# Guerreiro vs Dragão: Projeto de Jogo em Assembly MIPS

Bruno Alves - 147938
Reberth Kelvin Santos de Siqueira - 141589

21 de dezembro de 2025

**Resumo**

Este relatório detalha a implementação de "Guerreiro vs Dragão", um jogo de estratégia em turnos desenvolvido como trabalho final para a disciplina de Arquitetura e Organização de Computadores, ministrada pelo Prof. Dr. Fabio Augusto Menocci Cappa no segundo semestre de 2025 na Universidade Federal de São Paulo (UNIFESP). O projeto demonstra conceitos avançados de programação em assembly MIPS, incluindo arquitetura modular, renderização gráfica e lógica de jogo complexa. Uma característica única deste jogo é a mecânica de "Dívida de Juros Compostos", que serve como uma condição de vitória alternativa ao combate tradicional baseado em HP.

# Sumário

# 1 Introdução

"Guerreiro vs Dragão"é um jogo de batalha gráfico em turnos onde o jogador controla um guerreiro lutando contra um dragão. O projeto foi projetado para demonstrar as capacidades da linguagem Assembly MIPS em lidar com lógica, aritmética e E/S mapeada em memória para gráficos.

O objetivo principal é derrotar o dragão reduzindo seus Pontos de Vida (HP) a zero, ou alternativamente, acumular um "Contador de Dívida"de 10.000 através de uma mecânica de juros compostos, efetivamente dominando o inimigo com estratégia econômica.

# 2 Arquitetura do Sistema

O projeto segue uma arquitetura modular para garantir a manutenibilidade e organização do código. A base de código é dividida em vários módulos funcionais:

- **main.asm**: O ponto de entrada da aplicação. Gerencia o loop principal do jogo, verifica as condições de fim de jogo (Vitória/Derrota) e gerencia a lógica de turnos de alto nível.
- **data.asm**: Serve como o repositório central para todas as variáveis de estado do jogo (HP, dívida, contadores de turno), constantes (cores, endereços de memória) e strings de texto (mensagens de UI, perguntas do quiz).
- **macros.asm**: Define macros globais reutilizáveis em todo o projeto, como `draw_rectangle` que encapsula a chamada à função de desenho de retângulos, simplificando o código de renderização.
- **battle.asm**: Contém a lógica central de combate. Implementa as funções para os ataques do jogador, comportamento da IA do dragão, cálculos de dano e os algoritmos de juros compostos.
- **quiz.asm**: Implementa um sub-sistema educacional. Gerencia a habilidade "Conhecimento Arcano", lidando com a seleção de perguntas, validação de respostas e aplicação de recompensas especiais por respostas corretas.
- **rendering.asm**: O motor gráfico modular. Lida com escritas diretas no display mapeado em memória (0x10040000) para renderizar o céu, chão, sprites (Guerreiro e Dragão), barras de HP e barras de stamina. Inclui funções separadas para renderização de fundo, personagens e interface.
- **sprites.asm**: Armazena os dados gráficos dos personagens do jogo. Cada sprite é definido com um cabeçalho contendo largura e altura, seguido pelos valores de cor RGBA de cada pixel. Inclui onze sprites: `sprite_player` (guerreiro em pé), `sprite_player_defeated` (guerreiro derrotado), `warrior_shield` (guerreiro com escudo), `warrior_spear` (guerreiro lançando lança), `spear` (projétil lança), `sprite_dragon` (dragão padrão), `sprite_dragon_defeated` (dragão derrotado), `sprite_dragon_inferno` (dragão cuspindo fogo), `sprite_dragon_preparing_inferno` (dragão preparando Inferno), `sprite_dragon_defense` (dragão em postura defensiva) e `fireball` (projétil bola de fogo). A cor 0x00000000 é tratada como transparente pelo motor de renderização.

## 2.1 Geração de Sprites

Para facilitar a criação de sprites, foi desenvolvido o script Python `sprites/converter_sprites.py` que converte imagens PNG em formato compatível com MIPS Assembly. O conversor redimensiona as imagens para a largura desejada, extrai os valores RGB de cada pixel e gera automaticamente o código Assembly no formato `.word` adequado para inclusão no arquivo `sprites.asm`.

