UNIVERSIDADE FEDERAL DE SANTA CATARINA
ENGENHARIA DA COMPUTAÇÃO

DIOGO BORGES CORSO
JOÃO VITOR WAGNER PEREIRA
LIVIA SOUZA RECH

DESENVOLVIMENTO DE UM SISTEMA DE CONTROLE PARA MÁQUINA DE CAFÉ EM ASSEMBLY MIPS

ARARANGUÁ
JULHO - 2025

DIOGO BORGES CORSO
JOÃO VITOR WAGNER PEREIRA
LÍVIA SOUZA RECH

DESENVOLVIMENTO DE UM SISTEMA DE CONTROLE PARA MÁQUINA DE CAFÉ EM ASSEMBLY MIPS

Relatório de projeto apresentado como requisito parcial para a avaliação na disciplina de Organização e Arquitetura de Computadores I do curso de Engenharia da Computação da Universidade Federal de Santa Catarina.

ARARANGUÁ
JULHO - 2025

RESUMO

Este relatório detalha o processo de projeto e desenvolvimento de um software de controle para uma máquina de café comercial, implementado em Assembly MIPS e validado no simulador MARS. O objetivo do projeto foi criar um sistema embarcado capaz de gerenciar a preparação de três tipos de bebidas — Café Puro, Café com Leite e Mochaccino — com opções de dois tamanhos e adição de açúcar. A metodologia envolveu a estruturação do código em procedimentos para modularizar funções essenciais, como a interface com o usuário via console, o controle de estoque de ingredientes (café, leite, chocolate e açúcar), o cálculo de preços, a temporização do processo de preparo da bebida através de chamadas de sistema (syscall), e a geração automática de um cupom fiscal em formato de arquivo de texto (.txt) para cada transação. O sistema final implementa com sucesso todos os requisitos funcionais especificados, incluindo um modo de manutenção para reabastecimento dos insumos. A única divergência em relação à proposta original foi a não utilização da ferramenta Digital Lab Sim, optando-se pelo console do simulador MARS para a interação com o usuário. O projeto, portanto, representa uma aplicação prática e completa dos conceitos de programação em baixo nível, manipulação de memória e interação com o sistema operacional.

Palavras-chave: MIPS Assembly, Sistema de Controle, Máquina de Café, Sistemas Embarcados, Geração de Arquivos

SUMÁRIO

LISTA DE FIGURAS E TABELAS

1 INTRODUÇÃO

1.1. CONTEXTUALIZAÇÃO DO PROJETO

A arquitetura de computadores é a disciplina que estabelece a ponte entre o software e o hardware, definindo o modelo de programação e a interface que permite a execução de instruções por uma máquina. A programação em linguagens de baixo nível, como Assembly, oferece um controle direto sobre os recursos do processador e periféricos, sendo uma ferramenta essencial no desenvolvimento de sistemas embarcados, drivers de dispositivos e software de alta performance. Enquanto linguagens de alto nível abstraem a complexidade do hardware, o Assembly permite ao programador gerenciar a memória, os registradores e as operações da máquina de forma explícita, o que é fundamental para o controle de equipamentos específicos.

Nesse contexto, o presente trabalho detalha o desenvolvimento completo de um software de controle para uma máquina de café, utilizando Assembly MIPS. O projeto simula um sistema embarcado comercial, onde a lógica de operação, a interação com o usuário e o gerenciamento de recursos físicos são implementados em baixo nível. Esta aplicação prática serve para consolidar os conhecimentos teóricos da disciplina de Organização e Arquitetura de Computadores I, demonstrando como os conceitos de conjunto de instruções, manipulação de memória e chamadas de sistema são aplicados para resolver um problema do mundo real.

1.2. OBJETIVOS GERAIS E ESPECÍFICOS

O objetivo central deste projeto foi desenvolver um programa em Assembly para o processador MIPS, utilizando o simulador MARS, para realizar o controle completo de uma máquina de café para uso em ambientes comerciais ou empresariais.

Para atingir este objetivo principal, foram definidos os seguintes objetivos específicos:

- Oferecer múltiplas opções de bebidas: A máquina deveria ser capaz de preparar três tipos de bebidas (Café puro, Café com Leite e Mochaccino) em dois tamanhos diferentes (pequeno e grande).

