # carrega o valor 4 ao
registrador t4
    mul   s6, s6, t4                # multiplica o valor obtido no
cálculo anterior por 4 para obter o deslocamento
    add   t4, s0, s6                # armazena o valor do
deslocamento em t4

```

```

        add    t5, zero, zero                # armazena em t5 o valor 0

        beq    s2, zero, insere_navio_horizontal # valida se a posição do navio
é horizontal
        j      insere_navio_vertical          # chama a função que
insere navio vertical

#####
#####
# função: insere_navio_horizontal
#   função responsável por fazer a inserção de um navio na posição
horizontal
# entrada: t1 - recebe o número de colunas da matriz de navios
#   t4 - posição de inserção do navio na matriz de navios
#   t5 - contador de posições inseridas
#   s2 - posição do navio
#   s3 - comprimento do navio
#   s4 - linha inicial do navio
#   s5 - coluna inicial do navio
# saída: t4 - posição de inserção do navio na matriz de navios
#   t5 - contador de posições inseridas
#####
#####
insere_navio_horizontal:
    add    a5, s3, s5                        # soma o comprimento do
navio com a coluna inicial da inserção
    bgt    a5, t1, posicao_do_navio_invalida # valida se o valor
ultrapassa o número de colunas
    lbu    s2, (t4)                          # carrega em s2 o caractere
da posição a inserir o navio
    addi   a6, zero, 42                      # carrega em a6 o ascii do
caractere asterisco
    bne    s2, a6, posicao_do_navio_invalida # valida se na posição já
não foi inserido algum navio
    addi   t6, t0, 64                        # realiza a adição do navio
atual para obter o valor ascii do caractere referente
    sb     t6, (t4)                          # armazena no valor endereço t4 o
valor ascii do caractere referente ao navio
    addi   t5, t5, 1                          # acrescenta o contador de
posições inseridas
    beq    t5, s3, fim_insercao_navio        # verifica se o contador de
posições inseridas é igual ao comprimento do navio

```



```

        addi    t4, t4, 4                # aponta o próximo endereço
a ser inserido o caractere referente ao navio
        j      insere_navio_horizontal # retorna a função
insere navio horizontal

```

```

#####
#####

```

```

# função: insere_navio_vertical
#   função responsável por fazer a inserção de um navio na posição
vertical

```

```

# entrada: t1 - recebe o número de colunas da matriz de navios
#   t4 - posição de inserção do navio na matriz de navios
#   t5 - contador de posições inseridas
#   s2 - posição do navio
#   s3 - comprimento do navio
#   s4 - linha inicial do navio
#   s5 - coluna inicial do navio
# saída: t4 - posição de inserção do navio na matriz de navios
#   t5 - contador de posições inseridas

```

```

#####
#####

```

```

insere_navio_vertical:
    add    a5, s3, s4                # soma o comprimento do
navio com a linha inicial da inserção
    bgt    a5, t1, posicao_do_navio_invalida # valida se o valor
ultrapassa o número de linhas
    lbu    s2, (t4)                  # carrega em s2 o caractere
da posição a inserir o navio
    addi   a6, zero, 42               # carrega em a6 o ascii do
caractere asterísco
    bne    s2, a6, posicao_do_navio_invalida # valida se na posição já
não foi inserido algum navio
    addi   t6, t0, 64                # realiza a adição do navio
atual para obter o valor ascii do caractere referente
    sb     t6, (t4)                  # armazena no valor endereço t4 o
valor ascii do caractere referente ao navio
    addi   t5, t5, 1                  # acrescenta o contador t5 em
1
    beq    t5, s3, fim_insercao_navio # verifica se o contador t5 é
igual ao comprimento do navio
    addi   t4, t4, 40                # aponta o próximo endereço
a ser inserido o caractere referente ao navio

```

```
j      insere_navio_vertical      # retorna a função
insere navio vertical
```

```
#####
#####
```

```
# função: fim_insercao_navio
#      função responsável por fazer a inserção de um navio na posição
vertical
```

```
# entrada: t0 - contador de navios inseridos
#      t1 - valor ascii do caractere quebra de linha
#      t3 - caractere final das instruções de características do navio
#      s1 - quantidade de navios a serem inseridos
# saída: t4 - posição de inserção do navio na matriz de navios
#      t5 - contador de posições inseridas
```

