RELATÓRIO FINAL – PROJETO MIPS (Simulador IoT/Embarcado)

Disciplina: Arquitetura e Organização de Computadores – UNIFESP/ICT

Painel de Estação Ferroviária
Arquitetura e Propriedade
Antonio Carlos da Fonseca Dias Pereira 176.472
Leonardo Arantes Lopes Macedo 176.558
Luan Groppo Viana 177.103
Thiago Corso Capuano 163.996

https://github.com/capuano1/Faculdade/tree/main/Arquitetura%20e%20Organização%20de%20Computadores/Trabalho%20Final

---

1. Resumo

O objetivo deste trabalho foi desenvolver um simulador, em linguagem Assembly MIPS, de um sistema de painel de estação ferroviária, com as possibilidades básicas de adicionar, remover e listar trens, mas também com funcionalidades de buscar trens de acordo com seu destino ou código único do trem, ordenar a lista de trens em ordem crescente de horário de chegada e conseguir registrar a chegada de um trem. O sistema desenvolvido foi capaz de satisfazer todos os requisitos mínimos estabelecidos, além de alguns dos requisitos adicionais supracitados.

---

2. Introdução

Este projeto consiste no desenvolvimento de um simulador de painel de estação ferroviária em linguagem assembly MIPS, utilizando o simulador MARS. O tema escolhido, inspirado em sistemas reais de transporte, foca na gestão e exibição de informações de trens em uma estação, crucial para organização das linhas ferroviárias e para organização por parte dos clientes e/ou passageiros de cada um dos trens, seja de carga ou de transporte de pessoas.

O simulador desenvolvido permite ao usuário interagir com um painel ASCII através do console, inserindo dados como o horário atual, cadastrando novos trens, especificando seus destinos e horários de chegada. O sistema realiza cálculos para determinar o status de cada trem (A CAMINHO, ATRASADO), apresenta os dados em um formato de tabela, a qual

ordena os itens pelo horário de chegada para maior legibilidade, além de oferecer funcionalidades de busca e ordenação. Fora isso, o operador pode, manualmente, registrar a chegada de um trem (para que seu estado mude de A CAMINHO ou ATRASADO para CHEGOU).

O projeto possui relação direta com a disciplina através da programação de baixo nível utilizando a linguagem Assembly MIPS (programação de baixo nível) para desenvolvimento do sistema em questão.

O uso de inteligência artificial generativa foi controlado e limitado, conforme as regras estabelecidas, com uma seção própria dedicada à descrição do seu uso, correções e aprendizados.

---

3. Descrição Geral do Sistema

3.1 Objetivos do simulador

O simulador deve funcionar como um painel de gerenciamento de uma estação ferroviária, capaz de adicionar e remover trens da lista (até 10 trens podem estar na lista ao mesmo tempo), listar todos os trens no sistema, buscar trens de acordo com seu código de trem (único) ou de acordo com a sigla da cidade de destino e registrar a chegada de um trem.

3.2 Funcionalidades implementadas

- Interface ASCII
- Verificação de entrada (verificar se as entradas estão em formato válido)
- Possibilidade de cadastrar múltiplos trens (até 10)
- Cálculo simples de atraso (verificar se o trem está atrasado)
- Algoritmo de ordenação (Bubble Sort) implementado para manter a lista de trens em ordem crescente de horário de chegada
- Registro de chegada de trem
- Possibilidade de buscar trens de acordo com seu código (único) ou sigla da cidade de destino

3.3 Arquitetura geral do programa

- 4 vetores para armazenamento dos dados do trem:
    - Vetor para código único do trem
    - Vetor para sigla da cidade de destino do trem
    - Vetor para estado do trem (A CAMINHO, ATRASADO, CHEGOU)
    - Vetor para horário previsto de chegada do trem
- Diversas strings para navegar pelo sistema
- Verificação das strings inseridas, para garantir que sejam válidas dentro das especificações

- Módulo Bubble Sort, composto por 4 funções, utilizado pelas funções de adicionar e remover trem
- Função para listar trem
- Função para a busca dos trens, sendo que é possível optar por uma busca por cidade de destino ou por código de trem, com cada um possuindo sua própria função
- O fluxo esperado é: inserir data atual, adicionar trens, listar ou buscar trens, registrar chegadas de trens e, posteriormente, remover trens que já foram processados.

