

Trabalho de PSC – 2º. BIMESTRE

Objetivo

Utilização e avaliação de desempenho das instruções SSE disponíveis em processadores x86.

Introdução

Neste trabalho utilizaremos uma imagem onde cada pixel é descrito no formato RGB (vermelho, verde, azul), e implementar um algoritmo que realize algum tipo de operação sobre a imagem. Para facilitar a leitura e manipulação de imagens utilizaremos o formato TIF sem compressão.

A Figura 1(a) apresenta o conteúdo do arquivo TIF [1] para a imagem da Figura 1(b), com um pixel branco no canto superior esquerdo e outro no canto inferior direito (lembre-se a cor preta é representada por R= 00, G= 00 e B= 00 e a branca por R= FF, G= FF e B= FF) .

Os oitos primeiros bytes formam o cabeçalho do arquivo da seguinte forma:

- Os dois primeiros bytes representam o formato, sendo II para little endian ou MM para big endian.
- Os próximos dois bytes devem ser um número entre 0 e 0x2a. No exemplo o valor é 0x002a.
- Os próximos quatro bytes representam o deslocamento do início do arquivo até a primeira entrada no diretório da imagem, denominado de IFD (image file directory).
- O vetor contendo os pixels no formato RGB relativo à imagem é armazenado logo após do cabeçalho. Note que, o número total de bytes que representa a imagem pode ser obtida pela fórmula $I_{total_bytes} = \text{deslocamento} - 8$.


<pre>4d4d 002a 0000 ea68 ffff ff00 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 black 0's deleted 0000 0000 00ff ffff 000e 0100 0003 0000 0001 0064 0000 0101 0003 0000 0001 00c8 0000 0102 0003 0000 0003 0000 eb16 0103 0003 0000 0001 0001 0000 0106 0003 0000 0001 0002 0000 0111 0004 0000 0001 0000 0008 0112 0003 0000 0001 0001 0000 0115 0003 0000 0001 0003 0000 0116 0003 0000 0001 00c8 0000 0117 0004 0000 0001 0000 ea60 0118 0003 0000 0003 0000 eb1c 0119 0003 0000 0003 0000 eb22 011c 0003 0000 0001 0001 0000 0153 0003 0000 0003 0000 eb28 0000 0000 0008 0008 0008 0000 0000 0000 00ff 00ff 00ff 0001 0001 0001</pre>	
(a)	(b)

Figura 1- Imagem representada no formato TIF [4]

Logo após o final da seqüência de bytes representando a imagem, os dois bytes subsequentes representam o número de entradas no diretório da imagem. No exemplo o número de entradas é 0x000e, ou seja, 14 entradas.

Cada entrada é composta por 12 bytes, sendo o formato descrito na Figura 2.

0100	0003	0000 0001	0064 0000
TAG= cada valor representa um tipo de informação sobre a imagem. No caso o valor 0x0100 representa a largura	Informa o tipo de dado, 0001 =byte, 0002= ASCII, 0003= short int, 0004= int	Representa o número de valores representado por essa entrada	O valor da entrada. No exemplo é 0x00000064

Figura 2- Formato de uma entrada no diretório da imagem

Logo, a entrada da Figura 2 representa a largura da imagem que é 100 pixels (0x64). Os demais tags são descritos representam os seguintes parâmetros da imagem:

- 0100 - Image width
- 0101 - Image height
- 0102 - Bits per sample (8)
- 0103 - Compression method (1 = uncompressed)
- 0106 - Photometric Interpretation (2 = RGB)
- 0111 - Strip Offsets
- 0112 - Orientation (1 = 0 top, 0 left hand side)
- 0115 - Samples per pixel (1)
- 0116 - Rows per strip (200 = image height)
- 0117 - Strip Byte Counts (60000 = 100 x 200 x 3)
- 0118 - Minimum sample value (0,0,0)
- 0119 - Maximum sample value (255,255,255)
- 011c - Planar configuration (1 = single image plane)
- 0153 - Sample format

Além da representação em 24 bits (1 byte para cada cor RGB), outros formatos são possíveis tais como RGBA, onde o quarto byte representa a transparência. O RGBA é um formato é mais adequado, pois possibilita o acesso a cada pixel, em endereços múltiplos de 2ⁿ.

Descrição

Neste trabalho faremos algumas considerações para facilitar o processamento da imagem.