- Permitir a personalização: O usuário deve ter a opção de adicionar açúcar à sua bebida.
- Controlar os atuadores da máquina: Implementar um sistema de temporização (timer) para gerenciar o tempo de liberação de cada ingrediente em pó e da água, de acordo com o tamanho da bebida selecionada.
- Gerenciar o estoque de insumos: O sistema deveria controlar a quantidade de doses disponíveis em quatro contêineres (CAFÉ, LEITE, CHOCOLATE e AÇÚCAR), cada um com capacidade para 20 doses, e subtrair a quantidade usada a cada preparo.
- Implementar um sistema de bloqueio e manutenção: A máquina deveria bloquear a preparação de uma bebida caso não houvesse ingredientes suficientes e permitir o reabastecimento dos contêineres através de um comando no teclado.
- Gerar um comprovante da transação: Ao final de cada preparo, o programa deveria criar um cupom fiscal em um arquivo de texto (.txt), detalhando o pedido e o preço cobrado.
- Estruturar o código de forma modular: A programação deveria ser orientada a componentes, com o uso mandatório de procedimentos (funções) para organizar a lógica do software.

1.3. ESTRUTURA DO RELATÓRIO

Este relatório está estruturado para apresentar de forma clara e sequencial todas as etapas do projeto. A Seção 2 descreve a plataforma de desenvolvimento, abordando a arquitetura MIPS e o simulador MARS. A Seção 3 detalha a implementação do software, explicando a arquitetura do código, os módulos de interação, a lógica de negócio e o sistema de geração de arquivos. A Seção 4 apresenta os resultados obtidos, a validação funcional e uma discussão sobre os desafios encontrados. Por fim, a Seção 5 conclui o trabalho, sumarizando os resultados e os aprendizados do projeto.

2. PLATAFORMA E FERRAMENTAS DE DESENVOLVIMENTO

A realização deste projeto dependeu de uma plataforma de hardware e um conjunto de ferramentas de software específicas, que são fundamentais no campo de estudo de arquitetura de computadores. Esta seção descreve a arquitetura do processador MIPS, a linguagem Assembly correspondente e o ambiente de simulação MARS, que juntos formaram a base para o desenvolvimento do sistema de controle da máquina de café.

2.1. ARQUITETURA MIPS

MIPS (acrônimo para Microprocessor without Interlocked Pipeline Stages) é uma família de arquiteturas de microprocessadores do tipo RISC (Reduced Instruction Set Computer, ou Computador com Conjunto Reduzido de Instruções). Desenvolvida originalmente na Universidade de Stanford, a arquitetura MIPS se destaca por sua simplicidade e eficiência, características que a tornaram uma escolha popular tanto em ambientes acadêmicos quanto em sistemas embarcados, como roteadores e decodificadores de TV.

As principais características da filosofia RISC presentes na MIPS são:

- Conjunto de Instruções Reduzido: Possui um número menor de instruções, onde cada uma é projetada para executar uma operação simples e ser concluída rapidamente, idealmente em um único ciclo de clock.
- Arquitetura Load-Store: Apenas as instruções load e store podem acessar a memória principal. As operações aritméticas e lógicas são realizadas exclusivamente em dados armazenados nos registradores do processador. Isso simplifica o controle e acelera a execução.
- Formato de Instrução Fixo: Todas as instruções MIPS possuem o mesmo tamanho (32 bits), o que simplifica o processo de busca (fetch) e decodificação das instruções pelo hardware.
- Grande Conjunto de Registradores: A arquitetura disponibiliza um banco com 32 registradores de uso geral, incentivando os programadores e compiladores a

manterem os dados mais utilizados nos registradores para evitar acessos lentos à memória.

Devido a essa simplicidade e clareza de design, a arquitetura MIPS é frequentemente utilizada como ferramenta de ensino em cursos de organização e arquitetura de computadores.

2.2. LINGUAGEM ASSEMBLY MIPS

A linguagem Assembly é uma representação simbólica e legível por humanos da linguagem de máquina de um processador. Cada arquitetura de processador possui sua própria linguagem Assembly. No caso do MIPS, a linguagem fornece mnemônicos diretos para as instruções do hardware, como add para adição, lw para carregar uma palavra (load word) da memória e beq para desvio condicional (branch on equal).

O código-fonte do projeto (maquina_cafe.asm) foi inteiramente escrito em Assembly MIPS. A programação envolveu a manipulação direta de registradores (e.g., $t0, $a0, $v0) para armazenar dados temporários, passar argumentos para procedimentos e receber valores de retorno. Além das instruções, foram utilizadas diretivas do montador, como .data para definir a seção de dados do programa (variáveis e strings) e .text para indicar o início do código executável. O uso de rótulos (labels) foi essencial para implementar laços de repetição e desvios condicionais, estruturando o fluxo do programa.

