Universidade de Brasília

Departamento de Ciência da Computação - DCC

Matéria: Introdução aos Sistemas Computacionais - 2025.1

Professor: Marcus Vinicius Lamar

Alunos: Kauã Otaviano Teixeira, Thierry Luan Tenório De Jesus e Jennifer Carvalho Alves

Brasília, DF - 2025

Bomberman em Assembly

RESUMO:

Este presente artigo aborda sobre o processo de desenvolvimento em Assembly RISC-V do Bomberman, o famoso jogo criado pela Hudson Soft em 1983 que se baseia em fugir de um labirinto destruindo paredes e inimigos com bombas. Durante a leitura, serão abordadas técnicas de desenvolvimento em Assembly, metodologias utilizadas, peculiaridades e dificuldades encontradas na implementação no jogo em uma linguagem tão de baixo nível.

1. INTRODUÇÃO

Bomberman é um jogo de ação e estratégia originalmente lançado em 1983 pela Hudson Soft, e tornou-se um clássico dos videogames com sua jogabilidade simples e viciante. Nesta adaptação desenvolvida em Assembly RISC-V, o jogador assume o papel do icônico personagem Bomberman, que deve navegar por um labirinto repleto de blocos em busca de destruí-los com sua bomba e conseguir pontos suficientes para vencer.

2. METODOLOGIA

2.1 Tecnologias

Para o desenvolvimento do jogo, foram utilizados diversos recursos disponibilizados e ensinados pelo professor e ex-alunos de ISC para integrar corretamente vídeo, teclado e áudio.

A princípio, o desenvolvimento se iniciou com a IDE RARS o que permitiu entender melhor as peculiaridades do Assembly RISC-V e as ferramentas de Bitmap Display (vídeo), MIDI (áudio) e Keyboard MMIO (teclado). No entanto, viu-se que essa IDE era muito lenta e que era necessário trocar para o FPGRARS, um executor de arquivos assembly criado por um ex-aluno de CIC, para agilizar o desenvolvimento. Em decorrência, o desenvolvimento prosseguiu inteiramente pela IDE Visual Studio Code, FPGARS e Github.

Além disso, os sprites de todo o jogo foram pegos da internet ou feitos no Paint.NET e logo após convertidos para .data com o software bmp2oac3.exe disponibilizado pelo professor.

2.2 Técnica de desenvolvimento

O projeto foi planejado desde o princípio com o Framework Scrum (uma forma de gerenciar projetos com foco em ciclos de desenvolvimento e entrega de valor ao cliente) em mente. Isso contribuiu para diluir a carga de trabalho sobre os desenvolvedores durante todo o prazo de entrega e mesmo assim conseguir um produto e notas razoáveis.

Com o desenvolvimento em ciclos com objetivos claros e viáveis, também se viu necessário organizar os códigos dos desenvolvedores. A organização se baseou em seguir um padrão modular, genérico e comentado das funções de forma que o código se tornasse menos repetitivo, mais reutilizável e legível. Esse desenvolvimento só foi possível de ser posto em prática por causa do Git e Github (software e site de versionamento de projetos, respectivamente) que permitiu o incremento contínuo, assíncrono e observável de novas funcionalidades, sempre garantindo estabilidade em cada pull request.

2.2.1 Estrutura do código

A estrutura do código foi dividida em 3 partes: Setup, GameLoop e GameOver. No Setup são definidas configurações base para o funcionamento correto do GameLoop, como os elementos da matriz de colisão, tempos iniciais para a música e bomba, posição do bomberman e etc. O GameLoop é o jogo em si onde as funções de ações, música e renderização são chamadas em loop. E o GameOver é um loop que fica tocando música e espera o jogador teclar enter para terminar a execução. Essa organização se deu possível, pois as outras funções foram colocadas em arquivos com escopos similares e que são chamados no arquivo main.s com o comando “.include”, sendo extremamente necessário a configuração correta dos stack pointers no início e fim de cada função para não dar erro.

2.3 Mapas

Os mapas disponíveis para jogar têm praticamente o mesmo layout, mas com sprites diferentes. Quando o mapa é escolhido, todos os layouts associados a ele mudam também. O mapa a ser jogado é escolhido de forma aleatória no Setup.