- a) a imagem não utilizará nenhum método de compressão;
- b) o formato será RGB com 24 bits, sendo 1 byte para cada cor;
- c) não iremos alterar nenhum valor no diretório da imagem, portanto ele poderá ser escrito no arquivo de saída sem qualquer tipo de processamento.

O trabalho consiste em implementar alguma transformação na imagem, utilizando a linguagem C e uma segunda versão tentando otimizar as possíveis partes com instruções SSE.

Como idéia de implementação, seguem 2 exemplos de possíveis implementações, disponíveis em:

<http://www.koders.com/cpp/fidD5EC07C4BC766073944513F8CC93F85A479F71C1.aspx>

- a) Adicionar um ruído randômico

```

void Noise(int width, int height, RGBA* pixels)
{
    for (int i = 0; i < width * height; i++) {
        int red = pixels[i].red + rand() % 33 - 16;
        int green = pixels[i].green + rand() % 33 - 16;
        int blue = pixels[i].blue + rand() % 33 - 16;
        // clamp to [0,255]
        red = (red < 0 ? 0 : (red > 255 ? 255 : red));
        green = (green < 0 ? 0 : (green > 255 ? 255 : green));
        blue = (blue < 0 ? 0 : (blue > 255 ? 255 : blue));
        pixels[i].red = red;
        pixels[i].green = green;
        pixels[i].blue = blue;
    }
}

```

b) Ajuste de brilho

```

void AdjustBrightness(int width, int height, RGBA* pixels, int dred, int
dgreen, int dblue)
{
    for (int dx = 0; dx < width; ++dx) {
        for (int dy = 0; dy < height; ++dy) {
            red = pixels[dy * width + dx].red +dred;
            green = pixels[dy * width + dx].green + dgreen;
            blue = pixels[dy * width + dx].blue +dblue;
            pixels[i].red = (red < 0 ? 0 : (red > 255 ? 255 :
red));
            pixels[i].green = (green < 0 ? 0 : (green > 255 ? 255 :
green));
            pixels[i].blue = (blue < 0 ? 0 : (blue > 255 ? 255 :
blue));
        }
    }
}

```

Após definir e realizar a implementação, efetue o teste e obtenha os valores de desempenho, com imagens de diferentes tamanhos tais como 32x32, 640x 480, 720x480, 1280x960, 1600x1200, 2560x1920.

Faça a análise dos resultados obtidos durante os testes. - Para medir o tempo de execução utilize a instrução RDTSC

```

unsigned long long rdtsc(){
    asm("rdtsc" ); // armazena o numero de ticks do relógio em EDX:EAX
    asm("leave" );
    asm("ret"); // retorno de função de dados de 64 bits, são retornados
em EDX:EAX
}

int main(){
    unsigned long long inicio;
    inicio = rdtsc();
    printf (" RDTSC test" );
    printf (" Fim - Inicio : %lld" , rdtsc()- inicio);
}

```

O que entregar?

- Artigo descrevendo a implementação e analisando os resultados obtidos. (Impresso), no formato da SBC (www.inf.unioeste.br/~marcio/psc/FormatoArtigos.zip)
- Códigos implementados (por email: msoyamada@gmail.com)
- Trabalho em dupla
- Data da Entrega: 29/07/2009
- As defesas serão marcadas posteriormente

Referências e Links úteis

- [1] Intel Instruction Set Reference (Volume 2A- Instruções de A-M). <<http://www.intel.com/design/pentium4/manuals/253666.htm> > Acessado em Out/2006.
- [2] Intel Instruction Set Reference (Volume 2B- Instruções de N-Z). <<http://www.intel.com/design/pentium4/manuals/253667.htm> > Acessado em Out/2006.
- [3] AMD – AMD64 Architecture Programmer’s Manual (Volume 4- 128-Bit MediaInstructions). <http://www.amd.com/usen/assets/content_type/DownloadableAssets/dwamd_26568.pdf > Acessado em Out/2006
- [4] Formato TIF. <<http://local.wasp.uwa.edu.au/~pbourke/dataformats/tiff/> > Acessado em Out/2006.
- [5] RPG Engine. <<http://www.koders.com/cpp/fidD5EC07C4BC766073944513F8CC93F85A479F71C1.aspx> >Acessado em Out/2006.

Dicas:

- Utilize o aplicativo GIMP para gerar o arquivo TIF
- A imagem pode ser representada por um vetor de bytes contíguo