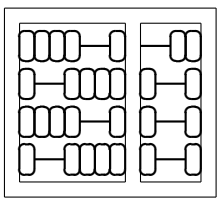

MC404 Organização de Computadores e Linguagem de Montagem

IC – UNICAMP

Primeiro Trabalho - 2S2008

1 Entrega

O trabalho deverá ser feito em grupo de até 2 alunos. O prazo de entrega é dia 13/10/2008. Tentaremos inicialmente coletar os trabalhos via sistema Teleduc.

2 Avaliação

A avaliação será baseada na corretude do programa, qualidade e documentação do código. O grupo também deverá entregar um relatório (máximo de duas páginas) descrevendo os passos que realizaram ao escrever o programa.

3 Descrição do Trabalho

Neste primeiro trabalho, você deve escrever um programa em linguagem de montagem que aceite como entrada um arquivo BMP preto/branco e gere um arquivo BMP de saída no qual as bordas dos objetos sólidos e pretos estejam pintadas de vermelho. Seu programa deve:

- *Aceitar como parâmetro de entrada um arquivo no formato BMP*
  Caso o arquivo não esteja nesse formato ou não existir, exiba uma mensagem informando ao usuário o ocorrido. Você deve limitar-se a arquivos BMP com 24 bits por pixel de cores e sem compressão (leia a seção 5 para informações sobre arquivos BMP). Você também pode assumir que o tamanho máximo dos arquivos de entrada sejam de 64Kb (um segmento no 80x86).

- *Gerar um arquivo BMP com as bordas de objetos pretos pintadas de vermelho*
  Para cada objeto contínuo preto na imagem de entrada, seu programa deve pintar a borda desse objeto com a cor vermelha. Para este trabalho, assuma que a borda de um pixel é constituída por todos os pixels adjacentes nas oitos direções (cima, baixo, direita, esquerda, superior direita, superior esquerda, inferior direita, inferior esquerda) que não fazem parte do objeto. A figura 1 mostra um exemplo de borda, com cada quadriculado representando um pixel da imagem. A imagem da esquerda é a original (de entrada) enquanto a da direita mostra como a imagem de saída deve ficar.

Para realizar este trabalho, você vai precisar de informações sobre argumentos de linha de comando (seção 4), funções do DOS para manipulação de arquivos (vistas no laboratório 5) e o formato BMP

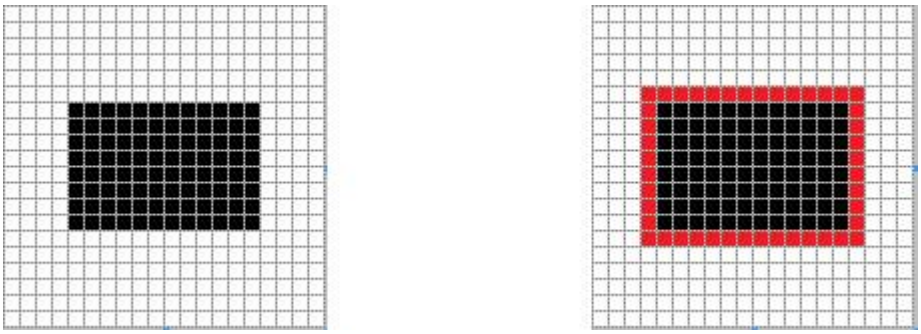

Figura 1: Exemplo de borda

(seção 5). Antes de começar a codificar, leia as próximas seções e tenha certeza que entendeu todos os conceitos que serão necessários neste trabalho. Como sugestão, faça programas testes para exercitar cada seção isoladamente.

Na página do projeto há figuras BMP feitas especialmente para essa atividade. Você deve usá-las para validar seu programa. As imagens estão com seus nomes no formato `imgXX.bmp`, onde `XX` é um índice. Teste primeiro as imagens de índices menores.

3.1 Extras

Para concorrer a pontos extras você pode implementar uma das tarefas adicionais abaixo. Note que você deve gerar um arquivo adicional com a figura resultante para cada tarefa extra.

- Cada borda deve ter uma cor diferente (0,5 ponto);
- A borda do objeto de maior área (maior número de pixels com a cor preta) deve ser pintada com uma cor especial (por exemplo, amarela) (1 ponto).

4 Argumentos de linha de comando no DOS

Antes de carregar um arquivo para execução, o DOS cria uma área chamada de PSP (*Program Segment Prefix*) com algumas informações a respeito do programa a ser executado. Para que um programa possa localizar seu PSP, o DOS carrega os registradores de segmento `DS` e `ES` com a localização do primeiro byte do PSP. A estrutura do PSP é apresentada a seguir:

Offset	Length	Description
0x00	2	INT 20h instruction.
0x02	2	Top of memory. Segment address of the top of the current program's memory allocation block.
0x04	1	Reserved.
0x05	1	INT 21h instruction.
0x06	2	Available memory--the number of bytes available in the segment.
0x08	2	Reserved.
0x0A	4	Terminate address (IP,CS). Address of the termination interrupt handler (INT 22h) inherited by the current program. DOS uses this value to restore the INT 22h vector when the program terminates.
0x0E	4	Ctrl-Break address (IP,CS). Address of the Ctrl-Break interrupt handler (INT 23h) inherited by the current program. DOS uses this value to restore the INT 23h vector when the program terminates.
0x12	4	Critical error address (IP,CS). Address of the Critical error interrupt handler (INT 24h) inherited by the current program. DOS uses this value to restore the INT 24h vector when the program terminates.
0x16	22	Reserved.
0x2C	2	Environment--the segment address of the DOS environment.
0x2E	46	Reserved.
0x5C	16	Unopened standard FCB1, as parsed from the first parameter in the command tail.
0x6C	20	Unopened standard FCB2, as parsed from the second parameter in the command tail.
0x80	1	Parameter length--the length, in bytes, of the command tail.
0x81	127	Command tail. Also used as default DTA for FCB functions.