2.3. SIMULADOR MARS

O desenvolvimento, teste e validação do programa foram realizados no MARS (MIPS Assembler and Runtime Simulator), conforme especificado nos requisitos do projeto. O MARS é um Ambiente de Desenvolvimento Integrado (IDE) projetado especificamente para a programação em Assembly MIPS, sendo amplamente utilizado para fins educacionais.

As principais funcionalidades do MARS utilizadas neste projeto foram:

- Editor de Código Integrado: O MARS oferece um editor de texto com destaque de sintaxe, facilitando a escrita e a leitura do código Assembly.

- Montador (Assembler): A ferramenta traduz o código-fonte em Assembly MIPS para o código de máquina binário correspondente que o processador MIPS seria capaz de executar.
- Simulador e Depurador: O MARS simula o comportamento de um processador MIPS, permitindo a execução do código passo a passo. Foi possível visualizar o estado de todos os 32 registradores e as posições de memória em tempo real, uma funcionalidade crucial para a depuração e verificação da lógica do programa.
- Suporte a Chamadas de Sistema (syscall): O simulador provê uma interface de chamadas de sistema que permite ao programa Assembly interagir com o ambiente, emulando um sistema operacional. Este recurso foi fundamental no projeto para realizar operações como imprimir texto no console, ler entradas do teclado, manipular arquivos e obter a hora do sistema para a implementação do timer.

3. DESENVOLVIMENTO DO SOFTWARE DE CONTROLE

Esta seção apresenta o processo de projeto e implementação do software da máquina de café. A abordagem seguiu a especificação do projeto, que exigia uma programação orientada a componentes, resultando em um código modular e legível, conforme detalhado a seguir.

3.1. ARQUITETURA GERAL DO CÓDIGO

O software foi estruturado de forma a separar os dados da lógica, utilizando as seções .data e .text do Assembly MIPS. A lógica principal foi centralizada no procedimento main, que orquestra as chamadas para outros procedimentos especializados, cada um responsável por uma tarefa específica.

3.1.1. Estrutura de Dados e Variáveis

Na seção .data, foram declaradas todas as variáveis globais, constantes e strings de texto necessárias para a operação da máquina. Os principais grupos de dados são:

- Controle de Estoque: Quatro variáveis do tipo .word (cafe, leite, chocolate, acucar) foram definidas para armazenar a quantidade de doses de cada ingrediente. Cada uma é iniciada com 20 doses.
- Contador de Cupons: Uma variável .word chamada contador é usada para garantir que cada cupom fiscal tenha um número sequencial único no nome do arquivo.
- Preços: Um array de .word (precos_base) armazena os preços base das três bebidas, enquanto a variável adicional_gde guarda o valor extra para o tamanho grande.
- Mensagens da Interface: Diversas strings no formato .asciiz foram declaradas para exibir os menus (menu_principal, menu_reabastecer), solicitar informações ao usuário (menu_tamanho, menu_acucar) e fornecer feedback (msg_reabastecido, erro_estoque, preparando).

- Buffers de Memória: Espaços de memória (.space) foram reservados para buffers como nome_arquivo (para construir o nome do cupom) e cupom_conteudo (para montar o texto a ser salvo no arquivo).

3.1.2. Modularização com Procedimentos

Conforme exigido pela proposta do projeto, a programação foi fortemente baseada em procedimentos. Essa abordagem não apenas organiza o código, mas também facilita a depuração e a reutilização de funções. Os principais procedimentos desenvolvidos foram:

- main: Rotina principal que contém o laço de controle do programa, exibindo o menu e direcionando o fluxo com base na escolha do usuário.
- inicializar_estoques e inicializar_contador: Funções chamadas no início para configurar o estado inicial da máquina.
- reabastecer_estoque: Gerencia a lógica de manutenção para repor os ingredientes.
- verificar_estoque e reduzir_estoque: Responsáveis por validar a disponibilidade de insumos antes do preparo e por decrementar as quantidades após o uso.
- timer_bebida: Calcula e executa o tempo de preparo da bebida usando a syscall 30.
- gerar_cupom: Orquestra todo o processo de criação do arquivo .txt, incluindo a formatação do conteúdo e o salvamento em disco.
- Procedimentos Auxiliares: Um conjunto de funções de propósito geral foi criado para tarefas como int_to_ascii (converter inteiro para string), strcat (concatenar strings), calcular_preco e formatar_preco.