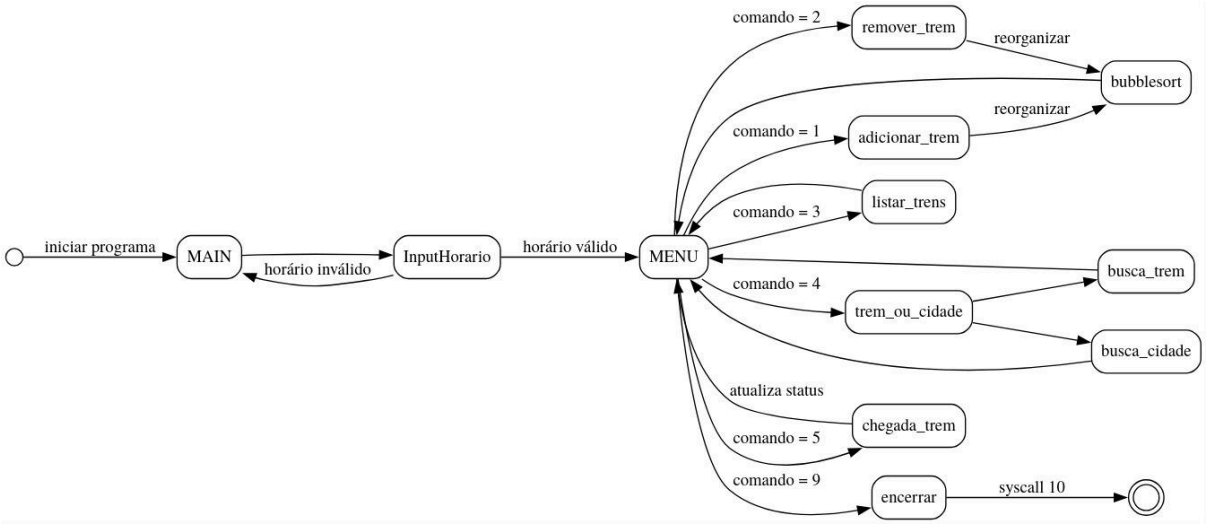
---

4. Máquina de Estados

4.1 Estados do sistema e transições

Os estados deste sistema e, por sua vez, suas transições, representam exatamente a navegação do sistema em si, com estados para o menu principal do sistema, para cada uma das funções apresentadas no menu e também para algumas funções extras, como o bubble sort, estado que é atingido por mais de um caminho (isto é, por mais de uma função, no caso, tanto pela função de adicionar trem quanto pela função de remover trem).

4.2 Diagrama de estados



---

5. Estruturas de Dados

As seguintes estruturas de dados foram utilizadas no desenvolvimento do sistema:

- Strings para armazenar as mensagens mostradas durante navegação do sistema
- Vetor: Foi usado um vetor de inteiros para armazenar os horários previstos de chegada de cada um dos trens
- Matrizes: Foram usadas matrizes (vetores de strings, sendo que strings são vetores de caracteres) para armazenar o código único do trem, sigla da cidade de destino e seu estado atual.
- Registradores globais:
    - $t8 foi usado como um registrador para armazenar o horário atual, informação repetidamente usada durante uso do sistema
    - $t9 foi usado como um contador do número de trens cadastrados no sistema, informação que também é repetidamente usada em diversas funções dentro do sistema, para controlar as iterações dos loops.
- Outros registradores foram usados de maneiras diversas e para informações diversas entre as diferentes funções implementadas
- Para leitura e armazenamento de novas strings (em particular, para armazenar novos códigos de trem e seus destinos), memória é alocada através do comando syscall 9
- Os vetores são iterados através da soma do imediato 4 (ou seus múltiplos) ao endereço base do vetor.

6. Funções principais

As funções principais e mais relevantes do sistema são as seguintes:

6.1 função: adicionar_trem

- Recebe do usuário: código único de trem, horário previsto de chegada e sigla da cidade de destino
- Registradores:
    - $s1: usado para armazenar a string de código único do trem (ou seja, armazena o endereço de memória do começo da string)
    - $s2: usado para armazenar a string da sigla da cidade de destino do trem (ou seja, armazena o endereço de memória do começo da string)
    - $t1: usado para analisar cada caractere da string durante sua validação
    - $t2: armazena o horário previsto de chegada do trem
    - $t4: armazena as horas do horário previsto de chegada do trem
    - $t5: armazena os minutos do horário previsto de chegada do trem
    - Outros registradores são usados de maneira auxiliar
- Após adição dos dados aos vetores correspondentes, a função bubble_sort é chamada para organizar a lista de trens.

6.2 função: buscar_trem

- Recebe do usuário o código único do trem sendo buscado
- Registradores:

    - $s2: usado para armazenar a string de código único do trem sendo buscado (ou seja, armazena o endereço de memória do começo da string)
    - $s3: usado para armazenar o vetor de códigos únicos dos trens cadastrados
    - $s4: usado para armazenar o vetor de horários de chegada dos trens cadastrados
    - $s5: usado para armazenar o vetor de estado dos trens cadastrados
    - $s6: usado para armazenar o vetor das siglas de destino dos trens cadastrados
    - Outros registradores são usados de maneira auxiliar
- Esta função é usada, também, como auxiliar para encontrar o trem de acordo com o seu código. Com isso, retorna o trem que foi buscado.

