# Robô seguidor de linha

Neste relatório iremos apresentar o desenvolvimento do código utilizado para a criação de um "robô" seguidor de linhas.

A principal ferramenta utilizada nesse processo fora o Bitmap presente no software MARS, no qual todos os comandos para devido funcionamento do "robô" foram implementados.

## 1º Passo: Criação do .data:

O .data consiste ,basicamente, na declaração das variáveis presentes no programa a ser implementado.

Nele nós definimos um array com o objetivo de armazenar os valores das coordenadas das linhas, em seguida definimos labels para as cores, para podermos identificar com maior facilidade o que está ocorrendo no bitmap, e por fim foram criados comandos '.asciiz' para podermos informar ao usuário qual informação, comando e valor que ele deverá fornecer a máquina.

## 2º Passo: .text - Loads, verificações sobre os dados fornecidos pelo usuário e armazenamento.

A partir da linha 19 até a linha 153, fizemos a implementação dos menus, das verificações das informações inseridas pelo usuário e o armazenamento dos dados fornecidos.

A princípio é possível ver que demos loads para as cores das linhas, do robô e das linhas pelas quais o robô já passou, em seguida fizemos comandos 'addi' para podermos contar a quantidade de linhas e para que possamos guardar a coordenadas da posição inicial do array.

Mais a frente iniciamos um loop para o menu, a priori nós criamos o bitmap e fizemos a distribuição dos pixels aos endereços do bitmap, depois utilizamos comandos Syscall para podermos mostrar na tela o menu, no qual o usuário deverá escolher o próximo passo a ser realizado. Após isso, fizemos loops para verificação das informações dadas pelo usuário, nos quais são analisados e indicado a validez das entradas para a escolha das opções no menu, para direção em que a linha irá ser criada, para o sentido dela e para as coordenadas, já que elas deverão permanecer dentro dos limites do bitmap essa verificação é necessária. E esses loops são chamados por comandos mais a frente no código: DesenharLinha, LoopSentido, LoopDirecao.

Em seguida fizemos um loop "DesenharLinha", cujo objetivo é armazenar as coordenadas, se válidas, em registradores, para que sejam utilizadas na

identificação pelo robô do local para qual ele deverá ir para começar a andar pela linha, o

"LoopSentido" que irá verificar o dado em relação ao sentido da linha e informar ao usuário caso seja inválida através da chamada dos loops de verificação do menu e um "LoopDirecao" para verificar a direção e informar a validez da direção ao usuário.

Após esses loops fizemos uma verificação para o tamanho da linha, já que, mesmo que as coordenadas estejam corretas, ao inserir o tamanho da linha, pode ocorrer de que ela seja projetada para fora do bitmap, essa verificação começa na linha 115 e caso a linha esteja nos conformes ela é armazenada através dos comandos "store word".

3ºPasso: Desenhar as linhas.

Primeiro multiplicamos x por quatro para poder acessar o tamanho correto do pixel do bitmap, seguido de um comando add para que ele possa "andar" no eixo x e depois somamos 256 para andar em y.

Após isso fizemos dois loops responsáveis por colorir os pixels no eixo X, dependendo do sinal armazenado em relação ao sentido da linha fornecido pelo usuário um respectivo loop é ativado para ir para esquerda ou para a direita pintando os pixels, e ambos os loops contém jumps para ContinuacaoDesenho, que será explicado mais à frente.

Agora para o eixo Y fizemos novamente, a multiplicação e a somas nos eixos X e Y, para que seja possível andar por eles. E também criamos outros dois loops responsáveis por colorir os pixels no eixo Y, que também dependerão do valor no registrador da direção para saber se deverão pintar o pixel acima ou abaixo, e ambos os loops tem o jump para ContinuacaoDesenho.

O ContinuacaoDesenho, basicamente, é a parte do código que verificará se o usuário deseja inserir mais linhas e para isso, também, fora utilizado um loop para que ele possa fazer quantas linhas quiser.

4ºPasso: Ajustar a tela.

