RELATÓRIO

Trabalho Prático

Tecnologias e Arquiteturas de Computadores

Licenciatura em Engenharia Informática

> **O jogo consiste num labirinto onde as letras de uma palavra estão espalhadas. Cabe ao jogador encontrá-las a todas antes do tempo terminar.**

Trabalho realizado por:

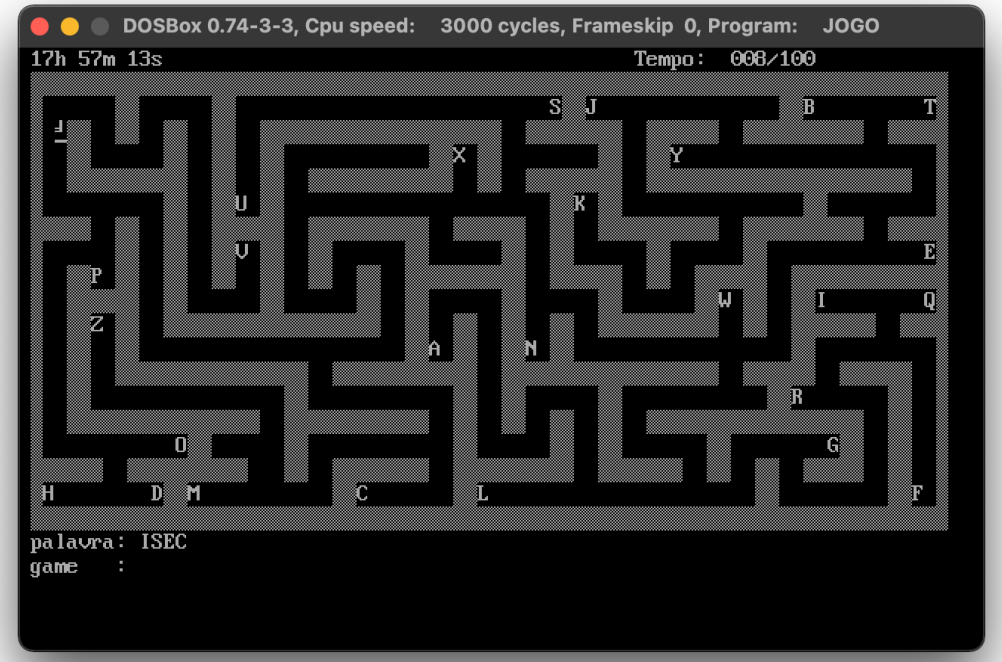
Tomás Gomes Silva - 2020143845

João Miguel Duarte dos Santos - 2020136093

Índice

Introdução

O trabalho prático de Tecnologias e Arquiteturas de Computadores consiste na criação de um jogo na linguagem de programação **Assembly** para processadores 8086.
Neste jogo, as letras de uma palavra encontram-se espalhadas por um labirinto e o objetivo é percorrer o labirinto e recolher as letras em falta para voltar a formar a palavra. O jogador tem que ser rápido e ágil pois existe um limite de tempo que não pode ser ultrapassado.
Neste relatório serão abordados alguns pontos essenciais relativos à realização do trabalho bem como a descrição das implementações que realizámos de modo a obter o nosso jogo final. A variante que implementámos corresponde à primeira variante.



Deteção de Colisões

Para obrigar o avatar a ficar dentro dos limites do labirinto e desse modo não atravessar paredes, implementámos um simples algoritmo de deteção de colisões entre o avatar e as paredes do labirinto.

Ao clicar numa tecla a posição é modificada (neste caso o utilizador clicou na tecla de cima e isso decrementou o valor de POSy). Logo a seguir, o avatar é movido para a posição modificada pelo passo anterior e começam então as comparações necessárias.

