# Projeto Aplicativo
# -Donkey Kong-
# Grupo 1

Eduardo Ferreira de Assis[1*] Alexandre Souza Costa Oliveira[2†] Thiaggo Ferreira Bispo de Souza[3‡]
Emanoel Johannes Cardim Lázaro[4§] Gabriel Pinheiro dos Santos[5¶]

[1]Universidade de Brasília, Departamento de Ciência da Computação, Brasil

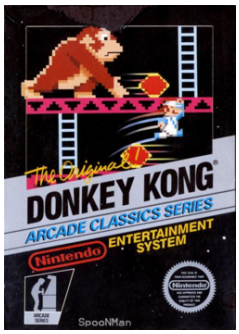


Figura 1: Capa do jogo NES Donkey Kong 1981.

**RESUMO**

A fim de aplicar os conhecimentos adquiridos ao longo do curso de Organização e Arquitetura de Computadores, foi pedida a realização do projeto Donkey Kong, o qual tem como objetivo recriar, por meio da linguagem Assembly, o jogo NES Donkey Kong de 1981, a adição de uma fase de autoria própria e sua execução em um dos processadores (uniciclo, multiciclo ou pipeline), implementados ao longo do semestre.

**Keywords:** Donkey Kong, Processador, Assembly, RISC-V, RARS, Quartus, Nintendo, Aseprite, Miyamoto.

## 1 INTRODUÇÃO

Donkey Kong é uma série de jogos criado por Shigeru Miyamoto que gira em torno do personagem Donkey Kong (o gorila), sendo o primeiro jogo da série criado em 1981 - sendo esse o recriado no projeto. Nesse jogo, Donkey Kong sequestra uma mulher chamada Pauline se tornando inimigo do Jumpman (rebatizado em 1984 para Mário). O objetivo do jogador é se desviar pulando de obstáculos que vêm na direção do personagem até chegar no Donkey Kong e derrotá-lo.

Donkey Kong foi o primeiro exemplo de jogo do tipo "plataforma". O jogo é tido como o primeiro jogo de plataforma da história a utilizar o pulo como habilidade, introduzindo a necessidade de pular entre brechas, obstáculos e inimigos próximos.

*e-mail: eduardoffassis@gmail.com
†e-mail: alexandresc45@gmail.com
‡e-mail: thiaggo98@gmail.com
§e-mail: emanoeljohannes@hotmail.com
¶e-mail: gabriel-santos-bs@hotmail.com


Figura 2: Shigeru Miyamoto, criador do Donkey Kong.

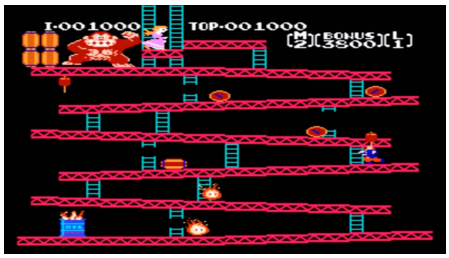


Figura 3: Modelo de inspiração para o desenvolvimento do jogo.

O jogo se inicia com a pontuação de 0, a qual sobe à medida em que o jogador pula barris e pega itens e acaba se o mesmo perder todas as vidas que o jogo dispõe, no caso, quatro. Os casos em que o jogador perde vida são aqueles em que o personagem entra em contato com os inimigos, sendo eles barris, fogos, tortas, molas e, na fase especial, com as tartarugas.

## 2 Fundamentação Teórica e Técnica

Ao longo do desenvolvimento do projeto, o mesmo foi dividido em diversas áreas:

### 2.1 Menu

Ao executar o programa, o menu do jogo é apresentado, criado por meio do código em Assembly.

O menu é composto por três opções:

Jogar: inicia o jogo;

Créditos: apresenta informações sobre os componentes do grupo.

Sair: finaliza a execução do jogo.


Figura 4: Tela inicial do jogo.