3.2. MÓDULO DE INTERAÇÃO COM O USUÁRIO

A comunicação com o usuário é feita inteiramente através do console do simulador MARS, utilizando chamadas de sistema para leitura e escrita.

3.2.1. Menu Principal e Seleção de Opções

O laço loop_menu_principal na rotina main é o ponto central da interação. Ele primeiro exibe o menu de opções usando a syscall 4 (impressão de string) com o texto da variável menu_principal. Em seguida, utiliza a syscall 5 (leitura de inteiro) para capturar a opção do usuário. Uma série de desvios condicionais (beq, bgt, blt) avalia a entrada e direciona a execução para a rotina apropriada, seja para selecionar uma bebida (opções 1-3), entrar no modo de manutenção (opção 5) ou encerrar o programa (opção 0). Se uma opção inválida for digitada, uma mensagem de erro é exibida e o menu é apresentado novamente.

3.2.2. Modo de Manutenção

Quando o usuário digita a opção 5 no menu principal, o programa desvia para o procedimento reabastecer_estoque. Esta rotina exibe um segundo menu, o menu_reabastecer, perguntando qual ingrediente deve ser reposto. Após o usuário selecionar o ingrediente (1 para café, 2 para leite, etc.), o software acessa a variável de estoque correspondente (cafe, leite, etc.) e restaura seu valor para 20, o máximo de doses. Uma mensagem de sucesso (msg_reabastecido) é exibida ao final do processo.

3.3. LÓGICA DE NEGÓCIO E PREPARO DA BEBIDA

Este conjunto de procedimentos constitui o núcleo funcional da máquina, gerenciando desde os insumos até a temporização.

3.3.1. Verificação e Redução de Estoque

Antes de iniciar qualquer preparo, o procedimento verificar_estoque é chamado. Ele determina a quantidade de doses necessárias (1 para o tamanho 'p' e 2 para o 'g') e então verifica se os contêineres dos ingredientes requeridos pela receita possuem essa quantidade. Se algum ingrediente for insuficiente, o procedimento retorna um código de

erro, e o main exibe a mensagem erro_estoque. Caso o estoque seja suficiente, o preparo prossegue e, ao final, o procedimento reduzir_estoque é invocado para subtrair as doses utilizadas das variáveis correspondentes.

3.3.2. Cálculo de Preços

O preço final da bebida é calculado no procedimento calcular_preco. A lógica acessa o array precos_base usando a opção da bebida (1, 2 ou 3) como índice para obter o valor inicial. Em seguida, ele verifica se o tamanho escolhido foi 'g' (grande). Em caso afirmativo, o valor da variável adicional_gde é somado ao preço base para formar o total.

3.3.3. Temporização do Preparo

A simulação do tempo de preparo é realizada pela rotina timer_bebida, conforme a especificação do projeto. O tempo total é calculado de forma incremental:

1. Inicia-se com o tempo de liberação de água: 5 segundos para bebida pequena ou 10 segundos para grande.
2. Para cada ingrediente em pó (café, leite, chocolate, açúcar) utilizado na receita, é adicionado 1 segundo por dose. Uma bebida grande, que requer duas doses, adiciona 2 segundos por ingrediente.

Para implementar o atraso, a rotina utiliza a syscall 30, que lê o relógio do sistema em milissegundos. Um laço de espera aguarda ativamente até que 1000 milissegundos (1 segundo) tenham se passado e repete esse processo até que o tempo total calculado seja atingido.

3.4. GERAÇÃO DO CUPOM FISCAL

Após o preparo da bebida, o programa gera um comprovante em um arquivo de texto, um requisito central do projeto.

3.4.1. Manipulação de Strings e Nomes de Arquivos

O procedimento gerar_cupom constrói dinamicamente o nome do arquivo para garantir sua unicidade. O processo utiliza a função auxiliar strcat para concatenar as seguintes partes em ordem:

1. A string base "cupom_" (da variável caminho_base).
2. O número atual do contador, que é primeiro convertido para uma string ASCII pela função int_to_ascii.
3. A extensão ".txt" (da variável extensao).

O resultado é um nome de arquivo como cupom_0.txt, cupom_1.txt, e assim por diante. Após a criação do nome, o valor da variável contador é incrementado e salvo para o próximo uso.

