PROJETO MIPS: Sistema de Monitoramento de Enchentes

Disciplina: Arquitetura e Organização de Computadores – UNIFESP/ICT

Aline Nataly Lima de Moura - 164905

Guilherme Araújo Mendes de Souza - 156437

GitHub: https://github.com/alinelmoura/Projeto_MIPS-AOC

São José dos Campos - SP

2025

1. Resumo

Este projeto apresenta o desenvolvimento de um simulador de sistema embarcado para o monitoramento e prevenção de enchentes, implementado em linguagem Assembly MIPS. O sistema simula uma estação remota capaz de coletar dados de sensores de nível da água e umidade do solo, processando essas informações em baixo nível. Uma máquina de estados classifica o risco de inundação em diferentes níveis. Além disso, a análise dos dados é realizada por meio do cálculo da média móvel das últimas cinco leituras, utilizando um vetor circular. O sistema aciona alertas visuais e sonoros conforme a gravidade da situação. A aplicação foi desenvolvida e testada no simulador MARS, evidenciando o uso de registradores, controle de fluxo e manipulação de memória.

2. Introdução

As enchentes urbanas representam um dos fenômenos naturais que mais afetam a segurança, a infraestrutura e a qualidade de vida nas cidades, sendo associadas ao aumento de eventos extremos de precipitação e à insuficiência dos sistemas de drenagem urbana, com impactos socioeconômicos significativos e crescente frequência devido às mudanças climáticas. Segundo dados do Instituto de Pesquisa Econômica Aplicada (IPEA), eventos de cheia foram responsáveis por cerca de 43% dos desastres naturais entre 1995 e 2015, afetando bilhões de pessoas em todo o mundo.

A Internet das Coisas (IoT) tem demonstrado grande potencial para coletar dados ambientais em tempo real e apoiar a gestão de riscos urbanos, fornecendo informações contínuas de parâmetros como nível da água e condições meteorológicas que podem anteceder inundações.

Além disso, projetos baseados em sensores e microcontroladores para monitoramento ambiental urbano contribuem diretamente para os Objetivos de Desenvolvimento Sustentável - ODS 11 (Cidades e Comunidades Sustentáveis) e ODS 13 (Ação contra a Mudança Global do Clima), pois ampliam a capacidade de detecção precoce de riscos.

Este projeto aprofunda a compreensão de como sistemas com recursos limitados (como microcontroladores e algoritmos de máquina de estados) podem ser utilizados para tomada de decisões críticas a partir de dados de sensores, integrando conceitos de sistemas embarcados e monitoramento remoto no contexto da disciplina de Arquitetura e Organização de Computadores (AOC).

3. Descrição Geral do Sistema

3.1. Objetivos do simulador

O propósito do simulador é receber leituras manuais que representam dados de sensores de umidade do solo e nível da água, processá-las em baixo nível e, a partir dessas informações, determinar o estado de alerta da área monitorada, exibindo as ações necessárias, como o acionamento de alertas sonoros.

3.2. Funcionalidades implementadas

- **Interface (ASCII):** Exibição de título, menu principal, mensagens de entrada e painel de status por meio de chamadas de sistema no console do simulador MARS.
- **Leitura manual dos sensores:** Entrada de valores via teclado simulando sensores de umidade do solo (0 = seco, 1 = úmido) e nível da água (valores inteiros de 0 a 10).
- **Cálculo da média móvel:** Armazenamento das últimas cinco leituras do nível da água em um vetor circular, com controle de índice e tamanho atual, permitindo o cálculo da média móvel das leituras armazenadas.
- **Classificação do estado de risco:** Determinação do estado do sistema (livre de risco, alerta ou risco elevado) a partir da comparação entre o nível da água e o estado de umidade do solo.
- **Sistema de alarme:** Ativação ou desativação do alerta sonoro com base no estado de risco calculado.

3.3. Arquitetura geral do programa

O programa é organizado em um loop principal responsável por controlar a execução do sistema. Dentro desse loop, são chamadas funções específicas para realizar tarefas como leitura dos dados, cálculo da média, verificação do nível de risco e exibição das informações.

Fluxo Principal:

1. **Menu:** Exibe opções e lê a escolha do usuário (Inserir uma nova leitura dos sensores ou Sair do Programa).