O "AjustaInicial" é uma função de extrema importância pois está bastante conectada com a parte de desenhar as linhas, já que é através dela que vamos setar a tela do bitmap toda vez que uma linha for desenhada e que vamos atualizar o X e o Y para o lugar onde a última linha foi desenhada.

Após atualizarmos x e y, fizemos uma sequência de operações responsáveis por somar ou subtrair 1 aos valores das coordenadas dependendo do sentido e da direção escolhida pelo usuário, se for pra direita nos somamos em x, para esquerda subtraimos em x, se for pra cima somamos em y e se for pra baixo subtraimos em

y, e todas essas funções estão conectadas com o loop linha no qual ele irá pedir o sentido da linha e a direção dela sempre que uma nova linha for criada, tudo isso para que a pintura dos pixel seja conforme o desejado.

5º Passo: Gerar o robô em uma posição aleatória e encontrar a primeira linha.

Gerar o robô, foi bem simples apenas utilizamos o comando "li $v0, 42" para que ele possa gerar aleatoriamente um número entre 0 e 63 duas vezes, uma para o X e outra para o Y. E para permitir que o robô possa andar em x e y , novamente,fizemos a multiplicação do valor de x por 4, para corresponder corretamente ao pixel do bitmap, seguido de um add  ao x e somamos 256 em y.

Em seguida fizemos uma função 'skip6' responsável por pintar o robô na tela e a função AndaNoArray, no qual ele vai pegar as posições X e Y iniciais da linha e vai compará-los com as coordenadas do robô através de uma subtração.

Após isso fizemos a função ProcuraX que será responsável por identificar para que sentido o robô irá se dirigir, analisando  o resultado da subtração feita no AndaNoArray, fizemos novamente as algebra em x e y para permitir que o robô ande. Em seguida implementamos dois loops responsáveis por identificar e pintar os bits que armazenam o valor da cor da linha, um identifica e pinta os bits da esquerda e o outro identifica e pinta os pixels da direita. E também fizemos uma função ProcuraY, semelhante ao ProcuraX, com princípio de identificar a direção que o robô deve ir para encontrar a linha, para cima ou para baixo, juntamente com dois loops responsáveis por pintar o pixels caso o robô esteja passando por cima de uma linha no eixo Y.

6ºPasso: Seguir linha.

Após encontrar a posição inicial nosso robô inicia agora uma série de funções de verificação de pixels adjacentes a posição inicial com o objetivo de encontrar pixel com a cor da linha, para isso utilizamos vários branch equals e load word para identificar os valores dos pixel ao redor e fazer um branch caso o valor respectivo a cor da linha seja encontrado nos pixels adjacentes.

Para isso criamos a função SegueLinha com os branch equals para outra funções sendo elas :SegueCima, SegueDireita, SegueBaixo e SegueEsquerda, responsáveis por pintar o bit identificado e reiniciar o loop do SegueLinha através de um jump.

E por fim, uma função ArrumaRobo, que simplesmente posiciona o robô para fora da linha, na direção que ele estava seguindo, para indicar que acabou aquela linha.

