RELATÓRIO FINAL – PROJETO MIPS - CineAOC

Disciplina: Arquitetura e Organização de Computadores – UNIFESP/ICT

Capa

CineAOC

Larissa Lumi Hayakawa de Sá - 176551

Laura Carvalho Dalsan - 176553

Vinícius Gonçalves - 176549

https://github.com/vinicius-g/CineAOC

---

1. Resumo (6–10 linhas)

O programa CineAOC é um projeto desenvolvido para que os atendentes reservem os assentos do cliente de uma forma fácil e visualmente agradável. Para isso, o simulador possibilita o cadastro das sessões disponíveis, sendo possível acessar cada uma delas individualmente para a escolha dos lugares, nessa etapa, a sala é visualmente representada com as respectivas coordenadas dos assentos para que o cliente escolha um assento disponível, que não está marcado com um “X”. É importante ressaltar que todas as ações são verificadas, ou seja, qualquer ação inválida é notificada na interface e o usuário pode realizar uma nova tentativa, garantindo a integridade das reservas e uma interação clara e segura com o sistema.

---

2. Introdução (meia página a 1 página)

O projeto desenvolvido tem como contexto um sistema de gerenciamento de salas de cinema, permitindo o cadastro de sessões, visualização da disposição dos assentos e a reserva de lugares pelos usuários, simulando o funcionamento de um cinema real, onde

cada sessão possui uma sala com fileiras e colunas de assentos que podem estar livres ou ocupados.

O simulador executado no MARS nos permite observar claramente o uso dos registradores, o acesso à memória, a organização dos dados em vetores linearizados e o funcionamento das chamadas de sistema para entrada e saída de dados. Além disso, o fluxo de execução do programa é controlado explicitamente por meio de estruturas condicionais e chamadas de função, evidenciando como decisões e estruturas de repetição são implementadas sem o auxílio de abstrações presentes em linguagens de mais alto nível.

Dessa forma, o CineAOC está diretamente ligado à disciplina Arquitetura e Organização de Dados à medida que explora conceitos como a manipulação direta de registradores, implementação de matrizes e linearização utilizando offset, uso de ponteiros e controle de desvios condicionais, além da máquina de estados representada em menus. Esse projeto foi bastante útil para compreender melhor a forma que os comandos são executados pelos computadores, já que, diferente das linguagens de médio ou alto nível, no MIPS não são realizadas grandes abstrações.

---

3. Descrição Geral do Sistema (1–2 páginas)

3.1 Objetivos do simulador

O objetivo do simulador CineAOC é representar o funcionamento de um dispositivo embarcado responsável pelo gerenciamento de salas de cinema, simulando o processo de cadastro de sessões e reserva de assentos de forma eficiente e organizada. O programa foi desenvolvido para permitir que o usuário interaja diretamente com o sistema por meio de menus, possibilitando a criação de sessões, a visualização da disposição dos assentos e a realização de reservas, respeitando as regras de disponibilidade de cada lugar. Além disso, o programa deve garantir que apenas assentos válidos e disponíveis possam ser reservados, notificando o usuário em casos de erro, como coordenadas inválidas ou tentativas de reserva duplicadas.

3.2 Funcionalidades implementadas

3.2.1 Interface ASCII:

- Menu Inicial

```
=== CINEAOC ===
1. Cadastrar Nova Sessao
2. Entrar em Sessao
3. Sair
```

- Menu de cadastro de sessão

```
Escolha: 1

Digite o Titulo do Filme: Wicked
```

- Menu de escolha de sessão

```
--- ESCOLHA A SESSAO ---
0 - Wicked
Digite o ID da sessao (0, 1 ou 2): 0
```

- Mapa da sessão

```
=== SALA: Wicked ===
     1   2   3   4   5
   +-------------------+
A |[ ] [ ] [X] [ ] [ ] |
B |[ ] [ ] [ ] [ ] [ ] |
C |[ ] [ ] [ ] [ ] [ ] |
D |[ ] [ ] [ ] [ ] [ ] |
E |[ ] [ ] [ ] [ ] [ ] |
   +-------------------+
        [ TELA ]
```

- Menu de reserva de assento

```
[ MENU DA SALA ]
1. Reservar Assento
2. Voltar ao Menu Principal
```

3.2.2 Cadastro de sessões

É possível cadastrar até 3 sessões após iniciar a execução do código, o id da sessão é salvo em um registrador do tipo $s, que ficam salvos e disponíveis ao longo da execução. Ao cadastrar uma nova sessão um título de até 20 caracteres deve ser informado. Esse título é apresentado no momento da escolha da sessão, e é exibido como título no menu de reserva de cadeiras.

