



UNIVERSIDADE DE ÉVORA

Licenciatura em Engenharia Informática

Remoção de ruído de imagens .GRAY em assembly



Tiago Pedro nº 57948
Bruno Moreira nº 59658

30 de Maio de 2024
Ano Letivo 2023/2024

Arquitetura de Computadores I
Professor Doutor Miguel Barão

Índice

1	Introdução	3
2	Funções	4
2.1	Função readGray	4
2.2	Função writeGray_mean	5
2.3	Função writeGray_median	6
2.4	Função meanFilter	7
2.5	Função medianFilter	9
2.6	Função main	11
3	Resultados	12

1 Introdução

O projeto final da unidade curricular de Arquitetura de Computadores I teve como objetivo a criação de um programa, no qual é aplicado o filtro da média e mediana a uma imagem .GRAY, em Assembly, utilizando os conhecimentos obtidos em aula ao longo do semestre e demonstrando a compreensão sobre o assunto e a capacidade da sua aplicação prática. O seguinte relatório tem como objetivo explicar a implementação das funções ao longo do código que constitui este projeto final.

2 Funções

2.1 Função readGray

A função 'readGray' lê uma imagem .GRAY a partir de um ficheiro e guarda os dados num array. A função usa os seguintes registos:

- a0, a1, a2 e a7 — Registos usados para argumentos.
- s6 e s1 — Registos usados para guardar o valor dos argumentos de chamada de função chamadora (nome do ficheiro e array da imagem, respetivamente).

As etapas da função seguem a seguinte ordem:

1. Alocar espaço na *stack*.
2. Guardar s6 e s1 na *stack*.
3. Mover a1 (array de imagem) para s1.
4. Abrir ficheiro para leitura com *system call* 1024 carregada em a7.
5. Mover a0 para s6 (para preservar o descritivo da imagem).
6. Verificar se houve algum erro a ler o ficheiro e saltar para a *label* readGray_error.
7. Em caso de sucesso, ler ficheiro com *system call* 63 carregada em a7.
8. Mover o descritivo (s6) para a0.
9. Mover o array da imagem (s1) para a1.
10. Carregar a2 com o tamanho do array da imagem - 239600 bytes.
11. Verificar se houve algum erro a ler o ficheiro com a *label* readGray_error.
12. Em caso de sucesso, fechar o ficheiro com *system call* 57 carregada em a7.
13. Imprimir na consola uma mensagem de sucesso com *system call* 4 carregada em a7.
14. Saltar para readGray_done, que desempilha os registos utilizados da *stack* e retorna à função chamadora.

No caso de ocorrer um erro, o programa salta para a *label* readGray_error, que mostra uma mensagem de erro na consola com *system call* 4, e salta para readGray_done.

2.2 Função `writeGray_mean`

A função '`writeGray_mean`' escreve uma imagem `.GRAY` a partir de dados de um array para o filtro da média. A função usa os seguintes registos:

- `a0`, `a1`, `a2` e `a7` — Registos usados para argumentos.
- `s6` e `s1` — Registos usados para guardar o valor dos argumentos de chamada de função chamadora (nome do ficheiro e array da imagem, respetivamente).

As etapas da função seguem a seguinte ordem:

1. Alocar espaço na *stack*.
2. Guardar `s6` e `s1` na *stack*.
3. Mover `a1` (array de imagem) para `s1`.
4. Abrir ficheiro para escrita com *system call* 1024 carregada em `a7`.
5. Mover `a0` para `s6` (para preservar o descritivo da imagem).
6. Verificar se houve algum erro a escrever em ficheiro e saltar para a *label* `writeGray_error`.
7. Em caso de sucesso, escrever no ficheiro com *system call* 64 carregada em `a7`.
8. Mover o descritivo para `a0`.
9. Mover o array da imagem para `a1`.
10. Carregar `a2` com o tamanho do array da imagem - 239600 bytes
11. Verificar se houve algum erro a escrever em ficheiro e saltar para a *label* `writeGray_error`.
12. Em caso de sucesso, fechar o ficheiro com *system call* 57 carregada em `a7`.
13. Imprimir na consola uma mensagem de sucesso com *system call* 4 carregada em `a7`.
14. Saltar para `writeGay_done`, que desempilha os registos utilizados da *stack* e retorna à função chamadora.