O algoritmo de comparação consiste em 3 simples passos:

1. É lido o caractere na posição atual do avatar (posição modificada) com recurso à interrupção 10h e à função 08h
2. O caractere lido é guardado em AL e com isto podemos verificar se o avatar se encontra em cima de uma parede (10110001b = '±')
    a. **Se o caractere lido for igual a '±'** isso quer dizer que o avatar se encontra em cima de uma parede. Sendo assim, a posição é incrementada e é chamada a macro GOTO_XY para que o jogador volte ao sítio onde estava previamente
    b. **Se o caractere lido for diferente de '±'** então o avatar encontra-se numa posição válida e a posição do avatar permanece inalterada
3. No fim deste algoritmo voltamos para o CICLO para que o jogador possa continuar a jogar.

Este exemplo teve por base a suposição de o utilizador ter clicado na tecla para cima mas o algoritmo é válido para qualquer tecla sendo apenas necessário ajustar que variáveis que são incrementadas e decrementadas (POSx e POSy).

```
5 references
ESTEND:
    cmp     al,48h
    jne     BAIXO
    dec     POSy        ;cima

    goto_xy POSx,POSy ; Move o jogador uma posição para cima
    mov     ah, 08h ; Ler o caractere que está no sítio do cursor
    int     10H ; Executar o interrupt
    cmp     al, 10110001b ; Comparar se na posição do cursor está o símbolo '±' que neste caso é a parede
    jne     CICLO ; Se não está então é uma jogada válida e o jogador fica lá
    inc     POSy ; Se está em cima de uma parece a posição é incrementada (para voltar à posição inicial)
    goto_xy POSx,POSy ; Move o jogador de volta para onde estava
    jmp     CICLO ; Volta para o ciclo sem ter alterado a posição em que estava
```

Posição Inicial Aleatória

De modo a adicionar um pouco de aleatoriedade ao jogo, implementámos uma funcionalidade que consiste em colocar o avatar numa posição aleatória sempre que o jogo começa e sempre que o jogador sobe de nível.
Utilizámos o procedimento CalcAleat fornecido pelo professor que gera números aleatórios de 16 bits e depois bastou-nos apenas ver se o número gerado era inferior ao tamanho do labirinto na vertical e na horizontal e se essa posição estava vazia.

Começamos por saltar para a label CALC_RANDOM_X que trata de gerar um número aleatório. Esse número é guardado na pilha pelo procedimento CalcAleat com recurso ao PUSH AX. Para recuperarmos o que estava em AX damos POP AX e temos finalmente o nosso valor aleatório em AX.
Comparamos o valor aleatório com 72 (largura máxima do labirinto) e se o valor for superior a 72 ou igual saltamos de volta para o CALC_RANDOM_X para que seja gerado outro número aleatório para X. No caso de o número aleatório ser inferior a 72 então movemos esse número para POSx e saltamos para CALC_RANDOM_Y para calcularmos um valor para Y.

O procedimento para gerar um valor para POSy é exatamente o mesmo só que no fim deste ciclo saltamos para CHECK_COORDS para verificar se o jogador não foi colocado em cima de uma parede ou de uma peça. Caso isso tenha acontecido saltamos de volta para CALC_RANDOM_X e repetimos todos os passos que descrevemos em cima até que estas condições estejam todas reunidas:

- POSx é menor do que 72 e POSy é menor do que 18
- No ponto (POSx, POSy) não se encontra nem uma parede nem uma letra

Com as duas variáveis de acordo com os requisitos em cima estipulados, procedemos então à movimentação do avatar para a posição aleatória utilizando o GOTO_XY.

```
3 references
CALC_RANDOM_X:
    call    CalcAleat
    pop ax
    CMP AX, 72
    JNBE    CALC_RANDOM_X
    MOV POSx, AL
    MOV POSxa, AL
    JMP CALC_RANDOM_Y
```

```
3 references
CALC_RANDOM_Y:
    call    CalcAleat
    pop ax
    CMP AX, 18
    JNBE    CALC_RANDOM_Y
    MOV POSy, AL
    MOV POSya, AL
    JMP CHECK_COORDS
```

```
1 reference
CHECK_COORDS:
    goto_xy POSx,POSy
    mov     ah, 08h
    int     10H
    CMP     AL, ' '
    JE      CICLO
    JMP     CALC_RANDOM_X
```

Horas e Tempo Limite

No canto superior esquerdo do jogo está representada a hora atual. Isto foi conseguido graças aos procedimentos fornecidos pelo professor, LER_TEMPO e TRATA_HORAS, que nos permitem obter as horas do computador e converter os dígitos para os seus caracteres correspondentes para que estes possam ser impressos no ecrã. O procedimento TRATA_HORAS está a ser chamado de segundo em segundo para atualizar as horas sempre que possível.