3.4.2. Escrita em Arquivo de Texto

Com o nome do arquivo e o conteúdo do cupom (uma longa string com os detalhes da bebida, tamanho, açúcar e preço formatado) prontos, o programa utiliza um trio de chamadas de sistema para a manipulação do arquivo:

- syscall 13 (Abrir Arquivo): É usada para criar e abrir o arquivo cupom_N.txt em modo de escrita.
- syscall 15 (Escrever em Arquivo): O conteúdo completo do cupom, previamente montado no buffer cupom_conteudo, é escrito no arquivo aberto.
- syscall 16 (Fechar Arquivo): O arquivo é fechado para garantir que todos os dados sejam salvos corretamente em disco.

Se a abertura do arquivo falhar por algum motivo, uma mensagem de erro (erro_arquivo) é exibida no console.

4. RESULTADOS E DISCUSSÃO

Nesta seção, são apresentados os resultados práticos obtidos com a execução do software no simulador MARS. É realizada uma análise comparativa entre os requisitos do projeto e a implementação final, seguida de uma discussão sobre os desafios encontrados durante o desenvolvimento e as soluções aplicadas.

4.1. VALIDAÇÃO FUNCIONAL NO SIMULADOR MARS

O software de controle da máquina de café foi integralmente montado e executado no ambiente de simulação MARS para a validação de seus requisitos funcionais. Os testes abrangeram todos os cenários de uso, desde um pedido padrão até o tratamento de erros e o modo de manutenção, confirmando que o programa se comporta conforme o esperado.

O fluxo de operação padrão inicia com a exibição do menu principal, conforme ilustrado na Figura 1, que apresenta as opções de bebidas e os modos de operação ao usuário. Após a seleção da bebida, o sistema guia o usuário através de solicitações sequenciais para a escolha do tamanho e a adição de açúcar, demonstrando uma interface de console funcional e intuitiva.

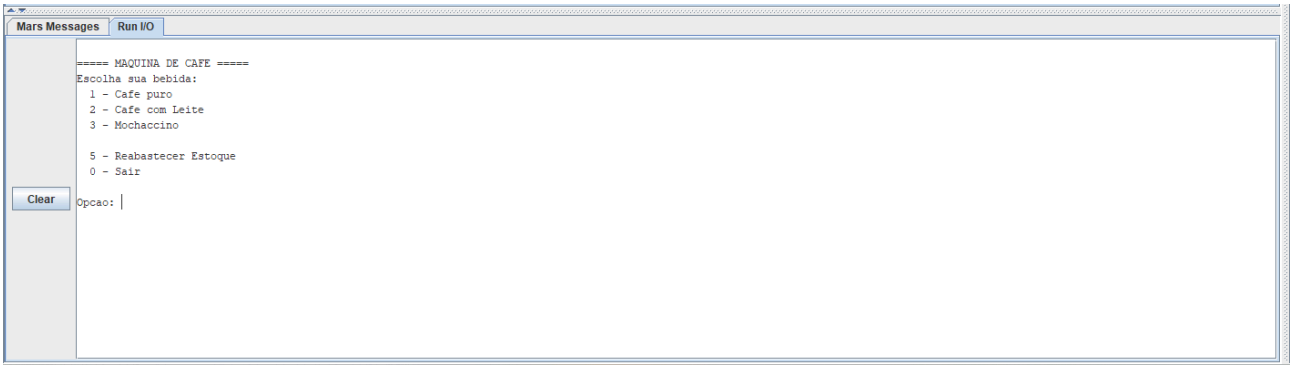

Figura 1 - Interação inicial com o usuário no console do MARS

Fonte: Autores

Uma vez que todas as opções são definidas, o sistema informa o início do processo e, após uma pausa simulada pelo timer, confirma a conclusão da bebida e a geração do comprovante. A Figura 2 demonstra as mensagens de status que fornecem feedback claro sobre o progresso da operação.