2. **Entrada:** Lê Umidade do Solo e Nível da água.
3. **Processamento (calcular_media):** Atualiza histórico e retorna a média.
4. **Lógica (verificar_status):** Define o estado de alerta.
5. **Saída (exibir_painel):** Mostra resultados e atua sobre o alarme.
6. **Repetição:** Retorna ao início.

4. Máquina de Estados

4.1. Estados do sistema e transições

O sistema opera com base em três estados principais, determinados pela função verificar_status:

1. **Estado VERDE (Livre de Risco):**
   - *Condição:* Nível da água.
   - *Ação:* Alarme sonoro desligado.
2. **Estado AMARELO (Alerta):**
   - *Condição:* Nível da água entre 4 e 6 (Quando a umidade do solo = 0).
   - *Ação:* Alarme sonoro desligado (apenas aviso visual).
3. **Estado VERMELHO (Risco/Crítico):**
   - *Condição:* Nível da água maior que 6 (Quando umidade do solo = 0) ou Nível da água maior que 3 (Quando umidade do solo = 1) .
   - *Ação:* Alarme sonoro Ligado.

4.2. Diagrama de estados

- Quando umidade do solo = 0:

Sensor de umidade	Nível da Água	LED Verde	LED Amarelo	LED Vermelho	Alarme
0	0	1	0	0	0
0	1	1	0	0	0
0	2	1	0	0	0
0	3	1	0	0	0
0	4	0	1	0	0
0	5	0	1	0	0
0	6	0	1	0	0
0	7	0	0	1	1
0	8	0	0	1	1
0	9	0	0	1	1
0	10	0	0	1	1

- Quando umidade do solo = 1:

Sensor de umidade	Nível da Água	LED Verde	LED Amarelo	LED Vermelho	Alarme
1	0	1	0	0	0
1	1	1	0	0	0
1	2	1	0	0	0
1	3	1	0	0	0
1	4	0	0	1	1
1	5	0	0	1	1
1	6	0	0	1	1
1	7	0	0	1	1
1	8	0	0	1	1
1	9	0	0	1	1
1	10	0	0	1	1

5. Estruturas de Dados

O sistema de prevenção de enchentes foi desenvolvido utilizando a arquitetura MIPS, fazendo uso eficiente do segmento de dados (.data) para armazenamento estático e registradores para manipulação dinâmica. Abaixo são detalhadas as estruturas utilizadas.

5.1. Vetores e Armazenamento de Histórico

A principal estrutura de dados composta utilizada no projeto é um **vetor** destinado a armazenar o histórico das leituras do sensor de nível de água.

- **Definição:** O vetor foi declarado na memória com o rótulo historico.
- **Capacidade:** Ele possui capacidade para armazenar 5 números inteiros de 32 bits.

A lógica de inserção neste vetor pode ser descrita como um tipo de “Fila circular”.Diferente de uma fila tradicional, quando o vetor atinge 5 leituras (capacidade máxima), o índice retorna à posição inicial, sobrescrevendo o dado mais antigo. Para controlar essa lógica, usamos duas variáveis auxiliares:

1. **indice_vetor:** Aponta para a posição atual onde o próximo dado será gravado.
2. **tamanho_atual:** Conta quantos elementos válidos existem no vetor, travando no valor 5 assim que o vetor é preenchido pela primeira vez.

5.2. Representação de Sensores e Atuadores

Como o simulador MARS não permite conectar sensores de verdade, a leitura dos dados e o acionamento dos alarmes foram simulados através do teclado e da tela do computador.

- **Sensores/Entradas:**
  - **Umidade do solo:** Representado por uma variável inteira booleana, onde o valor 0 indica solo seco e 1 indica solo úmido.

    - **Nível da água:** Representado por uma variável inteira com escala de 0 a 10.
- **Atuadores/Saída:**
    - **Painel de monitoramento:** Mapeamento de alerta do sistema (verde, amarelo, vermelho).
    - **Alarme Sonoro:** Ativado quando o retorno da verificar_status é vermelho.