A variável STR12 é um array que armazena espaço para ser preenchida com as horas. Os passos necessários para convertermos um número no seu caracter são:

1. Dividir o número por 10
2. Em AL vai estar o número que corresponde ao dígito das dezenas e em AH vai estar o número que corresponde ao dígito das unidades
3. Para obtermos o caractere para cada um desses números somar-lhe-emos 48 e teremos assim os dois caracteres necessários para formar as horas.

Repetimos estes procedimentos para os minutos e para os segundos.

Já no canto superior direito deparamo-nos com o tempo limite e o tempo decorrido do jogo. Para conseguir com que o tempo decorrido aumente colocámos um INC Tempo_tj dentro do procedimento TRATA_HORAS que está constantemente a ser chamado e desta forma o tempo decorrido é incrementado todos os segundos.
Usámos o mesmo método para gerar uma string a partir do valor numérico da variável. Quando o jogador sobe de nível o tempo limite é decrementado em 10 (tempo limite inicial: 99).

```
mov     ax,Tempo_limite
MOV     bl, 10
div     bl
add     al, 30h
add     ah, 30h
MOV     String_TJ[3],'/'
MOV     String_TJ[4],'0'
MOV     String_TJ[5], al
MOV     String_TJ[6], ah
MOV     String_TJ[7],'$'
GOTO_XY 59,0
MOSTRA  String_TJ
```

Preenchimento da Palavra

O preenchimento de uma palavra ocorre sempre que o avatar passa por cima de uma letra que pertença a essa palavra.
No nosso ciclo de jogo comparamos o caractere em que o avatar se encontra com o caractere " " (espaço). Isto serve para verificar se nos encontramos em cima de uma letra e se esse for o caso então saltamos para CHECK_LETRA, label esta que vai tratar de verificar se a letra em que nos encontramos pertence à palavra atual. Se o avatar não se encontrar em cima de uma letra então o ciclo continua normalmente.
Para além disso, colocamos SI a -1 para o podermos utilizar como índice no ciclo de verificação da letra.

O algoritmo que se encontra na label CHECK_LETRA consiste no seguinte:

1. Incrementar SI (este toma o valor de 0 na primeira iteração)
2. Movemos para AL o caractere que se encontra no índice SI da string que guarda a palavra que temos de que encontrar
3. Se essa letra for ' ' (espaço) isso quer dizer que iterámos pela palavra toda e sendo assim saltamos para FIM_LETRA que está encarregue de fazer com que o SI volte a -1 (pois este vai voltar a ser utilizado em CHECK_VITORIA) e saltamos incondicionalmente para a label CHECK_VITORIA. Essa parte do código verifica se, depois de o avatar ter comido uma letra, isso surtiu numa vitória (será explicado em detalhe no tópico seguinte referente às ***Condições de Vitória***)
4. No caso de a letra não ser ' ' (espaço), sabemos que o avatar se encontra em cima de uma letra
    a. **Se a letra em que nos encontramos NÃO for igual à letra do índice atual** voltamos ao início do ciclo para incrementar o SI e passar à próxima letra.
    b. **Se a letra em que nos encontramos for igual à letra do índice atual então** colocamos essa mesma letra no exato índice em que é indicado por SI. Com isto temos a certeza de que a letra fica na ordem certa.
5. Depois de mostrar a string com a nova letra adicionada voltamos ao início do ciclo para verificar a letra que vem a seguir

```
MOV     SI, -1
MOV     CL, DL
CMP     CL, ' '
JNE     CHECK_LETRA

goto_xy POSx,POSy
```

```
1 reference
FIM_LETRA:
    MOV SI, -1
    goto_xy     POSx,POSy
    JMP CHECK_VITORIA
```

```
3 references
CHECK_LETRA:
    INC SI
    MOV AL, String_nome[SI]
    CMP AL, ' '
    JE FIM_LETRA
    CMP AL, CL
    JNE CHECK_LETRA
    MOV Construir_nome[SI], CL
    goto_xy     10,21
    MOSTRA      Construir_nome
    JMP CHECK_LETRA
```