Sobre as telas finais, há duas importantes para o contexto do jogo: game-over e vitória. A tela de game-over aparece quando a vida do bomberman chega ao final e persiste tocando uma música até o jogador apertar outra tecla. A tela de vitória funciona exatamente da mesma forma, mas para a condição que a pontuação é maior que 50 pontos.

Figura 1: Layout dos mapas 1 e 2, respectivamente

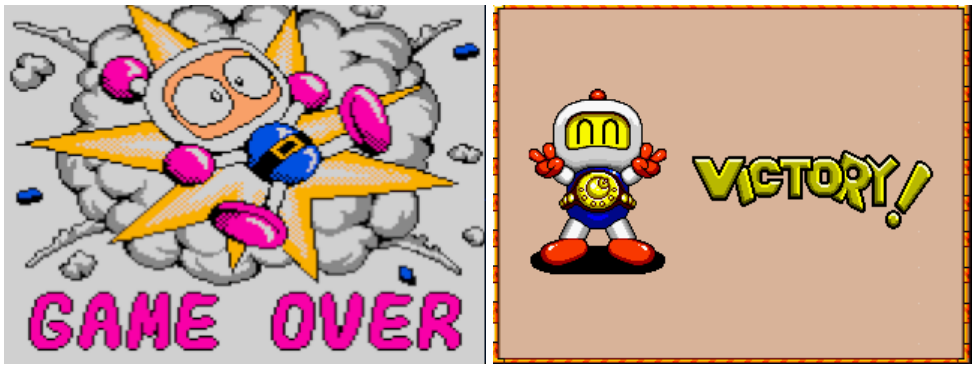

Figura 2: Telas de game over e vitória

2.4 Blocos

Os blocos, junto com o movimento do bomberman, foram umas das primeiras coisas a serem implementadas no jogo. Eles foram fundamentais para o desenvolvimento do resto dos componentes, pois as funções de setar elementos na matriz de colisão e renderizá-los foram feitas para funcionarem de forma genérica. Na função de renderizar elementos, foi configurado que a cor #00FF00 (verde) seria considerada como transparente, então todos os elementos que precisam desse atributo seguem a mesma cor de fundo.

Antes do jogo iniciar, é chamada uma função que coloca os blocos indestrutíveis (hard blocks) de maneira ordenada e os blocos destrutíveis (soft blocks) de maneira aleatória na

matriz de colisão. Os blocos são então renderizados a partir da matriz de colisão, cada um com seu sprite diferente de acordo com o tema do mapa. Com esse controle, é possível destruir um bloco facilmente, bastando alterar o valor na matriz de 2 (soft block) para 0 (espaço vazio)

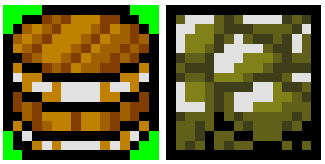

Figura 3: soft e hard block do mapa 1

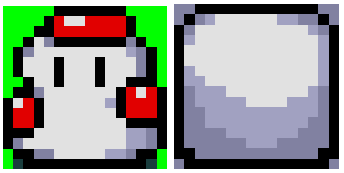

Figura 4: soft e hard block do mapa 2

2.5 Protagonista

O Bomberman é o protagonista do jogo e quem o jogador controla. Ele pode colocar bombas no chão com a tecla de espaço, perde vida pras bombas e recebe pontos quando coloca bombas e destroi blocos. O seu sprite foi criado no tamanho 16x16 para poder caber corretamente no layout quadriculado do mapa e, assim como todos os outros elementos, tem o fundo com a cor #00FF00 (Verde) que é considerada como transparente no momento de renderizar os elementos gráficos na tela. Isso possibilita utilizar o mesmo arquivo .data em diferentes mapas e fundos. Quando ele perde suas 3 vidas, o jogo termina com uma tela de game-over. Quando o bomberman atinge uma pontuação de 50 pontos,

Sobre a movimentação do bomberman, ela ocorre completamente baseada no que está na matriz de colisão, mas ele mesmo não faz parte dela, pois causaria problemas de elementos em cima de outros. O bomberman só pode andar em células com valor 0 (espaço vazio) e qualquer outro valor causa colisão, mas isso é algo que pode ser facilmente modificado.