ANEXO I

.data

```
Array: .space 400 # Espaço para armazenar os x e y das linhas

White: .word 0xFFFFFF

Blue: .word 0x0000FF

Red: .word 0xFF0000

Menu: .asciiz "1 - Desenhar linha\n2 - Prosseguir para o robô"

Escolha: .asciiz "\nEscolha uma opcao\n>> "

EntradaInvalida: .asciiz "Digite uma entrada válida.\n\n"

CoordInicial: .asciiz "\nO mapa e uma matriz de 64x64, cada posição (x,y) inidica um ponto na matriz com 0<=x<=63 e 0<=y<=63.\nDigite a posição inicial da linha"

xInicial: .asciiz "\nx inicial: "

yInicial: .asciiz "y inicial: "

Sentido: .asciiz "\nSelecione o sentido\n1-Positivo\n2-Negativo\n>> "

Direcao: .asciiz "\nSelecione a direcao\n1-Vertical\n2-Horizontal\n>> "

Tamanho: .asciiz "\nDigite o tamanho da linha (tenha em mente que a linha não pode cruzar os limites do mapa): "

ForaMapa: .asciiz "\nLinha cruzou os limites do mapa. Entre com uma nova coordenada."

Continuar: .asciiz "\nDeseja continuar a linha\n1-Sim\n2-Nao\n>> "

Barran: .asciiz "\n"


.text

lw $s0, White #Cor das linhas desenhadas

lw $s2, Blue #Cor do robô

lw $s3, Red #Cor das linhas ja visitadas


addi $t0, $t0, 0 #Conta quantidade de linhas desenhadas

addi $s1, $zero, 0 #Posicao inicial do array que guarda as coordernadas

Loop_Menu:

#Inicia todos os valores para a tela

addi $t6, $zero, 32768 #32768 = (512*512)/8 pixels

add $t7, $t6, $zero #Adicionar a distribuicao de pixels ao endereco
```

```
lui $t7, 0x1004 #Endereco base da tela no heap, pode mudar se quiser

li $v0, 4
la $a0, Menu
syscall

li $v0, 4
la $a0, Escolha
syscall

li $v0, 5
syscall
move $t1, $v0

blez $t1, Entrada_Invalida
bgt $t1, 2, Entrada_Invalida

beq $t1, 1, DesenharLinha
beq $t1, 2, Robo

Entrada_Invalida:
li $v0, 4
la $a0, EntradaInvalida
syscall
j Loop_Menu

DirecaoInvalida:
li $v0, 4
la $a0, EntradaInvalida
syscall
j LoopDirecao
```

```
SentidoInvalido:
li $v0, 4
la $a0, EntradaInvalida
syscall
j LoopSentido

CoordInvalida:
li $v0, 4
la $a0, ForaMapa
syscall
j DesenharLinha

DesenharLinha:
addi $s5, $zero, 0
li $v0, 4
la $a0, CoordInicial
syscall

li $v0, 4
la $a0, xInicial
syscall
li $v0, 5
syscall
move $t1, $v0 # x inicial e armazenado em t1

li $v0, 4
la $a0, yInicial
syscall
li $v0, 5
syscall
```

```
move $t2, $v0 # y inicial e armazenado em t2

LoopLinha:

LoopSentido: # Pede o sentido da linha
li $v0, 4
la $a0, Sentido
syscall
li $v0, 5
syscall
move $t3, $v0
blez $t3, SentidoInvalido
bgt $t3, 2, SentidoInvalido

LoopDirecao: # Pede a direção da linha
li $v0, 4
la $a0, Direcao
syscall
li $v0, 5
syscall
move $t4, $v0
blez $t4, DirecaoInvalida
bgt $t4, 2, DirecaoInvalida

li $v0, 4
la $a0, Tamanho
syscall
li $v0, 5
syscall
move $t5, $v0
```

```