# 3 Personagens

O jogo apresenta dois personagens principais em um confronto épico:

## 3.1 O Guerreiro (Jogador)

O protagonista é um guerreiro medieval controlado pelo jogador. Ele possui:

- **HP Inicial**: 100 pontos de vida
- **Stamina Inicial**: 100 pontos (regenera 15 por turno)
- **Posição no Display**: Lado esquerdo da tela (X=50, Y=185)
- **Sprites**: Múltiplas poses incluindo postura normal, com escudo, lançando lança e derrotado
- **Recursos**: 2 Estus Flasks para recuperação de HP
- **Habilidades**: 7 ações disponíveis (Pular Turno, Escudo, Rede, Flanco, Lança, Conhecimento Arcano e Estus Flask)

## 3.2 O Dragão (Inimigo)

O antagonista é um poderoso dragão controlado pela IA do jogo. Ele possui:

- **HP Inicial**: 1000 pontos de vida (compensando sua baixa taxa de acerto)
- **Stamina Inicial**: 100 pontos (regenera 15 por turno)
- **Posição no Display**: Lado direito da tela (X=180, Y=185), podendo voar para Y=140
- **Sprites**: Múltiplas poses incluindo postura normal, voando, cuspindo fogo, preparando Inferno, em defesa e derrotado
- **Comportamento**: Seleção aleatória entre 4 ataques (25% cada), com validação de stamina
- **Ataques**: Sopro de Fogo (20 stamina), Pisar (30 stamina), Voar (25 stamina) e Inferno (50 stamina)
- **Fallback**: Quando sem stamina suficiente, entra em postura de defesa

# 4 Mecânicas de Jogo

## 4.1 Apresentação de Informações

O jogo opera em um sistema de turnos alternados entre o jogador e o dragão. Para acompanhar a batalha completamente, é necessário observar duas interfaces simultaneamente:

**Console (Run I/O):**

- Exibe o menu de ações disponíveis para o jogador
- Mostra mensagens de ataque, dano causado e efeitos especiais
- Apresenta o status da batalha (HP do jogador, HP do dragão, contador de dívida)
- Recebe a entrada do jogador (números 1-6 para selecionar ações)
- Exibe as perguntas do quiz e suas opções de resposta

**Bitmap Display:**

- Renderiza os sprites do guerreiro e do dragão
- Mostra o cenário (céu azul e chão verde)
- Exibe as barras de HP de ambos os personagens
- Indica visualmente o turno atual através de um cursor amarelo
- Mostra o dragão em posição elevada quando está voando
- Exibe sprites de derrota quando um personagem é derrotado

Esta combinação de saída textual e gráfica proporciona uma experiência completa, onde o console fornece informações detalhadas sobre as mecânicas do jogo enquanto o display visual oferece feedback imediato sobre o estado da batalha.

## 4.2 Sistema de Combate

O combate é baseado em turnos. O jogador tem acesso a sete ações distintas, cada uma com custo de stamina específico:

0. **Pular Turno (Skip)**: O jogador não realiza nenhuma ação, útil para regenerar stamina. Custo: 0 stamina.
1. **Escudo (Shield)**: Ativa um escudo que absorve 50 HP de dano. Enquanto o escudo está ativo, ações ofensivas (2-5) são bloqueadas. Use novamente para cancelar. Custo: 0 stamina.
2. **Rede (Net)**: Um movimento tático que lança uma rede para capturar o dragão, fazendo-o perder um turno, e aplica juros compostos à dívida. Não causa dano direto. Taxa de acerto: 60% (40% se dragão estiver voando). Custo: 25 stamina.
3. **Flanco (Flank)**: Um ataque de alto risco e alta recompensa com 40% de chance de acerto crítico e dano aumentado (15-24 HP, 30 crítico). Taxa de acerto: 80%. Custo: 40 stamina.

4. **Lança (Spear)**: Ataque animado com projétil que causa menos dano (5-9 HP, 15 crítico), mas aumenta a evasão do jogador para o próximo turno. Taxa de acerto: 80%. Custo: 20 stamina.

5. **Conhecimento Arcano (Arcane Knowledge)**: Uma habilidade especial que aciona uma pergunta sobre arquitetura de computadores. Resposta correta aplica 5x juros compostos; resposta errada causa -5 HP de penalidade (absorvido pelo escudo se ativo). Custo: 50 stamina.

6. **Estus Flask**: Um item consumível inspirado na icônica mecânica de recuperação de HP do jogo *Dark Souls*. O jogador começa com 2 frascos disponíveis. Ao usar, regenera 25 HP imediatamente e mais 25 HP por turno durante 2 turnos adicionais. Custo: 0 stamina.