5.3. Organização de Dados na Memória

A memória do programa foi organizada no segmento .data.[

1. **Strings (ASCIIZ):** As mensagens de interface (msg_menu, str_verde, etc.) são armazenadas sequencialmente como cadeias de caracteres.
2. **Variáveis Globais (.word):** As variáveis de controle e o vetor são armazenados como inteiros de 4 bytes.

5.4. Convenção de Endereço Base + Deslocamento

Para acessar os elementos do vetor historico, o algoritmo utiliza o modo de endereçamento base mais deslocamento.

No procedimento calcular_media, essa convenção é explicitada através das seguintes instruções:

1. Carrega-se o endereço base do vetor (historico) em um registrador temporário.
2. O índice lógico (0 a 4) é multiplicado por 4 (instrução mul) para obter o offset em bytes.
3. Soma-se o offset ao endereço base (instrução add).
4. A instrução de acesso à memória (lw ou sw) utiliza esse endereço calculado para ler ou gravar a leitura do sensor.

Essa abordagem permite acesso aleatório (O(1)) a qualquer posição do histórico, essencial para a eficiência do cálculo da média e atualização cíclica dos dados.

6. Funções Principais

6.1. Função: loop_principal

Esta função controla o funcionamento geral do sistema. Ela funciona dentro de um laço de repetição infinito que exibe o menu, lê os dados digitados pelo usuário (umidade e nível da água) e chama as outras funções (calcular_media e verificar_status) para processar esses dados. O laço só é interrompido quando o usuário escolhe a opção de sair (0), encerrando o programa.

- **Parâmetros de entrada:** Recebe dados externos digitados pelo usuário.
- **Registradores usados:**
    1. $v0, $a0, $a1: Utilizados intensivamente para comunicação com o sistema.
    2. $t0: Usado na opção do menu, dado que não precisa ser memorizado pelo sistema.
    3. $s0, $s1, $s2, $s3: Foram escolhidos registradores do tipo saved para armazenar a Umidade, Nível de água e os resultados. Isso é necessário porque o loop faz chamadas a outras funções, logo garantimos a integridade dos dados durante todo o processamento.

6.2. Função: calcular_media

Esta função realiza o processamento matemático do sistema. Ela não apenas calcula uma média aritmética, mas gerencia a estrutura de dados na memória, garantindo que apenas as leituras válidas sejam consideradas e que o vetor sobrescreva os dados antigos automaticamente.

- **Parâmetros de entrada:** Recebe o valor inteiro da nova leitura do Nível da água.
- **Registradores usados:**
    1. $t0, $t1, $t2, $t3: Usados para manipulação de endereços de memória e controle de índices.
    2. $t5, $t6, $t7: Atuam como acumuladores temporários para realizar o loop de soma dos valores.
    3. $t8: Armazena o tamanho_atual, variável crucial para que a média seja calculada corretamente mesmo quando o vetor ainda não está cheio.
    4. $t9: Armazena a constante 5 (limite do vetor).
- **Uso da pilha:**Alocamos 16 bytes para salvar o endereço de retorno ($ra) e registradores de controle.
- **Lógica geral:**
    1. **Controle de Tamanho:** Verifica quantos itens existem no vetor. Se for menor que 5, incrementa o contador.
    2. **Inserção Circular:** Usa o resto da divisão (div/mfhi) para garantir que o índice do vetor volte a 0 após chegar a 4.
    3. **Loop de Soma:** Percorre o vetor somando os valores de 0 até o tamanho_atual.
    4. **Cálculo Final:** Divide a soma pelo tamanho atual e retorna o resultado.

6.3. Função: verificar_status

Esta função atua como a unidade lógica de decisão. Ela aplica as regras de segurança estabelecidas para determinar se a situação atual representa um risco de enchente, como estabelecidas anteriormente.

- **Parâmetros de entrada:**
    1. $a0: Nível da Água.
    2. $a1: Umidade do Solo.
- **Registradores usados:**
    1. $v0: Usado para retornar o código de status referente a cada resultado possível.

6.4. Função: exibir_painel

Esta função gerencia a interface de saída do sistema. Ela é responsável por traduzir os códigos numéricos calculados anteriormente em informações visuais para o usuário.