### 2.2 Mapa

Para a produção do mapa pelo qual o jogador interagirá, usamos um código em Assembly que aproveita os padrões do mapa para desenhá-lo; essa repetição de elementos no mapa para desenhá-lo permitiu uma grande otimização do uso da memória de dados.

Baseado no modelo do jogo Donkey Kong, é possível notar que o jogador é levado a seguir em apenas uma direção (pra cima) e que existem elementos que influenciam a jogabilidade, como escadas normais ou quebradas, plataformas e esteiras.

Existem, ao todo, 5 cenários no jogo, sendo 2 deles cenários autênticos, e foram implementados obstáculos e objetivos, sendo alguns deles estáticos e outros não.

Abaixo serão apresentados os cenários do mapa.

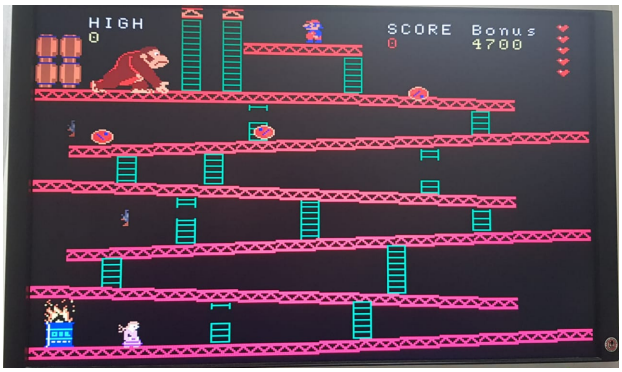


Figura 5: Primeiro cenário do jogo.

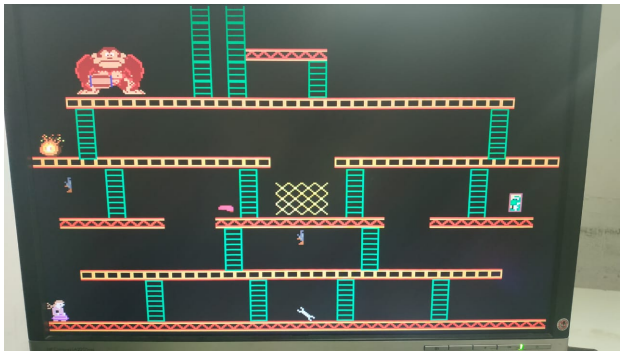

Figura 6: Segundo cenário do jogo.

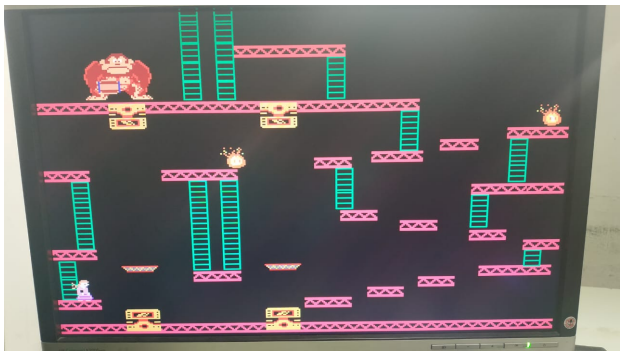

Figura 7: Terceiro cenário do jogo.

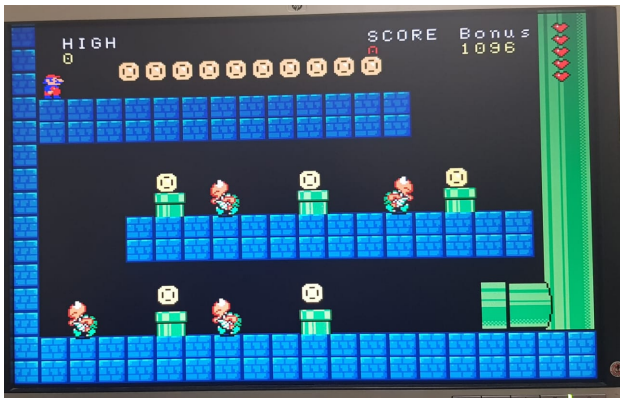


Figura 8: Primeira fase especial autêntica.