beq $t3, 1, VerificaPositivo # Verifica se a linha sera desenhada para fora do mapa
VerificaNegativo:
beq $t4, 1, VerificaVerticalNeg
VerificaHorizontalNeg:
sub $s7, $t1, $t5
bltz $s7, CoordInvalida
beqz $s5, Armazena
j SoDesenha
VerificaVerticalNeg:
add $s7, $t2, $t5
bgt $s7, 63, CoordInvalida
beqz $s5, Armazena
j SoDesenha
VerificaPositivo:
beq $t4, 1, VerificaVerticalPos
VerificaHorizontalPos:
add $s7, $t1, $t5
bgt $s7, 63, CoordInvalida
beqz $s5, Armazena
j SoDesenha
VerificaVerticalPos:
sub $s7, $t2, $t5
bltz $s7, CoordInvalida
bgtz $s5, SoDesenha

Armazena: # Se ela não for desenhada para fora, armazenamos seu x e y inicial

addi $s5, $s5, 1
addi $t0, $t0, 1 # Conta a quantidade de linhas
sw $t1, Array($s1) # Armazena x na posição do vetor, e soma 4 no indice
addi $s1, $s1, 4
```

```
sw $t2, Array($s1) # Armazena y na posição do vetor, e soma 4 no indice para prox linha
addi $s1, $s1, 4

SoDesenha:

add $t8, $zero, $t7

beq $t4, 1, DirecaoVertical # Faz verificação para a escolha qual sera o sentido da linha desenhada
# O sentido da linha é visto dentro do loop de desenho, pois se andaremos em positivo no x somaremos suas coord e subtrairemos
# se o sentido for negativo, o inverso acontece para o y, se for positivo subtraimos as coord e se for negativo somaremos

DirecaoHorizontal:
addi $t4, $zero, 0

#mult x por 4
add $t6,$t1,$t1
add $t6,$t6,$t6

#soma x
add $t8,$t8,$t6

#soma 256 em y
beq $t2,$zero,skip
loop_0:
addi $t8,$t8,256
addi $t4,$t4,1
bne $t2,$t4,loop_0
skip:
```

```
add $t7,$t5,$zero
addi $t4, $zero, 0

beq $t3, 1, loop_1.5
loop_1:
sw $s0, ($t8) #Pinto o pixel em $t8 com a cor seleciona armazenada em $s0
subi $t8, $t8, 4 #pixel - 4
addi $t4, $t4, 1 #Cont ++
bne $t4, $t5, loop_1
addi $t9, $zero, 0
j ContinuacaoDesenho

loop_1.5:
sw $s0, ($t8) #Pinto o pixel em $t8 com a cor seleciona armazenada em $s0
addi $t8, $t8, 4 #pixel + 4
addi $t4, $t4, 1 #Cont ++
bne $t4, $t5, loop_1.5
addi $t9, $zero, 1
j ContinuacaoDesenho
DirecaoVertical:
addi $t4, $zero, 0

#multipicla x por 4
add $t6,$t1,$t1
add $t6,$t6,$t6

#soma x
add $t8,$t8,$t6

#soma 256 para cada y
beq $t2,$zero,skip2
```

```
loop_2:
addi $t8,$t8,256
addi $t4,$t4,1
bne $t2,$t4,loop_2

skip2:
add $t7,$t5,$zero
addi $t4, $zero, 0

beq $t3, 1, loop_4
loop_3:
sw $s0, ($t8) #Pinto o pixel em $t0 com a cor seleciona armazenada em $s0
addi $t8, $t8, 256 #pixel + 256
addi $t4, $t4, 1 #Cont ++
bne $t4, $t5, loop_3
addi $t9, $zero, 2
j ContinuacaoDesenho

loop_4:
sw $s0, ($t8) #Pinto o pixel em $t0 com a cor seleciona armazenada em $s0
subi $t8, $t8, 256 #pixel - 256
addi $t4, $t4, 1 #Cont ++
bne $t4, $t5, loop_4
addi $t9, $zero, 3
ContinuacaoDesenho:
li $v0, 4
la $a0, Continuar
syscall
li $v0, 5
syscall
move $t4, $v0
```