6.3 função: bubblesort

- Esta é uma função auxiliar, porém, é um algoritmo não trivial para os requisitos do projeto. Ela recebe as listas de códigos dos trens, horários de chegada, destinos e estados, para ordenar de maneira crescente de acordo com horário previsto de chegada
- Registradores usados:
    - $t0: horário 1 para comparação
    - $t1: horário 2 para comparação
    - $t3: limite para o for de fora (outer_loop)
    - $t4: contador para o for de dentro (inner_loop)
    - $t5: limite para o for de dentro (inner_loop)
    - Outros registradores são usados de maneira auxiliar
- Após a execução desta função auxiliar, a lista de trens está organizada de maneira crescente de acordo com os horários de chegada.

6.4 função: remover_trem

- Esta função utiliza a função buscar_trem como auxiliar para encontrar o trem em questão e, após encontrá-lo e receber o retorno desta busca, a função limpa completamente os dados do trem removido, utiliza a função bubble sort como auxiliar para organizar os trens e, por fim, subtrai 1 do valor total de trens cadastrados

---

7. Uso de IA Generativa

7.1 Ferramentas de IA utilizadas

A única ferramenta de IA utilizada foi o Gemini AI, do Google.

7.2 Como a IA foi utilizada no projeto

A IA foi utilizada para que o grupo pudesse aprender e entender como trabalhar com vetores e matrizes em MIPS e também como receber e salvar strings inseridas pelo usuário

7.3 Tabela de trechos gerados pela IA

Local do código	Trecho gerado pela IA (resumo)	Explicação escrita pelo grupo	Ajustes feitos
adicionar_trem e buscar_trem	**Para salvar uma string dada por um usuário, é necessário realizar o equivalente à um malloc, utilizando o syscall 9**	**Precisamos alocar memória para que seja possível salvar as strings dadas pelo usuário, e isso é feito através do syscall 9**	**Não foram necessários ajustes, foi uma resposta simples**
.data	**Para salvar vetores de strings (matrizes de caracteres) é necessário declarar em .data como .words com um número de valores (geralmente 0) para representar o tamanho do vetor**	**Precisamos fazer .word para declararmos os vetores no começo do código assembly**	**Não foram necessários ajustes, foi uma resposta simples**

7.4 O que tivemos que corrigir no código gerado pela IA

Por conta da simplicidade das respostas buscadas, não foram necessárias mudanças em códigos gerados pela IA, sendo que foram usadas apenas 1-2 linhas para cada uma das duas perguntas feitas.

---

8. Testes e Resultados

8.1 Casos de teste utilizados

Foram utilizados os seguintes casos de teste:

- Adição de 3 trens em ordem crescente (ordenada) de horários de chegada
- Adição de 4 trens em ordem decrescente (pior caso) de horários de chegada, para verificar funcionamento do algoritmo de ordenação
- Tentativa de adicionar mais do que 10 trens ao sistema
- Adição de diferentes trens com horários previstos de chegada atrasados e dentro do tempo, para analisar algoritmo de verificação de atrasos do trem

8.2 Screenshots ou trechos de saída

```
Digite um comando ou 0 para ver a lista de comandos
TREM  DESTINO  PREVISAO  STATUS
Z999      XYZ      10:00     ATRASADO
L050      SJC      12:15     A CAMINHO
T007      RIO      14:00     A CAMINHO
C101      ABC      15:30     A CAMINHO
```

```
TREM   DESTINO   PREVISAO   STATUS
L234       SJC      13:00    ATRASADO
L345       CJD      14:00    ATRASADO
L123       GUA      17:00    A CAMINHO
L456       GUA      20:00    A CAMINHO
TREM   DESTINO   PREVISAO   STATUS
L234       SJC      13:00    ATRASADO
L345       CJD      14:00    CHEGOU
L123       GUA      17:00    A CAMINHO
L456       GUA      20:00    A CAMINHO
```

8.3 Discussão dos resultados

Todas as funcionalidades foram testadas e funcionam como previsto, após ajustes e correções de bugs. Uma sugestão de melhoria para o código seria a possibilidade de leitura de entrada através de arquivos, o que poderia ter facilitado o teste do sistema como um todo.

9. Conclusão

Através deste trabalho, o grupo foi capaz de desenvolver suas habilidades em linguagem Assembly MIPS, enfrentando o desafio de desenvolver um sistema maior do que o de costume, de acordo com os exercícios realizados na linguagem. Um desafio em particular ocorreu durante o desenvolvimento da função bubble sort. Apesar de ser um algoritmo de ordenação com implementação considerada fácil, por conta do baixo nível apresentado pela linguagem assembly MIPS, a dificuldade de implementação do algoritmo acaba aumentando.

---

10. Referências

BURCH, Carl. **MIPS Reference Card**. [S.l.: s.n.], [s.d.]. Disponível em: https://cburch.com/cs/330/reading/mips-ref.pdf. Acesso em: 01 dez. 2025.