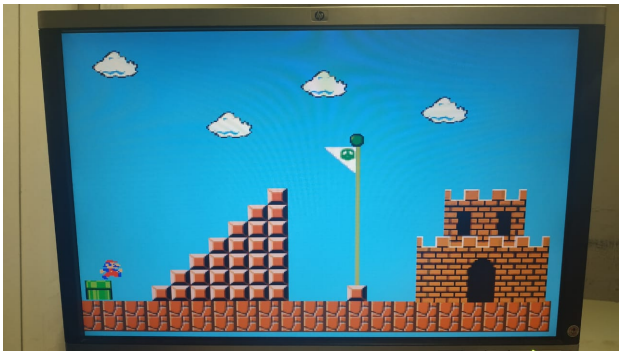

Figura 9: Segunda fase especial autêntica e última fase do jogo.

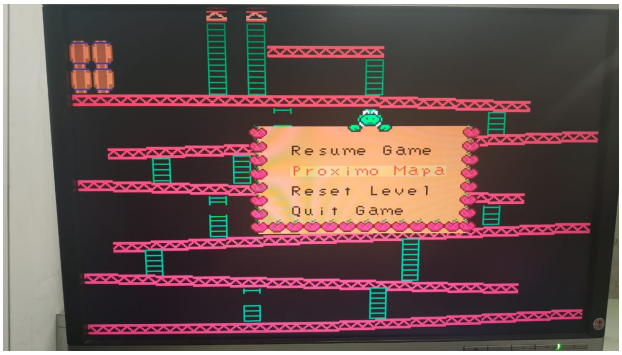


Figura 10: Menu de pausa dentro do jogo.

### 2.3 Modelos e Animação

Para a realização dos modelos de alguns personagens e itens, foram utilizados sprites originais disponibilizados pelo site `https://bit.ly/2XBambm`, o que possibilitou com que o jogo, nas suas fases originais, fosse bem fiel ao que foi feito pela Nintendo.

Abaixo temos os modelos utilizados, porém nem todos os sprites foram empregados no jogo.


Figura 11: Parte 1 dos sprites utilizados no jogo.


Figura 12: Parte 2 dos sprites utilizados no jogo.

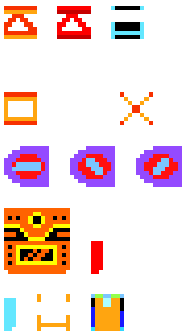

Figura 13: Parte 3 dos sprites utilizados no jogo.

Para a edição de alguns sprites usados no jogo original e as criações do menu, da princesa, do elevador, do Bowser e seus efeitos e de itens não pertencentes ao jogo original foi usado o programa Aseprite

Figura 14: Sprites feitos/editados no programa Aseprite.

Alguns sprites também foram tirados de *Sprite Charts*, porém de outros jogos.


Figura 15: Sprites pegos de jogos e sites diferentes.

Ao realizar os sprites, foi encontrada dificuldade em relação ao aplicativo Aseprite e a palheta de cores: foi necessário tomar cuidado ao escolher as cores pois a palheta de cores do RISC-V é diferente devido ao número de bits usados nos pixels (RISC-V com 8 bits por pixel e os bmps originais com 24 bits).

### 2.4 Sonorização

Ao longo do desenvolvimento do jogo, foram realizadas releituras de músicas para o formato aceito pela linguagem Assembly. Para a

criação das músicas contamos com a participação especial da Isabelle Alex dos Santos, ex-aluna de OAC, responsável pelos sons do jogo Pitfall! do semestre 2018.2.

Foram utilizadas as músicas de outro jogo da Nintendo: o Super Mario Bros, sendo elas Underworld Theme, Ending Theme, Castle Theme e Death Sound, além de músicas de autoria própria feitas diretamente em Assembly.