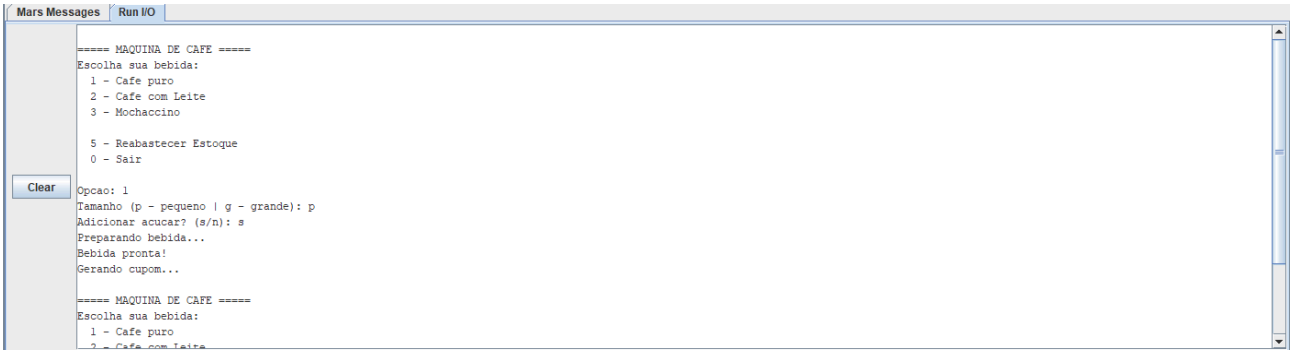

Figura 2 - Mensagens de status exibidas durante e após o preparo da bebida.

Fonte: Autores

A validação mais crítica do sucesso de uma transação é a correta geração do cupom fiscal em um arquivo .txt. A Figura 3 exibe o conteúdo de um arquivo de cupom gerado pelo sistema. A imagem comprova que todos os dados do pedido — nome da bebida, tamanho, opção de açúcar e preço final formatado — foram corretamente registrados, atendendo a um dos principais requisitos do projeto.

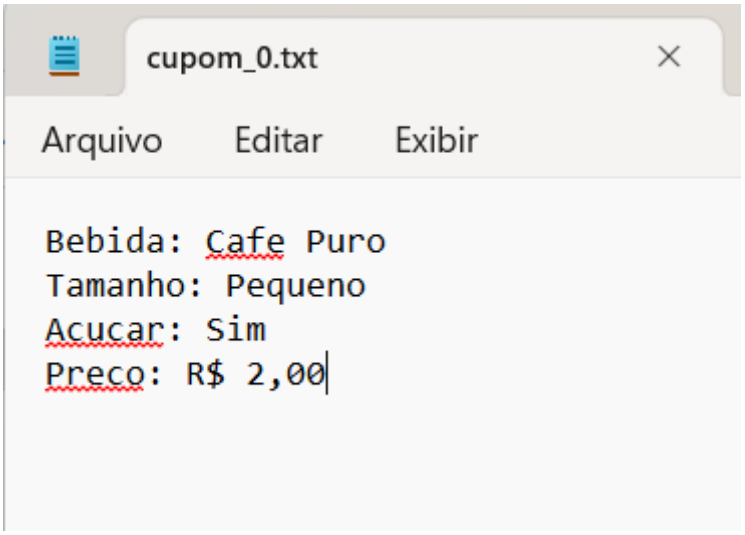

Figura 3 - Conteúdo de um cupom fiscal gerado com sucesso pelo programa.

Finalmente, o sistema de controle de estoque foi validado ao se forçar um cenário de erro. Ao tentar solicitar uma bebida que exigia mais doses de um ingrediente do que as disponíveis, o sistema corretamente bloqueou a operação e exibiu a mensagem de erro específica, conforme visto na Figura 4. Este teste confirmou a robustez da lógica de verificação de insumos.

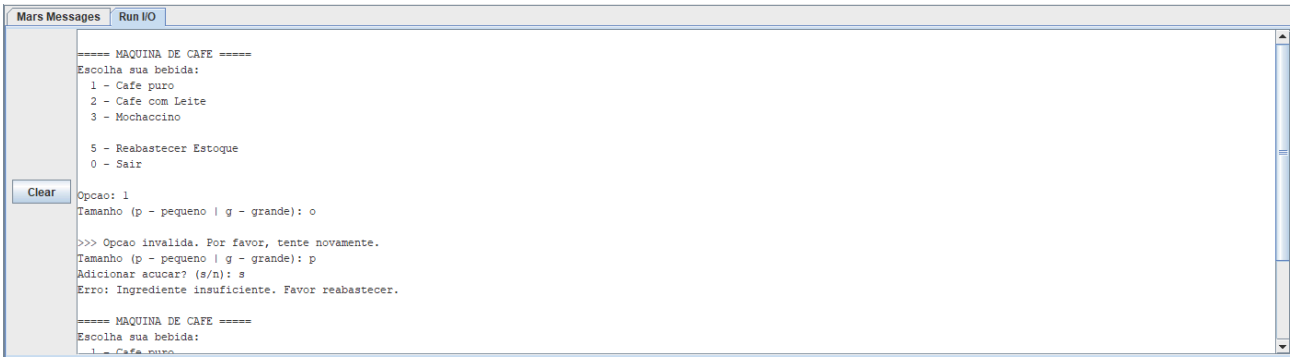

Figura 4 - Demonstração do mecanismo de bloqueio por falta de estoque.

