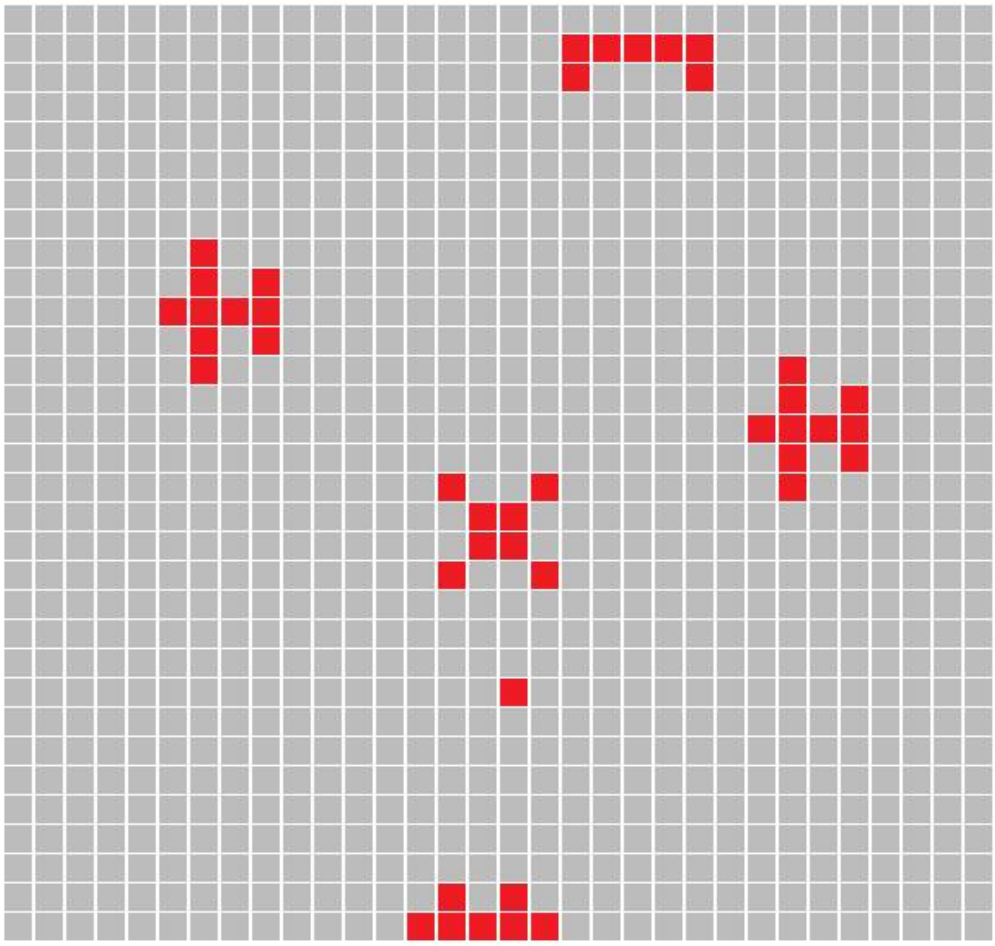
INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

ARQUITECTURA DE COMPUTADORES
LETI/LEE

IST-TAGUSPARK



Grupo 7

Carolina Neves - 82527
João Carlos - 82528
João Bernardo - 82547

O presente relatório está divido nas seguintes secções:

1. Introdução

Este trabalho tem como objectivo principal o desenvolvimento das competências fundamentais da disciplina de Arquitectura de Computadores, nomeadamente a programação em Linguagem A*ssembly*, os periféricos de entrada e saída e as interrupções.

Para tal, este Projeto tem como função concretizar um jogo de simulação de tiros em que o objetivo é atingir os aviões que se movimentam no Ecrã. Estes entram na parte superior direita do ecrã e movimentam-se para a esquerda. Na parte inferior do ecrã encontra-se um canhão que se movimenta em diversas direções (Norte, Sul, Este, Oeste, Nordeste, Noroeste, Sudeste e Sudoeste) e dispara tiros. Se um dos tiros acertar em algum avião, este explode e é adicionado um ponto à cotação do jogo. Contudo, para que o canhão não fique sem balas, deve-se acertar no cartucho que se encontra no limite superior do ecrã.

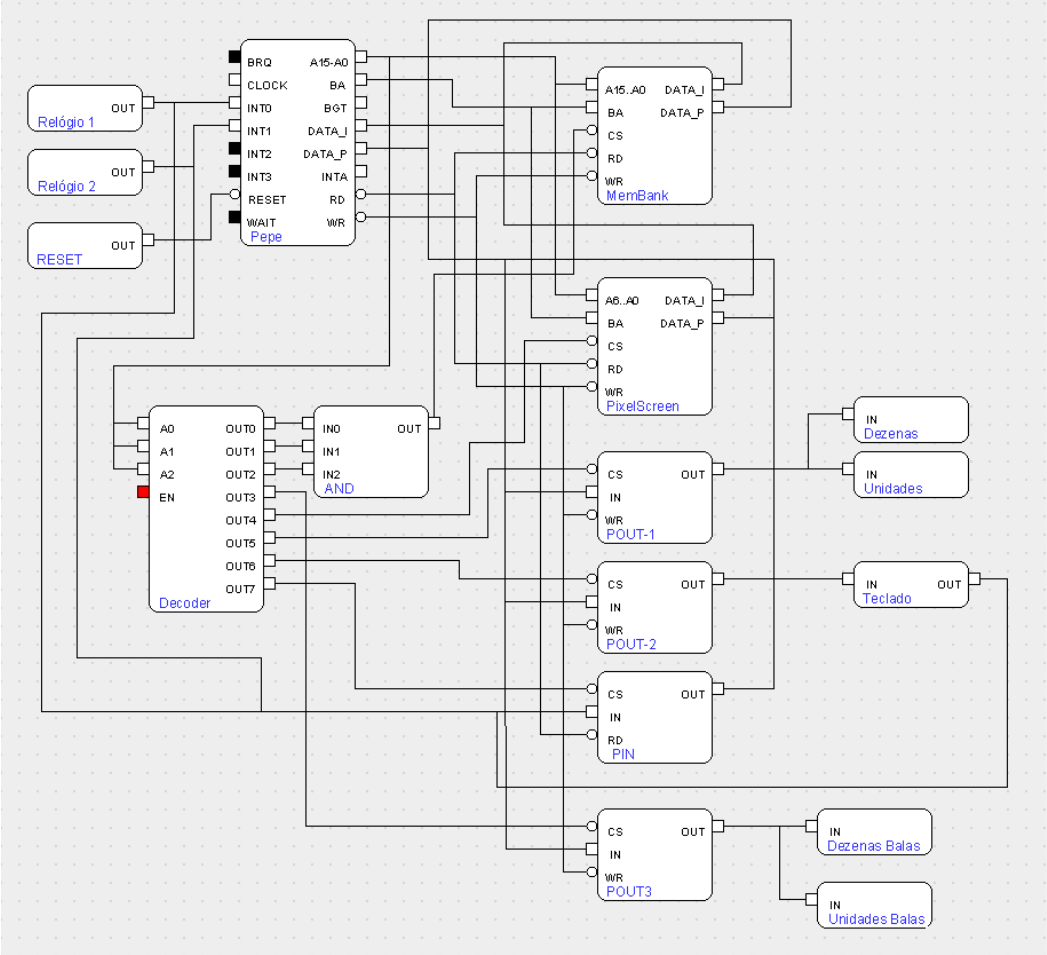
O jogo termina quando o canhão fica sem balas, quando o jogador chega aos 99 pontos (máximo permitido pelos displays) ou quando o utilizador carrega num botão para acabar o jogo.

