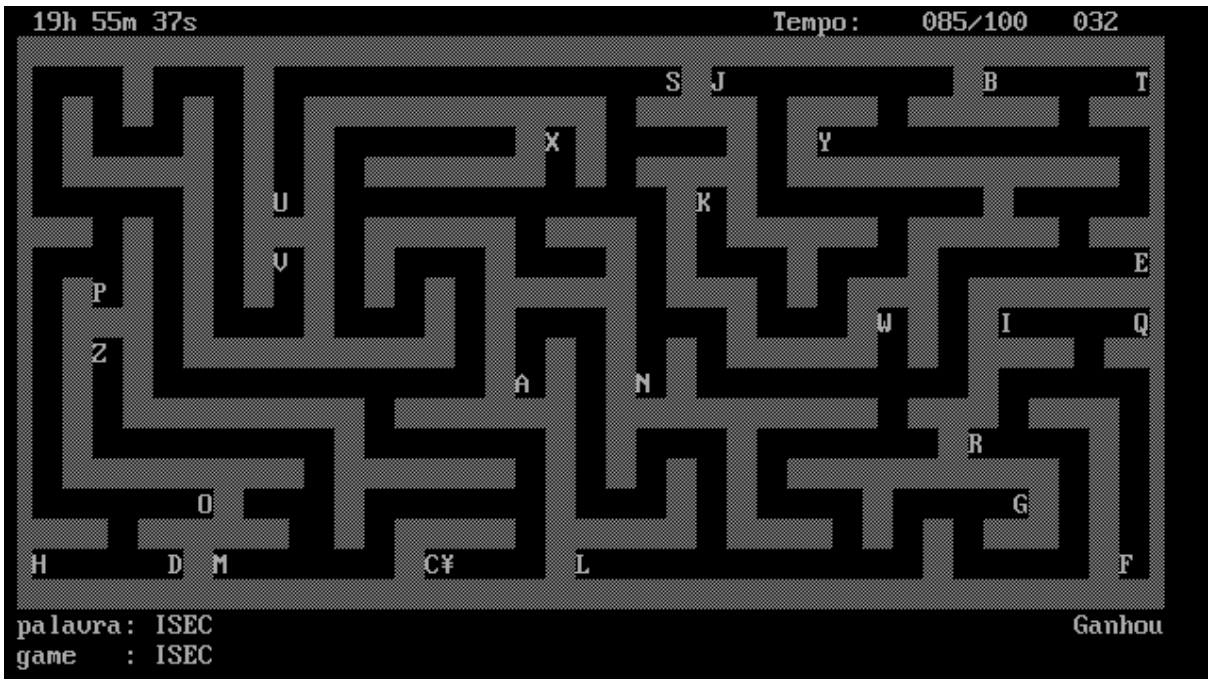
Licenciatura Engenharia Informática

Tecnologias e Arquiteturas de Computadores 2020/2021



Labirinto: O jogo que consiste em encontrar todas as letras duma palavra antes de acabar o tempo limite.

