Universidade Estadual de Maringá
Centro de Tecnologia
Departamento de Informática

Disciplina: Arquitetura e Organização de Computadores I

# Trabalho

**Objetivo:** O trabalho aqui proposto tem como objetivo a implementação de um simulador para uma arquitetura com pipeline simplificada, composta por um conjunto de instruções aritméticas, de desvio e de movimentação de dados entre registradores e memória.

**Instruções**

- O trabalho poderá ser feito em dupla.
- O trabalho poderá ser implementado em uma das seguintes linguagens:
  - C/C++;
  - Java;
  - Pascal;
  - Python;
  - Obs: caso a dupla deseje utilizar outra linguagem deverá comunicar o professor para verificar se a linguagem será aceita.
- O trabalho vale de 0,0 a 10,0 e corresponde a 3ª avaliação periódica.
- O trabalho deverá ser entregue via classroom na data combinada em sala.
- O nome do arquivo enviado deve seguir o padrão: NomeAluno1_RAXXXXXX_NomeAluno2_RAXXXXXX.zip;
  - O formato para submissão deve ser .zip, .rar, ou .tar;

**Descrição:** Deve-se implementar um simulador para uma arquitetura simples. As descrições para o hardware são as seguintes:

- **Processador:**
  - A arquitetura deverá ter um pipeline de 5 estágios, sendo eles:
    - Busca de instrução;
    - Decodificação de instrução;
    - Execução;
    - Acesso a memória;
    - Escrita do resultado nos registradores;
      Obs: Veja que nem todas instruções realizam ações em todos estágios do pipeline, porém para simplificar a implementação, faça com que todas as instruções passem por todos os estágios;
  - A forma de tratar *os hazards* de dados deverá ser escolhida pela dupla com uma das duas opções:
    - Inserção de *stalls;*
    - Encaminhamento *Forwarding*
  - Os *hazards* de controle deverão ser tratados prevendo que o desvio nunca será tomado.
  - Assuma que não existirão *hazards* estruturais;
  - A arquitetura deve ter 32 registradores de uso geral;
    - Os registradores devem ser nomeados de r0 até r31;
    - Inicialmente todos eles devem conter o valor 0;
  - É obrigatório o uso do registrador de estado *Program Counter* (PC).
- **Memória:**
  - A arquitetura deve ter memória de dados e instruções separadas;

- Assuma que o programa que está em execução pode ser armazenado inteiro na memória de instruções;
- Assuma que cada instrução ocupa uma posição na memória de instruções;
- O tamanho da memória de dados deve ser configurável (um arquivo de configuração ou um argumento via linha de comando)
  - Todos os dados da memória devem ser inicializados com o valor 0.
- Cada endereço de memória de dados deverá armazenar um valor inteiro;

- **Instruções:**
  - As instruções aritméticas e de desvio só podem ter seus operandos endereçados de duas maneiras:
    - Endereçamento direto por registrador;
    - Endereçamento por imediato;
  - Somente as instruções de load e store fazem acesso a memória;
  - O formato e significado das instruções que deverão ser implementadas são exibidos na Tabela 1;
  - Todas as instruções deverão operar somente sobre valores inteiros.
- **Entrada**
  - A entrada do programa deverá ser um arquivo de texto contendo um conjunto de instruções a serem executadas, onde cada instrução ocupa uma linha do arquivo (alguns exemplos estão disponíveis no classroom).
- **Saída:**
  - A cada ciclo deverá ser exibido:
    - Os valores armazenados em cada endereço da memória de dados;
      - Se a equipe preferir, valores zerados não precisam ser exibidos.
    - Os valores armazenados em cada um dos registradores de uso geral;
    - Os valores armazenados em cada um dos registradores de controle de estado;
    - Quais as instruções se encontram em cada estágio do pipeline;
  - Quando um estágio estiver sem instruções, a saída deve apresentar tal informação, preenchendo o campo com algo que indique tal estado, por exemplo o caractere "- ".

| Tipo de Instrução | Representação da instrução | Significado |
|---|---|---|
| Aritméticas | add rd, rs, rt | Atribui à rd a soma de rs e rt<br>rd←rs+rt |
| | addi rd, rs, imm | Atribui à rd a soma entre rs e um valor imediato<br>rd ← rs+imm |
| | sub rd, rs, rt | Atribui à rd a subtração de rs e rt<br>rd←rs-rt |
| | subi rd, rs, imm | Atribui à rd a subtração entre rs e um valor imediato<br>rd ←rs-imm |
| | mul rd, rs, rt | Atribui à rd o produto entre rs e rt<br>rd←rs*rt |
| | div rd, rs, rt | Atribui à rd o quociente da divisão de rs por rt |

| | | |
|---|---|---|
| | | rd←rs div rt |
| | mod rd, rs, rt | Atribui à rd o resto da divisão de rs por rt<br>rd←rs mod rt |
| Desvios | blt rs, rt, imm | Salta caso rs seja maior que rt<br>Se rs < rt então pc←imm |
| | bgt rs, rt, imm | Salta caso rs seja menor que rt<br>Se rs > rt então pc←imm |
| | beq rs, rt, imm | Salta caso rs e rt sejam iguais<br>Se rs = rt então pc←imm |
| | j imm | Salto incondicional<br>pc ←imm |
| Memória | lw rd, imm(rs) | Carrega da memória para o registrador rd<br>rd ←M[imm+rs] |
| | sw rs, imm(rt) | Armazena o valor de rs na memória<br>M[imm+rt] ←rs |
| Movimentação | mov rd, rs | Atribui o valor de rs à rd<br>rd ← rs |
| | movi rd, imm | Atribui o valor do imediato ao registrador rd<br>rd ←imm |

*Tabela 1: Conjunto de instruções da arquitetura a ser simulada.*

Obs: nas instruções imm é um valor constante inteiro.

# Problemas com Trabalhos COPIADOS:

Quem copiar terá o trabalho anulado (zerado), seja de outra dupla ou da internet.
Quem fornecer a cópia também terá o trabalho anulado (zerado).