3.2.3 Reserva de assentos

É possível reservar por vez, 1 entre 25 assentos disponíveis por sessão, referentes a uma matriz 5 por 5, sendo que as linhas são representadas por letras maiúsculas e as colunas vão de 1 até 5. Assentos reservados são marcados por um X, enquanto assentos livres permanecem como colchetes vazios, assentos reservados claramente não podem ser reservados por outra pessoa então o sistema exibe uma mensagem padrão para o tratamento deste erro.

3.3 Arquitetura geral do programa

Inclua uma descrição textual (ou diagrama simples) mostrando:

- Módulos/funções principais:
    - *create_session*: verifica se o programa já inseriu a quantidade máxima de sessões, se sim, aparece uma mensagem de erro para evitar o overflow da memória, senão, basta digitar o nome do filme para cadastrar a nova sessão.
    - *list_sessions*: em conjunto com *list_loop* é a função responsável por exibir o título e o ID de cada sessão cadastrada no cinema. Aqui foi usada uma lógica de abstração de um laço “for” simples.
    - *room_menu:* é responsável por exibir o mapa da sala atual, permitindo que o usuário veja quais assentos já estão reservados. Além de exibir o menu da sessão para que o usuário possa reservar assentos na sala selecionada.
    - *do_booking:* é uma das funções principais do sistema, responsável por permitir a reserva de assentos em conjunto com as funções *make_upper* e *check_upper*, que cuidam do tratamento da seleção da coordenada de linha do assento, a letra selecionada é convertida para um valor numérico que é multiplicado por 5 e somado a coluna, resultando na posição na memória referente a posição na matriz da sala, primeiro verifica se o lugar está ocupado e se não estiver, altera o estado para reservado.
    - *print_map*: uma das funções mais importantes e que é chamada com mais frequência, responsável por imprimir em ASCII o mapa da sala, com o título da sessão selecionada, o identificador das linhas e das colunas, cada cadeira exibindo se está reservada ou não, marcando com um “X” as que estejam reservadas. Para uma tarefa com tantas etapas são necessárias algumas funções e laços, configurados em *map_row_loop, map_col_loop, pr_free, map_nxt, map_row_end* e *map_end*.
    - *remove_newline:* a função responsável por permitir o uso correto de strings ao longo do código, por ser um sistema que recebe dados de entrada do computador e registra o envio através da tecla “enter”, quando a string é digitada ela salva o valor da tecla “enter” como um “/n”, essa função percorre a string e remove esse valor, substituindo pelo caractere padrão de final de linha “/0”.

- Fluxo principal do programa

    No fluxo ideal do sistema o usuário vai iniciar o programa, a função *main* irá exibir o menu principal do sistema com a opção de cadastrar uma nova sessão, usuário cadastra o novo título da sessão, seleciona a opção de exibir as sessões e seleciona a opção que acabou de cadastrar. Entrando na sessão ele reserva o lugar das pessoas que chegarem para assistir ao filme, informando sempre quando o lugar já está ocupado, já que ele consegue ter a visão geral da sala através do painel. Ao final do dia ele retorna para o menu principal e seleciona a opção de sair para encerrar o programa.