Ricardo Ferreira – 21260251

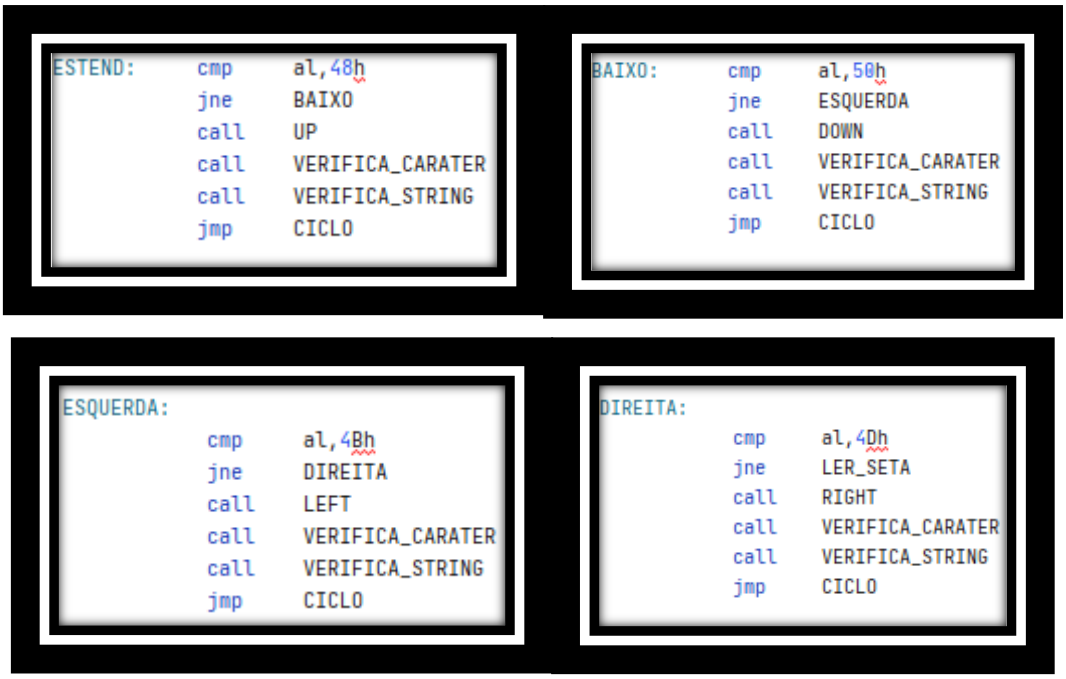
1. Introdução

Pretende-se desenvolver em linguagem Assembly um jogo que consiste em percorrer um labirinto fechado que além das paredes tem também no seu interior todas as letras do alfabeto. O objetivo é encontrar todas as letras duma palavra antes de acabar o tempo limite.

2. Funcionamento

A navegação do Avatar pelo labirinto é efectuada com recurso ao teclado, nomeadamente às setas (←, →, ↑, ↓)

O avatar é uma função que na Base lê uma seta, verifica se já chegou no tempo final e se não for verificado vai ler cada seta do teclado verificando, se encontrou uma parede, se na posição onde está, encontra um carater da palavra desse nível (verifica_carater) e se encontrou todas as letras da string (verifica_string).



Call UP: Movimento do avatar para cima, verificando uma parede;

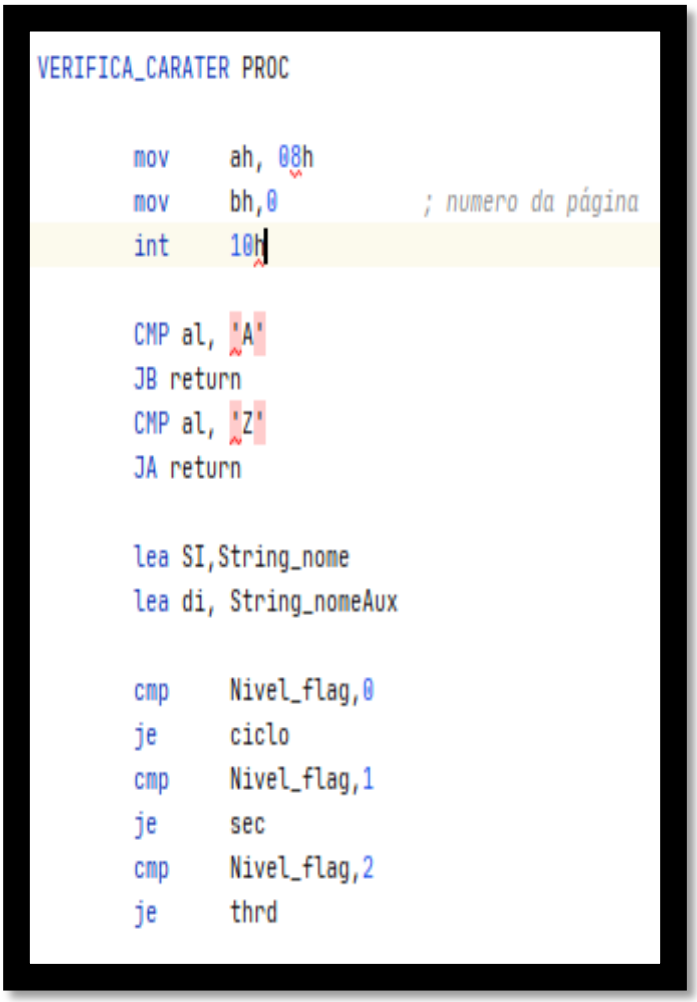
Call Down: Movimento do avatar para baixo, verificando uma parede;

