Trabalho Prático de Arquitectura de Computadores I

Trabalho realizado por:
José Serra 33289
Miguel Jesus 36926

Índice

Introdução

Pretende-se com este trabalho desenvolver um conjunto de funções em assembly MIPS para fazer detecção de contornos em imagens a cores RGB. Dado um ficheiro com uma imagem a cores RGB, o programa deverá gerar outro ficheiro de imagem, em tons de cinzento GRAY, com fundo branco e com traços escuros nos locais onde existem contornos na imagem original, através da utilização dos operadores Sobel.

Desenvolvimento do Trabalho

Funções:

main -

- Solução:
  Função Principal, na qual são chamadas todas as outras funções durante o decorrer do programa.
  Também contém algumas funções de controlo de erros, de forma a tornar o programa mais robusto.

read_rgb_image - esta função lê uma imagem no formato RGB para um array em memória.

- Argumentos:
  - $a0 - contém o endereço do buffer onde se encontra o conteúdo do user input.

- Retorno:
  - $v0 - o endereço do buffer que contém a imagem RGB

- Solução:
  A função abre o ficheiro introduzido pelo utilizador. De seguida, lê o ficheiro e caso exista algum tipo de erro em algum destes casos (ex: Input invalido) o retorno em $v0 será negativo, caso contrário, o retorno será o endereço do buffer que contém os dados do ficheiro.

write gray image - esta função escreve uma imagem em formato GRAY num ficheiro

- Argumentos:
  $a0 - nome do ficheiro GRAY
  $a1 - recebe o endereço de um buffer que contém dados
  $a2 - tamanho do ficheiro

- Retorno:
  void

- Solução:
  Esta função cria um ficheiro a partir do buffer obtido em contour, ou seja, este ficheiro terá a imagem final (deteção de contornos).

rgb_to_gray - esta função converte uma imagem RGB para uma imagem GRAY

- Argumentos:
  $a0 - recebe o endereço do buffer com a imagem RGB
  $a1 - recebe o endereço do buffer onde será colocada a imagem GRAY

- Retorno:
  void

- Solução:
  Esta função aplica sobre o buffer que contém a imagem RGB, a fórmula I = 0.30R+0.59G+0.11B nos respetivos Bytes. Possui um ciclo que aplica a cada três bytes(Pixel) a operação anterior cujo resultado é gravado no Buffer que no final desta função terá o formato GRAY.

Convolution - esta função calcula a convolução de uma imagem com o operador Sobel

- Argumentos:
  $a0 - recebe o endereço do buffer com a imagem GRAY
  $a1 - recebe o endereço da matriz Sobel
  $a2 - recebe o endereço do buffer onde será colocado o resultado das operações

- Retorno:
  void

- Solução:
  Em primeiro lugar, através da pilha guardámos, todos os registos que devem preservados e aos quais atribuimos os valores das matrizes Sobel, respectivamente.
  Como o operador Sobel não é aplicado nas margens considerámos apenas o interior da imagem, excluindo as margens (510x510).
  Através de um ciclo aplicámos o operador Sobel em todas as posições do buffer que contém a imagem em formato GRAY gravando o resultado das operações sobre a matriz Sobel na posição respetiva (O endereço de um valor do Buffer resultante é igual ao endereço do valor sobre qual são feitas as operações no buffer com o conteúdo do ficheiro GRAY.
  Utilizámos um contador de forma a percorrer as linhas no interior da imagem(510x510) de maneira a não operar sobre as margens.
  É feito um reset a este contador sempre que atinge a posição imediatamente antes da margem.
  Existe uma variável (posi) que nos indica a posição inicial de cada linha. Esta variável é incrementada em 512 sempre que o contador atinge a posição limite.
  Desta função resulta um buffer que contém todas as operações efetuadas pelo operador Sobel em GRAY.

Contour - esta função calcula a imagem final combinando duas imagens filtradas pelos op. Sobel

- Argumentos:
  - a0 - recebe o endereço do buffer que contém a convolução horizontal
  - a1 - recebe o endereço do buffer que contém a convolução vertical
  - a2 - recebe o endereço do buffer onde será colocado o resultado das operações, ou seja, a i imagem final.

- Retorno:

  - void

- Solução:

  Esta função recebe dois endereços dos buffers resultantes da operação de Sobel Horizontal e Vertical sobre o buffer que contém os dados do ficheiro em formato GRAY, e aplica-lhes a fórmula ½ (¼ *Bh + ¼ * Bv).
  A aplicação desta fórmula em cada Pixel é guardada através de um ciclo, num Buffer do qual resultará a imagem final.

correct_string_terminator - troca o caracter terminador de uma string ("\n") por "0".

- Argumentos:

  a0 – contém o endereço do buffer onde se encontra uma string com caractér terminador “\n”.

- Solução:

  Esta função destina-se a substituir o caractér terminador de uma String(User Input) “\n” por um caractér terminador “0”, pois os Sistemas Operativos baseados em Unix não admitem “\n” como terminador mas sim “0”.
  Através de um ciclo é percorrida a string introduzida pelo utilizador até encontrar o caractér terminador “\n” e de seguida substituí o mesmo por “0”.

Bibliografia

http://www.mrc.uidaho.edu/mrc/people/jff/digital/MIPSir.html

Conclusão

Deparámo-nos com vários desafios ao longo do desenvolvimento do trabalho, sendo o maior deles sem dúvida, o Debugging. Este aspeto da programação mostrou-se um desafio à altura em quase todas as funções que criámos, no entanto, pensamos ter ultrapassado todos os obstáculos que nos foram impostos.
Na função Convolution, tivemos alguma dificuldade em chegar à solução do problema, portanto como temos alguma facilidade em programar em JAVA e C, decidimos criar um pequeno programa em C em forma de teste. Isto ajudou-nos a chegar mais rapidamente à solução.
Pensamos ter cumprido todos os objetivos propostos pelo professor e estamos bastante satisfeitos com o resultado final.
Retiramos deste trabalho profunda aprendizagem da programação em Assembly e de alguns aspetos da Arquitectura de Computadores tais como a utilização de registos e funcionamento da memória.