Tabela 1: Habilidades do Jogador

| **Ação** | **Stamina** | **Acerto** | **Acerto (Voando)** | **Crítico** | **Dano/Efeito** |
|---|---|---|---|---|---|
| Pular | 0 | – | – | – | Nenhum |
| Escudo | 0 | – | – | – | Absorve 50 HP |
| Rede | 25 | 60% | 40% | – | Captura dragão |
| Flanco | 40 | 80% | 50% | 40% | 15-24 (30 crit) |
| Lança | 20 | 80% | 50% | 15% | 5-9 (15 crit) + Evasão |
| C. Arcano | 50 | – | – | – | 5x juros / -5 HP |
| Estus | 0 | – | – | – | +25 HP/turno (3x) |

O Dragão atua como o oponente de IA. A cada turno, seleciona aleatoriamente entre quatro ataques (25% cada), mas deve ter stamina suficiente para executá-los. Se não houver stamina para nenhum ataque, entra em postura de defesa.

- **Sopro de Fogo** (20 stamina): Ataque animado com projétil de bola de fogo. Dano de 25-40 HP (60 crítico). Taxa de acerto: 35% (50% se jogador tiver evasão) com 15% de chance de crítico. Reduz a dívida em 5% ao acertar.

- **Pisar (Stomp)** (30 stamina): Atordoa o jogador, fazendo-o perder um turno. Reduz a dívida em 5%. Sem dano direto.

- **Voar (Fly)** (25 stamina): Aumenta a evasão do dragão. O jogador precisa de 50+ para acertar no próximo turno (reduz acerto de 80% para 50%). O dragão é renderizado em posição elevada (Y=140).

- **Inferno** (50 stamina): Ataque em duas fases. No primeiro turno, o dragão "acumula fogo" (exibe sprite de preparação). No turno seguinte, desencadeia o ataque devastador com 80% de acerto e 45-65 HP de dano. Reduz a dívida em 5% ao acertar.

- **Postura de Defesa** (0 stamina): Fallback automático quando o dragão não tem stamina para nenhum ataque. Exibe sprite de defesa e aguarda regeneração de stamina.

Tabela 2: Habilidades do Dragão

| Ataque | Stamina | Acerto | Acerto (Evasivo) | Crítico | Dano/Efeito |
|---|---|---|---|---|---|
| Sopro de Fogo | 20 | 35% | 50% | 15% | 25-40 (60 crit) |
| Pisar | 30 | 100% | 100% | – | Atordoa jogador |
| Voar | 25 | 100% | 100% | – | +Evasão dragão |
| Inferno | 50 | 80% | 80% | – | 45-65 (2 turnos) |
| Defesa | 0 | – | – | – | Aguarda stamina |

## 4.3 Sistema de Stamina

Uma mecânica de gerenciamento de recursos que adiciona profundidade estratégica ao combate:

- **Stamina Máxima**: Tanto o jogador quanto o dragão possuem 100 pontos de stamina.
- **Regeneração**: A cada início de turno, 15 pontos de stamina são regenerados automaticamente (não excede 100).
- **Custo de Habilidades**: Cada habilidade possui um custo específico de stamina. Se o personagem não tiver stamina suficiente, a ação falha e exibe uma mensagem de erro.
- **Custos do Jogador**: Pular/Escudo/Estus (0), Lança (20), Espada (25), Flanco (40), Conhecimento Arcano (50).
- **Custos do Dragão**: Sopro de Fogo (20), Voar (25), Pisar (30), Inferno (50).
- **Barra Visual**: Barras de stamina azuis são exibidas abaixo das barras de HP no display gráfico.
- **Fallback do Dragão**: Quando o dragão não possui stamina para nenhum ataque, automaticamente entra em postura de defesa.

## 4.4 Sistema de Dívida de Juros Compostos

Uma mecânica única envolvendo um "Contador de Dívida".

- **Crescimento**: Cada vez que o jogador acerta um golpe, o contador de dívida cresce 10% mais um valor base de 100.
- **Redução**: Quando o dragão atinge o jogador, a dívida é reduzida em 5%, simulando um revés.
- **Vitória**: Se o contador de dívida atingir 10.000, o jogador vence imediatamente via "Vitória por Juros Compostos".
- **Inspiração**: Esta mecânica foi inspirada nas habilidades do personagem Knuckle Bine, do anime *Hunter x Hunter*, onde o acúmulo de "juros" de aura leva à derrota do oponente.

