Universidade de Brasília
Departamento de Ciência da Computação

Disciplina: CIC0099 – Organização e Arquitetura de Computadores – Unificado 2025/2
Equipes de até 5 pessoas.
Entrega do link do relatório (em .pdf) e fontes (.s) em um único arquivo zip pelo Moodle até às 23h55 do dia **19/10/2025**.
Não serão aceitos relatórios depois do dia e horário definidos. ('pelamordezeus' 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).

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;
- Familiarizar o aluno com a compilação C para Assembly RISC-V RV32IMF

(2.5) 1) Simulador/Montador Rars

Faça o download e deszipe o arquivo Lab1.zip disponível no Moodle.

(0.0) 1.1) No diretório `Arquivos`, abra o `Rars16_Custom1` e carregue o programa de ordenamento `sort.s`. Dado o vetor: V[30]={9,2,5,1,8,2,4,3,6,7,10,2,32,54,2,12,6,3,1,78,54,23,1,54,2,65,3,6,55,31},

a) 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?

b) Modifique o programa para ordenar o vetor em ordem decrescente e contar o número de instruções por tipo e o número total exigido pelo procedimento `sort`.

c) Usando os contadores de instruções e tempo do Banco de Registradores CSR (veja no final), meça novamente a quantidade de instruções executadas e o tempo de execução dos itens a) e b).

(2.5) 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}:

(1.5) a) Para o procedimento `sort`, escreva as equações dos tempos de execução, $t_o(n)$ e $t_i(n)$, em função de n.

(1.0) b) Para n={10,20,30,40,50,60,70,80,90,100}, plote (em escala!) as duas curvas, $t_o(n)$ e $t_i(n)$, em um mesmo gráfico n×t. `Comente os resultados obtidos.`

(2.5) 2) Compilador cruzado GCC

Um compilador cruzado (*cross compiler*) compila um código fonte para uma arquitetura diferente daquela da máquina em que está sendo utilizado. 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 disponibiliza estes (e vários outros) compiladores C (com diretiva –S) *on-line* para as arquiteturas RISC-V, ARM, x86 e x86-64 e outras.

(0.0) 2.1) Teste a compilação para Assembly RISC-V com programas triviais em C disponíveis no diretório 'Arquivos', 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`.

Dica: Use o compilador "`RISC-V (32-bits) gcc 15.2.0`"

(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.

Dica: Uso de Assembly em um programa em C. Use a função `show` definida no `sort.s` para não precisar implementar a função `printf`, conforme mostrado no `sortc_mod.c`

(1.0) 2.3) Compile o programa `sortc_mod.c` e, com a ajuda do Rars, monte uma tabela comparativa com o número total de instruções executadas pelo **programa todo**, e o tamanho em bytes dos códigos em linguagem de máquina gerados para cada diretiva de otimização da compilação {-O0, -O3, -Os}. Compare ainda com os resultados obtidos no item 1.1) com o programa `sort.s` que foi implementado diretamente em Assembly. Analise os resultados obtidos usando o mesmo vetor de entrada.

(1,0) 2.4) Pesquise na internet e explique as diferenças entre as otimizações -O0, -O1, -O2, -O3 e -Os.

(5.0) 3) Transformada Discreta de Fourier:

A Transformada Discreta de Fourier (DFT) converte os sinais amostrados no domínio do tempo (amostra) para o domínio frequência complexa (espectro) e é definida por

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{\frac{-2\pi.i.k.n}{N}}$$

onde, x[n] são as amostras do sinal x no domínio da amostra (tempo), X[k] são as amostras complexas do espectro no domínio frequência, N é o número de pontos de x[n] e X[k], $i = \sqrt{-1}$.

Dica: Fórmula de Euler $e^{i.\theta} = cos(\theta) + i.sin(\theta)$

3.1) (0,5) Escreva um procedimento que receba um ângulo em radianos θ (em fa0) e retorne $cos(\theta)$ (em fa0) e $sin(\theta)$ (em fa1).

```
{fa0,fa1} = sincos(float theta)
```

Dica: use aproximação por séries para o cálculo das funções trigonométricas

3.2) (1.0) Escreva um procedimento em Assembly RISC-V com a seguinte definição

```
void DFT(float *x, float *X_real, float *X_imag, int N)
```

que dado o endereço do vetor x[n] de floats (em a0) de tamanho N na memória, os endereços dos espaços reservados para o vetor complexo X[k] (parte real e parte imaginária) (em a1 e a2) e o número de pontos N (em a3), calcule a DFT de N pontos de x[n] e coloque o resultado no espaço alocado para X_real[k] e X_imag[k].

3.3) (0,5) Escreva um programa `main` que defina no .data o vetor x[n], o espaço para o vetor X[K], o valor de N, e chame o procedimento DFT.

```
.data
N: .word 8
x: .float 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0
X_real: .float 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
X_imag: .float 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
.text
…
jal DFT
...
```

A seguir, apresente no console a saída dos N pontos no formato:

```
x[n]  X[k]
1.0   8.0+0.0i
1.0   0.0+0.0i
1.0   0.0+0.0i
1.0   0.0+0.0i
1.0   0.0+0.0i
1.0   0.0+0.0i
1.0   0.0+0.0i
1.0   0.0+0.0i
```