Fonte: Autores

4.2. ANÁLISE DA IMPLEMENTAÇÃO VERSUS REQUISITOS

A implementação final atendeu à grande maioria dos requisitos definidos no documento da proposta do projeto. A tabela abaixo resume a comparação:

Requisito	Status	Observações
Três tipos de bebidas	Atendido	Implementado via menu de opções 1, 2 e 3.
Dois tamanhos de copo (p/g)	Atendido	O sistema solicita ao usuário a entrada dos caracteres 'p' ou 'g'.
Opção de adicionar açúcar (s/n)	Atendido	O sistema solicita ao usuário a entrada dos caracteres 's' ou 'n'.
Uso de timer para o preparo	Atendido	O procedimento timer_bebida utiliza a syscall 30 para simular o tempo de preparo.
Gerenciamento de estoque (20 doses)	Atendido	As variáveis no segmento .data controlam o estoque, que é devidamente reduzido.
Bloqueio por falta de ingrediente	Atendido	O procedimento verificar_estoque impede a preparação se não houver insumos.

Modo de reabastecimento	Atendido	A opção 5 do menu principal permite restaurar os estoques para o valor máximo.
Geração de cupom fiscal em .txt	Atendido	O procedimento gerar_cupom cria e salva um arquivo .txt com os detalhes do pedido.
Programação modular com procedimentos	Atendido	O código foi estruturado em múltiplos procedimentos para cada tarefa específica.
Uso da ferramenta Digital Lab Sim	Não Atendido	Conforme observado, esta ferramenta não foi utilizada. A interação foi adaptada para o console padrão do simulador MARS.

Tabela 1 - Requisitos Implementados

Fonte: Autores

A divergência principal foi a não utilização do Digital Lab Sim. Contudo, essa adaptação não comprometeu a lógica central nem os demais requisitos funcionais do projeto, que foram todos validados com sucesso usando as ferramentas de I/O do próprio MARS.

4.3. DESAFIOS ENCONTRADOS E SOLUÇÕES ADOTADAS

O desenvolvimento em Assembly MIPS apresentou desafios inerentes à programação de baixo nível, que exigiram soluções específicas:

- Manipulação de Strings e Conversão de Dados: Tarefas triviais em linguagens de alto nível, como concatenar strings ou converter um número inteiro para sua representação em texto, são complexas em Assembly. Para superar isso, foram criados procedimentos auxiliares dedicados e reutilizáveis: strcat para a concatenação, e int_to_ascii para a conversão. A criação do nome do arquivo cupom_N.txt e a formatação do conteúdo do cupom dependem diretamente dessas rotinas.
- Formatação de Moeda: Exibir o preço, armazenado em centavos (ex: 350), no formato monetário "R$ 3,50" exigiu uma lógica de formatação cuidadosa. O procedimento formatar_preco foi desenvolvido para essa finalidade, utilizando operações de divisão por 100 para separar a parte inteira (reais) e o resto (centavos). Foi necessário um tratamento especial para valores de centavos menores que 10, a fim de inserir o '0' à esquerda (ex: R$ 2,05).

- Persistência do Contador de Cupons: Para evitar a sobreescrita de cupons antigos a cada nova execução do programa, era necessário que o contador não iniciasse sempre em '1'. A solução implementada na rotina inicializar_contador foi criar um laço que, ao iniciar o programa, verifica a existência dos arquivos cupom_1.txt, cupom_2.txt, e assim por diante, até encontrar um número de arquivo que não existe. O contador é então inicializado com este primeiro número vago, garantindo a unicidade dos cupons entre diferentes sessões de uso da máquina.

5. CONCLUSÃO

O desenvolvimento do software de controle para a máquina de café em Assembly MIPS representou uma jornada completa pelo ciclo de vida de um projeto de sistema embarcado, desde a interpretação de requisitos até a implementação, teste e validação de uma solução funcional.