Restrições e Observações sobre o Projeto

O jogo está quase completo, falta-lhe ter um contador de balas e sempre que uma bala é lançada, a esse contador é subtraído uma unidade. Sempre que uma bala atinge o cartucho, deveria ser adicionado 3 balas ao contador.

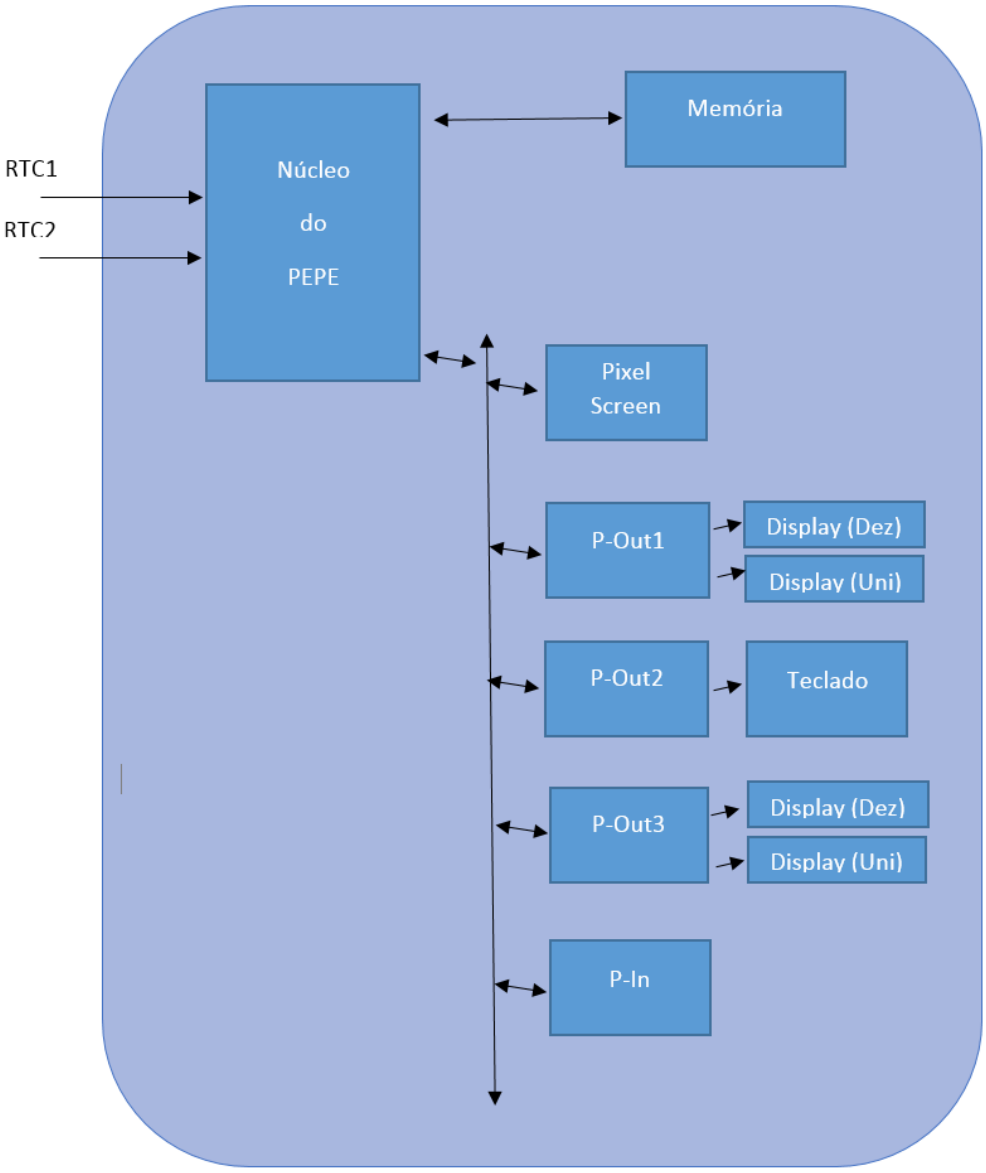
2. Conceção e Implementação

2.1 Estrutura Geral



- Processador PEPE de 16 bits que tem como função correr o jogo
- Memória RAM que guarda as informações que lhe são transmitidas não só pelos periféricos mas também pelo código
- Relógio 1 que está encarregue pelo deslocamento das balas
- Relógio 2 que está encarregue pela progressão dos aviões e deslocamento do cartucho
- Teclado, deteta qual o botão pressionado e de acordo com essa tecla desempenha determinada função