No caso de ocorrer um erro, o programa salta para a *label* `writeGray_error`, que mostra uma mensagem de erro na consola com *system call* 4, e salta para `writeGray_done`.

2.3 Função `writeGray_median`

A função '`writeGray_median`' escreve uma imagem `.GRAY` a partir de dados de um array para o filtro da mediana. A função usa os seguintes registos:

- `a0`, `a1`, `a2` e `a7` — Registos usados para argumentos.
- `s6` e `s1` — Registos usados para guardar o valor dos argumentos de chamada de função chamadora (nome do ficheiro e array da imagem, respetivamente).

As etapas da função seguem a seguinte ordem:

1. Alocar espaço na *stack*.
2. Guardar `s6` e `s1` na *stack*.
3. Mover `a1` (array de imagem) para `s1`.
4. Abrir ficheiro para escrita com *system call* 1024 carregada em `a7`.
5. Mover `a0` para `s6` (para preservar o descritivo da imagem).
6. Verificar se houve algum erro a escrever em ficheiro e saltar para a *label* `writeGray_median_error`.
7. Em caso de sucesso, escrever no ficheiro com *system call* 64 carregada em `a7`.
8. Mover o descritivo para `a0`.
9. Mover o array da imagem para `a1`.
10. Carregar `a2` com o tamanho do array da imagem - 239600 bytes.
11. Verificar se houve algum erro a escrever em ficheiro e saltar para a *label* `writeGray_median_error`.
12. Em caso de sucesso, fechar o ficheiro com *system call* 57 carregada em `a7`.
13. Imprimir na consola uma mensagem de sucesso com *system call* 4 carregada em `a7`.
14. Saltar para `writeGay_median_done`, que desempilha os registos utilizados da *stack* e retorna à função chamadora.

No caso de ocorrer um erro, o programa salta para a *label* `writeGray_median_error`, que mostra uma mensagem de erro na consola com *system call* 4, e salta para `writeGray_median_done`.

2.4 Função meanFilter

A função 'meanFilter' aplica o filtro da média a uma imagem .GRAY. A função usa os seguintes registros:

- a0 — Armazena o nome do ficheiro a aplicar o filtro.
- a1 — Armazena o array com os dados da imagem.
- s0-s5 — Armazenam variáveis constantes e para controlar loops.
- t0-t6 — Armazenam valores importantes no decorrer da função.
- s7 e s8 — Espaço que a imagem ocupa em memória, 2396000 bytes.

As etapas da função seguem a seguinte ordem:

1. Guardar o comprimento da imagem (400) em t0.
2. Carregar t0 em s1.
3. Guardar a largura da imagem (599) em t0.
4. Carregar t0 em s2.
5. Guardar o tamanho do lado do kernel (3) em t0.
6. Carregar t0 em s3.
7. Guardar 0 em s4 — contador i.
8. Iniciar um loop for que incrementa s4 de 0 até 399 — s1 - 1.
9. Guardar 0 em s5 — contador j.
10. Iniciar um loop for que incrementa s5 de 0 até 598 — s2 - 1.
11. Guardar 0 em t1 — soma.
12. Guardar 0 em t2 — contador m.
13. Iniciar um loop for que incrementa t2 de 0 até 2 — s3 - 1.
14. Guardar t3 como a soma de s4 e t2 — linha do *kernel*.
15. Iniciar um loop for que verifica se $t3 < 0$ e $t3 \geq s1$.
16. Guardar 0 em t4 — contador n;
17. Guardar t5 como a soma de s5 e t4 — coluna do *kernel*.
18. Iniciar um loop for que verifica se $t5 < 0$ e $t5 \geq s2$.

19. Calcular o endereço atual do array da imagem de input e guardar esse valor em t6.
20. Carregar a2 com o valor de t6.
21. Adicionar à soma o valor de a2.
22. Calcular o endereço atual do array da imagem de output e guardar esse valor em t6.
23. Calcular a média ao dividir o valor da soma por 9 (lado do *kernel*).
24. Guardar o valor da média em cada pixel da imagem de output.
25. Retornar à função chamadora.

