# Relatório do projeto R-Type

Durante a execução do nosso projeto, tivemos várias preocupações quanto à eficiência do código, de forma a tornar a experiência de jogo mais eficaz e confortável para o utilizador. Assim, esforçámos-nos por evitar a repetição de código, criando várias rotinas auxiliares que executavam as tarefas que várias rotinas principais exigiam. Para tornar a chamada das rotinas mais eficientes, decidimos passar os argumentos pela pilha, preservando o conteúdo dos registos.

Rotinas de Escrita

Para a visualização do jogo no ecrã temos as rotinas básicas de escrita de carateres e de strings, como a **EscCar**, **EscString**, **EscreveNave**, **EscreveMsgIni**, **EscreveMsgFin**, **EscreveLimite**, etc. Para o deslocamento da nave, dos tiros e dos obstáculos substituímos os carateres da posição antiga por espaços, *‘ ‘*. A rotina que escreve no LCD é a **EscTextoLCD.** No LCD para a escrita da posição da nave é chamada a rotina **LCDCanhaoPos**, que verifica o valor da coluna e da linha da nave a cada deslocação. Cada um destes valor tem de ser primeiro convertido de hexadecimal para decimal primeiro, pela rotina **CnvrtDec** e depois para ASCII.

Botões e suas respetivas Rotinas de Tratamento

Os botões I0, I1, I2 E I3 ativam a mesma flag, que verifica quando ocorre a movimentação da nave. Nas rotinas de tratamento de cada botão é ainda alterado um endereço de memória com o valor a ser acrescentado à palavra de memória que contém a posição do canhão, de forma a que esta seja atualizada, executando de seguida a rotina **Nave** que processa o deslocamento da nave. O botão I4 permite a execução da rotina **CriaTiro,** gerando os tiros e o Temporizador ativa uma flag que permitirá o deslocamento dos tiros e dos obstáculos, de uma forma de 1:2, verificada por um endereço de memória que atua como contador. É no ciclo de **Jogo** que todas estas flags são verificadas para a execução das rotinas.

Com a escrita da mensagem final, alteramos a máscara das interrupções para que todos os botões façam reset do jogo, corremos uma rotina que aguarda que alguma flag seja acionada para voltar à mensagem inicial, **EsperaFim.** No regresso à mensagem inicial, a máscara é de novo atualizada para responder aos comandos de jogo.

Tiros

Os tiros são executados ao premir o botão I4, podendo-se disparar até 5 tiros. Para que isto seja possível, a rotina de criação de tiros só pode ser executada se o número de tiros no mapa for menor a 5; quando um tiro colide com um obstáculo ou com o limite final, a sua posição é apagada e quando um novo tiro tiro é criado, temos uma variável que percorre a tabela, procurando uma posição livre e escreve a nova lá.

Colisões

Para o funcionamento do jogo, temos uma rotina que deteta as colisões entre os tiros com os obstáculos e a nave com os os obstáculos. Para isto verificamos as posições de cada um com os outros elementos, na movimentação de cada um. Ou seja, na movimentação dos tiros verificamos se a posição seguinte, a próxima coluna, está ocupada com um obstáculo e um procedimento análogo entre os asteroides com os tiros. Assim é chamada a mesma rotina **VeriColisoes,** com parâmetros diferentes. Na movimentação da nave chamamos uma rotina de deteção de colisões diferente mas com o mesmo efeito, uma vez que a nave tem quatro caracteres, implicando que sejam verificadas as colisões com todos estes caracteres, usando a rotina **HaNaveLa.**

Obstáculos

Na criação dos obstáculos, havia várias restrições do enunciado, nomeadamente quanto à distância entre os mesmos e aos momentos de criação de cada tipo de obstáculo. Assim temos duas variáveis que são incrementadas com a geração dos obstáculos e que verificam quando deve ser o número de criação e de que obstáculo. Para sabermos qual obstáculo a ser criado e em que posição, definimos dois vetores com 16 posições cada um, em que um contém a posição e outro o carácter do obstáculo. Assim ao percorrermos um dos vetores à procura de uma posição livre para criação de um novo obstáculo, após uma colisão do obstáculo com o limite esquerdo ou com a sua destruição pelos tiros, também é percorrido o outro vetor, procedendo-se ao registo da informação do obstáculo de forma organizada.

Rotina Pseudo Random

Esta rotina foi a implementação em Assembly da rotina fornecida no enunciado, na rotina **PseudoRanom.** Para gerarmos a raiz da sequência, fomos incrementando uma endereço de memória com o número de ciclos de espera até ao início do jogo, como sugerido no enunciado. Uma vez gerado este número, a rotina gera um novo número aleatório das próximas vezes que for chamada, pela **CriaObstaculo**

## Divergências do enunciado

Na movimentação da nave, decidimos que se esta chocar com os limites de jogo, então os limites atuariam como uma parede, apenas impedindo o movimento da nave. Quanto ao número de tiros, o nosso código permitia definir qualquer número que escolhêssemos, contudo para tornar a dificuldade do jogo num nível mais elevado, definimos este número para 5. Na escrita do LCD, dividimos a escrita da posição em que na primeira linha escrevemos ‘Linhas: XX’ e na segunda linha ‘Colunas: XX’.