Universidade de Brasília
Departamento de Ciência da Computação

Disciplina: CIC 116394 – Organização e Arquitetura de Computadores – Turma A 2019/1
Prof. Marcus Vinicius Lamar
Equipes de até 5 pessoas (6 pessoas apenas com choro com lágrimas de sangue).
Entrega do relatório (pdf) e fontes em um único arquivo zip pelo Moodle até às 23h55 do dia 08/05/2019
Não serão aceitos relatórios depois do dia e horário definidos. ('PelamordeDeus' não deixe para enviar às 23h54, pois mesmo que o Moodle esteja com problemas ou fora do ar o relatório não poderá mais ser enviado. O melhor é ir enviando à medida que ele for sendo feito).

Laboratório 1
- Assembly RISC-V -

Objetivos:

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

(1.5) 1) Simulador/Montador Rars

Faça o download e deszipe o arquivo Lab1.zip disponível no Moodle. Serão criados 2 diretórios.

(0.0) 1.1) No diretório System_Rars, abra o `Rars12_Custom2` e carregue o programa `sort.s`. Dado 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 e o número total exigido pelo procedimento SORT. Qual o tamanho em bytes do código executável? E da memória de dados usada?

(1.0) 1.2) Considere a execução deste algoritmo em um processador RISC-V 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}, escreva as equações dos tempos de execução dado n, $t_o(n)$ e $t_i(n)$, e, para n={10,20,30,40,50,60,70,80,90,100}, plote (em escala!) as duas curvas em um mesmo gráfico n×t. Comente os resultados obtidos.

(0.0) 1.3) Sabendo que as chamadas do sistema padrão do Rars usam um console (parte do SO) para entrada e saída de dados, execute o programa testeECALLv13.s e analise como está sendo feita as chamadas Enviroment CALL (`ecall`) via macros. Note que essas chamadas usam as ferramentas KDMMIO e BITMAP DISPLAY sem precisar de console de comandos.

(0.5) 1.4) Faça um programa que desenhe a bandeira do seu time de futebol na tela.

(2.5) 2) Compilador cruzado GCC

Um compilador cruzado (*cross compiler*) compila um código fonte para uma arquitetura diferente daquela da máquina que está sendo utilizada. Você pode baixar gratuitamente os compiladores gcc para todas as arquiteturas (RISC-V, ARM, MIPS, x86, etc) e instalar na sua máquina, sendo que o código executável gerado apenas poderá ser executado em uma máquina que possuir o processador para qual foi compilado. No gcc, a diretiva de compilação `-S` faz com que o processo pare com a geração do arquivo em assembly e a diretiva `-march` permite definir a arquitetura a ser utilizada.

Ex.:
```
riscv64-unknown-elf-gcc -S -march=rv32imf  -mabi=ilp32f  # RV32IMF
arm-eabi-gcc -S -march=armv7                             # ARMv7
gcc -S -m32                                              # x86
```

Para fins didáticos, o site Compiler Explorer (https://cx.rv8.io/) disponibiliza estes compiladores C (com diretiva `-S`) *on-line* para as arquiteturas RISC-V, ARM e x-86 de 32 e 64 bits.

(0.0) 2.1) Teste a compilação para Assembly RISC-V com programas triviais em C disponíveis no diretório 'ArquivosC', para entender a convenção do uso dos registradores e memória utilizada pelo gcc para a geração do código Assembly, usando as diretivas de otimização `-O0` e `-O3`.

(0.5) 2.2) Dado o programa sortc.c, compile-o com a diretiva `-O0` e obtenha o arquivo sortc.s. Indique as modificações necessárias no código Assembly gerado para que possa ser executado corretamente no Rars.

(2.0) 2.3) Compile os programas sortc.c e sortc2.c e, com a ajuda do Rars, monte uma tabela comparativa com o número total de instruções executadas e o tamanho em bytes dos códigos em linguagem de máquina gerados para cada diretiva de otimização da compilação {-O0, -O1, -O2, -O3, -Os}. Compare ainda com os resultados obtidos no item 1.1) do programa sort.s que foi implementado diretamente em Assembly. Analise os resultados obtidos.