Figura 5: Sprites do bomberman

Figura 6: Sprite da vida

2.6 Bomba

A bomba é um dos elementos mais interessantes quando se trata de Bomberman, pois ela é o núcleo do jogo e o que o faz ser memorável. Por causa disso, um foco especial na consistência da bomba foi necessário, pois é preciso especificar o que ela afeta e o que afeta ela, como, por exemplo, destruir blocos, dar dano e colisão. Infelizmente a frequência das bombas foi implementada de forma estática tanto em tamanho quanto em animação, mas não impediu o seu propósito. O seu sprite segue o mesmo padrão do bomberman de 16x16, ela é colocada embaixo do jogador no mapa quando a tecla de espaço é pressionada e explode após 3 segundos com a explosão persistindo por meio segundo.

Figura 7: Sprites da bomba e explosão

2.7 Display reativo

Há dois elementos no display na parte superior do mapa: a vida do bomberman e a pontuação. A vida do bomberman foi implementada printando a quantidade de corações da esquerda para a direita a partir do número de vidas restantes do bomberman. O display de pontuação foi feito de maneira similar, mas sendo necessário primeiro pegar o dígito a ser impresso da variável de pontuação e escolher o .data correto para imprimir.

Figura 8: Display da vida e pontuação no jogo

2.8 Música e efeitos sonoros

Foram implementadas três músicas e um efeito sonoro no jogo. As músicas tocam em loop quando chamadas dentro de seus contextos: jogo normal, game-over e vitória. Para que elas possam ser executadas corretamente com o game-loop, se viu necessário o comando system ecall 33 para que a nota fosse tocada de forma assíncrona e de um controlador do tempo para saber quando tocar a próxima nota. O efeito sonoro implementado, foi um som grave mais baixo para a explosão da bomba.

3. RESULTADOS OBTIDOS

Todas as funcionalidades que nos pusemos a implementar estão funcionando corretamente e sem bugs aparentes: o Bomberman consegue se movimentar com uma animação

simples, tem colisão, ele coloca a bomba, a bomba dá dano e destroi os blocos, há uma condição simples de vitória e o display de vida e pontuação são atualizados em tempo real. Infelizmente não conseguimos implementar corretamente os power-ups e inimigos, mas isso esteve dentro do planejado para o nosso objetivo, então podemos considerar esse projeto um sucesso!

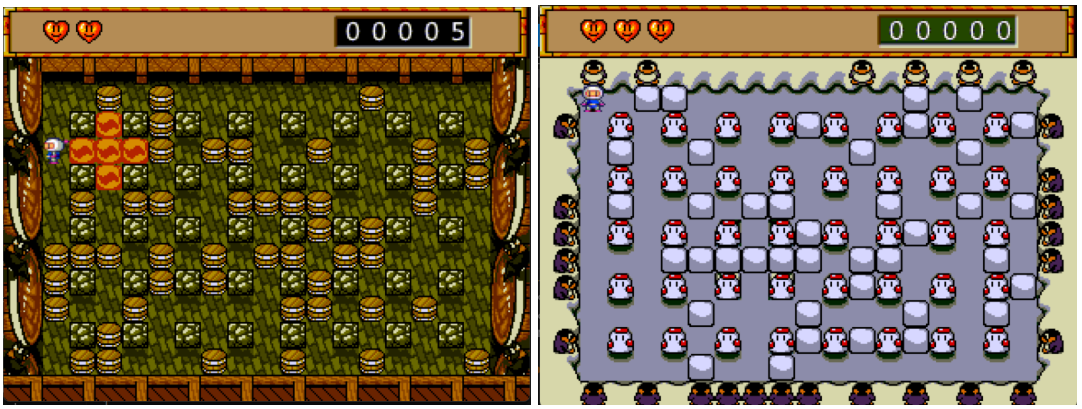

Figura 9: Mapa 1 e Mapa 2 em jogo

4. CONCLUSÃO

Em resumo, esse foi um projeto extremamente difícil e oneroso por se tratar de uma linguagem nova e de tão baixo nível para a gente. Apesar disso tudo, conseguimos entregar algo que nos agradou e trouxe resultados. Certamente essa é uma experiência valiosa que nos acompanhará como recordação durante todo o trajeto daqui em diante.

5. REFERÊNCIAS BIBLIOGRÁFICAS

- Vídeo de ex-aluno explicando sobre renderização: https://youtu.be/2BBPNgLP6_s?si=T1dXTjPRUglwKEbE
- Repositório de jogos de outros semestres: https://github.com/victorlisboa/LAMAR
- Jogo original: https://online.oldgames.sk/play/nes/bomberman/6762