Universidade de Brasília

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Organização e Arquitetura de Computadores - Turma C

Relatório - Trabalho III: Programação Assembler

Aluno:
Lucas Santos - 14/0151010

Professor:
Ricardo Jacobi

25 de setembro de 2016

1 Objetivo

O trabalho consiste no desenvolvimento de algumas funções para manipulação de imagens, em assembler do *MIPS*. As operações a serem implementadas são:

- **read_img**: Leitura de um arquivo binário com uma descrição de uma figura colorida, no formato RGB utilizado pelo *MIPS*;
- **load_img**: Exibição da figura no mostrador gráfico do *MIPS*;
- **get_pixel**: Leitura de pixels da imagem;
- **set_pixel**: Escrita de pixels na imagem;
- **grey**: Conversão para tons de cinza.

Estas operações deverão ser disponibilizadas a partir de uma interface textual com o usuário. O arquivo contendo a imagem deve chamar-se **"image.raw"**.

Figura 1: Imagem **"image.raw"** utilizada para executar o código.

2 Documentação do Código

O código foi desenvolvido em forma de *labels*, com o objetivo de aumentar a modularizaração. As funções implementadas são descritas a seguir:

2.1 *menu*

Representa a interface textual com o usuário, onde as operações estão designadas da seguinte forma:

```
********* RAW Images in MIPS *********
1 - Load Img
2 - Get Pixel
3 - Set Pixel
4 - 255 Grey Scale
5 - Exit
```

De acordo com a escolha do usuário, o programa executa uma operação. Se a escolha for 1, a função **read_img** é executada; Se 2, a função **get_pixel** é executada; Se 3, a função **set_pixel** é executada; Se a escolha for 4, a função **grey** é executada; E finalmente, se 5, o programa é encerrado.

Algumas variáveis globais são definidas nesta *label*, são elas:

- Endereço inicial do *buffer*, armazenado no registrador $s1;
- Endereço inicial do *display*, armazenado no registrador $s2;
- Contador para realização de *loops*, armazenado no registrador $t0;
- Tamanho, em bytes, de um pixel no arquivo **"image.raw"**, armazenado no registrador $t1;

2.2 *read_img*

Realiza a abertura do arquivo **"image.raw"**, e executa a *label* **load_img**, armazenando o *File Descriptor* no registrador $s0.

2.3 *load_img*

Realiza um laço, até o fim do arquivo **"image.raw"**, onde um pixel é lido do arquivo (1 pixel = $0x$RGB), e armazena o pixel lido no *display* utilizando o formato RGB do *MIPS* (1 pixel = $0x0$RGB).

Utilizando o *File Descriptor*, os 3 bytes que representam RGB são armazenados no $s1, o valor 0 é armazenado no byte mais significativo de $s1, após estas operações, o valor de $s1 é armazenado no *display*. As variáveis $t0 (contador) e $s2 (endereço do display) são incrementadas para o percorrimento do arquivo de leitura e gravação no *display*, respectivamente.

Figura 2: Resultado do *load_img* aplicado na imagem **"image.raw"**.

2.4 *close_img*

Realiza o fechamento do arquivo **"image.raw"**, utilizando o *File Descriptor* no registrador $s0.

2.5 *get_pixel*

Realiza a leitura dos valores RGB de um pixel da imagem especificado pelo usuário, onde x representa a linha, e y representa a coluna.

O valor de x obtido do usuário é multiplicado por 4 (endereços em bytes) e armazenado em $s3. O valor de y obtido do usuário é multiplicado por 4 (endereços em bytes) e depois multiplicado por 64 (pixels por linha), e finalmente armazenado em $s4.

A soma de x e y é armazenada em $t3 e somada ao endereco inicial do *display*, o resultado representa o endereço onde se encontra o pixel desejado. O pixel desejado é carregado em $t4, e os valores de RGB são exibidos na interface da seguinte forma:

```
Digite o valor da linha x (de 0 a 63): 0
Digite o valor da coluna y (de 0 a 63): 0

Valor do pixel: 0x00492f32
R: 73
G: 47
B: 50
```

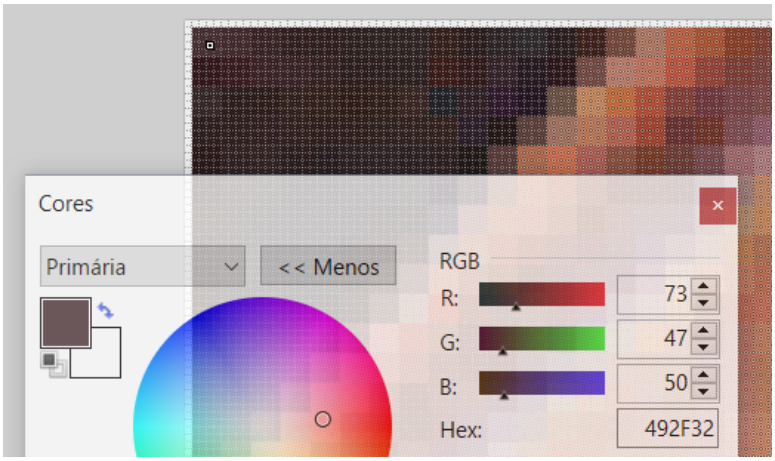

Figura 3: Resultado obtido por um editor de imagens, comprovando os valores obtidos pelo programa desenvolvido.