5.1. SÍNTESE DOS RESULTADOS

O objetivo primordial do projeto, que consistia em desenvolver um programa de controle para uma máquina de café utilizando a arquitetura MIPS, foi alcançado com sucesso. O software final, executado no simulador MARS, implementa de forma robusta todas as funcionalidades essenciais especificadas: oferece um menu com três tipos de bebidas e dois tamanhos, permite a personalização com açúcar, gerencia o estoque de quatro ingredientes distintos, bloqueia a operação em caso de insumos insuficientes e possui um modo de manutenção para reabastecimento.

Além disso, foram atendidos os requisitos técnicos cruciais, como a utilização de um timer baseado em chamadas de sistema para o preparo das bebidas e a geração de um cupom fiscal em formato .txt para cada transação. A arquitetura do código foi desenvolvida de maneira estritamente modular, com o uso intensivo de procedimentos, atendendo a uma das principais diretrizes do projeto e resultando em um programa organizado e de fácil manutenção. A única divergência em relação ao plano original foi a não utilização do Digital Lab Sim, o que não impediu a validação completa da lógica do programa.

5.2. APRENDIZADOS OBTIDOS

Este projeto proporcionou uma valiosa experiência prática, solidificando os conceitos teóricos da disciplina de Organização e Arquitetura de Computadores. A programação em Assembly MIPS exigiu uma compreensão profunda sobre como o processador opera em seu nível mais fundamental, incluindo o gerenciamento direto de registradores, o acesso à memória e a utilização do conjunto de instruções da máquina.

O desenvolvimento de funcionalidades como a manipulação de strings, a conversão entre tipos de dados e a interação com o sistema de arquivos através de syscalls destacou a complexidade que é normalmente abstraída por linguagens de programação de alto nível. A necessidade de construir essas rotinas a partir de operações básicas reforçou a importância das abstrações de software e o valor das bibliotecas padrão. A depuração, realizada através da observação passo a passo do estado dos registradores e da memória no MARS, foi uma lição fundamental sobre a importância de metodologias de teste rigorosas em programação de baixo nível.

5.3. PROPOSTAS DE MELHORIAS E TRABALHOS FUTUROS

Embora o projeto tenha sido concluído com sucesso, ele abre portas para diversas melhorias e expansões futuras que poderiam torná-lo ainda mais completo:

- Persistência de Dados: Atualmente, o estoque e o contador de cupons são reinicializados a cada execução. Uma melhoria significativa seria salvar o estado da máquina (níveis de estoque, último número de cupom) em um arquivo de configuração ao sair e carregá-lo ao iniciar, tornando a simulação mais realista.
- Interface com o Usuário: A interface via console poderia ser substituída por uma interface visual mais rica, utilizando o display de bitmap do MARS ou, como planejado originalmente, integrando a ferramenta Digital Lab Sim para criar uma experiência de usuário mais interativa.
- Sistema de Pagamento: Poderia ser implementado um módulo para simular um sistema de pagamento, no qual o usuário precisaria inserir créditos virtuais antes de poder selecionar uma bebida, adicionando uma nova camada de complexidade à lógica do programa.
- Relatórios Gerenciais: Uma funcionalidade adicional no modo de manutenção poderia gerar relatórios de vendas, salvando em um arquivo de texto um resumo de quantas bebidas de cada tipo foram vendidas e o faturamento total do período.

Em suma, o projeto atingiu plenamente seus objetivos educacionais, servindo como uma demonstração prática e abrangente dos desafios e das recompensas da programação de sistemas em baixo nível.

REFERÊNCIAS

1 UNIVERSIDADE FEDERAL DE SANTA CATARINA (UFSC). **Proposta de projeto: Máquina de Café.** Material da disciplina de Organização e Arquitetura de Computadores I. Araranguá, 2024. (Fornecido pelo professor via Moodle).

2 PATTERSON, David A.; HENNESSY, John L. **Organização e Projeto de Computadores: A Interface Hardware/Software.** 5ª Edição MIPS. Rio de Janeiro: LTC, 2017.

3 VOLLMAR, Ken; SANDERSON, Pete. **MARS (MIPS Assembler and Runtime Simulator).** Versão 4.5. Distribuído na disciplina de Organização e Arquitetura de Computadores I. Araranguá: UFSC, 2024. (Fornecido pelo professor via Moodle).

APÊNDICE A — CÓDIGO FONTE COMPLETO

O código-fonte completo do projeto, desenvolvido em Assembly MIPS, está disponível para consulta no seguinte repositório online:

Disponível em: https://github.com/daiogocrs/maquina-de-cafe-mars