

UNIVERSIDADE FEDERAL DE SÃO PAULO
INSTITUTO DE CIÊNCIA E TECNOLOGIA



Disciplina: Arquitetura e Organização de Computadores

Professora Dra. Denise Stringhini

Robô Seguidor de Linha

Álvaro Cardoso Vicente de Souza RA: 133536

Gabriel Angelo Cabral Neves RA: 136124

Jonathan Andrade de Oliveira RA: 133702

São José dos Campos

2019

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 validade 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 validade 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.

4º Passo: 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 subtraímos em x, se for pra cima somamos em y e se for pra baixo subtraímos 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 outras 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.