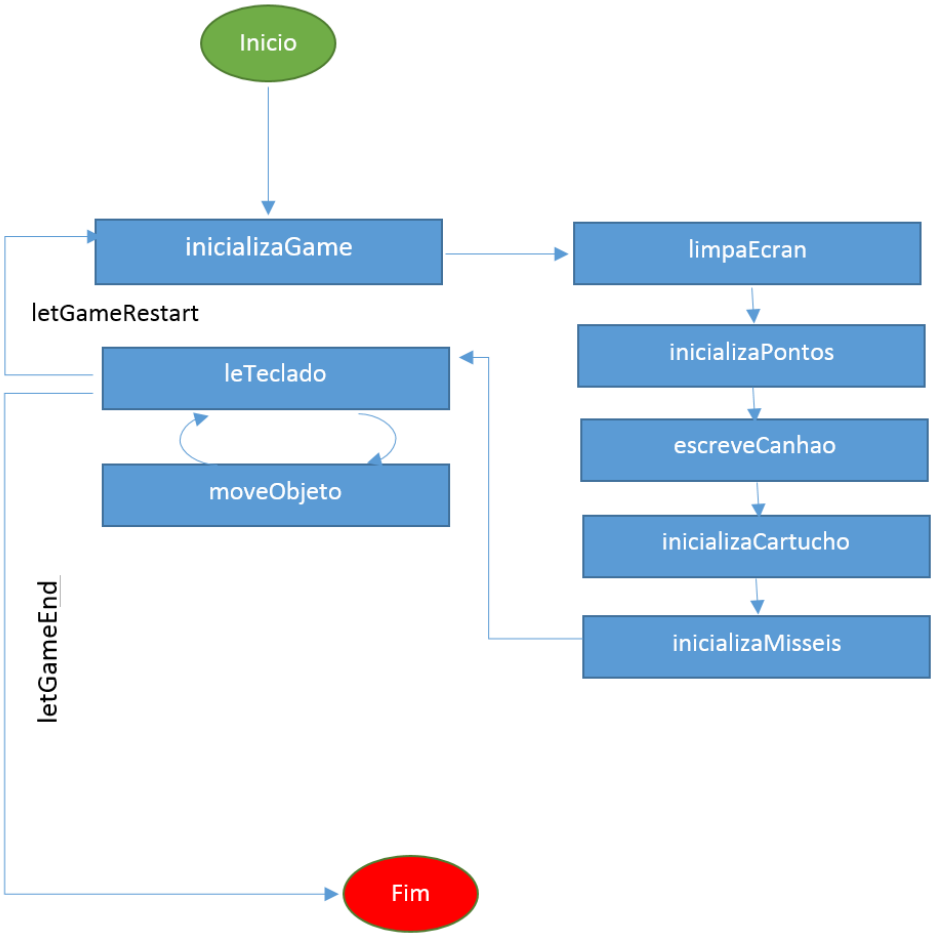
- 2 Displays de 7 segmentos que mostram o conteúdo da pontuação, o qual é incrementado de uma unidade sempre que uma bala atinge o avião (número total de aviões abatidos)
- 2 Displays de 7 segmentos que mostram o número de balas disponíveis para serem lançadas
- Pixel Screen, onde se observa o decorrer do jogo.

