Arquitectura de Sistemas e Computadores I

Docente: Miguel Barão

Trabalho elaborado por:
Rúben Teimas - nº 39868
Pedro Claudino - nº 39870

Índice

Introdução

Neste trabalho foi-nos pedido que desenvolvessemos um conjunto de funções em assembly MIPS com a finalidade de remover o ruído de uma imagem em Gray Scale. O trabalho deverá receber uma imagem nesse formato, ler a respetiva imagem para um buffer, aplicar o algoritmo de remoção de ruído, e por ultimo escrever a nova imagem num novo ficheiro.
Para a realização do trabalho foi-nos facultada uma imagem em formato ***.png*** e de dimensões **512x512** mas, devido a alguns problemas, acabou por ser usada uma imagem 4x4 de modo a facilitar o *debugging* e a encontrar esses mesmos problemas.

Desenvolvimento

Foram criados 3 buffers de acordo com as dimensões da imagem (512x512).
No main é carregada a imagem original para de seguida serem efetudadas as chamadas das funções.

Funções:

1. Read gray image
   - Guarda o "file discriptor" no endereço $v0, coloca no $a1 o endereço do buffer para onde é lida a imagem, no $a2 é guardado o número de bytes a ler de seguida a imagem é lida e guardada no endereço $a1 novamente por fim o ficheiro é fechado.
2. Write gray image
   - É efetuada a escrita da imagem num ficheiro em formato "Gray", que está divídida em três partes que são abrir, escrever e por fim fecha o ficheiro.
3. Mean filter
   - Esta função começa por guardar o pixel inicial de coordenadas (2,2) no endereço $s0, de seguida calcula o pixel da penultima coluna e penultima linha e guarda esse valor no endereço $s5.

     Com o pixel inicial é somado o valor de cada pixel para a matrix 3x3 e ao fim de uma linha da matrix é feito um salto de 512 pixeis para mudar para a próxima linha.

     De seguida o somatório de todos os pixeis da matrix é divídido por nove e é guardado no registo $s4, de seguida é executada uma verificção num jump chamado **border**, que serve para para colocar ambos os buffers na posição correta caso se encontrem na margem mais à direita, soma dois ao pixel corrente para ignorar a margem esquerda e direita..

Por fim entra na condição cycle que irá verificar enquanto o buffer for mais pequeno que o endereço $s5 o ciclo continua a correr.

4. Median filter

   - Criado um array de nove posições para a organição dos pixeis da matriz, é retirado o quinto pixel que corresponde à mediana.

     Guardado também esse mesmo pixel para a imagem é executada também uma verificação de margem para limitar o alcance da matriz, de seguida é começado um ciclo para percorrer todas as posições do buffer da imagem.

Funções Auxiliares

1. Sort

   - Nesta função auxiliar é passado o array onde os pixeis anteriores estão guardados e irá ordena-los por ordem crescente recorrendo a um bubble sort, pois, embora não seja o algoritmo de ordenação com a melhor complexidade, pareceu-nos ser o mais fácil de implementar.

     Existem três jumps nesta função auxiliar, um chamado "sort cycle" é o ciclo que percorre o array e que começa por verificar se o array original já está organizado, caso não esteja, irá entrar no jump swap, que será responsável por trocar os 2 valores que se encontram em ordem contrária. Após o swap, o array volta à posição inicial e começa a verificar novamente a ordenação do array

     O último jump denominado "out" é executado quando o array chega à ultima posição, ou seja, está ordenado. Ao entrar nesse jump, volta à posição inicial do array e devolve a 5º posição do array. Por fim, volta para a função Median Filter.

Imagens

Imagem com filtro de media.

Imagem com filtro de mediana.

Conclusão

Após a realização deste trabalho podemos afirmar que o objetivo inicial foi atingido, dado que conseguimos, a partir da imagem original, obter 2 novas imagens aplicando o filtro de média e mediana.
Depois de alguma discussão pensamos que seja correto afirmar que a melhoria de qualidade do filtro de mediana em relação ao filtro de média se deve a uma razão puramente estatistica, pois em alguns casos, como este, a mediana é um valor mais representativo do intervalo de valores do que a média.
Ainda assim, podiamos ter tido uma melhor estruturação do código de forma a evitar repetição do mesmo. Uma possivel solução seria ter uma "função central" onde seria feito load dos bytes e depois chamadas as funcoes de média e mediana, que no seu corpo teriam exclusivamente as diferenças entre si.