ARM Invaders: Relatório do Projeto

Antonio Fassini – NUSP 12551032
Pedro Fassini – NUSP 1256571

August 16, 2025

Contents

Chapter 1

Introdução

O projeto **ARM Invaders** é um mini-jogo/simulador interativo, inspirado no clássico *Space Invaders*, mas com foco didático na manipulação de registradores ARM e flags do processador.

Ele foi desenvolvido na linguagem C, com uso de trechos de *inline Assembly*, e é executado diretamente no terminal da Raspberry Pi ou em ambientes Linux/WSL. A proposta é transformar conceitos abstratos de Organização e Arquitetura de Computadores em uma experiência visual e interativa, permitindo ao aluno observar em tempo real o funcionamento de registradores, flags e instruções básicas da ISA ARM.

Chapter 2

Objetivos e Motivação

O ARM Invaders busca:

- Tornar o estudo de registradores ARM mais visual e acessível, facilitando a fixação de conceitos fundamentais.
- Demonstrar, de forma prática, como instruções simples modificam o estado interno de um processador.
- Oferecer um recurso multiplataforma, rodando apenas no terminal, sem dependências gráficas pesadas.
- Aproximar conteúdos da disciplina (Arquitetura e Redes) do funcionamento real de hardware e sistemas.

Chapter 3

Relação com a Disciplina

Embora o foco da disciplina esteja em protocolos e comunicação em rede, todo sistema computacional depende de operações internas da CPU. O simulador permite:

- Entender como dados são processados internamente na CPU.
- Visualizar a influência de instruções de baixo nível sobre desempenho e eficiência.
- Conectar hardware, sistema operacional e aplicações em rede.
- Explorar o ciclo *fetch-decode-execute* em um contexto lúdico.

Chapter 4

Decisões de Design

As principais escolhas foram:

- **Simplicidade intencional:** operações básicas (ADD, SUB, MUL, MOV) garantem curva de aprendizado suave.
- **Terminal puro:** compatibilidade com qualquer ambiente Linux/WSL/Raspberry Pi.
- **Código comentado:** cada função foi documentada para incentivar exploração e modificações.
- **Extensibilidade:** comandos adicionais podem ser facilmente implementados no interpretador.

Chapter 5

Descrição da Lógica e Funcionalidades

A implementação segue esta lógica:

1. **Inicialização:** cria registradores e flags com valores iniciais.
2. **HUD:** mostra registradores como "naves" com barras de vida, números e símbolos.
3. **Entrada:** usuário escolhe registrador e operação (ADD, SUB, MUL, MOV).
4. **Execução:** a operação é realizada em C com *inline Assembly* exibido no terminal.
5. **Flags:** N, Z, C, V são atualizadas após cada instrução.
6. **Explosão:** quando o valor chega a zero, a nave "explode" em ASCII-art.
7. **Loop principal:** mantém o jogo ativo enquanto houver registradores com vida.

Funcionalidades Implementadas

- Visualização contínua dos registradores e flags.
- Execução das operações aritméticas: ADD, SUB, MUL, MOV.
- Uso de cores ANSI para realce.
- HUD com animações simples em ASCII.
- Geração de valores aleatórios (`rand`).
- Salvamento e carregamento de estado (save/load).
- Execução automatizada de comandos (scripts).

Chapter 6

Conexão com Conteúdos Teóricos

- **Registradores:** simulam armazenamento temporário de dados.
- **Flags:** N/Z/C/V exibem resultados das operações.
- **Memória:** gerenciamento de estados e persistência via arquivos de texto.
- **ISA ARM:** instruções básicas demonstradas com trechos reais de Assembly.
- **Ciclo de Execução:** o loop do jogo reflete o ciclo *fetch-decode-execute*.

Chapter 7

Estrutura do Repositório

```
arm_invaders/
 src/        # Código-fonte
 doc/        # Relatórios e documentação
 media/      # Imagens, vídeos e capturas
 scripts/    # Exemplos de automação
 LICENSE     # Licença do projeto
 Makefile    # Script de compilação
 README.md   # Documentação resumida
```

Chapter 8

Como Compilar e Executar

Requisitos

- `gcc` (instalar via: `sudo apt install build-essential`)

Compilação

```
make
```

Execução

```
make run
```

Chapter 9

Resultados e Discussão

O ARM Invaders permitiu observar:

- Impacto imediato de instruções em registradores e flags.
- Relação entre baixo nível (Assembly) e lógica de alto nível (C).
- Desafios de sincronização e atualização em tempo real.
- Potencial de expansão para novos comandos e interfaces.

Chapter 10

Conclusão

O projeto cumpriu sua função didática: aproximar a teoria da prática em Arquitetura ARM. Além de reforçar conceitos como registradores, flags e instruções, mostrou que é possível criar experiências motivadoras e visuais mesmo em ambientes simples como o terminal.

Chapter 11

Licença

Este projeto é distribuído sob a licença MIT. Consulte o arquivo LICENSE para mais detalhes.