- Relação entre funções e dados

O projeto possui a separação entre funções e dados através das palavras reservadas .data e .text, os dados e informações principais do programa estão na parte de .data, como as string principai que são impressas, as variáveis de controle, contadores que devem se manter ao longo de toda a execução e a definição da matriz linearizada que salva as informações de reserva de cada um dos assentos, bem como o vetor que guarda os títulos. Já na parte do código que executa as funções, realiza as chamadas e altera os valores, temos funções que salvam novos dados, como o vetor de títulos que reserva 3 espaços de 20 bytes para salvar até 3 títulos com 20 caracteres, ou a matriz linearizada de assentos que possui 75 bytes, que usamos para dividir entre 3 sessões com 25 cadeiras cada, ou seja cada byte representa “0” como assento livre e “1” como assento ocupado. Ao decorrer do código por várias vezes utilizamos offsets justamente para controlar a alocação de valores para cada espaço de memória de forma correta, a lógica para alocação dos assentos consiste na atualização do registrador que armazena o id da sessão atual, e multiplicação desse valor por 25, dessa forma temos o offset necessário para que apenas a parte da memória desta sessão seja atualizado, o assento é reservado após isso, multiplicando o valor da linha selecionada por 5 e somando o valor da coluna, dessa forma o endereço acessado é atualizado na memória como “assento reservado”.

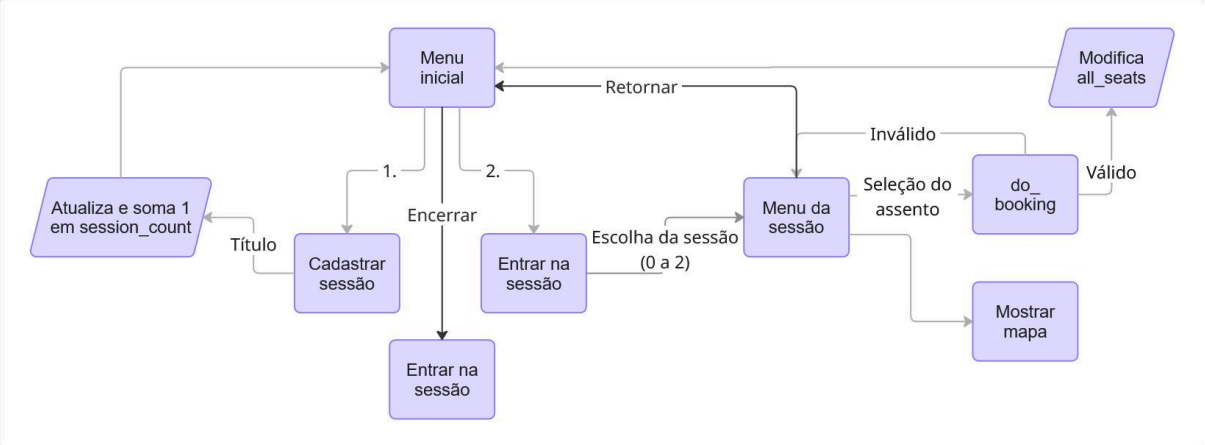
---

4. Máquina de Estados (1–2 páginas)

4.1 Estados do sistema e transições

Os estados principais do programa são dados pelos menus mostrados na seção 3.2 deste documento. O menu principal controla 3 estados referentes às funções principais, onde é possível através do estado 1 cadastrar uma nova sessão, o estado 2 é responsável por escolher uma sessão e o estado 3 é responsável por encerrar o programa. Entrando no menu da sessão, utilizado para reservar os assentos temos outro menu com mais 2 estados, o estado 1 é responsável por iniciar o processo de reserva de um assento, onde o usuário irá escolher as coordenadas da cadeira que deseja e o estado 2 é responsável por retornar de forma segura para o menu inicial a qualquer momento.

4.2 Diagrama de estados



---

5. Estruturas de Dados (1–2 páginas)

Descreva as estruturas usadas:

- Vetores (ex.: histórico de leituras): titles (títulos dos filmes), vetor de todos os assentos (3 salas x 25 lugares = 75 bytes)
- Matrizes (se houver): linearizada pelo vetor de todos os assentos
- Representação de sensores e atuadores: Não se aplica
- Organização de dados na memória: estruturas estaticamente alocadas
- Convenção de endereço base + deslocamento: usada na variável *all_seats*, o valor em *current_base* multiplicado por 25 é a base, e o deslocamento é a multiplicação da linha por 5, somado com a coluna.