```
#####
#####
```

```
    fim_insercao_navio:
        beq    t0, s1, fim_instrucoes      # valida se existe mais
um navio a ser inserido
        beq    t3, t1, loop_navio_especs   # valida se a instrução atual
está encerrada por uma quebra de linha
        j      instrucao_invalida          # chama a função de
instrução inválida
```

```
#####
#####
```

```
# função: fim_instrucoes
#      função responsável por retornar após todos os navios serem inseridos
# entrada: t2 - somatório do comprimento dos navios inseridos
# saída: s3 - somatório do comprimento dos navios inseridos
```

```
#####
#####
```

```
    fim_instrucoes:
        add    s3, zero, t2                # adiciona a s3 o somatório
do comprimento dos navios (posições inseridas)
        ret
```

```
#####
#####
```

```
# função: qtde_invalida
#      função responsável por imprimir a mensagem de quantidade de navios
inválida
# entrada: não recebe entradas
```

```

# saida: não retorna nenhum valor
#####
#####
qtde_invalida:
    la    a0, msg_qtde_invalida        # carrega em a0 a
string de quantidade inválida
    li    a7, 4                        # informa impressão de string
    ecall                                # chama impressão
    j     fim                          # encerra o programa

#####
#####
# função: instrucao_invalida
#     função responsável por imprimir a mensagem de instrução inválida
# entrada: não recebe entradas
# saida: não retorna nenhum valor
#####
#####
instrucao_invalida:
    la    a0, msg_instrucao_invalida    # carrega em a0 a
string de instrução inválida
    li    a7, 4                        # informa impressão de string
    ecall                                # chama impressão

    j     fim                          # encerra o programa

#####
#####
# função: posicao_do_navio_invalida
#     função responsável por imprimir a mensagem de posição do navio
inválida
# entrada: não recebe entradas
# saida: não retorna nenhum valor
#####
#####
posicao_do_navio_invalida:
    la    a0, msg_posicao_do_navio_invalida # carrega em a0 a
string de posição inválida
    li    a7, 4                        # informa impressão de string
    ecall                                # chama impressão
    j     fim                          # encerra o programa

```

```

#####
#####
# função: imprime_matriz
#     função responsável por imprimir inicializar a impressão de uma matriz
# entrada: a4 - Endereço inicial da matriz a ser imprimida
# saída: t0 - contador de posições impressas
#     t1 - valor total de posições a serem impressas
#     t2 - endereço inicial da matriz a ser imprimida
#     t3 - contador de posições impressas por na linha
#     t4 - total de posições por linha
#     t5 - índice da linha a ser impressa
#####
#####
imprime_matriz:
    la    a0, string_colunas      # carrega em a0 a string de
indices das colunas
    li    a7, 4                  # informa impressão de string
    ecall                                # chama impressão
    add    t0, zero, zero        # carrega em t0 o valor 0
    addi   t1, zero, 100         # carrega em t1 o total de
posições da matriz
    add    t2, zero, a4          # carrega em t2 a posição
inicial do vetor da matriz
    add    t3, zero, zero        # carrega em t3 o valor 0
    addi   t4, zero, 10          # carrega em t4 o valor 10
    add    t5, zero, zero        # carrega em t5 o valor 0
    add    a0, zero, t5          # carrega em a0 o valor 0
    addi   a0, a0, 48            # transforma o valor decimal
para o valor ascii
    li    a7, 11                # informa impressão de
caractere
    ecall                                # chama impressão
    addi   a0, zero, 32          # carrega em a0 o ascii do
asterísco
    li    a7, 11                # informa impressão de
caractere
    ecall                                # chama impressão
    j      loop_imprime_matriz    # chama função loop
imprime matriz

#####

```

```
#####
# função: loop_imprime_matriz
#   função responsável por iterar a impressão de uma matriz
# entrada: t0 - contador de posições impressas
#   t1 - valor total de posições a serem impressas
#   t2 - endereço inicial da matriz a ser imprimida
#   t3 - contador de posições impressas por na linha
#   t4 - total de posições por linha
#   t5 - índice da linha a ser impressa
# saída: t0 - contador de posições impressas
#   t1 - valor total de posições a serem impressas
#   t2 - endereço inicial da matriz a ser imprimida
#   t3 - contador de posições impressas por na linha
#   t4 - total de posições por linha
#   t5 - índice da linha a ser impressa
#####
#####
loop_imprime_matriz:
    lbu    a0, (t2)                # carrega em a0 o valor
contido na posição da matriz
    li     a7, 11                  # informa impressão de
caractere
    ecall                                # chama impressão
    addi   a0, zero, 32
    li     a7, 11                  # informa impressão de
caractere
    ecall                                # chama impressão
    addi   t0, t0, 1
    addi   t3, t3, 1
    addi   t2, t2, 4
    beq    t3, t4, quebra_linha_matriz
    beq    t0, t1, fim_impressao
    j      loop_imprime_matriz