2.5 Função medianFilter

A função 'medianFilter' aplica o filtro da mediana a uma imagem .GRAY. A função usa os seguintes registros:

- a0 — Armazena o nome do ficheiro a aplicar o filtro.
- a1 — Armazena o array com os dados da imagem.
- s0-s5 — Armazenam variáveis constantes e para controlar loops.
- t0-t6 — Armazenam valores importantes no decorrer da função.
- s7 e s8 — Espaço que a imagem ocupa em memória, 2396000 bytes.

As etapas da função seguem a seguinte ordem:

1. Alocar espaço na *stack*.
2. Guardar ra e s0-s10 na *stack*.
3. Guardar o comprimento da imagem (400) em t0.
4. Carregar t0 em s0.
5. Guardar a largura da imagem (599) em t0.
6. Carregar t0 em s1.
7. Guardar o tamanho do lado do kernel (3) em t0.
8. Carregar t0 em s2.
9. Guardar $(s2 * s2)$ em s3.
10. Guardar 0 em s4 — contador i.
11. Iniciar um loop for que incrementa s4 de 0 até 399 — $s1 - 1$.
12. Guardar 0 em s5 — contador j.
13. Iniciar um loop for que incrementa s5 de 0 até 598 — $s2 - 1$.
14. Guardar 0 em s6 — soma.
15. Guardar 0 em s9 (endereço da janela da mediana, usado para calcular a mesma).
16. Guardar 0 em t1 — contador m.
17. Iniciar um loop for que incrementa t1 de 0 até 2 — $s3 - 1$.
18. Guardar t2 como a soma de s4 e t1 — linha do *kernel*.

19. Iniciar um loop for que verifica se $t2 < 0$ e $t2 \geq s0$.
20. Guardar 0 em $t3$ — contador n ;
21. Guardar $t4$ como a soma de $s5$ e $t3$ — coluna do *kernel*.
22. Iniciar um loop for que verifica se $t4 < 0$ e $t4 \geq s1$.
23. Calcular o endereço atual do array da imagem de input e guardar esse valor em $t5$.
24. Carregar $t6$ com o valor de $t5$.
25. Guardar $t6$ no array de janela do kernel (guardado em $s11$).
26. Incrementar $s9$.
27. Percorrer a janela do *kernel* e organizar os valores com um algoritmo de *bubble sort*, de modo a organizá-los por ordem crescente.
28. Calcular o valor da mediana e guardar esse valor em $t2$.
29. Calcular o endereço atual do array da imagem de output e guardar esse valor em $t3$.
30. Carregar o valor da mediana em cada pixel da imagem de output.
31. Retornar à função chamadora.

2.6 Função main

A função 'main' é responsável pela chamada das funções de escrita e leitura de ficheiros, bem como as funções que aplicam os filtros de média e mediana à imagem .GRAY. A função usa os seguintes registos:

- a0, a1, a7 — Registos usados para argumentos.
- s7 e s8 — Registos usados para guardar o espaço que a imagem ocupa em memória, 2396000 bytes.

As etapas da função seguem a seguinte ordem:

1. Carregar em s7 o espaço do array da imagem de input (239600 bytes).
2. Carregar em s8 o espaço do array da imagem de output (239600 bytes).
3. Carregar em a0 o nome do ficheiro da imagem a ler.
4. Carregar em a1 o espaço da imagem de input (239600 bytes).
5. Chamar a função readGray para ler a imagem.
6. Chamar a função meanFilter, e aplicar o filtro da média.
7. Carregar em a0 o nome do ficheiro da imagem de média.
8. Carregar em a1 o espaço da imagem de output (239600 bytes).
9. Chamar a função writeGray_mean para escrever a imagem de média.
10. Chamar a função medianFilter, e aplicar o filtro da mediana.
11. Carregar em a0 o nome do ficheiro da imagem de mediana.
12. Carregar em a1 o espaço da imagem de output (239600 bytes).
13. Chamar a função writeGray_median para escrever a imagem de mediana.
14. Sair do programa com a system call 10 carregada em a7.

3 Resultados

Após a execução do programa são produzidas duas imagens .GRAY, que convertidas para .PNG usando ImageMAgick, têm o seguinte aspeto:



Figure 1: Imagem original, com ruído para remover.



(a) Imagem com filtro da média aplicado.



(b) Imagem com filtro de mediana aplicado.