Condições de Vitória

Quando acabamos de verificar se uma letra pertence à palavra que queremos encontrar é necessário averiguar se as strings String_nome e Construir_nome são exatamente iguais. Caso sejam, isto indica que o jogador completou a palavra com sucesso.
Como referido no tópico anterior, sabemos que SI se encontra a -1. Na primeira iteração SI toma o valor de 0.

A label CHECK_VITORIA consiste numa sequência de comparações bastante simples:

1. Movemos para AL o caractere que se encontra na variável String_nome de índice SI
2. Movemos para AH o caractere que se encontra na variável Construir_nome de índice SI
3. Comparamos o caractere que está em AL com o caractere ‘ ‘ (espaço) para verificar se terminámos de iterar pela string
    a. **Se o caractere em AL for igual a ‘ ‘ (espaço)**, então acabámos de iterar pela string e sabemos que, se chegámos a esse ponto, então as strings String_nome e Construir_nome são iguais uma à outra. Isto quer dizer que o jogador encontrou todas as letras da palavra e salta, consequentemente, para o procedimento JOGO_TERMINOU_VITORIA que vai avançar para o próximo nível ou terminar o jogo se for esse o caso
    b. **Se o caractere em AL for diferente de ‘ ‘ (espaço)**, então resta-nos comparar AL com AH para observar se os caracteres nas duas strings no índice SI são iguais. Se forem iguais voltamos para o início do ciclo e repetimos todos os procedimentos em cima descritos para verificar se a próxima letra também já foi coletada. Se forem diferentes então a palavra está incompleta e saímos do ciclo saltando para a label IMPRIME que vai dar continuidade ao jogo

No caso de o jogador ter apanhado todas as letras necessárias para completar a palavra saltamos para o procedimento JOGO_TERMINOU_VITORIA. Este procedimento termina o jogo mostrando uma mensagem de vitória ou salta para o nível seguinte dependendo do nível em que nos encontrávamos e reduz 10 segundos no tempo limite do jogo.
As labels CARREGAR_NÍVEL_XPTO alteram a palavra e limpam a string Construir_nome.

```
2 references
CHECK_VITORIA:
    INC SI
    MOV AL, String_nome[SI]
    MOV AH, Construir_nome[SI]
    CMP AL, ' '
    JE JOGO_TERMINOU_VITORIA
    CMP AL, AH
    JNE IMPRIME
    JMP CHECK_VITORIA
```

```
2 references
JOGO_TERMINOU_VITORIA PROC
    MOV         AX, 10
    SUB         Tempo_limite, AX
    XOR         AX, AX
    MOV         Tempo_j, AX
    CALL        Trata_Horas
    INC         Nivel

    MOV AX, 2
    CMP AX, Nivel
    JE  CARREGAR_NIVEL_2
    MOV AX, 3
```

Conclusão

Este trabalho permitiu-nos consolidar bem a matéria das aulas de Tecnologias e Arquiteturas de Computadores e colocar-nos à prova relativamente àquilo que realmente sabíamos. Fomos confrontados com problemas que nos obrigaram a investigar mais sobre determinados assuntos e consideramos que isso é também bastante importante.
Por se tratar de um jogo creio que nos deu um pouco mais de motivação para realizarmos este trabalho prático e de qualquer das formas foi uma excelente oportunidade de construir algo com uma aplicação prática.