Universidade de Brasília
Departamento de Ciência da Computação

Disciplina: CIC 116394 – Organização e Arquitetura de Computadores – Turma A 2016/2
Prof. Marcus Vinicius Lamar
Equipes de até 5 pessoas.
Entrega do relatório (pdf) e fontes em um arquivo zip pelo Moodle até as 23h55 do dia 03/10/2016

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 sort.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 para poder ser executado corretamente no Mars.

(1.0) 2.3) Compile novamente o programa sort.c e 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) Plotador de Gráficos no Mars

Uma das principais ferramentas para engenheiros e cientistas são os programas usados para desenhar gráficos de funções F(x).

(5.0) 3.1) Escreva um procedimento em assembly MIPS 'plot' que receba um intervalo real [LINF, LSUP] e desenhe na tela gráfica do Mars o gráfico da função 'funcao' nesse intervalo. O procedimento 'funcao' deve receber um valor real em ponto flutuante precisão simples IEEE 754 no registrador $f0 e devolver um valor em ponto flutuante precisão simples no registrador $f12. Obs.: Note que valores NaN e Inf podem ser retornados pela função.

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 no formato {BBGGGRRR}

(1.0) 3.2) Desenhe e capture a tela dos gráficos das seguintes funções:

a) F(x)=-x; x ∈[-1,1]
b) F(x)=x^2+1; x ∈ [-2,2]
c) F(x)=sqrt(x) x ∈ [-1,10]
d) F(x)=(x+1)^2*(x-1)*(x-2)/(x-1.5) x ∈ [-2,3]

e) F(x) = crie duas funções que plotadas juntas desenhem o símbolo do Batman (parte superior e inferior)