

**UNIVERSIDADE FEDERAL DA BAHIA**

**DEPARTAMENTO DE CIÊNCIA DE COMPUTAÇÃO**

**LUCAS MENEZES PEREIRA**

**MATHEUS OLIVER DE CARVALHO CERQUEIRA**

**RELATÓRIO DE ANÁLISE DE JOGO CONSTRUÍDO EM ASSEMBLY ARQUITETURA X86**

Salvador

2017

1. **APRESENTAÇÃO**

*Assembly* é uma notação legível por humanos para o código de máquina, utilizada para programar códigos entendidos por dispositivos computacionais. O código de máquina torna-se legível pela substituição dos valores em bruto por símbolos chamados mnemónicos. Cada mnemônico é associado a uma operação do computador.

A Linguagem *Assembly* é muito versátil, sendo constituinte de software básico e muito variada, devido às várias arquiteturas disponíveis, criadas por empresas como fabricantes de Microcontroladores, Microprocessadores, e outros Chips de Circuitos Integrados para garantir a exclusividade de seu Hardware através do Software.

Nesse cenário de diversas formas de *Assembly*, A arquitetura x86, da Intel® é uma das mais utilizadas, sendo possivelmente utilizada para criar, também, softwares de Jogos.

Será desenvolvida, portanto, neste relatório, uma análise do código *Assembly* de uma versão do clássico da década de 80, *Space Invaders*™, recriado por fãs, sem fins lucrativos.

O jogo, originalmente lançado para *Arcades*, consiste de uma nave, cujos canhões atiram sobre uma horda de alienígenas, que se movimentam horizontalmente, descendo uma linha da tela quando chegam à lateral do monitor. A versão escolhida não utiliza imagens *bitmap* similares, mas caracteres, reproduzindo a mesma mecânica (funcionamento do jogo definido pelo código escrito por desenvolvedores).

1. **MATERIAIS UTILIZADOS**
   1. **Sistema Operacional:**
   * Windows 10 64 bits, com processador em x64;
   * Linux Lubuntu 32 bits, 1,90GB de memória RAM.
   1. **Hardware:**
   * 4GB de memória RAM (Utilizável: 3,89GB);
   * Intel Core i3-3010M CPU @2.40GHz.
   1. **Software:**

* Oracle VM VirtualBox;
* Terminal (Linux);
* Jogo *Space Invaders*™(Coleção de arquivos);
* Nasm
* QEmu.

1. **OBJETIVOS**
   1. **Geral**

A partir do jogo escolhido, analisar como o jogo foi construído, e como a linguagem *Assembly* é utilizada para garantir o pleno funcionamento do jogo, associando cada ação ou recurso específico a seus blocos de código correspondentes.

* 1. **Específicos**
* Montar o jogo;
* Verificar a execução do programa (garantir o funcionamento e validação do código), destacando as ações mais relevantes;
* Verificar, para cada ação, quais trechos de código contribuem para que ela aconteça, e como essa contribuição acontece.

1. **METODOLOGIA**

Aqui são mencionados e discutido quais itens e como serão analisados neste relatório. As medidas de pré-execução serão esclarecidas na próxima seção. Dada a situação do jogo executado, os itens seguintes serão considerados na análise.

* 1. **Mecânica do Jogo**

A partir da apresentação do jogo, em que foi explicitado o conceito do jogo, podem ser destacados os seguintes funcionamentos:

* A nave pode se movimentar para a esquerda ou para a direita;
* A nave atira de acordo com a entrada do usuário;
* Os alienígenas se locomovem em uma única direção por vez, em conjunto;
* Os alienígenas atiram no jogador;
* Cada alienígena que toma um tiro, é abatido;
* Caso o jogador seja atingido, o jogo termina, e ele perde;
* Caso o jogador atinja todos os alienígenas, o jogo termina e ele ganha.

Considerando os itens apresentados, o teste de funcionamento deve comprovar que todos, sem exceção, são válidos. Após a confirmação, terá início a análise dos programas, ressaltando as mecânicas explicitadas.