#####
#####
# função: quebra_linha_matriz
#   função responsável por realizar a quebra de linha durante a impressão
da matriz
# entrada: t0 - contador de posições impressas
#   t1 - valor total de posições a serem impressas
#   t3 - contador de posições impressas por na linha
```

```

#      t5 - indice da linha a ser impressa
# saida: t3 - contador de posições impressas por na linha
#      t5 - indice da linha a ser impressa
#####
#####
        quebra_linha_matriz:
            addi    t3, zero, 0                # zera o contador de posições
impressas por linha
            addi    t5, t5, 1                  # incrementa o indice de
linhas
            addi    a0, zero, 10               # carrega em a0 o ascii da
quebra de linha
            li      a7, 11                     # informa impressão de
caractere
            ecall                                # chama impressão
            beq     t0, t1, fim_impressao      # verifica se imprimiu
todas as posições
            add     a0, zero, zero             # zera o contador a0
            add     a0, zero, t5               # adiciona a a0 o valor do
indice da linha impressa
            addi    a0, a0, 48                 # transforma o valor decimal
no valor ascii do caractere
            li      a7, 11                     # informa impressão de
caractere
            ecall                                # chama impressão
            addi    a0, zero, 32               # carrega em a0 o valor ascii
do caractere espaço
            li      a7, 11                     # informa impressão de
caractere
            ecall                                # chama impressão
            j       loop_imprime_matriz

#####
#####
# função: fim_impressao
#      função responsável por retorar após o fim da impressão da matriz
# entrada: não recebe entradas
# saida: não retorna nenhum valor
#####
#####
fim_impressao:
            addi    a0, zero, 10               # carrega em a0 o valor ascii

```

da quebra de linha

li a7, 11 # informa impressão de
caractere

ecall # chama impressão

ret # retorna

#####

função: inicializa_matriz
função responsável por inicializar a alocação das posições dos vetores
das matrizes

entrada: a4 - posição inicial do vetor da matriz a ser inicializada

saída: t0 - contador de posições alocadas

t1 - valor total de posições a serem alocadas

a3 - posição da matriz a ser alocada

#####

inicializa_matriz:

add t0, zero, zero # carrega em t0 o valor 0

addi t1, zero, 100 # carrega em t1 o total de
posições

add a3, zero, a4 # carrega em a3 a posição
inicial da matriz

addi t4, zero, 10 # carrega em a4 o valor 10

j loop_inicializa_matriz # chama a função loop
inicializa matriz

#####

função: loop_inicializa_matriz
função responsável por iterar a alocação das posições dos vetores das
matrizes

entrada: t0 - contador de posições alocadas

t1 - valor total de posições a serem alocadas

a3 - posição da matriz a ser alocada

saída: t0 - contador de posições alocadas

a3 - posição da matriz a ser alocada

#####

loop_inicializa_matriz:

addi t6, zero, 42 # carrega em t6 o ascii do
asterisco

```

        sb    t6, (a3)                # insere o valor do ascii do
asterísco na posição da matriz
        addi  t0,t0, 1                # incrementa em 1 o contador
de posições alocadas
        addi  a3,a3, 4                # pula para próxima posição a
ser alocada
        beq   t0, t1, fim_inicializa_matriz  # verifica se alocou todas as
posições
        j     loop_inicaliza_matriz      # retorna para a função
loop inicializa matriz

```

```

#####
#####

```

```

# função: fim_inicializa_matriz
#   função responsável por retornar após a alocação das posições dos
vetores das matrizes
# entrada: a3 - posição da matriz a ser alocada
# saída: a3 - posição da matriz a ser alocada
#####
#####

```

```

fim_inicializa_matriz:
    ret                                # retorna

```

```

#####
#####

```

```

# função: fim
#   função responsável por encerrar a execução do programa
# entrada: não recebe entradas
# saída: não retorna nenhum valor
#####
#####

```

```

fim:
    nop                                # encerra

```