6. Funções principais (2–3 páginas)

6.1 função: create_session

- Parâmetros de entrada: variáveis globais
- Registradores usados: $t2 (deslocamento do título), $t1 (base do vetor de títulos), $t0 (deslocamento do título), $a1 e $a0 (parâmetros) e $v0 (syscall)
- O que retorna: atualiza a variável *titles* (vetor que guarda os títulos das sessões)
- Lógica geral: Se já existem 3 sessões ou nenhuma sessão retorna uma mensagem de erro padrão, evitando que haja overflow ou acesso incorreto, caso contrário, você poderá digitar o título do filme e ele será armazenado no vetor *titles* com a base sendo a multiplicação do contador de sessões por 20, em seguida, a quantidade de sessões é incrementada, aparece uma mensagem de sucesso e finalmente volta ao menu inicial.

6.2 função: list_sessions

- Parâmetros de entrada: variáveis globais
- Registradores usados: $t0 (total de sessões e depois ID escolhido), $t1 (contador de laço), $t2 (endereço base dos vetores), $t3 (deslocamento calculado), $s7 (ponteiro calculado para os assentos).
- O que retorna: Atualiza os registradores globais *current_base* e *current_ptr.*
- Lógica geral: A função lista as sessões cadastradas exibindo o ID e o título de cada uma, após a listagem, solicita ao usuário o ID da sessão desejada e valida a entrada. Caso o ID seja válido, a função calcula o endereço base dos assentos da sessão selecionada e armazena esse ponteiro para uso posterior. Em caso de erro, uma mensagem é exibida e o sistema retorna ao menu principal.

6.3 função: do_booking

- Parâmetros de entrada: letra representante da linha e número da coluna
- Registradores usados: $t0 (linha), $t1 (coluna), $t2 (cálculo da posição), $t3 (endereço final na memória), $t4 (status atual do assento), $t5 (valor 1 para ocupação).
- O que retorna: Altera um byte específico na memória do vetor *all_seats* de 0 para 1.
- Lógica geral: A função normaliza as coordenadas de entrada e verifica se estão dentro dos limites da sala de 5 por 5. Em seguida, converte a coordenada em um endereço linear, somando o deslocamento ao ponteiro base da sessão $s7. O sistema lê o byte nesse endereço: se for 0, grava o valor 1 para reservar se já for 1, informa que o lugar está ocupado para evitar erros na gravação.

6.4 função: room_menu

- Parâmetros de entrada: Número da sessão
- Registradores usados: $t0 (opção do menu), $a0 (parâmetros), $v0 (syscalls)
- O que retorna: Não retorna valores.
- Lógica geral: A função exibe o mapa da sala da sessão selecionada e apresenta um menu com as opções de reservar o assento (ou retornar ao menu principal). Com base na escolha do usuário, a função redireciona a execução para a função de reserva *do_booking* ou retorna ao estado inicial do programa.

6.5 função: check_upper

- Parâmetros de entrada: Número da sessão
- Registradores usados: $t0 (linha), $t1 (coluna), $t2 (cálculo do deslocamento), $t3 (endereço final na memória), $t4 (valor lido da memória), $s7 (endereço base da sessão).
- O que retorna: Realiza a gravação do valor 1 na memória se o assento estiver livre ou desvia para erro.

- Lógica geral: A função normaliza a letra da fileira subtraindo 65 para obter um índice numérico e valida se a linha e coluna estão dentro dos limites da matriz. Em seguida converte as coordenadas, multiplicando a linha por 5 e somando a coluna. O resultado é adicionado ao ponteiro base em $s7 para acessar o byte exato na memória, onde o sistema verifica se o valor é zero antes de confirmar a reserva gravando o valor 1.

6.6 função: print_map

- Parâmetros de entrada: Strings do programa
- Registradores usados: $t0 (contador de linhas), $t1 (contador de colunas), $s1 (ponteiro auxiliar de percurso), $t3 (valor do assento atual).
- O que retorna**:** Desenha a representação visual da sala em ASCII.
- Lógica geral: A função exibe o cabeçalho e inicializa um ponteiro temporário $s1 com o endereço base da sessão para percorrer a memória. Utilizando dois laços de repetição, ela lê cada byte e imprime a cadeira livre ou ocupada. Ao final de cada ciclo de colunas a função insere uma quebra de linha e bordas para manter o formato visual de grade 5 por 5.