3.4) (1.0) Calcule a DFT dos seguintes vetores x[n], com N=8

x1: .float 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0

x2: .float 1.0, 0.7071, 0.0, -0.7071, -1.0, -0.7071, 0.0, 0.7071

x3: .float 0.0, 0.7071. 1.0, 0.7071, 0.0, -0.7071, -1.0, -0.7071

x4: .float 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0

3.5) Para os sinais x[n] abaixo

a) N=8, x[n]={1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0 ,0.0}

b) N=12, x[n]={1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0 , ..., 0.0}

c) N=16, x[n]={1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0 , ..., 0.0}

d) N=20, x[n]={1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0 , ..., 0.0}

e) N=24, x[n]={1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0 , ..., 0.0}

f) N=28, x[n]={1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0 , ..., 0.0}

g) N=32, x[n]={1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0 , ..., 0.0}

h) N=36, x[n]={1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0 , ..., 0.0}

i) N=40, x[n]={1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0 , ..., 0.0}

j) N=44, x[n]={1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0 , ..., 0.0}

onde ... são todos valores 0.0

(1,0) 3.5.1) Para cada item: Meça o tempo de execução do procedimento DFT e calcule a frequência do processador RISC-V Uniciclo simulado pelo Rars.

(1,0) 3.5.2) Faça um gráfico em escala de $N \times t_{exec}$

Que conclusões podemos tirar desta análise?

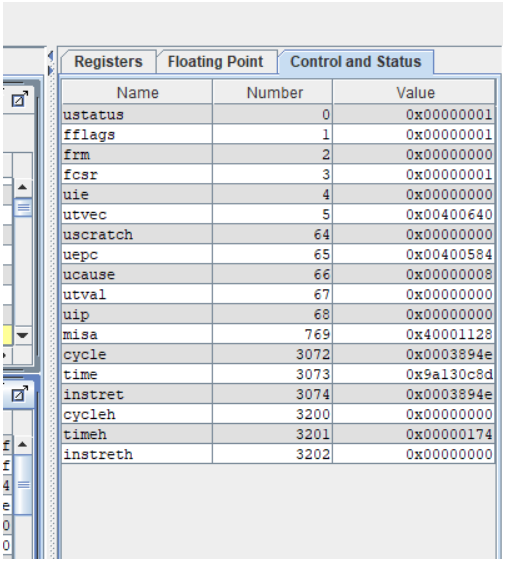
Dicas para medir o desempenho

O RISC-V possui o banco de registradores de Status e Controle (visto mais tarde em aula) no qual armazena continuamente diversas informações úteis, e que podem ser lidas pela instrução:

```
csrr t1, fcsr       # Control and Status Register Read
```

onde t1 é o registrador de destino da leitura e fcsr é um imediato de 12 bits correspondente ao registrador a ser lido.

Os registradores abaixo são registradores de 64 bits que contém as informações:



Name	Number	Value
ustatus	0	0x00000001
fflags	1	0x00000001
frm	2	0x00000000
fcsr	3	0x00000001
uie	4	0x00000000
utvec	5	0x00400640
uscratch	64	0x00000000
uepc	65	0x00400584
ucause	66	0x00000008
utval	67	0x00000000
uip	68	0x00000000
misa	769	0x40001128
cycle	3072	0x0003894e
time	3073	0x9a130c8d
instret	3074	0x0003894e
cycleh	3200	0x00000000
timeh	3201	0x00000174
instreth	3202	0x00000000

{`timeh`, `time`} = tempo do sistema em ms
{`instreth`, `instret`} = número de instruções executadas
{`cycleh`, `cycle`} = número de ciclos executados (se CPI=1 é igual ao instret)

Geralmente nossos programas não precisarão dessa precisão de 64 bits. Podemos usar então apenas os 32 bits menos significativos: `time`, `instret` e `cycle`.

Ex.: Para medir o tempo e o número de instruções do procedimento PROC para os registradores s0 e s1 respectivamente.

```
main: ...
      ...
      csrr s1,3074    # le o num instr atual
      csrr s0,3073    # le o time atual
      jal PROC
      csrr t0,3073    # le o time atual
      csrr t1,3074    # le o num instr atual
      sub s0,t0,s0    # calcula o tempo de execução texec em ms
      sub s1,t1,s1    # calcula o número de instruções I executadas
      ...
```

Note que terá um erro de 3 instruções na medida do número de instruções. Por quê?

Como o Rars simula um processador RISC-V Uniciclo (CPI=1), conhecendo o número de Instruções executadas (I), pode-se calcular a frequência do processador equivalente: $f = \frac{I}{T_{exec}}$

O relatório deve ser escrito na forma resposta ao item, contendo apenas os itens que valem ponto. No final deverá constar a URL clicável de um vídeo da apresentação gravada usando o PowerPoint onde o grupo apresenta os resultados obtidos dos itens pedidos. O grupo deve gravar a apresentação usando uma videoconferência no Teams (usem a equipe OAC-U), com a participação por câmera de TODOS os componentes. É importante que todos os participantes falem sobre os itens, quem não participar ativamente da apresentação receberá nota 0 no Laboratório.

Sugestão: crie um canal para o seu grupo no YouTube e poste os vídeos dos relatórios sempre com o nome 'UnB – OAC Unificado – 2025-2' – Grupo Y - Laboratório X - <palavras-chaves que identifiquem este vídeo em uma busca>'.

Passos do vídeo:

i) Apresente o grupo, a disciplina, a turma, o semestre e o Laboratório;
ii) Apresente cada item solicitado que vale ponto. Sugerimos que cada componente apresente um item;
iii) Apresente as conclusões o grupo.