* 1. **Análise do código**

A análise dos programas se dará individualmente, para funcionamento explicitado acima. Para facilitar o acompanhamento da análise, os códigos foram ‘quebrados’ em blocos, trazendo dos arquivos listados abaixo, trechos responsáveis por cada ação. É importante lembrar que não será feita a análise de todos os códigos dos programas, mas apenas das funções destacadas como mais relevantes, que preservam a identidade e execução do jogo.

São os arquivos:

* Space-invaders.asm;
* Arena.asm;
* Bullets.asm;
* Display.asm;
* Game.asm;
* Invaders.asm;
* Keyboard.asm;
* Player.asm.
* Bootloader.asm

Sabe-se que estes não são os únicos arquivos inclusos no Jogo, também havendo um LICENSE, um README.md, um .gitignore e dois S*hell Script*, create.sh e start.sh. Como a intenção no relatório é perceber como a Linguagem *Assembly* implica no funcionamento do jogo, apenas estes, .asm, serão abertos e, consequentemente, analisados.

1. **OPERACIONAL**
   1. **Medidas Pré-Execução**

Antes de executar o código do jogo baixado, é preciso atender também a certas condições. Para garantir a integridade e segurança do sistema operacional nativo, foi criada uma partição de disco por máquina virtual, através do *Software* da *Oracle*, *Virtualbox*, para instalação da distribuição *Linux,* *Lubuntu*. A escolha do SO tem razão pela interface gráfica amigável a desenvolvedores e por ser leve, bastando pouco menos de 2GB de RAM alocadas da máquina hospedeira para a virtual. Assim, a execução da máquina virtual fica mais otimizada, evitando lentidão de processos, ou até travamentos.

É importante ressaltar que o *Windows* não possuía desvantagens significativas para o cumprimento dos objetivos, sendo a alternativa considerada apenas para isolar a execução dos programas ao *Hardware* da máquina hospedeira.

* 1. **Conhecendo a Pasta projeto**

Ao extrair a pasta principal, raiz, tem-se dois arquivos *Assembly*, “space\_invaders.asm” (o arquivo principal, que chama o código contido nos outros .asm) “bootloader.asm”, que será detalhado mais à frente, um documento de texto, “.gitgnore”, um “LICENSE”, um “README.md” e duas pastas, “src” (contendo os códigos modulados do jogo, a serem usados por “space\_invaders.asm”) e “scripts”, que contém dois *Shell Scripts*, “create.sh” e “start.sh”. Estes dois últimos arquivos serão utilizados apenas na montagem do jogo dentro do sistema operacional.

* 1. **Montagem do Jogo**

Após inicializar o *Lubuntu*, e instalar os *Softwares* QEmu e Nasm (não será detalhado o processo de instalação neste relatório), deve-se abrir o *Terminal* (nativo do *Linux Lubuntu*) e digitar, em ordem, as seguintes linhas de comando:

*mkdir -p build*

*nasm bootloader.asm -f bin -o build/bootloader.bin*

*nasm space-invaders.asm -f bin -o build/space-invaders.bin*

*dd if=build/bootloader.bin of=build/image.img bs=512 count=1*

*dd if=build/space-invaders.bin of=build/image.img bs=512 count=4 seek=1*

O código acima, contido no arquivo "create.sh" na pasta "scripts", foi rodado para criar o arquivo de imagem emulável (.img) do jogo. A segunda e a terceira linhas acima utilizam o montador *Nasm* para criar os arquivos binários de execução de *Boot* a partir dos arquivos .asm.

Dos arquivos .bin recém-criados, foi extraído o arquivo de imagem "space.img".

* 1. **Emulação**

Após a extração do arquivo "space.img" foi preciso executá-lo usando o emulador QEmu, de funcionamento similar a uma máquina virtual comum, mas com a vantagem de simular o boot de um computador e assim permitir a execução dos programas assembly feitos para tal plataforma. Após a execução do comando "qemu-system-x86\_64 -fda space.img" a simulação do boot foi feita e a tela do jogo foi aberta, disponibilizando-o para ser testado.