(6.0) 3) Problema do Entregador de Pizzas

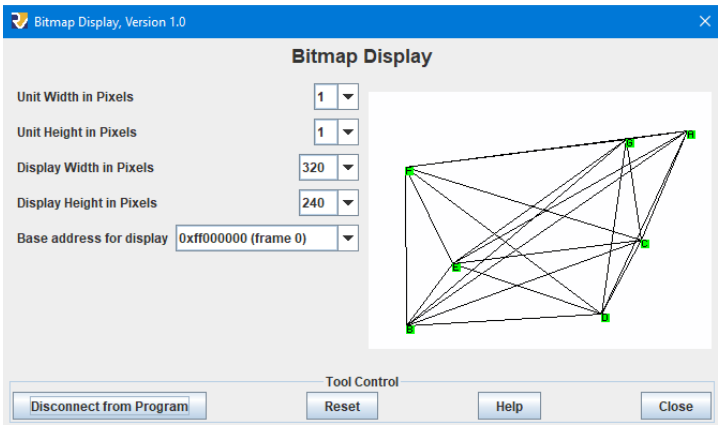
Dado o clássico problema computacional do ~~Caixeiro Viajante~~ Entregador de Pizzas: Um motoboy recebe a tarefa de entregar N pizzas a N clientes e retornar à loja, gastando o menor tempo possível (estimado aqui pela distância percorrida).

(0.5) 3.1) Crie um procedimento SORTEIO que dado um número inteiro N (`a0`) e um ponteiro C(`a1`) crie aleatoriamente (`ecall 41`) na memória de dados, um conjunto $C = \{c_1, c_2, \dots, c_N\}$ de N casas de clientes c_n em que cada casa é caracterizada pelas suas coordenadas (x, y) onde $x \in (0{,}310)$ *e* $y \in (0{,}230)$.
Exemplo de definição manual: `C: .word 10,10, 10,20, 5,30  # conjunto C de N=3 pares x,y`

```
.data
N: .word 6            # Número de Casas/Clientes
C: .space 160         # Espaço em bytes correspondente a 2 coordenadas x 20 casas (máx) x 4 bytes,

.text
       la t0,N
       lw a0,0(t0)
       la a1,C
       jal SORTEIO
```

(0.5) 3.2) Crie um procedimento DESENHA que, dado N(`a0`) e C(`a1`) desenhe na tela as posições das casas sorteadas usando a posição do pixel superior esquerdo de um quadrado verde de 8 pixels de lado contendo seu número como o caractere ASCII $(n + 64)$, conforme o exemplo abaixo.
Dica: Use `ecall 111`



(1.0) 3.3) Crie um procedimento ROTAS que, dado N(`a0`) e C(`a1`), desenhe as linhas que ligam uma a todas as outras casas e monte uma matriz $D=d_{ij}$ de dimensão NxN com suas distâncias Euclidianas (obs.: o motoboy anda onde quiser!)

$$d_{i,j} = \sqrt{\left(c_i(x) - c_j(x)\right)^2 + \left(c_i(y) - c_j(y)\right)^2}$$

```
.data
D: .space 1600        # máximo de 20x20 casas x 4 bytes (float precisão simples)
.text
la a2,D
```

(3.0) 3.4) Crie um procedimento ORDENA, que, dado N(`a0`), C(`a1`) e D(`a2`), ordene o conjunto C e a matriz D, e desenhe em vermelho a linha que inicia em uma casa c_1 (loja), une todas as casas, sem passar duas vezes em uma mesma casa, e volte à loja c_1, de modo a ter menor distância total f percorrida possível, isto é:

$$f = min\left\{\sum_{i=1}^{N-1} d_{i,i+1} + d_{N,1}\right\}$$

Filme as execuções para $N = \{2,3,4,5,6,7,8,9,10\}$.

(1.0) 3.5) Para $N = \{2,3,4,5,\dots,19,20\}$ faça um gráfico $t_{exec} \times n$ do tempo necessário para que um processador RISC-V Uniciclo de frequência de clock de 50MHz finalize o procedimento ORDENA para cada `n`. Quais as suas conclusões?

Para a apresentação da verificação dos laboratórios (e projeto) nesta disciplina, crie um canal para o seu grupo no YouTube e poste os vídeos dos testes (sempre com o nome 'UnB – OAC Turma A - 2019/1 – Grupo Y - Laboratório X - <palavras-chaves que identifiquem este vídeo em uma busca>'), coloque os links clicáveis no relatório.

Passos do vídeo:
i) Apresente o grupo e seus membros;
ii) Explique o projeto a ser realizado;
iii) Apresente os testes solicitados;
iv) Apresente suas conclusões.