6.7 função: remove_newline

- Parâmetros de entrada: Uma string qualquer formatada incorretamente
- Registradores usados: $t0 (endereço atual na string), $t1 (caractere carregado).
- O que retorna: uma string formatada corretamente, sem o “/n” e com o “/0”.
- Lógica geral: A função passa pela string procurando pelo valor ASCII de 10 que corresponde à tecla “enter”. Ao encontrar esse caractere ela o substitui imediatamente por um “/0”. Isso garante que ao imprimir o nome do filme posteriormente o texto não cause uma quebra de linha indesejada que desmontava o layout do mapa.

---

7. Uso de IA Generativa (1–2 páginas)

7.1 Ferramentas de IA utilizadas

A ferramenta de IA mais utilizada foi o Google Gemini Pro, que foi oferecido para vários estudantes durante uma promoção com vagas limitadas, e como membros da Universidade Federal de São Paulo um dos membros do nosso grupo conseguiu adquirir o acesso, o raciocínio da versão pro é muito melhor e o contexto se mantém por uma conversa extensa, o que foi útil para que a IA recordasse de outros trechos do código para ajudar a desenvolver

7.2 Como a IA foi utilizada no projeto

Utilizamos a IA majoritariamente para corrigir erros ao longo do desenvolvimento, bem como um “co piloto”, enquanto o código era desenvolvido e a lógica era pensada, caso houvesse algum erro, ou comportamento inesperado o trecho do código era passado na IA para que ela dissesse o que estava incorreto, sempre passando um contexto geral do código até então. Não houve um trecho específico em que usamos IA, foi mais como um desenvolvimento conjunto, isso não exclui nosso esforço para o desenvolvimento, apenas agiliza encontrar erros comuns enfrentados como: registrador incorreto utilizado, nome de variável errado, mal posicionamento das funções, falta de “;” no final das linhas, erro nos cálculos de forma manual para alocação da memória, erro na chamada de funções, sejam parâmetros incorretos, escrita incorreta ou mal entendimento de como a função efetivamente funcionava, dentre outros erros que ocorreram menos.

A principal dificuldade era que mesmo que a IA mantivesse um bom contexto, por vezes esquecemos de atualizar o contexto, ou seja, alteramos grandes porções do código porque tivemos novas ideias, mas não passamos isso para a IA, então a gente acabava travando em alguma outra parte do desenvolvimento e quando solicitamos apoio as respostas eram incongruentes, já que a IA não tinha a informação de que alteramos o início ou o meio do código, então é uma questão de erro mais humano do que da máquina efetivamente, soluções como o github Copilot, que agora está integrado no Visual Studio Code por exemplo, são muito eficientes para esse tipo de uso da IA, já que a IA sempre sabe o contexto completo do seu código, seja de configuração, código em outras pastas e arquivos completos, isso é muito útil para desenvolvimento de aplicações gigantescas e até para desenvolvimento de sistemas já muito bem estabelecidos, que às vezes precisam de atualização em apenas uma função, mas que faz parte de uma arquitetura muito maior e que necessita de cuidado, com essa integração completa com códigos que estão até em outras pastas, o desenvolvimento com a IA pode ser de grande apoio para evitar “boilerplates” que são códigos comuns mas que demandam algum tempo para escrever, e até para melhor entendimento de um sistema muito grande.

Por mais que nosso código não fosse tão extenso e nem tivesse tanto contexto para ser aprendido, optamos por utilizar essa estratégia de desenvolvimento, que se demonstrou muito útil.

7.3 Tabela de trechos gerados pela IA

Não se aplica, metodologia de uso de IA explicada no tópico anterior.

7.4 O que tivemos que corrigir no código gerado pela IA

Os principais erros apontados foram em questão de contexto, quando nós desenvolvemos uma função inteira e não passamos para a IA, então ela acaba gerando respostas alucinadas que não batiam com o nosso código atual, algumas vezes mesmo assim acabava sendo útil, porque a forma como o código estava estruturado já nos dava uma visão melhor de onde estava o nosso erro.