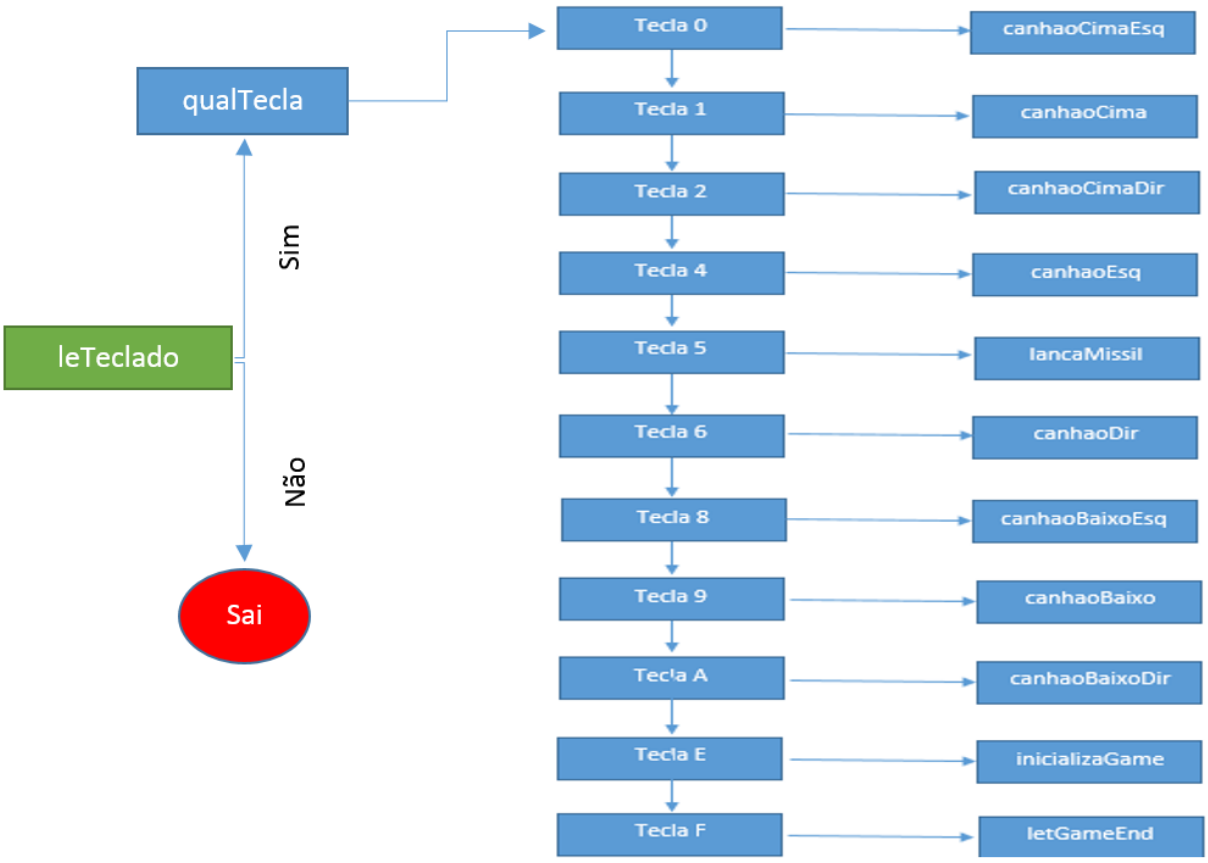


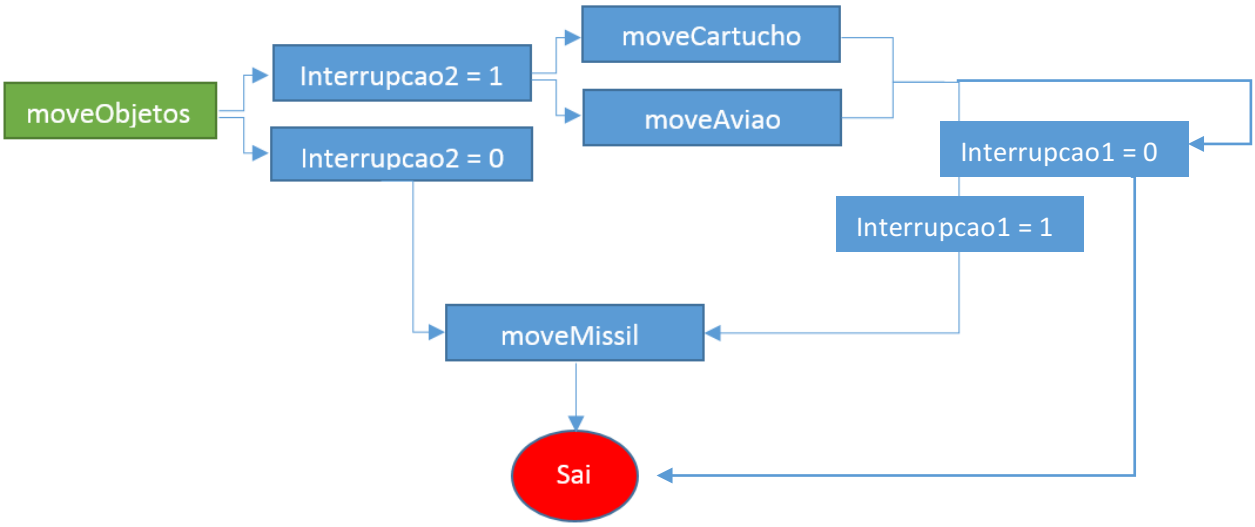
2.2 Mapa de endereçamento escolhido

Dispositivo	Endereços
RAM (MemBank)	0000H a 5FFFH
POUT-3 (periférico de saída de 8 bits)	06000H
PixelScreen	8000H a 807FH
POUT-1 (periférico de saída de 8 bits)	0A000H
POUT-2 (periférico de saída de 8 bits)	0C000H
PIN (periférico de entrada de 8 bits)	0E000H

2.3 Comunicações entre processos




qualTecla
leTeclado
Sim
Não
Sai
Tecla 0
Tecla 1
Tecla 2
Tecla 4
Tecla 5
Tecla 6
Tecla 8
Tecla 9
Tecla A
Tecla E
Tecla F
canhaoCimaEsq
canhaoCima
canhaoCimaDir
canhaoEsq
lancaMissil
canhaoDir
canhaoBaixoEsq
canhaoBaixo
canhaoBaixoDir
inicializaGame
letGameEnd


moveObjetos
Interrupcao2 = 1
Interrupcao2 = 0
moveCartucho
moveAviao
Interrupcao1 = 0
Interrupcao1 = 1
moveMissil
Sai

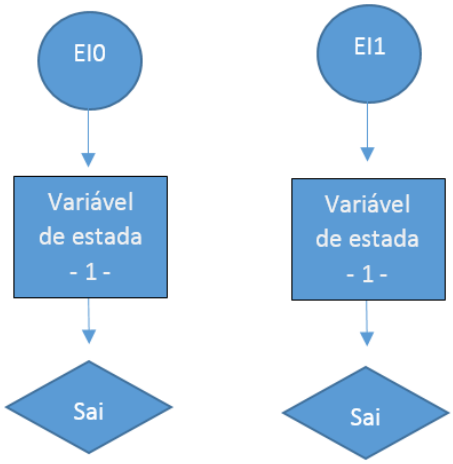
2.4 Variáveis de Estado

- As 2 rotinas de interrupção (interrupcao1 e interrupcao2) colocam duas variáveis de estado a 1 (uma para a Interrupção 1 e outra para a interrupção 2). Estas variáveis, quando a 1, vão permitir o movimento dos aviões e do cartucho. Quando a 0, não há movimento destes objetos.
- A função escrevePixel recebe uma variável de estado. Se esta variável estiver a 1, vai ser aceso um determinado pixel, se tiver a 0 vai apagar o pixel.
- A função colisaoPix determina se houve colisão entre dois objetos (avião e bala, ou cartucho e bala) e caso tenha ocorrido coloca o valor 1 na variável de estado. Esta variável de estado vai dar informação à rotina colisoes.

2.5 Rotinas de Interrupção

Para controlar o movimento das balas, dos aviões e dos cartuchos, recorre-se a interrupções, as quais interrompem o decorrer do programa, retomando no final da interrupção.

A Interrupção 1 tem como função colocar uma variável de estado referente às balas a 1 para que o movimento das mesmas seja possível, e a Interrupção 2 tem como função colocar a variável de estado referente aos aviões e cartucho a 1 para que estes possam ser movidos.



2.6 Rotinas

letGameBegin

Rotina responsável pelo desenvolvimento do jogo. Esta inicializa o Ecrá para que se possa dar inicio ao jogo, inicializa o Stack Pointer para os Push e Pops ativa as interrupções 1 e 2 (responsáveis pelos movimentos dos objetos (balas, cartucho e aviões), e entra num ciclo que tem como função ler a tecla premida do teclado, e de acordo com essa tecla desempenhar determinada função, e mover os objectos.

Chama as seguintes rotinas:

- inicializaGame
- leTeclado
- moveObjectos

inicializaGame

Faz as inicializações iniciais do jogo, para que este possa dar início. Chama as seguintes rotinas:

- limpaEcran
- inicializaPontos
- escreveCanhao
- inicializaAvioes
- inicializaCartucho
- inicializaMisseis

limpaEcran

Apaga o conteúdo existente na memória referente ao Pixel Screen. Esta rotina introduz o valor 0 em todos os endereços do mesmo, começando no endereço 8000H (pixelScreenIn) e acabando no 807FH (pixelScreenOut).

leTeclado

Rotina que faz o varrimento do teclado e verifica se alguma tecla foi premida. Esta função verifica tecla a tecla, começando na última linha e passando para a linha de cima (caso não tenha sido premida alguma tecla dessa linha). Se não foi premida nenhuma das teclas das quatro linhas, sai da rotina, caso contrário entra na rotina que vai decidir o que fazer de acordo com a tecla premida. A função converte o valor da linha premida para hexadecimal e guarda o mesmo na memória para uma consulta posterior.

