Arquitetura de Computadores I
Licenciatura em Engenharia Informática

Ano Letivo 2023–2024

1 Objectivo

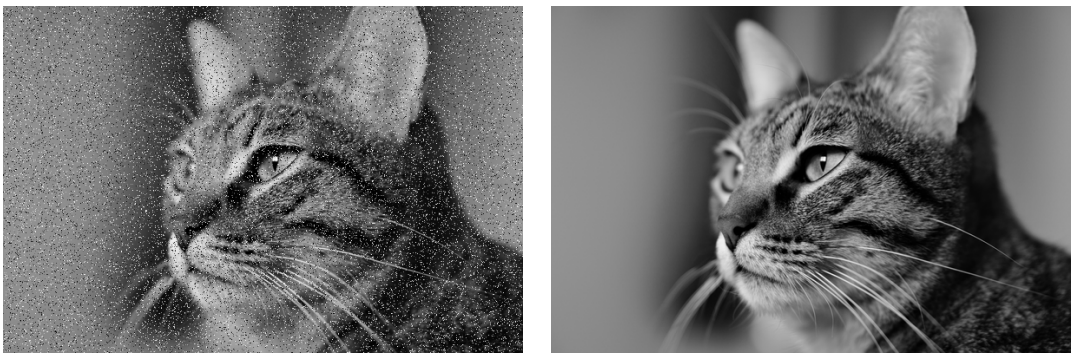
Pretende-se desenvolver um conjunto de funções em assembly RISC-V para remover ruído em imagens em tons de cinzento. Dado um ficheiro com uma imagem, o programa deverá abrir a imagem, correr um algoritmo de remoção de ruído (*denoising*) e guardar a imagem resultante noutro ficheiro com o mesmo formato e dimensão.

Na figura 1 ilustra-se o que se pretende obter. A figura 1(a) contém a imagem de um gato corrompida por ruído. O programa deverá processar esta imagem e gerar uma nova imagem o mais parecida possível com a 1(b), sem ruído.

2 Representação de imagens monocromáticas

Uma imagem em tons de cinzento consiste numa matriz de píxeis onde cada elemento da matriz contém a intensidade de um píxel:

$$\begin{bmatrix} I_{11} & I_{12} & \cdots & I_{1n} \\ I_{21} & I_{22} & \cdots & I_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ I_{m1} & I_{m2} & \cdots & I_{mn} \end{bmatrix}.$$



(a) Com ruído.

(b) Sem ruído.

Figura 1: Imagens em tons de cinzento com dimensão 599×400 píxeis.

Para guardar a imagem em memória, é necessário guardar os píxeis num *array* em posições consecutivas. Uma maneira possível é organizá-los sequencialmente em linhas:

$$\big[\underbrace{I_{11}\cdots I_{1n}}_{1^{\text{a}}\text{ linha}}\ \underbrace{I_{21}\cdots I_{2n}}_{2^{\text{a}}\text{ linha}}\cdots\ \underbrace{I_{m1}\cdots I_{mn}}_{m\text{-ésima linha}}\big].$$

Esta forma de organização denomina-se *row major* e é a usada na linguagem C. Outra organização possível, usada noutras linguagens, é a *column major* onde a arrumação dos elementos segue ao longo de colunas.

3 Formatos de ficheiros de imagem

A representação de uma imagem em memória consiste num *array* de píxeis e noutras variáveis contendo os parâmetros da imagem, tais como as dimensões e a profundidade de cor (número de bits por píxel).

Quando se guarda a imagem num ficheiro, todos os dados têm obrigatoriamente de ser guardados consecutivamente. A maneira como a informação é organizada e codificada designa-se genericamente por formato do ficheiro. Existem diversos formatos para ficheiros de imagem: JPEG, PNG, BMP, GIF, TIFF, PGM, PBM, *etc.*

Neste trabalho vamos usar o formato GRAY, que consiste simplesmente em escrever as linhas da imagem consecutivamente (*row major*). Os ficheiros GRAY não contêm informação acerca do tamanho da imagem nem do número de bits por píxel, essa informação é perdida na conversão. Para conseguir descodificar uma imagem neste formato, é necessário conhecer previamente as características da imagem.

A conversão entre formatos de imagem pode ser realizada com o programa `convert` disponibilizado pelo pacote de software `ImageMagick` em Linux[1]. Por exemplo, a conversão entre os formatos PNG e GRAY pode realizar-se com os comandos:

```
convert imagem.png imagem.gray                          # PNG  -> GRAY
convert -size 599x400 -depth 8 imagem.gray imagem.png   # GRAY -> PNG
```

No segundo caso, correspondente à conversão de GRAY para PNG, é necessário passar como opções a dimensão da imagem `-size 599x400`, e a profundidade de cor `-depth 8`, uma vez que o ficheiro GRAY não contém essa informação.

4 Processamento de imagem

O processamento de imagem consiste essencialmente em transformar uma imagem A numa imagem B processada. Os algoritmos usados nessa transformação variam consoante o objetivo que se pretende atingir. No nosso caso, pretendemos transformar imagens com ruído em imagens com o mínimo de ruído possível, operação essa chamada *denoising*.

Nas secções 4.1 e 4.2 apresentam-se duas técnicas de remoção de ruído, o filtro de média e o filtro de mediana. A figura 2 mostra os resultados obtidos com cada um dos filtros.