A IA não deu erros notáveis ao longo do desenvolvimento, nomes mantinham a constância, funções eram usadas corretamente, jump, jal, jar, funções complexas como

essas que envolviam entendimento do código, não apresentavam dificuldades para serem entendidas pela IA, então como pontos de correção do grupo, o que é possível ressaltar era adaptação para nossa linguagem, seja para convenção de Camel Case utilizada pelo grupo ou até correções de ASCII que o grupo achou mais interessante, mas de forma geral, poucos erros observados no desenvolvimento dos trechos de IA.

---

8. Testes e Resultados (1–2 páginas)

8.1 Casos de teste utilizados

- Um caso típico:

```
--- ESCOLHA A SESSAO ---
0 - Wicked
Digite o ID da sessao (0, 1 ou 2): 0


=== SALA: Wicked ===
     1   2   3   4   5
   +-------------------+
A |[ ] [ ] [X] [ ] [ ] |
B |[ ] [ ] [ ] [ ] [ ] |
C |[ ] [ ] [ ] [ ] [ ] |
D |[ ] [ ] [ ] [ ] [ ] |
E |[ ] [ ] [ ] [ ] [ ] |
   +-------------------+
         [ TELA ]
```

  A sessão foi selecionada e foi possível abrir e exibir os assentos ocupados e disponíveis da sessão específica.

- Um caso extremo ou adverso:

```
[ MENU DA SALA ]
1. Reservar Assento
2. Voltar ao Menu Principal
Escolha: 1

Fileira (A-E): r
Coluna  (1-5): 1
[ERRO] Coordenada invalida!


=== SALA: Wicked ===
     1   2   3   4   5
   +-------------------+
A |[ ] [ ] [X] [ ] [ ] |
B |[ ] [ ] [ ] [ ] [ ] |
C |[ ] [ ] [ ] [ ] [ ] |
D |[ ] [ ] [ ] [ ] [ ] |
E |[ ] [ ] [ ] [ ] [ ] |
   +-------------------+
        [ TELA ]
```

Coordenada inválida, selecionando uma letra que não está entre as disponíveis na sessão.

```
[ MENU DA SALA ]
1. Reservar Assento
2. Voltar ao Menu Principal
Escolha: 1

Fileira (A-E): r
Coluna  (1-5): a
```

Coordenada inválida, mas dessa vez foi selecionada uma letra no local que se estava esperando um número, isso faz com que o sistema encerre de forma inesperada. Como é um caso muito atípico, que ocorre apenas após um erro não esperado, não gastei mais tempo tentando validar essa entrada.

- Um caso estrutural quando o vetor de sessão está “cheio”:

```
=== CINEAOC ===
1. Cadastrar Nova Sessao
2. Entrar em Sessao
3. Sair
Escolha: 1

[ERRO] Limite de 3 sessoes atingido! Impossivel criar mais.
```

Na versão inicial tentar cadastrar a quarta sessão quebrava a execução do código, agora há uma validação da quantidade de sessões para evitar esse erro de overflow, garantindo maior integridade do sistema.

8.2 Screenshots ou trechos de saída

Reserva realizada:

```
=== SALA: Wicked ===
     1   2   3   4   5
   +-------------------+
A |[ ] [ ] [ ] [ ] [ ] |
B |[ ] [ ] [ ] [ ] [ ] |
C |[ ] [ ] [ ] [ ] [ ] |
D |[ ] [ ] [ ] [ ] [ ] |
E |[ ] [ ] [ ] [X] [ ] |
   +-------------------+
         [ TELA ]

[ MENU DA SALA ]
1. Reservar Assento
2. Voltar ao Menu Principal
Escolha: 1

Fileira (A-E): e
Coluna  (1-5): 5
Reserva realizada com SUCESSO!
```

Impossibilidade de reservar um assento já reservado:

```
=== SALA: Wicked ===
     1   2   3   4   5
   +-------------------+
A |[ ] [ ] [ ] [ ] [ ] |
B |[ ] [ ] [ ] [ ] [ ] |
C |[ ] [ ] [ ] [ ] [ ] |
D |[ ] [ ] [ ] [ ] [ ] |
E |[ ] [ ] [ ] [X] [X] |
   +-------------------+
         [ TELA ]

[ MENU DA SALA ]
1. Reservar Assento
2. Voltar ao Menu Principal
Escolha: 1

Fileira (A-E): e
Coluna  (1-5): 4
[ERRO] Este lugar ja esta ocupado!
```