- **Parâmetros de entrada:**
    1. $a0: Valor da Média calculada (inteiro).
    2. $a1: Código do Status (0, 1 ou 2).
- **Registradores usados:**
    1. $v0, $a0: Utilizados intensivamente para realizar as chamadas de sistema.
    2. $t0: Atua como um backup temporário.
- **Lógica geral:** Imprime a média atual do nível da água, o status do sistema, e se foi necessário acionar o alarme ou não

7. Uso de IA Generativa

7.1 Ferramentas de IA utilizadas

IA utilizada: Gemini.

7.2 Como a IA foi utilizada no projeto

A ferramenta foi empregada como um assistente de codificação e documentação, atuando principalmente em:

- Implementação de Algoritmos: Geração do esqueleto inicial para a lógica da Fila Circular e Média móvel na função calcular_media, traduzindo a lógica matemática para instruções MIPS.
- Refatoração de Lógica: Otimização da estrutura de decisão a função verificar_status para corrigir falhas de prioridade entre os estados de alerta.
- Documentação Técnica: Auxílio na descrição textual das estruturas de dados e funcionamento interno dos registradores para o relatório.

7.3 Tabela de trechos gerados pela IA

Prompt	**Função Associada**	**O que foi gerado?**
"Gere um código MIPS que insere um valor em um vetor 5 posições e calcula a média desses valores"	calcular_media, loop_soma, fim_soma	A IA forneceu três blocos de código: um para inserção no vetor usando lógica circular, um loop para realizar a soma dos elementos e a divisão final para obter a média.
"Crie uma lógica para verificar status baseada em nível e umidade"	verificar_status	Um esboço de estrutura condicional (if/else) comparando os registradores de entrada e retornando valores 0, 1 ou 2.

7.4 O que tivemos que corrigir no código gerado pela IA

Embora a IA tenha fornecido a sintaxe correta e a estrutura base, o código gerado precisou de mudanças manuais olhando para as especificações do projeto:

1. **Correção da Média (uso da váriavel tamanho_atual):**
    - **Problema:** O código original gerado pela IA assumia que o vetor sempre estava cheio, dividindo a soma por 5 desde a primeira leitura. Isso gerava médias incorretas no início da execução (ex: lia 10, dividia por 5 e dava 2, em vez de 10).
    - **Ajuste Manual:** Foi implementada manualmente a lógica da variável tamanho_atual, criando um contador que incrementa até 5 e trava. O código de divisão foi alterado para usar esse registrador dinâmico na hora da divisão.

8. Testes e Resultados

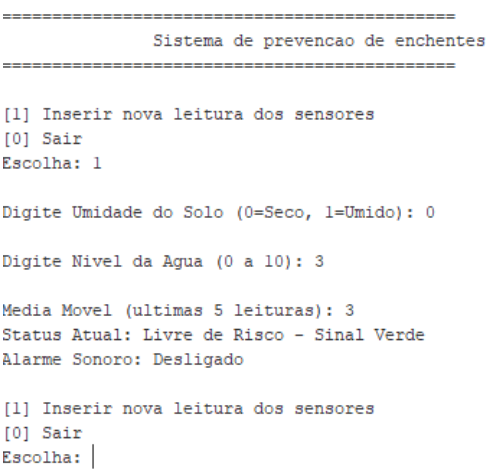
8.1 Casos de teste utilizados

1. **Caso Típico (Normal):** Umidade 0, Nível 2.
    - *Esperado:* Média atualizada, Status Verde, Alarme Off.

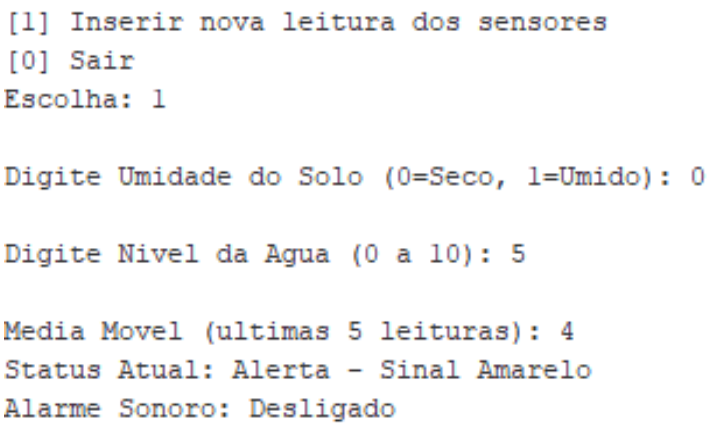
2. **Caso de Transição (Amarelo):** Umidade 0, Nível 5.
    - *Esperado:* Status Amarelo.
3. **Caso Extremo (Crítico com Solo Úmido):** Umidade 1, Nível 4.
    - *Esperado:* Status Vermelho (devido ao solo úmido agravar o risco), Alarme On.
4. **Teste de Média Móvel:** Inserir sequência 10, 10, 10, 10, 10.
    - *Esperado:* Média convergir para 10 após 5 iterações.

