

Alunos:

João Almeida- 90119

Rafael Cordeiro-90171

# Introdução

O objectivo deste trabalho é criar um jogo básico de Pacman através do uso da linguagem Assembly, servindo, por isso, para aperfeiçoar as nossas capacidades no desenvolvimento de jogos nesta linguagem, no uso de periféricos e interrupções, algo que ainda desconhecíamos.

Para a melhor compreensão do nosso processo de desenvolvimento, vamos realizar o relatório abrangendo as características gerais do programa mencionadas no ponto 2.

# 2. Trabalho realizado na 1º semana

**Desenrolar-se num recinto de jogo pré-definido na janela de texto**

Para a criação do labirinto usámos uma rotina que lia sucessivamente as strings que constituem o tabuleiro. Para o efeito, utilizámos a rotina print\_labi para “obter” a string e, de seguida a EscString para escrever no ecrã.

**Permitir movimentos do PacMan nas 4 direções (direita, esquerda, cima, baixo) com recurso ao mecanismo de interrupções activados por pressão do teclado**

**->Leitura e Avaliação da jogada**

Para isto, o raciocínio utilizado já é mais complexos, pois tivemos de catalogar os movimentos(por exemplo, ocorria MoveUp se o 4 bit do conteúdo armazenado por Mexe pacman estiver ativo, estes valores variavam dependendo da interrupção).Para a avaliação de jogadas, simplesmente analisávamos se o conteúdo da memória nessa posição estavam ‘-‘ ou ‘|’,caso isto acontecesse a jogada era inválida.

**->Movimentos Pacman**

Para os movimentos do Pacman, por exemplo, se fosse MoveUp, decrementava 0001h,pois como quero andar uma linha para trás, como os primeiros 8 bits representam as linhas e os outros as colunas, basta decrementar.

**Registar em tempo real o nº de vidas e a pontuação na janela de texto**

Nesta parte utilizamos duas funções, uma print\_stats e a outra é O\_PacMan\_foi\_comido\_pelo\_bicho, a primeira serve para imprimir a pontuação na janela de texto através da utilização das funções EscCar e EscString. A segunda consiste em analisar se o pacman foi ou não comido por um monstro , caso tenha sido, decrementa as vidas.

**Registar o tempo de jogo no display de 7 segmentos**

A base consiste em utilizar a função update display que vai atualizando os LCD´s. Contudo, tivemos de converter os números do display para decimal.

Nesta parte dividimos em duas funções uma para separar o número nos vários LCD´s, a outra que se chama conversor para ASCII, onde aqui aplicamos a conversão, incrementando 48, que corresponde à posição do ´0´ no código ASCII.

**Utilizar o temporizador para controlar o tempo de jogo, determinar a velocidade de movimento dos monstros e controlar o tempo de bónus em que o PacMan pode apanhar os monstros**

Nós utilizámos o temporizador para controlar todo o tipo de velocidades dos monstros, bem como no aparecimento de eventos. Nas velocidades, utilizamos registos com valores pré-definidos que iam decrementando à medida que o temporizador atualizava, por exemplo, se quiséssemos que os Monstros tivessem uma velocidade de 1 segundo, como o Clock\_tick está a 000Ah, a velocidade variava ao segundo .No caso do aparecimento dos eventos bastava comparar o tempo atual, que estava guardado em M[N\_to\_Display], com o tempo que deveriam aparecer os eventos.

NOTA: Nós decidimos que o nosso código não tem classificações, mas apenas pontuações que vão sendo acumuladas ao longo do jogo e guardadas no final de cada jogo, assim ninguém perde ou ganha, simplesmente tem mais ou menos pontos.

# Conclusões

Com este trabalho foi possível ter um primeiro contacto com a linguagem Assembly e permitiu-nos expandir os nossos conhecimentos na resolução e tratamento de erros em linguagens diferentes. Para além disso, permitiu-nos trabalhar com o processador P3 e algumas das suas funcionalidades.

Ao longo do desenvolvimento do jogo, fomos enfrentando algumas dificuldades, visto que não estávamos habituados a lidar com a nova linguagem e, por isso, foi necessário algum tempo para a implementação de funções. Para além disso, enfrentámos algumas dificuldades na correção de erros, visto que só possuíamos os breakpoints para detetar a localização de falhas.

No final, conseguimos concluir o trabalho proposto com sucesso, abrangendo todas a funcionalidades pedidas no enunciado.