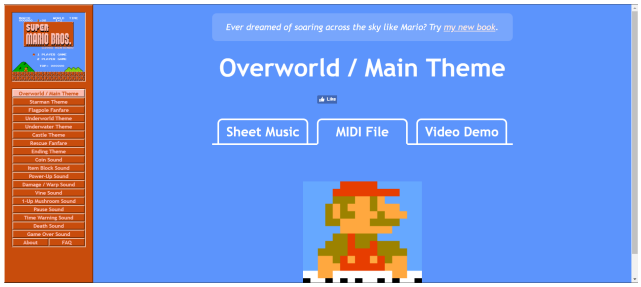


Figura 16: Site usado como base das músicas no jogo.

## 2.5 Movimentação

Para fazer a movimentação foi necessário foi necessário criar uma função que move todos os objetos. O jogo possui gravidade, cada objeto consegue se movimentar em direções diferentes e existem funções de detecção de chão e parede que são responsável pelas colisões com objetos estáticos dentro do jogo, assim como a movimentação do personagem em esteiras.

O Analog Digital Converter (ADC) foi implementado com sucesso nesse projeto, funcionando com sensibilidade no processador Uniciclo implementado durante o semestre e comentado adiante.

Existem também funções de colisão com objetos e inimigos dentro do jogo; funções importantíssimas para a definição do número de vidas e da pontuação.

Figura 17: Analog Digital Converter utilizado para a movimentação no jogo.

## 2.6 Placar e Temporizador

O placar do jogo apresenta quatro indicadores, os quais são: a maior pontuação alcançada (*High Score*), pontuação do jogo atual, o bônus e as vidas, sendo essas representadas por corações.

Todos os valores apresentam comportamento similar ao Donkey Kong original: a pontuação, iniciada como 0, essa pontuação aumenta de acordo com as ações que o jogador realiza (ex: pulando barris, pegando itens, etc.) e caso o jogador atinja a maior pontuação já atingida no jogo, ela fica salva no indicador de High Score; o bonûs decresce de acordo com o tempo e para ser reivindicado pelo jogador ele deve vencer aquela fase o quanto antes .

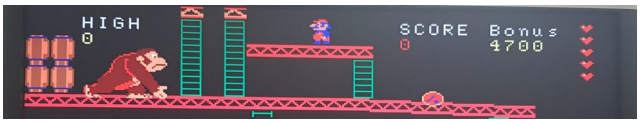


Figura 18: Placar do jogo.

## 2.7 Implementação no Processador

Nesta etapa, entre os processadores uniciclo, multiciclo e pipeline, foi escolhido o primeiro pois estava mais estável que o resto dos processadores. No entanto, só foi possível chegar a trinta frames por segundo no jogo com este processador, o que não é incomum já que não é o processador com a organização mais eficiente.

Durante o semestre, além das funções em inteiro e ponto flutuante que já tínhamos no processador, foram adicionadas as funções que trabalham com instruções CSR, ebreak, ecall, uret e o boa parte do tratamento de exceções.

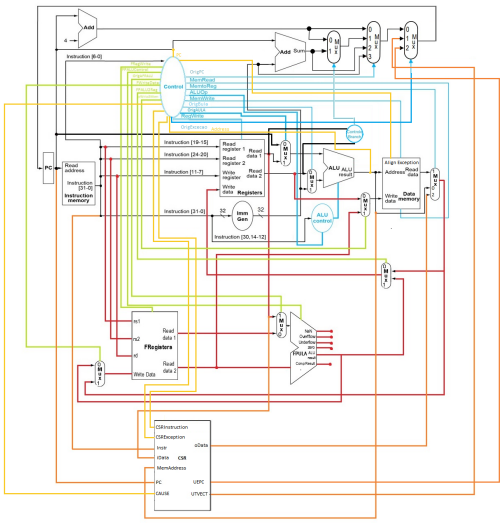

Figura 19: Caminho de dados da CPU Uniciclo utilizada.