8.2 Screenshots ou trechos de saída

Status: Livre de Risco



Status: Alerta



Status: Risco (Com umidade do solo = 0)

```
[1] Inserir nova leitura dos sensores
[0] Sair
Escolha: 1

Digite Umidade do Solo (0=Seco, 1=Umido): 0

Digite Nivel da Agua (0 a 10): 9

Media Movel (ultimas 5 leituras): 5
Status Atual: Risco - Sinal Vermelho
Alarme Sonoro: Ligado
```

Status: Risco (Com umidade do solo = 1)

```
[1] Inserir nova leitura dos sensores
[0] Sair
Escolha: 1

Digite Umidade do Solo (0=Seco, 1=Umido): 1

Digite Nivel da Agua (0 a 10): 7

Media Movel (ultimas 5 leituras): 6
Status Atual: Risco - Sinal Vermelho
Alarme Sonoro: Ligado
```

Encerramento do programa

```
[1] Inserir nova leitura dos sensores
[0] Sair
Escolha: 0

-- program is finished running --
```

8.3 Discussão dos resultados

Os resultados obtidos com a execução do simulador ocorreram conforme o esperado. O sistema foi capaz de receber corretamente as leituras dos sensores, calcular a média móvel do nível da água e classificar o estado de risco de acordo com as regras definidas. O acionamento do alarme sonoro também funcionou conforme o esperado, sendo ativado apenas nas situações de risco elevado. Dessa forma, o comportamento observado confirma que a lógica implementada atende aos objetivos propostos.

9. Conclusão

Este projeto permitiu aplicar, de forma prática, os conceitos fundamentais de Arquitetura e Organização de Computadores por meio do desenvolvimento de um simulador de monitoramento de enchentes em Assembly MIPS. A implementação do sistema demonstrou como dados de sensores podem ser processados em baixo nível, utilizando estruturas como vetores, registradores, pilha e sub-rotinas. O programa apresentou funcionamento correto na leitura dos dados, no cálculo da média móvel e na classificação do nível de risco, além do acionamento adequado do alarme sonoro. Dessa forma, o trabalho atingiu seus objetivos, evidenciando a importância dos sistemas embarcados no monitoramento ambiental e na prevenção de desastres naturais.

10. Referências

Guia Rápido MIPS Tipos de Dados e Formatações. Disponível em: <https://www.inf.ufpr.br/wagner/ci243/GuiaMIPS.pdf>. Acesso em: 07 dez. 2025.

PATTERSON, David A; HENNESSY, John L. Organização e projeto de computadores: a interface hardware/ software. 3.ed. Rio
de Janeiro: Campus, 2005. 484 p. ISBN 9788535215212. Acesso em: 07 dez. 2025.

HERNANDEZ, Luis Carlos; SZIGETHY, Leonardo. Controle de enchentes: exemplos do uso da tecnologia e inovação para o controle de enchentes. *Centro de Pesquisa em Ciência, Tecnologia e Sociedade*, Instituto de Pesquisa Econômica Aplicada (IPEA), 03 dez. 2020. Disponível em:
https://www.ipea.gov.br/cts/pt/central-de-conteudo/artigos/artigos/231-controle-de-enchentes**.** Acesso em: 06 dez. 2025.

UNIVERSIDADE FEDERAL DE SÃO PAULO (UNIFESP). Slides da disciplina de Arquitetura e Organização de Computadores (AOC). São Paulo: UNIFESP, [s.d.].

GOOGLE. Gemini. [s.d.]. Disponível em: https://gemini.google.com/