Chama as seguintes rotinas:

- qualTecla10

qualTecla10

Rotina que de acordo com a tecla premida vai desempenhar determinada função. Primeiro acede ao local da memória onde se encontra a tecla premida e coloca esse valor num registo. Após primeiro passo, multiplica o valor por 4 e soma ao endereço que contém a tabela para as teclas (no conteúdo desta tabela encontra-se máscaras para decidir qual a tecla que vai lançar a bala, a tecla que vai terminar o jogo e a tecla que vai reiniciar o mesmo). Após a soma, retira o primeiro conteúdo para um registo, e o conteúdo do endereço a seguir para um outro. Estes dois registos contêm o valor a somar e a subtrair à linha e à coluna do canhão (caso tenha os valores -1, 0 e 1).

Ao comparar o conteúdo da tabela referente à tecla com as máscaras, decidi se vai para alguma das rotinas responsáveis por estas funções, caso contrário vai mover o canhão.

Chama as seguintes rotinas:

- moveCanhao
- lancaMissil
- inicializaGame
- letGameEnd

escrevePixel

Rotina responsável pelo acender/apagar o pixel de acordo com a linha e coluna que recebe e uma variável de estado (0 para apagar e 1 para acender). De acordo com a linha e a coluna, sabe-se o pixel a acender/apagar e as fórmulas são (8000+4(linha)+coluna//8) dando-nos o endereço do pixel, e (coluna%8) para saber qual o número da coluna dentro do endereço. Para saber qual o valor em binário desse pixel, soma-se o número da coluna ao endereço que contém todos os pixéis em binário.

Caso se queira acender o pixel, faz-se um OR com o valor do endereço já existente, caso contrário nega-se o valor em binário do pixel a acender e faz-se um AND com o conteúdo do endereço. No final volta a guardar esse endereço na memória.

esceveLinha

Rotina responsável por escrever o objeto pretendido. Esta vai decidir se o Pixel Screen recebe na variável de estado 0 ou 1. Faz esta verificação para todos os bits dentro de uma linha.

Chama as seguintes rotinas:

- escrevePixel

apagaLinha

Rotina responsável por apagar o objeto pretendido. Esta vai colocar a variável de estado que o Pixel Screen recebe sempre a 0, para todos os pixéis onde se encontra o objeto passem para 0.

Chama as seguintes rotinas:

- escrevePixel

P.S. Devido a problemas no correr do programa teve-se que criar duas funções em separado, uma para apagar o avião e outra para apagar o cartucho e o canhão. São precisamente iguais, apenas o registo que contém o comprimento do objeto é que difere. Tentou-se que esta rotina recebe-se já este parâmetro da função anterior.

escreveCanhao

Rotina que tem como função enviar parâmetros para a escreveLinha, parâmetros estes que são o comprimento e altura do canhão, coordenadas de referência onde este vai começar a ser escrito e a linha que se quer desenhar. A função envia uma só linha de cada vez, terminado só quando envia todas.

Chama as seguintes rotinas:

- escreveLinha

apagaCanhao

Rotina que tal como a escreveCanhao, tem como função enviar parâmetros para a apagaLinha, parâmetros estes que são a altura do canhão e a variável de estado a 0, a qual decide que vai apagar os bits em vez de os acender.

Chama as seguintes rotinas:

- apagaLinha

moveCanhao

Rotina encarregue mover o canhão. Tem uma segunda função que é decidir para que direção o canhão se vai mover. Esta recebe da função qualTecla10 o valor a somar ou subtrair à linha e coluna do avião. Para decidir para onde se vai mover faz comparações dos valores que recebe da função anterior com -1, 0 e 1.

Para mover o Canhão, este primeiro é apagado nas coordenadas antigas, nas respetivas rotinas que movem o canhão nas diferentes direções são alterados os valores da coordenada de referência e por final este é escrito nas novas coordenadas.

Chama as seguintes rotinas:

- apagaCanhao
- canhaoCimaEsq (Subtrai 1 à linha e coluna)
- canhaoCima (Subtrai 1 à linha)
- canhaoCimaDir (Subtrai 1 à linha e soma 1 à coluna)
- canhaoBaixoEsq (Adiciona 1 à linha e subtrai 1 à coluna)
- canhaoBaixo (Adiciona 1 à linha)
- canhaoBaixoDir (Adiciona 1 à linha e à coluna)
- canhaoEsquerda (Subtrai 1 à coluna)
- canhaoDireita (Adiciona 1 à coluna)
- escreveCanhao

canhaoCima

Rotina encarregue por alterar o valor das coordenadas do canhão. A estas vai subtrair 1 à linha e mantém inalterada a coluna (mover para cima). Tem também a função de verificar se já se encontra o mais acima possível, se sim não se move, fica no mesmo sítio.

canhaoCimaDir

Rotina encarregue por alterar o valor das coordenadas do canhão. A estas vai subtrair 1 à linha e adicionar 1 à coluna (mover para cima e para a direita). Tem também a função de verificar se já se contra o mais acima ou o mais à direita possível. Caso um destes casos aconteça, o canhão não se move.

canhaoCimaEsq

Rotina encarregue por alterar o valor das coordenadas do canhão. A estas vai subtrair 1 à coluna e à linha (mover para cima e para a esquerda). Tem também a função de verificar se já se contra o mais acima ou o mais à esquerda possível. Caso um destes casos aconteça, o canhão não se move.

canhaoBaixo

Rotina encarregue por alterar o valor das coordenadas do canhão. A estas vai adicionar 1 à linha e mantém inalterada a coluna (mover para baixo). Tem também a função de verificar se já se encontra o mais abaixo possível, se sim não se move, fica no mesmo sítio.

canhaoBaixoDir

Rotina encarregue por alterar o valor das coordenadas do canhão. A estas vai adicionar 1 à linha à coluna (mover para baixo e para a direita). Tem também a função de verificar se já se contra o mais abaixo ou o mais à direita possível. Caso um destes casos aconteça, o canhão não se move.

canhaoBaixoEsq

Rotina encarregue por alterar o valor das coordenadas do canhão. A estas vai adicionar 1 à coluna e subtrair 1 à linha (mover para baixo e para a esquerda). Tem também a função de verificar se já se contra o mais abaixo ou o mais à esquerda possível. Caso um destes casos aconteça, o canhão não se move.

