Relatório de Arquitetura de Computadores I

Tiago Pinto 54718, Henrique Anacleto 54897

28/05/2023

0.1 Introdução

Com este trabalho pretendemos desenvolver um programa, em Assembly RISC-V, para localizar personagens da saga StarWars, a partir de uma imagem rgb. A imagem que foi utilizada possuía três personagens diferentes (mestre Yoda, Darth Maul e Mandalorian), cada um com uma tonalidade de cor diferente. Então como esses personagens eram distínguiveis pela cor, bastava procurar os pixeis correspondentes a cada personagem, a partir das tonalidades correspondentes, e depois calcular "o centro de massa", ou posição média, desses pixeis. Após descobrir-se as coordenadas do centro de massa de cada personagem, podia-se então marcar com uma cruz o centro do personagem escolhido.

0.2 Desenvolvimento do Projeto

Para o desenvolvimento deste projeto, foram inicialmente criadas algumas variáveis globais:

imagem: imagem do StarWars utilizada no input, em formato rgb, para poderem ser lidos os pixeis da mesma;

imagem2: imagem do StarWars utilizada no output, em formato rgb, que vai ser escrita com uma cruz, no centro do personagem escolhido;

frase: string com a frase inicial, pedindo ao utilizador para escolher um dos 3 personagens;

pixeis array: array com o número de bytes da imagem rgb.

Após a declaração das variavéis globais, foram implementadas algumas funções.

0.2.1 Função: read rgb image

Objetivo: Lê o ficheiro starwars.rgb e coloca cada pixel num array.

Esta função foi a primeira a ser implementada, e para isso, inicialmente começamos por abrir o ficheiro com a imagem rgb e defini-lo como leitura, para isto utilizamos o comando syscall (a7 = 1024 para abrir o ficheiro e a0 = 0 para definir leitura do ficheiro).

Depois utilizamos outro comando syscall, mas desta vez, para ler o ficheiro e colocar os valores num array (pixeis array). Para isso foi necessário igualar o a7 a 63. Também tivemos que definir o limite máximo do array, e para

isso, igualamos o a2 a 172800, que é o número de bytes da imagem rgb, e pôr o a1 como endereço do array.

Por fim fechamos o ficheiro com o comando syscall (a7 = 57), e pusemos o endereço a0 a retornar a função, para termos "feedback" se havia algum erro (a0 = -1) ou não.

0.2.2 Função: hue

Objetivo: calcula a componente Hue a partir das componentes R, G e B de um pixel.

Neste trabalho, o ficheiro de imagem original contém uma imagem a cores no formato RGB com 8 bits de profundidade de cor. Assim, cada componente de cor é codificada por um byte e portanto cada pixel tem três bytes.

Numa outra função (location), extraímos os valores RGB de um pixel, e guardamos nos registos a3 (R), a4 (G) e a5 (B). E após isso, fomos calcular a componente Hue, que representa a tonalidade num círculo de cores, e que iria servir para conseguirmos identificar, a qual pixel cada personagem correspondia.

A componente Hue pode ser calculada de forma aproximada, resultando num valor inteiro entre 0 e 359 graus.

Para isso utilizamos branchs, para comparar os valores RGB dos pixeis, e depois, dependendo das comparações, realizava as respetivas operações para calcular a Hue. Foi necessário colocar alguns valores inteiros em alguns registos, visto que não dá para realizar subtração, divisão e multiplicação com immediate. Os branches utilizados foram blt e bge. Por fim a componente Hue ficaria guardada no registo a6. Caso as componentes R, G, B fossem iguais, a Hue seria igual a 0.

Após a Hue ter sido calculada, a função atual, salta para a função indicator, que iria servir para identificar qual personagem, o pixel pertence. Por fim, depois da execução da função indicator, a função hue retorna a6 (registo com o valor da Hue).

Como realizamos uma função dentro de outra, quando iniciamos o loop, alocamos espaço na stack e guardamos o return address dentro dela, e antes da função terminar, recuperamos o return address e libertamos o espaço da stack.

0.2.3 Função: indicator

Objetivo: dado um personagem e um pixel com componentes R, G, B, indica se esse pixel pertence ou não à personagem.

Como cada personagem tem tonalidades próprias diferentes dos restantes, podemos distingui-los apenas pela componente Hue. Então, após esta ter sido calculada, esta função iria compará-la com os intervalos que pertencem a cada personagem.

Para isso, também foram utilizados branches (blt), e uma label para cada personagem. Foram colocados os valores inteiros em alguns registos visto que não é possivel realizar instruções branch com immediate. O registo s0 iria ser o retorno da função e iria ser igual a:

1 se o pixel pertencer ao Yoda;

2 se o pixel pertencer ao Darth Maul;

3 se o pixel pertencer ao Mandalorian;

0 se não pertencer a nenhum dos personagens.

Este registo é sempre resetado quando a função inicia.

0.2.4 Função: location

Objetivo: calcula o centro de massa para um certo personagem.

Após a realização das funções hue e indicator em todos pixeis, podemos calcular o "centro de massa" de cada personagem.

Para a implementação desta função, inicialmente colocamos o valor 320 (número de colunas da imagem), no registo s4. Depois, inicia-se um loop, que só iria terminar, quando o programa lê-se os pixeis todos (a2 = 0) onde começamos por separar os bytes do registo a1 (endereço que lê os pixeis), em três registos diferentes, para obtermos os valores rgb do pixel; (a3 = R, a4 = G; a5 = B). Agora que já tinhamos os valores rgb do pixel, a função irá realizar um jump para a função hue, para calcular essa componente para o pixel e consequentemente, realizar a função indicator, para procurar qual o personagem que esse pixel pertence.