beq $t4, 1, AjustaInicial # Se quiser continuar a linha, ele vai para AjustaInicial que atualiza o x e o y da linha para o lugar onde paramos

j Loop_Menu

AjustaInicial: # Essas 3 prox linhas são para setar a tela, toda vez que desenharmos precisamos usa-las

addi $t6, $zero, 32768 #32768 = (512*512)/8 pixels

add $t7, $t6, $zero #Adicionar a distribuicao de pixels ao endereco

lui $t7, 0x1004 #Endereco base da tela no heap, pode mudar se quiseraddi $t6, $zero, #32768 = (512*512)/8 pixels

beqz $t9, SubX # Essa parte arruma o x e y da linha dependendo dos parametros usados para desenhar a linha, por exemplo,

beq $t9, 1, SomaX # se a linha foi desenhada para a direita precisamos somar o x com o tamanho da linha para continuar o desenho

beq $t9, 2, SubY

SomaY:

sub $t2, $t2, $t5

addi $t2, $t2, 1

j LoopLinha

SubY:

add $t2, $t2, $t5

subi $t2, $t2, 1

j LoopLinha

SomaX:

add $t1, $t1, $t5

subi $t1, $t1, 1

j LoopLinha

SubX:

sub $t1, $t1, $t5

addi $t1, $t1, 1

```
j LoopLinha


Robo:
li $a1, 63 # Gera aleatoriamente um numero entre 0 e 63 para a posicao inicial do robo
li $v0, 42
syscall
add $t1, $a0, $zero #t1 posicao x inical do robo

li $a1, 63
li $v0, 42
syscall
add $t2, $a0, $zero #t2 posicao y inical do robo

addi $t6, $zero, 32768 #32768 = (512*512)/8 pixels
add $t7, $t6, $zero #Adicionar a distribuicao de pixels ao endereco
lui $t7, 0x1004 #Endereco base da tela no heap, pode mudar se quiser

add $t8, $zero, $t7
addi $t4, $zero, 0

#mult x por 4
add $t6,$t1,$t1
add $t6,$t6,$t6

#soma x
add $t8,$t8,$t6

beq $t2,$zero,skip6
loop_5:
addi $t8,$t8,256
```

```
addi $t4,$t4,1
bne $t2,$t4,loop_5

skip6:
add $t7,$t5,$zero
addi $t4, $zero, 0

sw $s2, ($t8) #Pinto o pixel em $t8 com a cor seleciona armazenada em $s2, pintando o robo na tela
subi $t8, $t8, 4 #pixel + 4
addi $t4, $t4, 1 #Cont ++
addi $t9, $zero, 0

addi $s4, $zero, 0
AndaNoArray: # Pega os x e y iniciais das linhas
lw $s5, Array($s4) #s5 posicao inicial x da linha
addi $s4, $s4, 4
lw $s6, Array($s4) #s6 posicao inicial y da linha
addi $s4, $s4, 4

sgt $t3, $s5, $t1 # Verifica se o x da linha é maior do que ox do robo
addi $t5, $zero, 0
sub $t5, $t1, $s5 # Pega o tamanho subtraindo um x do outro
abs $t5, $t5 # Tamanho em modulo

ProcuraX: # Anda com o robo em x para o x inicial da linha
addi $t6, $zero, 32768 #32768 = (512*512)/8 pixels
add $t7, $t6, $zero #Adicionar a distribuicao de pixels ao endereco
lui $t7, 0x1004 #Endereco base da tela no heap, pode mudar se quiser

add $t8, $zero, $t7
```

```
addi $t4, $zero, 0

#mult x por 4
add $t6,$t1,$t1
add $t6,$t6,$t6

#soma x
add $t8,$t8,$t6

#soma 256 em y
beq $t2,$zero,skip7
loop_7:
addi $t8,$t8,256
addi $t4,$t4,1
bne $t2,$t4,loop_7
skip7:
add $t7,$t5,$zero
addi $t4, $zero, 0

beqz $t5, Saix
beq $t3, 1, xqdl
beq $s5, $t1, loop_8
sub $t1, $t1, $t5

loop_8:
beq $t9, $s0, NaoPinta
sw $t9, ($t8) #Pinto o pixel em $t8 com a cor seleciona armazenada em $s0
j ContinuarPinta
NaoPinta:
sw $t9, ($t8)
ContinuarPinta:
```

```
subi $t8, $t8, 4 #pixel - 4
lw $t9, ($t8)