2.6 *set_pixel*

Realiza a escrita dos valores RGB de um pixel da imagem especificado pelo usuário, onde x representa a linha, e y representa a coluna.

O valor de x obtido do usuário é multiplicado por 4 (endereços em bytes) e armazenado em $s3. O valor de y obtido do usuário é multiplicado por 4 (endereços em bytes) e depois multiplicado por 64 (pixels por linha), e finalmente armazenado em $s4.

A soma de x e y é armazenada em $t3 e somada ao endereco inicial do *display*, o resultado representa o endereço onde se encontra o pixel desejado. O pixel desejado é carregado em $t4.

Os valores de RGB que o usuário deseja alterar são lidos do teclado e armazenados em $t4, colocando o 0 no byte mais significativo, e depois são passados para o endereço relativo do *display*, armazenado em $t3. Na interface com o usuário a operação é representada da seguinte forma:

```
Digite o valor da linha x (de 0 a 63): 63
Digite o valor da coluna y (de 0 a 63): 63

Valor do pixel: 0x009f5645
Digite o valor de R (de 0 a 255): 255
Digite o valor de G (de 0 a 255): 255
Digite o valor de B (de 0 a 255): 255

Valor do pixel: 0x00ffffff
```

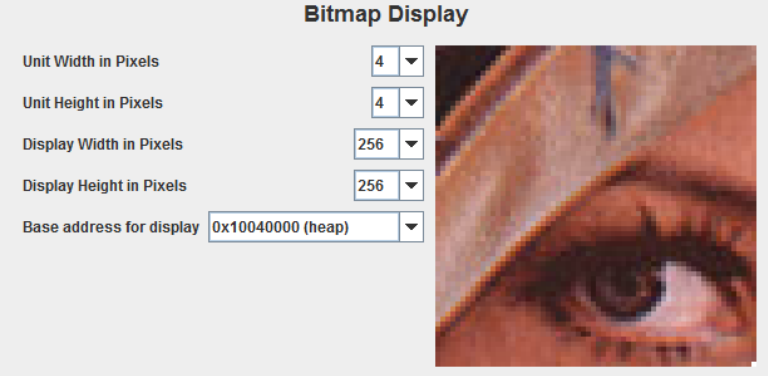

Figura 4: Resultado da operação no *display*.

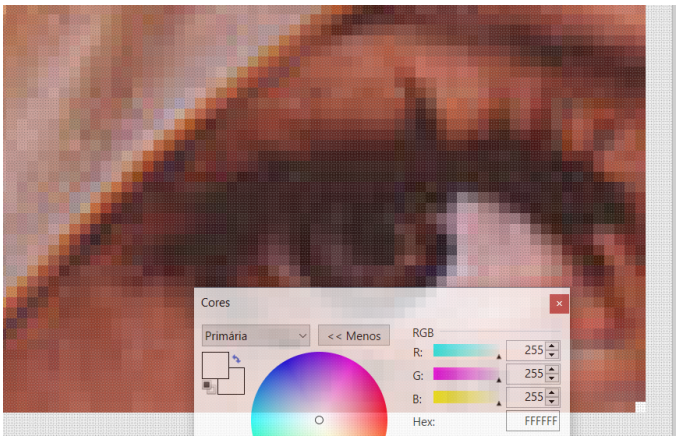

Figura 5: Resultado obtido por um editor de imagens, comprovando os valores obtidos pelo programa desenvolvido.

2.7 *grey*

Realiza um laço, até o fim do arquivo **"image.raw"**, onde um pixel armazenado no *display* é transformado para tons de cinza.

O valor de B é armazenado em $t2, o de G em $t3, e o de R em $t4, o cálculo da média entre estes valores é armazenado em $t2. O byte $t2 é armazenado no endereço correspondente no *display* nos bytes $0x0bbb$, e as variáveis $t0 (contador), $s1 (endereço de leitura do display) e $s2 (endereço de escrita do display) são incrementadas para o percorrimento do arquivo de leitura, leitura do *display* e gravação no *display*, respectivamente.

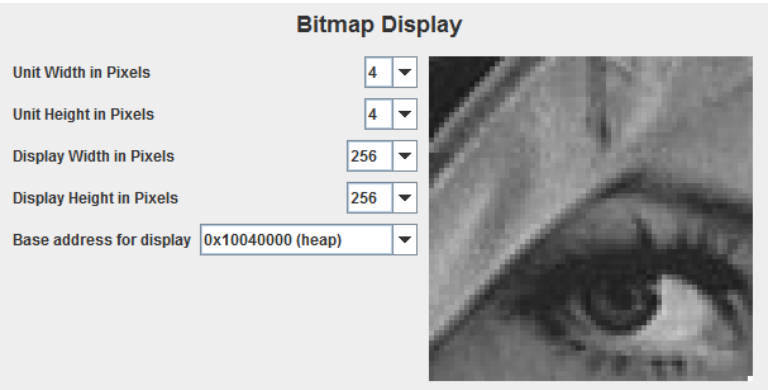

Figura 6: Resultado da operação no *display*.

2.8 *erro_pixel*

Tratamento de erros, caso: O usuário digite uma linha ou coluna inválida; O usuario digitou um valor de RGB invalido. O programa volta para o menu.

2.9 *erro_arquivo*

Encerra o programa, caso a leitura do arquivo resulte em erro.

2.10 *exit*

Encerra o programa.