Universidade de Brasília
Departamento de Ciência da Computação

Disciplina: CIC 116394 – Organização e Arquitetura de Computadores – Turma A 2017/1
Prof. Marcus Vinicius Lamar
Equipes de até 5 pessoas.
Entrega do relatório (pdf) e fontes em um arquivo zip pelo Moodle até às 23h55 do dia 03/05/2017

Laboratório 1
- Assembly MIPS –

Objetivos:

- Familiarizar o aluno com o Simulador/Montador MARS;
- Desenvolver a capacidade de codificação de algoritmos em linguagem Assembly MIPS;
- Desenvolver a capacidade de análise de desempenho de algoritmos em Assembly;

(2.0) 1) Simulador/Montador MARS

Instale em sua máquina o simulador/montador MARS v.4.5 Custom 5 disponível no Moodle.

(0.0) 1.1) Dado o programa sort.s e o vetor: V[10]={5,8,3,4,7,6,8,0,1,9}, ordená-lo em ordem crescente e contar o número de instruções por tipo, por estatística e o número total exigido pelo algoritmo. Qual o tamanho em bytes do código executável?

(2.0) 1.2) Considere a execução deste algoritmo em um processador MIPS com frequência de *clock* de 50MHz que necessita 1 ciclo de *clock* para a execução de cada instrução (CPI=1). Para os vetores de entrada de n elementos já ordenados v_o[n]={1,2,3,4...n} e ordenados inversamente v_i[n]{n, n-1, n-2,...,2,1}, obtenha o número de instruções, calcule o tempo de execução para n={1,2,3,4,5,6,7,8,9,10,20,30,40,50,60,70,80,90,100} e plote esses dados em um mesmo gráfico n x t_{exec}. Comente os resultados obtidos.

(2.0) 2) Compilador GCC

Instale na sua máquina o *cross compiler* MIPS GCC disponível no Moodle.
Forma de utilização: **mips-sde-elf-gcc –S teste.c** #diretiva –S para gerar o arquivo Assembly teste.s
Inicialmente, teste com programas triviais em C para entender a convenção utilizada para a geração do código Assembly.

(0.5) 2.1) Dado o programa sortc.c, compile-o e comente o código em Assembly obtido indicando a função de cada uma das diretivas do montador usadas no código Assembly (.file .section .mdebug .previous .nan .gnu_attribute .globl .data .align .type .size .word .rdata .align .ascii .text .ent .frame .mask .fmask .set).

(0.5) 2.2) Indique as modificações necessárias no código Assembly gerado pelo gcc para poder ser executado corretamente no Mars.

(1.0) 2.3) Compile novamente o programa sortc.c e com a ajuda do Mars compare o número de instruções executadas e o tamanho em bytes dos códigos obtidos com os dados do item 1.1) para cada diretiva de otimização da compilação {-O0, -O1, -O2, -O3, -Os}.

(6.0) 3) Figuras Geométricas:

Hoje em dia existem diversas bibliotecas nas mais diversas linguagens de programação que permitem realizar, de maneira fácil, o desenho das principais figuras geométricas. Crie um conjunto de procedimentos (biblioteca) que permita desenhar as seguintes figuras dados seus parâmetros:
3.0) int ponto (x1,y1,cor) :: define a cor do pixel na posição (x1,y1)
3.1) int reta (x1,y1,x2,y2,cor) :: reta unindo os pontos (x1,y1) a (x2,y2) de cor 'cor'. Obs: conectando todos os pixels
3.2) int poligono (Num, x1,y1,x2,y2,x3,y3,…,cor) :: Polígono definido pelos 'Num' vértices (x1,y2),(x2,y2),...
3.3) int elipse(x1,y1,r1,x2,y2,r2,cor) :: Elipse com focos (x1,y1), (x2,y2) e raios r1 e r2
3.4) int preenchimento(x1,y1,cor1,cor2) :: preencher o menor polígono fechado de cor cor1 a partir do ponto (x1,y1) com a cor cor2. Opcional: int preenchimento2(x1,y1,cor) onde a cor do ponto (x1,y1) na tela indica a cor da superfície contínua deve ser preenchida.
Em todas as funções o retorno do valor 0 indica que o procedimento foi corretamente executado e o retorno de um valor diferente de zero indica que aconteceu algum problema na execução do procedimento.

Dado os procedimentos primitivos anteriores, escrever os procedimentos para:
3.5) int quadrado (x1,y1,lado, cor) :: Quadrado com centro em (x1,y1) e lado 'lado'
3.6) int circulo(x1,y1,raio, cor) :: círculo com centro em (x1,y1) e raio 'raio'
3.7) int trieq(x1,y1,lado,cor) :: triângulo equilátero centrado em (x1,y1) e lado 'lado'
3.8) criar procedimentos para desenhar as bandeiras do Japão, dos Estados Unidos, do Brasil e do Butão no centro da tela.

A tela gráfica do Mars, acessível pelo BitMap Display Tool, possui resolução 320x240 e 8 bits/pixels para a codificação das cores. O pixel na posição (x,y) pode ser plotado através do comando `sb $t0,0($t1)`:

$t1 = 0xFF000000+320*y+x é o endereço do pixel na memória de vídeo VGA
$t0 = 1 byte que define a cor no formato {BBGGGRRR}