canhaoDireita

Rotina encarregue por alterar o valor das coordenadas do canhão. A estas vai adicionar 1 à coluna e mantém inalterada a linha (mover para a direita). Tem também a função de verificar se já se encontra o mais à direita possível, se sim não se move, fica no mesmo sítio.

canhaoEsquerda

Rotina encarregue por alterar o valor das coordenadas do canhão. A estas vai subtrair 1 à coluna e mantém inalterada a linha (mover para a esquerda). Tem também a função de verificar se já se encontra o mais à esquerda possível, se sim não se move, fica no mesmo sítio.

inicializaAvioes

Rotina que tem como função desenhar os aviões nas coordenadas iniciais, coordenadas que já estão colocadas na memória. Percorre este ciclo o número de vezes que foi estipulado para o máximo de aviões (o número de aviões foi estipulado de 3, mas pode ser alterado, adicionando à tabela das coordenadas do avião mais duas words por avião e alterando a constante do número máximo de aviões de 3 para o número pretendidos).

Chama as seguintes rotinas:

- escreveAviao

escreveAviao

Rotina responsável por escrever o avião nas coordenadas indicadas. Esta envia como parâmetros para a escreveLinha, a altura e o comprimento do mesmo e o valor da linha a desenhar. Faz isto para todos os aviões e no final chama uma rotina responsável por geral o um numero aleatório para as linhas do avião.

Chama as seguintes rotinas:

- escreveLinha
- geraAvioesAlea

apagaAviao

Rotina que tem a mesma função que a apagaCanhao. Esta envia parâmetros para a apagaLinhaA (= apagaLinha), os quais são a altura do mesmo e a variável de estado a 0, que decide que vai apagar.

Chama as seguintes rotinas:

- apagaLinhaA

moveAvioes

Rotina encarregue por movimentar os aviões da direita para a esquerda de acordo com a interrupção 2. Se a variável de estado desta interrupção estiver a 1, os aviões vão-se movimentar uma coluna para a esquerda, mantendo a mesma linha. Caso já se encontre o mais à esquerda possível, este vai ser reposto no ecrã mas o mais à direita possível para continuar o seu movimento para a esquerda.

A rotina apaga o avião nas coordenadas antigas, altera para as novas, subtraindo 1 à coluna e escrevendo nas novas coordenadas. Executa estes procedimentos para todos os aviões que se encontrem em movimento.

Chama as seguintes rotinas:

- apagaAviao
- repoeAviao
- escreveAviao

repoeAviao

Rotina encarregue por fazer reaparecer o avião do lado direito do Pixel Screen assim que ele seja abatido por uma bala ou assim que chegue ao limite esquerdo do Pixel Screen.

De acordo com o número guardado na memória para gerar uma linha aleatória para o novo posicionamento do avião, este valor é somado ao endereço da tabela que contém todas as linhas possíveis para ser reintroduzido. Este novo valor da linha vai ser substituído pelo anterior e a coluna volta à coluna inicial antes do movimento.

geraAvioesAlea

Rotina que trata de criar um número aleatório para adicionar à tabela que contém os vários valores da linha onde o novo avião vai ser escrito.

inicializaCartucho

Rotina que tem como função desenhar o cartucho nas coordenadas iniciais, coordenadas que são constantes, facilmente podem ser alteradas.

Chama as seguintes rotinas:

- escreveCartucho

P.S. Devido a falta de tempo foi a única função de inicializar em que as coordenadas são constantes. As outras, depois de se fazer Restar ao jogo, os objetos são escritos precisamente no último local onde estavam antes do reinício.

escreveCartucho

Rotina responsável por escrever o cartucho nas coordenadas indicadas. Esta envia como parâmetros para a escreveLinha, a altura e o comprimento do mesmo e o valor da linha a desenhar.

Chama as seguintes rotinas:

- escreveLinha

apagaCartucho

Rotina que tem a mesma função que a apagaCanhao e a apagaAviao. Esta envia parâmetros para a apagaLinha, os quais são a altura do mesmo e a variável de estado a 0, que decide que vai apagar.

Chama as seguintes rotinas:

- apagaLinha

moveCartucho

Rotina encarregue por mover o cartucho para a esquerda e para a direita, de acordo com o valor gerado aleatoriamente pela função geraCartuchoAlea. Anteriormente foi verificado se a variável de estado da interrupção se encontra a 1 ou não. Se não se encontrar, não se movimenta, caso contrário movimenta-se. O princípio da obtenção deste valor é o mesmo que para as linhas do avião, a diferença é que este valor é para as colunas.

O cartucho mantém-se no mesmo sítio durante 10 clocks, e no final destes move-se para uma nova posição. Sempre que o PC passa por este rotina incrementa uma unidade ao tempo que ele se encontra no sítio e quando chega ao limite de tempo, volta a ficar a zero.

A maneira de mover o cartucho é como todas as outras movimentações - apagar o cartucho nas coordenadas anteriores, alterar as mesmas e voltar a escrevê-lo nas novas.

Chama as seguintes funções:

- geraCartuchoAlea
- apagaCartucho
- esceveCartucho

geraCartuchoAlea

Rotina que trata de criar um número aleatório para adicionar à tabela que contém os vários valores da coluna onde o cartucho vai ser escrito.

inicializaMisseis

Rotina que tem como função colocar todas as coordenadas das balas a zero (ao terem estas coordenadas significa que podem ser lançadas e não movidas).

lancaMissil

Rotina encarregue de lançar um míssil de acordo com o valor da variável de estado da Interrupção 1. Caso o valor seja 1, o míssil vai ser lançado, caso contrário não.

Primeiro esta rotina verifica se existe algum míssil em condições de ser lançado e para isso compara a coluna dos mesmos com 0. Caso seja 0 é porque está pronto (compara com 0 porque é uma posição do míssil que nunca vai existir quando este se estiver a mover, uma vez que o canhão tem algum comprimento). Após verificar se existe algum míssil pronto a ser lançado, ao valor do canhão soma-se X à linha e Y à coluna para que o míssil saia da parte de cima do canhão.