As informações de interesse para o trabalho estão nos offsets `0x80` e `0x81`. No offset `0x80` está o tamanho do *command tail*, que é tudo o que aparece na linha de comando após o nome do comando. Os caracteres do *command tail* estão armazenados a partir do offset `0x81`, em ASCII. O último byte do *command tail* é sempre o valor `0x0D` (*carriage return*).

Como exemplo, suponha que temos a seguinte linha de comando:

```
C:\>projeto1 img01.bmp
```

No offset `0x80` do PSP vai estar armazenado o valor 10, e a partir do endereço `0x81` estarão os caracteres ASCII ‘ `img01.bmp`’ (note que o espaço também faz parte do *command tail*).

5 Formato BMP

O formato de arquivo BMP foi desenvolvido pela Microsoft como formato nativo para imagens no Windows 3.0. A partir de então o formato ficou amplamente conhecido (e difundido), sendo ainda usado nos sistemas operacionais atuais da Microsoft. Arquivos BMP são independentes de dispositivos de visualização (DIB - `Device Independent Bitmap`), ou seja, o formato especifica as cores dos pixels de forma independente do método usado pelo dispositivo para representá-las.

Um arquivo BMP é um arquivo binário, separado em 4 seções principais:

1. **File Header**
   Informações sobre o arquivo em geral: identificação do arquivo, tamanho, localização da imagem no arquivo.

2. **Image Header**
   Informações sobre a imagem: largura e altura da imagem, quantidade de bits usados por pixel, etc.

3. **Color Table**
   Esse campo pode não estar presente em alguns arquivos, dependendo do formato da imagem. Quando presente, este campo em geral contém os valores RGB para cada índice de uma paleta de cores. Nota: não usaremos este campo neste projeto.

4. **Pixel Data**
   A imagem propriamente dita. O formato dos dados varia dependendo do modo de cor da imagem, e se existe compressão ou não.

Os campos presentes em um arquivo BMP são apresentados a seguir.

Section	Offset	Size	Name	Contents	Meaning
	0x00	02	bfType	"BM"	Microsofts' BMP ID word
File	0x02	04	bfSize	varies	Size in bytes of the file
Header	0x06	04	bfRes	00,00	Reserved
	0x0A	04	bfOffBits	varies	Offset in file where image starts
	0x0E	04	biSize	40	Size of bitmap header
	0x12	04	biWidth	varies	Width in pixels
	0x16	04	biHeight	varies	Height in pixels
	0x1A	02	biPlanes	1	Number of image planes (only one)
Image	0x1C	02	biBitCount	varies	Bits per pixel (1,4,8, or 24)
Header	0x1E	04	biComp	varies	Compression type (0 = uncompressed)
	0x22	04	biSizeImg	varies	Size of compressed image (or zero)
	0x26	04	biHorRes	varies	Horizontal Res. in pixels/meter
	0x2A	04	biVerRes	varies	Vertical Res. in pixels/meter
	0x2E	04	biClrUsed	varies	Number of colors used
	0x32	04	biClrImp	varies	Number of 'important' colors
Color Map	0x36	varies	ColorMap	varies	Not present if biBitCount = 24
Pixel Data	Follows Color Map		(specified by bfOffBits)		contents varies

O campo *biBitCount* determina a resolução de cores do bitmap. Note que se esse campo por 24, a seção *Color Map* não estará presente no arquivo (esse é o caso das figuras que usaremos neste projeto).

O campo *biClrUsed* refere-se a quantidade de cores presentes, e não necessariamente é igual ao máximo que a resolução permite. Por exemplo, pode ser que a resolução de cores seja 256, mas somente 200 sejam usadas. **Se *biClrUsed* for zero, então o máximo de cores da resolução é usado.**

A formato da seção `Pixel Data` depende do modo de cor e se existe ou não compressão. Em um modo de 24 bits e sem compressão (que deve ser usado no trabalho), `Pixel Data` contém conjuntos de 3 bytes para cada pixel da imagem, com 1 byte para o azul, 1 byte para o verde e 1 para o vermelho. Cada byte representa a intensidade dessas cores, com 0 significando ausência e 0xFF a máxima intensidade. Assim, o pixel preto é representado por 3 bytes 0x00 e o branco por 3 bytes 0xFF.

O número de bytes que forma uma linha da imagem deve ser múltiplo de 4. Ou seja, se a imagem tem uma largura de 159 pixels, 1 byte (com o valor 0) deve ser adicionado a cada linha, de forma a completar 160 bytes por linha. Os primeiros bytes da imagem em `Pixel Data` referecem-se aos pixels da última linha da imagem, da esquerda para direita. Isso mesmo, a imagem é armazenada de ponta cabeça no formato BMP!