Como vamos realizar uma função dentro de outra, quando iniciamos o loop, alocamos espaço na stack e guardamos o return address dentro dela.

Quando a função hue retorna para a location, avançamos o a1, para o próximo pixel (adicionamos 3, porque são 3 bytes), e subtraimos 3 a a2, que é o tamanho máximo do array da imagem. Depois vimos se o pixel pertence ou não ao personagem escolhido, neste caso s0 é o registo que possui o personagem a que o pixel pertence, e o s7 é o registo que possui o personagem escolhido pelo usuário. Se estes valores forem diferentes, então significa que o pixel não pertence ao personagem escolhido, e então o programa iria pular para uma label, que apenas iria somar mais um ao contador, recuperar o return address e libertar o espaço da stack, e depois terminar ou voltar ao loop, dependendo se todos os pixeis já foram lidos ou não.

Caso s0 e s7 sejam iguais, então, soma-se um valor ao registo s1. Este registo conta o número de pixeis do personagem escolhido. Depois, calcula-se o X e o Y. O X é o resultado do resto do contador por 320, e Y é o resultado da divisão do contador por 320. Por fim soma-se o X e o Y, aos valores das coordenadas do centro de massa.

Após todos os pixeis terem sido lidos, pudemos então terminar de calcular as coordenadas do centro de massa, realizando a divisão dos mesmos pelo número de pixeis do personagem, que estava guardado no registo s1.

0.2.5 Função: write rgb image

Objetivo: cria um ficheiro novo com uma imagem no formato RGB com uma mira em forma de cruz sobreposta no centro de massa do personagem escolhido.

Para implementação desta função, iniciamos por abrir o ficheiro, mas desta vez colocar no modo escrita, para isso utilizamos o comando syscall, e colocamos o registo a7 = 1024 e o a1 = 1. De seguida, colocamos no a1, o tamanho do array onde serião lido os pixeis, e guardamos a descrição do ficheiro em s6.

Antes de desenharmos a cruz, ainda colocamos no registo t6, o valor do fim do array dos pixeis, que é a soma de a2 (tamanho do array), com a1 (endereço do pixeis array, onde são lidos os pixeis), e colocamos as variaveis X, Y e contador, a 0.

Para desenhar a cruz, inicialmente calculamos o X e o Y, da mesma forma que calculamos na função location, só que depois disso comparamos com os valores do centro de massa. Se o valor de X for igual ao valor da coordenada

Y do centro de massa, então a função é enviada para uma label nomeada de Cruz 2. De forma análoga, acontece se o valor de Y for igual ao valor da coordenada X do centro de massa, só que o programa será enviado para a label Cruz 3.

Nas labels Cruz 2 e Cruz 3, incialmente são inseridos em 2 registos diferentes os valores da coordenada do centro de massa do personagem, (X para Cruz 2, Y para Cruz 3). Depois a um dos registos adiciona-se 10 e noutro subtrai-se 10, e depois se X/Y tiverem no intervalo desses valores obtidos então um pixel verde será pintado, de forma a que quando todos pixeis forem lidos, uma cruz é desenhada.

Caso o X/Y não pertençam ao intervalo, então o programa irá avançar para o próximo pixel (somar 3 ao a1), o contador irá avançar, e se todos os pixéis não tiverem sido lidos, então o programa retorna ao loop.

Após os pixéis terem sido todos lidos, novamente o ficheiro coloca os valores para o pixeis arrays, syscall (a7 = 64), e inserimos em a1 o endereço onde serão lidos os pixeis da imagem. Por fim, o ficheiro fecha, com o comando syscall (a7 = 57) e o endereço a0 retorna a função, para termos "feedback" se havia algum erro (a0 = -1) ou não.

0.2.6 Função: main

Objetivo: Pede ao utilizador para escolher um personagem e depois executar as funções implementadas.

Para finalizar temos a função main, esta que foi a primeira a começar a ser implementada.

Incialmente colocamos os valores 1 e 3, nos registos t0 e t1, respetivamente.

Depois executamos a frase de inicio que pede ao utilizador para digitar um número de 1 a 3, cada um correspondente a um dos personagens da imagem. Para isso igualamos o a7 a 4 e colocamos o a0 com o enderenço onde seria lido a string.

Depois o programa inicia um loop até o utilizador escolher um personagem válido, para isso utilizamos o "scanf" do assembly, que é o comando syscall (a7 = 5).

Quando o personagem é válido. O número do personagem escolhido será movido para o registo s7, que depois será utilizado na função location, para comparar com o registo s0 que é o retorno da função indicator, para identificar se o pixel pertence ao personagem escolhido.

De seguida, colocamos o endereço da imagem de input em a0, para depois iniciar-se a função read rgb image, e depois desta retornar, executar a location.

Após o cálculo do centro de massa do personagem escolhido, a partir da função location, colocamos o enderenço da imagem de output para realizar a função write rgb image, para então desenhar uma nova imagem igual à original, mas com uma cruz a mirar no personagem escolhido.

0.3 Conclusão

Com este trabalho, podemos concluir, que conseguimos aplicar todos os conhecimentos aprendidos ao longo deste semestre, em Arquitetura de Computadores I.

Incialmente, tivemos alguma dificuldade, ao realizar o código, visto que nunca tinhamos mexido com manipulação de imagens rgb, mas pensamos que foi bem formulado e desenvolvido, uma vez que ao longo do tempo essa dificuldade foi diminuindo e conseguimos deixar o trabalho funcional.

Por fim, apesar de desafiador, gostamos de realizar este trabalho, e pretendemos melhorar os nossos conhecimentos e adquirir mais, futuramente.