### 4.5 Conhecimento Arcano (Sistema Educacional)

O jogo integra conteúdo educacional diretamente na jogabilidade. A ação "Conhecimento Arcano" apresenta perguntas aleatórias sobre Arquitetura de Computadores (ex: sobre ULA, RAM, Barramentos).

- **Resposta Correta**: Aplica a fórmula de juros compostos 5 vezes instantaneamente, fornecendo um grande impulso para a condição de vitória por dívida.
- **Resposta Errada**: Penaliza o jogador com perda de HP.

### 4.6 Condições de Vitória e Derrota

O jogo apresenta múltiplas formas de conclusão da batalha:

**Vitória do Jogador:**

- **Vitória por HP**: Reduzir o HP do dragão a zero ou menos através de ataques diretos.
- **Vitória por Juros Compostos**: Acumular o contador de dívida até atingir 10.000 pontos. Esta é a vitória estratégica que recompensa jogadores que conseguem manter pressão constante enquanto evitam dano.

**Derrota do Jogador:**

- **Derrota por HP**: O jogador perde quando seu HP chega a zero ou menos. Neste momento, o guerreiro é exibido em sua sprite de derrota (deitado no chão) e a barra de HP muda para vermelho.

## 5 Implementação Técnica

### 5.1 Motor Gráfico

O jogo roda em um display de 256x256 pixels com profundidade de cor de 32 bits. O módulo `rendering.asm` usa endereçamento de memória eficiente para desenhar os pixels.

- **Cálculo de Endereço**: $Base + (Y \times 256 + X) \times 4$. A multiplicação por 256 é otimizada usando um deslocamento lógico à esquerda (`sll`) de 8 bits.
- **Sprites**: Sprites são armazenados com cabeçalhos de largura e altura, e o loop de renderização ignora a cor de transparência (0x00000000) para sobrepor os personagens no fundo.

### 5.2 Geração de Números Aleatórios

O jogo utiliza extensivamente a chamada de sistema (syscall) 42 para gerar números aleatórios para determinar:

- Sucesso do ataque (cálculos de Acerto/Erro).
- Acertos críticos.
- Escolhas da IA do Dragão.
- Seleção de perguntas do Conhecimento Arcano.

### 5.3 Ambiente de Teste

O projeto foi desenvolvido e validado utilizando o simulador MARS (MIPS Assembler and Runtime Simulator). Para a saída gráfica, foi utilizada a ferramenta *Bitmap Display* incluída no simulador, com as seguintes configurações específicas para garantir a visualização correta:

- **Unit Width in Pixels**: 1
- **Unit Height in Pixels**: 1
- **Display Width in Pixels**: 256
- **Display Height in Pixels**: 256
- **Base address for display**: 0x10040000 (heap)

## 6 Desafios e Decisões de Projeto

Durante o desenvolvimento do projeto, foram identificados desafios técnicos significativos, especialmente relacionados à renderização gráfica. Um dos principais obstáculos foi o desenho do *bitmap*, onde inicialmente tentou-se utilizar um endereço de memória estático para a manipulação dos pixels. No entanto, para o correto funcionamento com a ferramenta de display gráfico do simulador, deveríamos ter utilizado o endereço de memória da *heap* (dinâmica). Essa divergência causou dificuldades iniciais na exibição correta das sprites e cores na tela, exigindo uma refatoração do código de renderização para apontar para o endereço base correto (0x10040000).

Além disso, devido à complexidade inerente ao desenvolvimento em baixo nível com Assembly, foi tomada a decisão de priorizar a profundidade e robustez das mecânicas de jogo — como o sistema de combate, as perguntas do quiz e o cálculo de juros compostos — em detrimento de uma apresentação visual mais elaborada. O foco principal foi garantir que a lógica do jogo funcionasse perfeitamente, mantendo os gráficos funcionais, porém simples, para assegurar a entrega de um sistema estável e livre de bugs críticos.

## 7 Conclusão

O projeto "Guerreiro vs Dragão"cria com sucesso um jogo RPG envolvente usando linguagem Assembly de baixo nível. Ele demonstra que lógica complexa, design de software modular e interfaces gráficas podem ser efetivamente implementados mesmo sem abstrações de alto nível, fornecendo insights profundos sobre arquitetura de computadores e operações em nível de máquina.