O teclado e o display funcionam normalmente, porém há uma diferença em relação ao áudio que sai da placa e o áudio que sai quando o jogo é executado no RARS.

# 3 METODOLOGIA

O método utilizado para o desenvolvimento do trabalho foi passado durante as aulas de Organização e Arquitetura de Computadores. Com o uso do RARS foi possível produzir o código dos efeitos

sonoros e da música, assim como implementar as imagens produzidas a partir do Aseprite ou aproveitadas do jogo original, ou seja, as telas, os objetos animados e as vidas e também implementar a movimentação do personagem; para utilizarmos imagens utilizamos o programa bmp2isc, que converte o um arquivo em .bmp pra .s, utilizamos o Bitmap Display para projetar as imagens na tela e o Keyboard and Display MMIO simulator para permitir a interação do teclado. Após implementar o código foi necessário adaptá-lo à execução da DE1-SoC, adicionando programas que foram implementados durante o semestre de forma que funcionasse no processador Uniciclo que foi desenvolvido por meio do programa Quartus Prime na linguagem de descrição de hardware Verilog.

### 3.1 A Arquitetura RISC-V

Para juntar e organizar cada parte do desenvolvimento do jogo foi utilizado o programa RARS (RISC-V Assembler and Runtime Simulator) e todo o processo foi feito na linguagem Assembly RISC-V. O RARS é responsável por montar e simular a execução da linguagem Assembly RISC-V e foi feito a partir do MARS (MIPS Assembler and Runtime Simulator), um programa que realiza a mesma função só que para a linguagem Assembly MIPS.

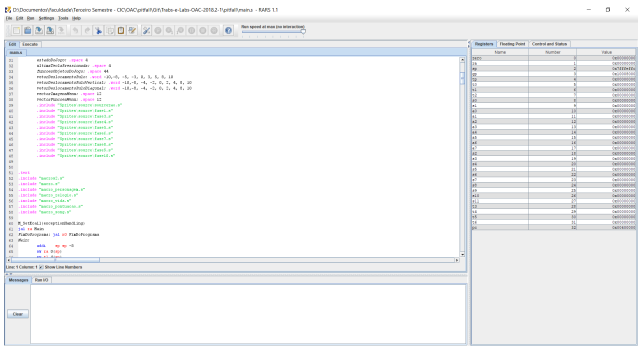

Figura 20: Imagem da interface do Rars.

O RISC-V é um conjunto de instruções (ISA - Instruction Set Architecture) baseado e estabelecido nos princípios RISC (Reduced Instruction Set Computer) e é livre para ser utilizado para qualquer finalidade. Destaca-se por ser projetado com foco para dispositivos computadorizados modernos, como computação em nuvem, aparelhos móveis, sistemas embarcados e internet das coisas (IoT - Interner of Things). O projeto começou em 2010 na Universidade da Califórnia, em Berkeley, mas muitos colaboradores trabalham no projeto de fora da universidade; é voltado para implementações de alto desempenho e baixo consumo de energia, sendo um conjunto limpo e modular, trabalhando com bases de 32, 64 e 128 bits, com várias opções de extensão em ponto flutuante.

Existem produtos disponíveis comercialmente que usam desta tecnologia, como SoC's (System-on-Chip) 64 bits de quatro núcleos e processadores para sistemas embarcados e outros tipos de processadores; e existem ainda produtos em desenvolvimento como CPUs com aplicações em IoT e servidores, processadores de placas de vídeo, SoCs abertos, entre outros.

### 3.2 Quartus Prime

O Quartus Prime foi o programa utilizado na confecção do processador uniciclo selecionado para executar o jogo. É um *software* de programação lógica de dispositivos produzido pela Intel. Antes da Intel adquirir a empresa Altera Corporation (fabricante de dispositivos lógicos programáveis) o *software* se chamava Altera Quartus II. O Quartus possibilita a análise e síntese de linguagens de descrição de hardware, permitindo que o desenvolvedor compile seus *designs* e faça análises temporais, examine diagramas RTL (Register Transfer Level), simule reações a diferentes estímulos e ajuste o dispositivo de acordo com o programador.

