Universidade Federal de Santa Maria
UFSM – CT – DELC
ELC1011 – Organização de Computadores
Prof. Giovani Baratto

Trabalho 1: Simulador de Instruções MIPS
Proposta Inicial do Trabalho

2025 – 1º semestre

Este primeiro trabalho prático da disciplina de ELC1011 – Organização de Computadores visa aprofundar a compreensão da arquitetura MIPS através da implementação de um simulador educacional para um subconjunto específico de suas instruções. O desenvolvimento deste simulador em linguagem ***assembly*** para MIPS proporcionará uma visão detalhada do ciclo de execução de instruções e da interação entre os componentes de um processador. O trabalho poderá ser realizado individualmente ou em equipes de até dois alunos. A colaboração é encorajada para a troca de conhecimento e o desenvolvimento de soluções mais robustas. A entrega do trabalho será efetuada via Moodle, na data a ser definida em comum acordo, entre alunos e professor. Para a avaliação, cada equipe (ou aluno individual) deverá submeter um arquivo compactado no formato ZIP contendo:

- Um relatório completo do trabalho em formato PDF, seguindo a estrutura padrão de trabalhos acadêmicos: introdução, objetivos (geral e específicos), revisão bibliográfica (conceitos relevantes para o simulador MIPS), metodologia (detalhes da implementação do simulador), experimento (exemplos de testes realizados com o simulador), resultados (saídas do simulador para os testes), discussão (análise dos resultados e desafios encontrados), conclusões e perspectivas (aprendizados e possíveis extensões futuras).
- Os arquivos-fonte dos programas desenvolvidos em linguagem *assembly* MIPS, devidamente organizados. O código-fonte deverá ser completo, funcional e apresentar uma organização clara, facilitando a análise e compreensão do seu funcionamento.

A Figura 1 apresenta uma ilustração conceitual do projeto.

Para a implementação do simulador, serão utilizadas as seguintes estruturas de dados para modelar a arquitetura MIPS:

1. `reg`: Um vetor de 32 elementos (indexados de 0 a 31), do tipo inteiro de 32 bits, representando o banco de registradores de propósito geral do MIPS. O registrador `$0` (índice 0) deve sempre retornar o valor zero.
2. `PC`: Uma variável inteira de 32 bits que armazena o endereço da próxima instrução a ser buscada na memória (Contador de programa[1]).
3. `IR`: Uma variável inteira de 32 bits que contém a instrução atualmente em processo de decodificação e execução (Registrador de instruçoes[2]).
4. `mem_text`: Um vetor de inteiros de 32 bits que simula o segmento de texto da memória, onde o código do programa é armazenado. O endereço base para este segmento será `0x0040 0000`.
5. `mem_data`: Um vetor de inteiros de 32 bits que simula o segmento de dados estáticos da memória. O endereço base para este segmento será `0x1001 0000`.

[1] Program Counter
[2] Instruction Register

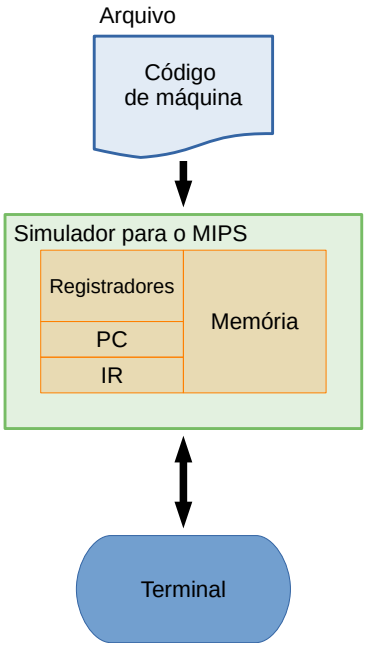

Figura 1: Ilustração do simulador do primeiro projeto da disciplina ELC1011.

6. `mem_stack`: Um vetor de inteiros de 32 bits que simula o segmento da pilha da memória. O endereço base para este segmento será `0x0x7FFF EFFC`, crescendo para endereços menores.
7. Variáveis para armazenar os campos da instrução corrente, como `opcode`, `rs`, `rt`, `rd`, `shamt`, `funct`, `immediate` e `address`, de acordo com os formatos de instrução MIPS (R-type, I-type, J-type).