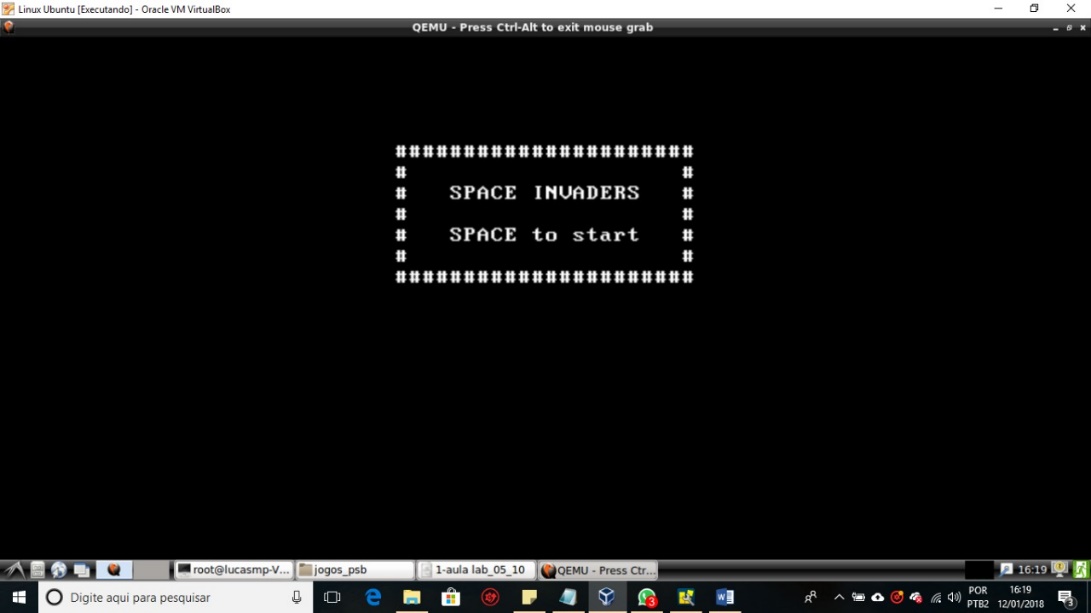


Imagem 1 – Tela inicial do Jogo em execução, rodando no QEmu, VurtualBox.

* 1. **Teste de Funcionamento**

O teste deve conferir se os itens citados em 4.1 estão de acordo. Assim deve-se avaliar, ao executar o jogo, a partir das entradas do usuário, ou avaliação do *Software*, se a ação ocorre corretamente. Em caso confirmativo, os códigos são analisados.

As representações da nave e dos aliens são dados pelos caracteres “M” e “T”, respectivamente.

* + 1. **Movimento da Nave**

Segundo a interface gráfica do jogo, as teclas para movimentar a nave são: “A” para mover para a esquerda e “D” para a direita. Logo ao iniciar o jogo, as teclas foram testadas e corresponderam ao informado.

* + 1. **Tiro da Nave**

Para atirar com a nave, deve-se apertar a tecla espaço. Também funcionou corretamente.

* + 1. **Movimento dos Alienígenas**

Os alienígenas estão organizados em fileiras alternados como no xadrez (diferindo um pouco da versão clássica, mas tal fato é meramente estético, não comprometendo a execução do programa). Foi verificado o movimento, controlado pelo sistema, de uma direção por vez, descendo na tela ao chegar ao limite lateral e refletir o movimento horizontalmente (direita, desce, esquerda, desce...).

* + 1. **Tiro dos Alienígenas**

Os tiros dos alienígenas também são controlados pelo sistema, ocorrendo por fileira, 1 tiro para cada alienígena, a cada unidade regular de tempo. A diferença em relação ao clássico se dá pela multiplicidade de tiros, que no original era de um por nave aleatória, sendo, novamente, apenas uma mudança devido ao gosto do desenvolvedor, sendo a versão presente, a mecânica que deve ser analisada.

* + 1. **Morte dos Alienígenas**

Os aliens morrem caso acertado o tiro da nave, como era previsto anteriormente.

* + 1. **Morte do Jogador**

O jogador morre se for atingido por um tiro de alien, validando o caso de perda. Para reiniciar o jogo, basta apertar a tecla “R”.