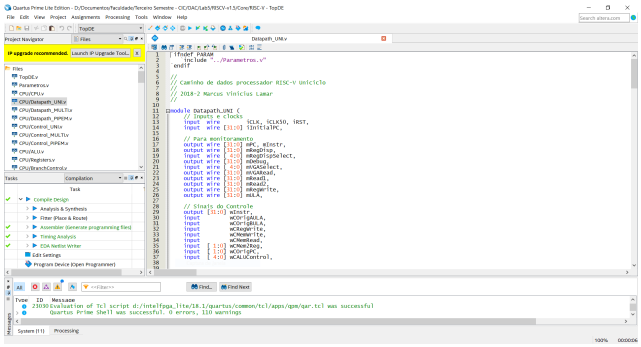

Figura 21: Imagem da interface do Quartus Prime.

## 4 Resultados Obtidos

Após a realização do projeto, atingimos os seguintes objetivos: o personagem anda para cima, para a esquerda e a direita, com animação; existem efeitos sonoros durante pulos, além de música; o personagem pode cair entre os espaços das plataformas, está sujeito a movimentos das esteiras, pode subir e descer escadas pegar itens, usá-los e morrer para inimigos. Também foram implementados *cross-overs* com Super Mario Bros em certas fases do jogo e itens adicionais. O jogo funciona com o teclado e o ADC, temos sons para diversos acontecimentos do jogo e duas fases especiais.

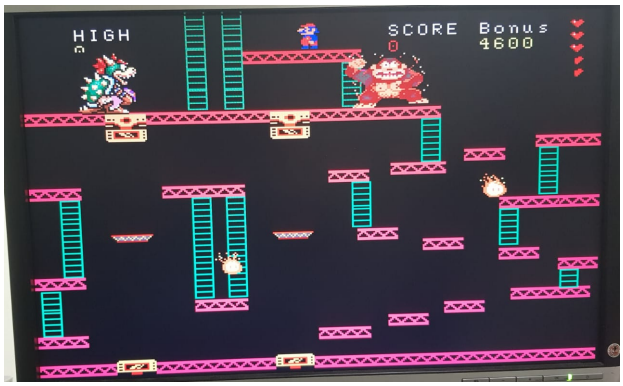


Figura 22: Fase do jogo em que há cross-over com um personagem de Super Mario Bros.


Figura 23: Jogador subindo a escada.

## 5 Conclusão e Trabalhos Futuros

Por fim, pode-se concluir que o projeto Donkey Kong proporcionou um melhor entendimento do funcionamento de diversos elementos aprendidos ao longo do semestre de Organização e Arquitetura de Computadores, como a linguagem Assembly Risc-V, processadores e programas utilizados ao longo do desenvolvimento do jogo. Também, no decorrer do curso, foi possível implementar funções de ponto flutuante para os processadores uniciclo, multiciclo e pipeline, desenvolvendo um novo caminho de dados e projetando um controle para cada modelo de processador.

Como trabalhos futuros temos a implementação de mais fases especiais, com *cross-overs* com mais jogos da Nintendo, incluindo novos personagens, sons, e itens, assim como mais níveis de dificuldade dentro do jogo.

## Referências

www.spriters-resource.com
https://www.npr.org/sections/alltechconsidered/2015/06/19/415456813/the-legendary-mr-miyamoto-father-of-mario-and-donkey-kong
pt.wikipedia.org/wiki/RISC-V
github.com/TheThirdOne/rars
en.wikipedia.org/wiki/Intel_Quartus_Prime
https://bojoga.com.br/retroplay/analises-de-jogos/arcade-pinball/donkey-kong-nintendo-1981/
https://pt.wikipedia.org/wiki/Altera
https://pt.wikipedia.org/wiki/Donkey_Kong
http://pixelartmaker.com/art/4ca4553e5b770fc