8.3 Discussão dos resultados

O sistema se apresentou muito eficiente e cumpre os principais requisitos que o grupo estabeleceu no início do projeto, a reserva pode ser feita sem problemas, as sessões podem ser cadastradas e são completamente individuais, os assentos de uma sessão não

interferem nos assentos de outra sessão e é possível navegar entre sessões através do uso de menus de forma muito eficiente, como um sistema real.

As limitações observadas são principalmente as limitações da quantidade de sessões, por mais que 3 seja um número baixo de sessões aumentar esse número não trará benefícios reais para esse simulador, e o grupo não conseguiu desenvolver um sistema dinâmico para controlar a reserva de assentos.

Como pontos de melhorias observados a falta de uma funcionalidade para remoção de sessões se mostrou um problema para realização de testes posteriores com o sistema, já que era necessário reiniciar o sistema sempre que queríamos cadastrar uma outra sessão. O mesmo vale para a reserva de assentos, não era possível desmarcar um assento já selecionado, o que para o mundo real não faz muito sentido, deveria ser possível realizar a liberação de um assento a qualquer momento, seria uma funcionalidade real e muito interessante, mas o grupo não conseguiu implementar a função de “pop” para devolver o valor para 0 ou limpar uma sessão inteira, vale manter essas ideias como pontos de melhoria que iriam destacar muito o sistema.

---

9. Conclusão (meia página a 1 página)

Desenvolver o projeto CINEAOC foi uma ótima forma de entender como o MIPS funciona na prática em um projeto um pouco maior do que os judges desenvolvidos em aula. Foi uma tarefa bastante desafiadora entender como um computador processa os comandos diretamente, sem a facilidade e abstração que a gente tem em linguagens modernas, que com certeza facilitam muito o desenvolvimento e entendimento, até mesmo a manutenção do código, visto a dificuldade de leitura do código enfrentada pelo grupo em diversos momentos

Uma das maiores lições foi perceber que, na memória do computador, não existe uma matriz 3D, já que a memória é apenas linear. Tivemos que achar uma forma de contornar nosso entendimento e a lidar com a linearidade da memória, fazendo diversas contas para organizar o salvamento de dados, de forma bem manual, já que a linguagem não permitia que fizéssemos isso de forma mais fácil.

O uso da inteligência artificial também foi um ponto chave, funcionando como uma parceira de estudo que ajudou a destravar dúvidas e entender melhor a lógica quando travamos em algum erro.

Para o futuro, a ideia não é só aumentar o tamanho do cinema, mas deixar o sistema mais esperto e interativo. Queremos trazer recursos para melhorar o visual do mapa, mostrando até onde ficam as saídas de emergência, para deixar a simulação mais completa, bem como maior interatividade com a remoção de salas e liberação de assentos, para que o sistema possa lidar com diferentes situações que poderiam acontecer em ambientes reais.

---

10. Referências

Stringhini, Denise. Slides de aula disponíveis para acesso no classroom. Acesso em: 14 dez. 2025.

PATTERSON, David A.; HENNESSY, John L. ***Organização e Projeto de Computadores****: A Interface Hardware/Software. 5. ed. Rio de Janeiro: Elsevier, 2014.*

VOLLMAR, Ken; SANDERSON, Pete. ***MARS: MIPS Assembler and Runtime Simulator****. Versão 4.5. Missouri State University. Disponível em: https://computerscience.missouristate.edu/mars-mips-simulator.htm Acesso em: 14 dez. 2025.*

IMAGINATION TECHNOLOGIES. ***MIPS32 Architecture for Programmers Volume II****: The MIPS32 Instruction Set. Revision 6.06. [S.l.]: MIPS Technologies, Inc., 2016.*

GOOGLE. Gemini. Disponível em: https://gemini.google.com. Acesso em: 14 dez. 2025.
