UNIVERSIDADE DE BRASÍLIA
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

116394 ORGANIZAÇÃO E ARQUITETURA DE COMPUTADORES

Trabalho I: Programação Assembler

OBJETIVO

Este trabalho objetiva a prática da programação em *assembler* do MIPS. O trabalho consiste em desenvolver um sistema de desenho de primitivas gráficas no ambiente MARS, com o auxílio da ferramenta de exibição de saída gráfica mapeada em memória.

DESCRIÇÃO

A. Display gráfico mapeado em memória

- O MARS oferece, dentre as suas ferramentas de apoio ao desenvolvimento de aplicações em *assembler* MIPS, uma janela gráfica baseada em *pixels*. Suas principais características são as seguintes:
  - Resolução configurável, *default* 512 x 256 *pixels*.
  - Cada *pixel* representado por uma palavra de 32 bits, no formato RGB, um *byte* para cada cor. *Red* = 0x00FF0000, *Green* = 0x0000FF00, *Blue* = 0x000000FF.
  - *Display* mapeado em memória. O ponto superior esquerdo da tela corresponde ao pixel com coordenadas (0, 0). A coordenada *x* cresce para a direita e a coordenada *y* cresce para baixo.
  - O endereço correspondente ao primeiro pixel é configurável. Opções: *global data, global pointer, static data, heap, memory map.*
  - O desenho de um pixel na tela é realizado pela escrita de uma palavra contendo a descrição de sua cor RGB na posição de memória correspondente.

B. Primitivas Gráficas a serem implementadas

- `void setColor(unsigned int cor)`: especifica a cor com que serão desenhadas as primitivas
- `void point(x, y)`: desenha um ponto na posição indicada
- `void line(x0, y0, x1, y1)`: desenha uma linha através do algoritmo de Bresenham utilizando a função ponto:

```
plotLine(x0,y0, x1,y1) {

  dx=x1-x0;
  dy=y1-y0;

  D = 2*dy - dx;
  plot(x0,y0);
  y=y0;

  for x from x0+1 to x1
    if (D > 0) {
      y = y+1;
      plot(x,y);
      D = D + (2*dy-2*dx);
    }
    else {
      plot(x,y);
      D = D + (2*dy);
    }
}
```

- `void rect(x, y, l, a)`: desenha um retângulo, sendo dados o ponto inicial e a largura e altura, utilizando a função linha.
- `void fillRect(x, y, l, a)`: desenha um retângulo preenchido com a cor corrente a partir da chamada da função linha.
- `void circle(x, y, r)`: desenha um círculo com centro *x, y* e raio *r*.

obs 1: círculo é opcional. É requisito para alcançar a nota 10,0 no trabalho. Sem a implementação de círculo, o trabalho vale até 9,0.

obs 2: as funções `rect()` e `fillRect()` devem chamar a função `line()`, que por sua vez deve chamar a função `point()`. Não é opcional, é requisito do trabalho.

C. Interface com o usuário

- Utilizando as chamadas do sistema, escrever uma mensagem com um cardápio de opções para o usuário. Ex:
    1. Seta cor
    2. Desenha ponto (x, y)
    3. Desenha linha (x0, y0, x1, y1)
    4. Desenha retângulo (x, y, l, a)
    5. Preenche retângulo (x, y, l, a)
    6. Desenha círculo (x, y, r)
- O usuário escolhe a opção através do número e depois entra com os parâmetros requeridos.
-

ENTREGA

Entregar no Moodle em um arquivo compactado:

- Relatório de implementação:
    - *cabeçalho*: com título do trabalho, nome e matrícula do aluno, identificação da turma
    - *objetivo*: sumarize os objetivos principais do trabalho
    - *cálculo do endereço do ponto na tela*: apresente a fórmula utilizada para estabelecer a correspondência entre um ponto na tela e seu respectivo endereço em memória
    - *configuração do MARS*: apresente a configuração de memória utilizada. Indique qual a área de memória necessária para armazenar o *bitmap* da tela.
    - *análise dos resultados*: avalie o número de instruções utilizadas em cada função e sua complexidade relativa. Comente eventuais restrições dos algoritmos.
    - *documentação do código*: indique quais as funções implementadas (todas), seus parâmetros e funcionamento.
- código assembler: arquivo asm

Prazo de entrega: 11/04/14