[1] Instalação no Ubuntu: `sudo apt install imagemagick`

(a) Filtro de média.

(b) Filtro de mediana.

Figura 2: Resultados obtidos com a aplicação dos filtros.

4.1 Filtro de média

Consideremos uma imagem ruidosa A, em tons de cinzento, como a da figura 1(a). O ruído na imagem consiste essencialmente em píxeis que foram corrompidos, apresentando intensidades adulteradas. Uma técnica rudimentar de remoção de ruído, chamada *filtro de média*, consiste em substituir cada píxel pela média dos píxeis numa vizinhança à sua volta. A ideia é que, como os píxeis vizinhos têm tipicamente intensidades parecidas, a média terá o efeito de substituir os píxeis de ruído por valores mais perto das intensidades dos píxeis vizinhos.

Matematicamente, o filtro de média consiste em definir um *kernel*, a matriz 3×3

$$M = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix},$$

e aplicá-lo do seguinte modo: para calcular o píxel $B(i, j)$ da imagem filtrada, sobrepõe-se a matriz M à imagem original A centrada na posição (i, j) e multiplicam-se os pesos da matriz pelos píxeis que ficam "por baixo". Somando os 9 resultados, obtém-se a média que se guarda na imagem filtrada. O processo repete-se para cada píxel da imagem.

Esta operação chama-se *convolução 2D* e representa-se por um asterisco (não é um produto matricial):

$$B = M * A.$$

Cada elemento B_{ij} da matriz B é obtido pelo *somatório de convolução*

$$B(i, j) = \sum_{p=-1}^{1} \sum_{q=-1}^{1} A(i + p, j + q)\, M(2 - p, 2 - q).$$

A imagem 2(a) mostra o resultado da aplicação do filtro de média. Comparando com a imagem original ruidosa 1(a), verificamos que esta técnica esbate um pouco o ruído, mas também esbate a própria imagem (tente perceber a razão).

4.2 Filtro de mediana

Outra técnica de remoção de ruído consiste em usar um *filtro de mediana*. O filtro de mediana é semelhante ao filtro de média, mas em vez de substituir cada píxel pela média da vizinhança, substitui pela mediana.

Dados 9 píxeis da vizinhança, a mediana é um valor x tal que existem tantos valores maiores como menores do que x. Um método para obter a mediana consiste em ordenar os valores e escolher o que fica no meio. Por exemplo, se a vizinhança de um píxel for formada pelos valores

$$\begin{bmatrix} 97 & 101 & 99 \\ 96 & 149 & 102 \\ 89 & 93 & 94 \end{bmatrix},$$

então ordenando os valores obtemos $(89, 93, 94, 96, \mathbf{97}, 99, 101, 102, 149)$. A mediana é o valor do meio 97. Este processo repete-se para cada píxel da imagem.

A imagem 2(b) mostra o resultado da aplicação do filtro de mediana. Como se pode observar, a imagem produzida não apresenta o esbatimento do filtro de média e o resultado é consideravelmente melhor.

5 Implementação do trabalho

O programa deve estar estruturado em várias funções para as várias tarefas que tem de realizar: leitura e escrita de ficheiros, filtro de média, filtro de mediana, e outras que considerar necessárias.

Antes de começar a programar em Assembly, sugere-se que primeiro implemente o programa na linguagem C. Esta versão não é para avaliação, mas é muito útil para compreender bem o problema e estruturar o programa.

O código deverá estar comentado. Em particular, as funções deverão ter uma descrição da funcionalidade implementada e indicar os argumentos e o resultado. Veja o seguinte exemplo:

```
#########################################################
# Funcao: media
# Descricao: Esta funcao calcula a media de 2 numeros.
# Argumentos:
#   a0 - primeiro numero
#   a1 - segundo numero
# Retorna:
#   a0 - media
#########################################################
media:
        add a0, a0, a1
        srai a0, a0, 1
        ret
```

Para realizar I/O, nomeadamente leitura e escrita de ficheiros, consulte a ajuda do simulador RARS. Contém lá exemplos de código para realizar essas operações, que poderá adaptar para o seu trabalho.

O trabalho deverá ser acompanhado de um relatório, escrito em LaTeX. O relatório deverá explicar a estrutura do programa (sem incluir código) e mostrar os resultados obtidos, incluindo a imagem original, com ruído, e as obtidas com o filtro de média e mediana.

6 Submissão dos trabalhos

- Os trabalhos devem ser realizados em **grupos de 2 alunos**.
- Cada grupo deve submeter, no moodle, um único ficheiro comprimido contendo o código Assembly RISC-V, as imagens geradas e o relatório em PDF.
- O nome do ficheiro comprimido deve ser formado pelos números dos alunos por ordem crescente, por exemplo `11111-22222.zip`.
- Apenas um dos alunos do grupo deve submeter o trabalho.
- Não é permitida a utilização de ferramentas que automatizem a escrita de código, tais como compiladores para RISC-V ou inteligência artificial como o chatGPT.
- Não é permitido partilhar código com outros grupos ou usar código já existente.
- Qualquer situação de fraude implica a reprovação à unidade curricular e será obrigatoriamente reportada para instauração de procedimento disciplinar, conforme estipulado no regulamento académico.
- O trabalho é longo. Devem começar já!

Bom trabalho! 👍
Miguel Barão