O simulador deverá implementar o seguinte ciclo de execução de instruções:

1. **Inicialização:** Inicializar todos os registradores em `reg` com zero, exceto o ponteiro de pilha (geralmente `$sp`, ou `reg[29]`), que deve ser inicializado com o endereço base da pilha (`0x0x7FFF EFFC`). Inicializar `PC` com o endereço inicial do programa (`0x0040 0000`).
2. **Carregamento dos Arquivos:**
   - Abrir o arquivo binário `trabalho_01-2025_1.bin`, com as instruções em linguagem de máquina do programa que será simulado. Ler sequencialmente os bytes deste arquivo e armazená-los no vetor `mem_text`, a partir do endereço `0x0040 0000`. Assumir a arquitetura MIPS como **little-endian**.
   - Abrir o arquivo binário `trabalho_01-2025_1.dat`, com os dados estáticos do programa que será simulado. Ler sequencialmente os bytes deste arquivo e armazená-los no vetor `mem_data`, a partir do endereço `0x1000 0000`, também considerando a ordem **little-endian**.
3. **Loop de Execução:** Repetir os seguintes passos até que uma instrução de saída seja executada:
   (a) **Busca da Instrução**[3]**:** Ler a palavra de 32 bits da memória `mem_text` no endereço apontado por `PC` e armazená-la em `IR`.
   (b) **Decodificação da Instrução**[4]**:** Extrair os campos da instrução presente em `IR` (opcode, operandos, etc.). Incrementar o valor de `PC` em 4 para apontar para a próxima instrução. Identificar o tipo da instrução (R-type, I-type, J-type) utilizando os campos `opcode` e `funct`.
   (c) **Execução da Instrução**[5]**:** Desviamos ao procedimento associado à instrução que será simulada. No procedimento realizamos as operações aritméticas, lógicas, de acesso à memória, desvios, etc., necessárias para a simulação da instrução. Se a instrução for `syscall`, verificar

[3]Instruction Fetch
[4]Instruction Decode
[5]Execute

o código de serviço presente no registrador `$v0` (`reg[2]`). Implementar pelo menos os serviços para saída do programa (`exit` com código 1 e `exit2` com código 10). Outros serviços poderão ser definidos posteriormente.

O subconjunto inicial de instruções MIPS a ser implementado inclui (mas não se limita a):

- **Aritméticas:** `add`, `sub`, `addi`.
- **Lógicas:** `and`, `or`, `andi`, `ori`.
- **Transferência de Dados:** `lw`, `sw`.
- **Desvios:** `beq`, `bne`, `j`.
- **Outras:** `sll`, `srl`, `syscall`.

A lista completa e detalhada das instruções suportadas, incluindo seus formatos e funcionalidades, depende das instruções presentes no programa simulado no arquivo binário `trabalho_01-2025_1.bin`.

Os arquivos `trabalho_01-2025_1.bin`, `trabalho_01-2025_1.dat` e `trabalho_01-2025_1.asm` estarão disponíveis no Moodle até 11/04/2025, na pasta 'arquivos de entrada'. O arquivo `.asm` contém o código fonte do programa em *assembly* que foi utilizado para gerar os arquivos binários, servindo como referência para entender o programa a ser simulado.

O professor fornecerá suporte contínuo durante o desenvolvimento do trabalho, tanto em sala de aula quanto online. O documento de ajuda (**FAQ**) será uma fonte importante de informações e será atualizado regularmente com as dúvidas mais frequentes. Para questões não abordadas no FAQ, utilizem os canais de comunicação (Moodle ou fórum).

Como ponto de partida, elaborem as especificações detalhadas do projeto, o fluxograma do simulador em *assembly* MIPS, a metodologia de implementação, um cronograma de desenvolvimento e uma lista inicial com as dúvidas do trabalho.

No dia 11/04/2025, discutiremos o trabalho e definiremos os prazos finais para a entrega do simulador e do relatório.

Este trabalho é uma oportunidade valiosa para aplicar os conhecimentos teóricos da disciplina e desenvolver habilidades práticas em organização de computadores.