Chama as seguintes funções:

- escreveMissil

escreveMissil

Rotina encarregue de escrever um míssil nas coordenadas que lhe indicarem. Como um míssil é só de um bit, apenas precisa de chamar a função que escreve o Pixel e colocar a variável de estado a 1 para acender o pixel.

apagaMissil

Rotina encarregue de apaga um míssil nas coordenadas que lhe indicarem. Como um míssil é só de um bit, apenas precisa de chamar a função que escreve o Pixel e colocar a variável de estado a 0 para apagar o pixel.

moveMissil

Rotina encarregue por mover os mísseis que se encontram no Pixel Screen. Caso a variável de estado da Interrupção 1 esteja a 1, os mísseis vão ser movidos, caso contrário não.

Para fazer a seleção dos misseis que vão ser movidos, vai-se verificar as coordenadas dos mesmos. Caso estejam a 0, não há movimento pois estes mísseis estão prontos a serem lançados. Caso contrário vai movê-los e após mover verifica se houve colisão com algum dos objetos (aviões ou cartucho). Se houve colisão o mesmo vai ficar novamente disponível para ser lançado.

Numa última verificação, caso não tenha colidido com nenhum objeto, a linha do míssil vai ser comparada com a linha 0, limite superior do écran e caso tenha chegado lá, o míssil vai ser reposto e pronto a ser lançado.

No final da rotina, coloca a variável de estado da Interrupção 1 a 0.

Chama as seguintes funções:

- colisoes
- apagaMissil
- escreveMissil
- repoeMissil

repoeMissil

Rotina que tem como função apagar o míssil e colocar as suas coordenadas a 0 (prontos a serem disparados novamente).

adicionaMissil (não está em funcionamento)

Rotina que tinha como objectvo adicionar ao número de mísseis já existentes 3 – Quando se atinge o Cartucho – Caso o resultado da soma fosse superior a 9, o valor de misseis prontos a serem disparados seria 9.

escreveExplosao

Rotina responsável por escrever a explosão nas coordenadas onde o avião foi abatido. Esta envia como parâmetros para a escreveLinha, a altura e o comprimento da mesma e o valor da linha a desenhar.

Chama as seguintes rotinas:

- escreveLinha

apagaExplosao

Rotina que tem como função apagar a explosão do avião. Esta envia parâmetros para a apagaLinha, os quais são a altura do mesmo e a variável de estado a 0, que decide que vai apagar.

Chama as seguintes rotinas:

- apagaLinha

moveObjectos

Rotina encarregue por mover todos os objetos do jogo. Caso a variável de estado da Interrupção 2 esteja a 0, os aviões e o cartucho permanecerão no mesmo sítio. A verificação da variável de estado das balas encontra-se dentro da rotina moveMissil.

Chama as seguintes rotinas:

- moveAvioes
- moveCartucho
- moveMissil

colisoes

Rotina encarregue de todas as colisões do jogo. Primeiro começa por verificar se houve colisão de uma bala com os aviões e caso se tenha sucedido, o mesmo vai ser apagado, vai ser reposto no início do ecrã, ser feita a explosão e no final apagada a mesma. Por sua vez também vai ser atualizada a pontuação (vai ser somado mais um valor à pontuação já existente).

Por sua vez averigua se um dos misseis colidiu com o cartucho e caso se tenha sucedido, seria adicionado 3 balas às disponíveis para serem lançadas.

Chama as seguintes rotinas:

- aviaoColisao
- apagaAviao
- repoeAviao
- escreveExplosao
- pontuacao
- apagaExplosao
- cartuchoColisao
- adicionaMissil

aviaoColisao

Rotina que enviará como parâmetros as coordenadas de cada pixel aceso do avião e as coordenadas da bala para a função colisaoPix que por sua vez averiguará se houve colisão ou não.

Para saber as coordenadas de cada pixel, utiliza-se a tabela colisaoAviao que tem o valor a adicionar e a subtrair à coordenada de referência do avião.

Chama as seguintes rotinas:

- colisaoPix

P.S. As rotinas que escrevem os objetos poderiam ter sido feitos assim, à coordenada de referência somava-se X à linha e Y à coluna para se acender apenas os bits pretendidos. Deste modo ficaria mais percetível a sobreposição de aviões.

cartuchoColisao

Rotina igual à aviaoColisao que tem como objetivo enviar como parâmetros as coordenadas de cada pixel aceso do cartucho e as coordenadas da bala para a função colisaoPix que por sua vez averiguará se houve colisão entre estes dois objetos ou não.

O método para obter as coordenadas dos pixéis acesos é o mesmo que o método utilizada na função aviaoColisao.

Chama as seguintes rotinas:

- colisaoPix

colisaoPix

Rotina que vai verificar se as coordenadas das balas são iguais às coordendas de cada pixel aceso dos aviões e dos cartuchos. Caso sejam iguais é porque houve colisão entre os dois objetos e não verifica mais nenhuma coordenada. Esta função por sua vez devolve uma variável de estado que indica se houve colisão ou não.

inicializaPontos

Rotina que tem como função colocar os pontos do jogo a 0. Guarda em dois locais, o primeiro é no Periférico de Saída 1 (Display para os pontos) e o segundo é na memória.

P.S. Decidiu-se guardar na memória pois quando se ia atualizar o valor da pontuação já existente a partir do Periférico de Saída 1, este colocava valores diferentes aos esperados. Desta maneira conseguiu-se contornar o problema.

pontuacao

Rotina que tem como objetivo atualizar a pontuação sempre que há uma colisão de uma bala com um avião. Uma vez que se pretendia um Display em decimal, quando o valor dos 4 bits menos significativos chegavam a 9H, a este valor somava-se 7H para passar de 9H para 10H.

No final de atualizar a pontuação, o novo valor é guardado na memória e colocado no Periférico de Saída 1. Quando o valor da pontuação do jogo chega a 99, o jogo termina.

3. Conclusões

O objetivo do trabalho era a construção de um jogo em Assembly utilizando todas as matérias que se aprenderam ao longo do semestre. Este jogo consiste numa simulação de tiro aos aviões, em que o jogador destrói o maior número de aviões possíveis até chegar aos 99 pontos ou até ficar sem balas (momentos em que o jogo acaba).

Referente à execução do jogo, o projeto está incompleto, tendo um número infinito de balas uma vez que quando uma chega ao limite superior do ecrã ou atinge algum objeto, fica pronta para ser lançada novamente. Falta também a 2 teclas, a Restart e a Pausa. No que diz respeito ao visual do jogo, falta um ecrã de início, pausa e fim de jogo.