* + 1. **Vitória do Jogador**

O jogador vence se todos os aliens forem abatidos, como era esperado.

* 1. **Análise dos Códigos**

Aqui, será verificado como cada uma das funções anteriores é definida pelos códigos dos programas. Apresentando um panorama geral, o programa “space\_invaders.asm” serve como motor e suporte do jogo, sendo o arquivo que chama a função dos outros, contidos na pasta “src”.

O “bootloader.asm” torna o jogo capaz de ser carregado no boot de um computador. A execução no boot é uma habilidade especial de programas em Assembly, pois, sendo linguagem de baixo nível e especificando diretamente os registradores com os quais trabalhará, não é preciso que o Sistema Operacional, que, por si, é mais robusto que a BIOS (esteja rodando para que a aplicação seja suportada. Esse desempenho, portanto, é vital para que o jogo funcione, mas, como se relaciona apenas ao sistema, não será considerado na análise.

Além destes dois, tem-se os explicitamente nomeados da pasta “src”. São eles:

* “Arena.asm”: O arquivo com menor código. Desempenha a função de criar o cenário, consistente de um fundo preto, delimitado de “#” por todos os lados, com largura definida no próprio “space\_invaders.asm”;
* “Bullets.asm”: Estabelece a propriedade de tiro, tanto da nave, quanto dos alienígenas;
* “Display.asm”: Atualiza a tela de acordo com as movimentações gráficas (atualiza posição da nave, tiros e aliens) e verifica a exibição de caracteres;
* “Game.asm”: Carrega os elementos como os aliens, nave, verifica as condições de vitória ou derrota do jogador (consequentemente, se o jogo acabou ou não) ;
* “Invaders.asm”: Cria o objeto alien invasor e suas propriedades e mecânicas;
* “Keyboard.asm”: Cria interface para as entradas do jogo, captando do teclado do jogador e enviando para o sistema;
* “Player.asm”: Cria o objeto nave e suas propriedades e mecânicas.

Essas caracterizações dos arquivos foram feitas após leitura dos códigos internos a eles, sendo um resumo sucinto, mas que ajuda a compreender as funções a partir deles.

Portanto, a seguir, serão descritas as mecânicas, antes evidenciadas, em como a codificação *Assembly* garante o funcionamento de cada.

* + 1. **Movimento da Nave**
    2. **Tiro da Nave**
    3. **Movimento dos Alienígenas**
    4. **Tiro dos Alienígenas**
    5. **Morte dos Alienígenas**
    6. **Morte do Jogador**
    7. **Vitória do Jogador**

1. **RESULTADOS E DISCUSSÕES**

O Jogo funcionou perfeitamente, o que possibilitou a análise e serviu como base para o entendimento dos códigos (A ideia de como funciona o jogo visualmente possibilita prever, com conhecimento anteriormente adquirido, como foi desenvolvido o game).

1. **CONCLUSÃO**
2. **REFERÊNCIAS**

GitHub. Disponível em: <https://github.com/flxbe/asm-space-invaders>. Acesso em 08 de Janeiro de 2018.

Nasm. Disponível em:< http://www.nasm.us>. Acesso em 10 de Janeiro de 2018

Ubuntu. Disponível em: <https://www.ubuntu.com/download/desktop>. Acesso em 13 de Janeiro de 2018

University of Virginia Computer Science. Disponível em: <http://www.cs.virginia.edu/~evans/cs216/guides/x86.html>. Acesso em 12 de Janeiro de 2018.­­

VirtualBox. Disponível em <https://www.virtualbox.org>. Acesso em 13 de Janeiro de 2018.

Wikipedia. Disponível em: <https://pt.wikipedia.org/wiki/Assembly#Arquitetura>. Acesso em 12 de Janeiro de 2018.

Wikipedia. Disponível em: <https://pt.wikipedia.org/wiki/Boot>. Acesso em 09 de Janeiro de 2018

Wikipedia. Disponível em: <https://pt.wikipedia.org/wiki/Space\_Invaders>. Acesso em 12 de Janeiro de 2018.