sw $s2, ($t8) #Pinto o pixel em $t8 com a cor seleciona armazenada em $s0
li $v0, 32
addi $a0, $zero, 200
syscall
addi $t4, $t4, 1 #Cont ++
bne $t4, $t5, loop_8
j Saix
xqdl:
add $t1, $t1, $t5
loop_9:
beq $t9, $s0, NaoPinta1
sw $t9, ($t8) #Pinto o pixel em $t8 com a cor seleciona armazenada em $s0
j ContinuarPinta1
NaoPinta1:
sw $t9, ($t8)
ContinuarPinta1:
addi $t8, $t8, 4 #pixel + 4
lw $t9, ($t8)
sw $s2, ($t8) #Pinto o pixel em $t8 com a cor seleciona armazenada em $s0
li $v0, 32
addi $a0, $zero, 200
syscall
addi $t4, $t4, 1 #Cont ++
bne $t4, $t5, loop_9
Saix:

sgt $t3, $s6, $t2
addi $t5, $zero, 0
sub $t5, $t2, $s6
```

```
abs $t5, $t5

ProcuraY: # Anda com o robo em y para o y inicial da linha
addi $t6, $zero, 32768 #32768 = (512*512)/8 pixels
add $t7, $t6, $zero #Adicionar a distribuicao de pixels ao endereco
lui $t7, 0x1004 #Endereco base da tela no heap, pode mudar se quiser

add $t8, $zero, $t7
addi $t4, $zero, 0

#mult x por 4
add $t6,$t1,$t1
add $t6,$t6,$t6

#soma x
add $t8,$t8,$t6

#soma 256 em y
beq $t2,$zero,skip8
loop_10:
addi $t8,$t8,256
addi $t4,$t4,1
bne $t2,$t4,loop_10
skip8:
add $t7,$t5,$zero
addi $t4, $zero, 0

beqz $t5, Saiy
beq $t3, 1, xqdl2
beq $s6, $t2, loop_11
sub $t2, $t2, $t5
```

```
loop_11:
beq $t9, $s0, NaoPinta2
sw $t9, ($t8) #Pinto o pixel em $t8 com a cor seleciona armazenada em $s0
j ContinuarPinta2
NaoPinta2:
sw $t9, ($t8)
ContinuarPinta2:
subi $t8, $t8, 256 #pixel - 4
lw $t9, ($t8)
sw $s2, ($t8) #Pinto o pixel em $t8 com a cor seleciona armazenada em $s0
li $v0, 32
addi $a0, $zero, 200
syscall
addi $t4, $t4, 1 #Cont ++
bne $t4, $t5, loop_11
j Saiy
xqdl2:
add $t2, $t2, $t5
loop_12:
beq $t9, $s0, NaoPinta3
sw $t9, ($t8) #Pinto o pixel em $t8 com a cor seleciona armazenada em $s0
j ContinuarPinta3
NaoPinta3:
sw $t9, ($t8)
ContinuarPinta3:
addi $t8, $t8, 256 #pixel + 4
lw $t9, ($t8)
sw $s2, ($t8) #Pinto o pixel em $t8 com a cor seleciona armazenada em $s0
li $v0, 32
addi $a0, $zero, 200
syscall
```

```
addi $t4, $t4, 1 #Cont ++
bne $t4, $t5, loop_12

Saiy:

addi $t5, $zero, 0
SegueLinha: # Verifica em qual posição adjacente tem uma linha e a segue
lw $t9, -256($t8)
beq $t9, $s0, SegueCima
lw $t9, 4($t8)
beq $t9, $s0, SegueDireita
lw $t9, 256($t8)
beq $t9, $s0, SegueBaixo
lw $t9, -4($t8)
beq $t9, $s0, SegueEsquerda
j ProcuraNovaLinha

SegueCima:
sw $s3, ($t8)
subi $t8, $t8, 256
sw $s2, ($t8)
li $v0, 32
addi $a0, $zero, 200
syscall
subi $t2, $t2, 1
j SegueLinha

SegueDireita:
sw $s3, ($t8)
addi $t8, $t8, 4
sw $s2, ($t8)
```

```
li $v0, 32
addi $a0, $zero, 200
syscall
addi $t1, $t1, 1
j SegueLinha

SegueBaixo:
sw $s3, ($t8)
addi $t8, $t8, 256
sw $s2, ($t8)
li $v0, 32
addi $a0, $zero, 200
syscall
addi $t2, $t2, 1
j SegueLinha

SegueEsquerda:
sw $s3, ($t8)
subi $t8, $t8, 4
sw $s2, ($t8)
li $v0, 32
addi $a0, $zero, 200
syscall
subi $t1, $t1, 1
j SegueLinha

ProcuraNovaLinha: # Se ainda possuir linhas no mapa continua, se não finaliza o programa
addi $t5, $zero, 0
subi $t0, $t0, 1
beqz $t0, Fim
```

```
j AndaNoArray

Fim:
li $v0, 10
syscall
```