A forma dos canhões, aviões e cartuchos foi alterado do que inicialmente foi dado no enunciado, podendo ser facilmente alteradas. As novas ilustrações encontram-se abaixo apresentadas

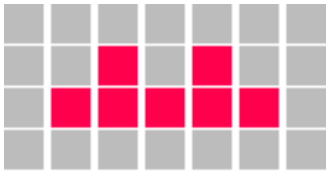


Fig1. Nova ilustração do canhão.

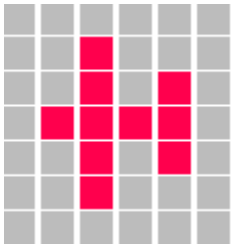


Fig2. Nova ilustração do avião.

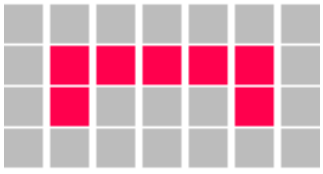


Fig3. Nova ilustração do cartucho.

A solução do jogo apresentada funciona bem apesar de estar em falha alguns pormenores anteriormente referidos. No que diz respeito ao funcionamento do mesmo, temos aviões a moverem-se e quando atingem o limite esquerdo do ecrã os mesmos voltam à parte direita do ecrã aparecendo em linhas aleatórias. Temos múltiplas balas, todas em movimento e a serem disparadas. O canhão move-se em todas as direções com os respetivos limites e quando existe uma colisão com o avião, o mesmo explode.

Por falta de tempo não foi possível incluir 4 ecrãs de escrita no Pixel Screen, mas seriam facilmente introduzidos. Criar-se-iam words como para o desenho dos objectos e escrever-se linha a linha, pixel a pixel. Caso os desenhos ultrapassassem o tamanho permitido para o PEPE, escrever-se-ia duas letras primeiro e depois de escritas incluía-se as restantes.

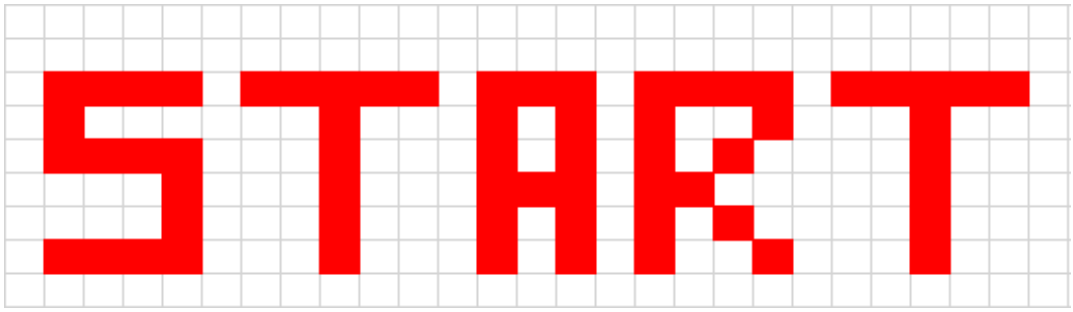


Fig4. Ecrã inicial.

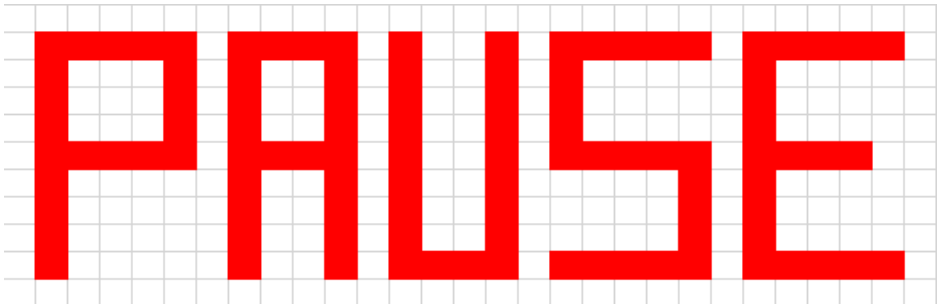


Fig5. Ecrã de pausa

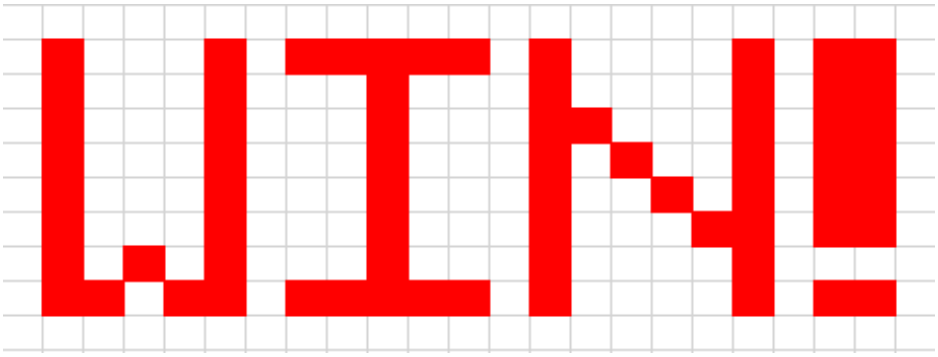


Fig6. Ecrã de vitória.

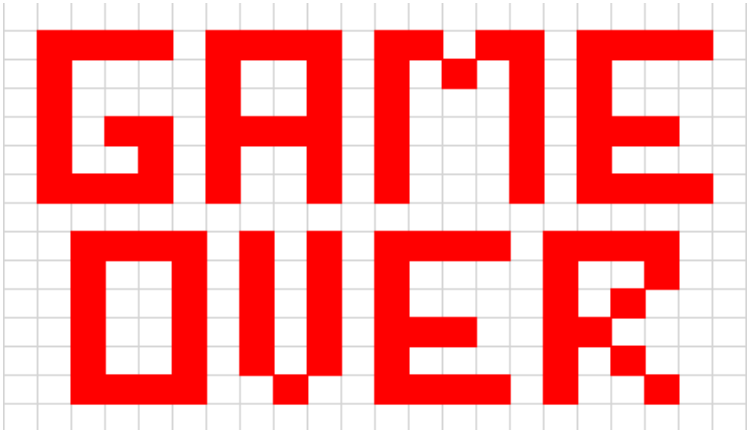


Fig7. Ecrã de Game Over

4. Código Assembly

Segue em anexo no ficheiro Projecto.asm