Call Left: Movimento do avatar para a esquerda, verificando uma parede;

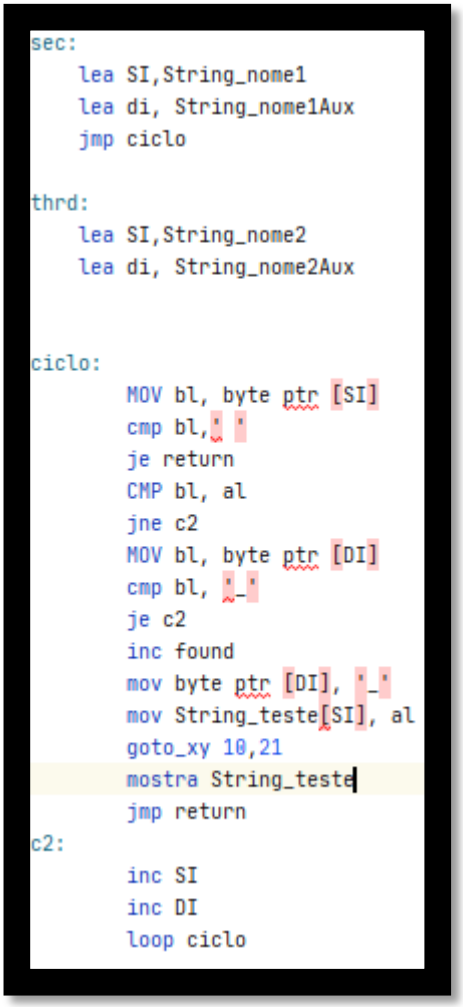
Call Right: Movimento do avatar para a direita, verificando uma parede;

3. Lógica

O jogo é jogado na 1ª Variante, ou seja, o labirinto é sempre igual para todos os níveis do jogo apresentando todas as letras do alfabeto, sendo que a dificuldade aumenta com a diminuição do tempo e com palavras de maior dimensão. O avatar que respeita a ordem das sucessivas letras da palavra a encontrar.



```
VERIFICA_CARATER PROC

        mov     ah, 08h
        mov     bh,0            ; numero da página
        int     10H

        CMP al, 'A'
        JB return
        CMP al, 'Z'
        JA return

        lea SI,String_nome
        lea di, String_nomeAux

        cmp     Nivel_flag,0
        je      ciclo
        cmp     Nivel_flag,1
        je      sec
        cmp     Nivel_flag,2
        je      thrd
```



```
sec:
    lea SI,String_nome1
    lea di, String_nome1Aux
    jmp ciclo

thrd:
    lea SI,String_nome2
    lea di, String_nome2Aux


ciclo:
        MOV bl, byte ptr [SI]
        cmp bl,' '
        je return
        CMP bl, al
        jne c2
        MOV bl, byte ptr [DI]
        cmp bl, '_'
        je c2
        inc found
        mov byte ptr [DI], '_'
        mov String_teste[SI], al
        goto_xy 10,21
        mostra String_teste
        jmp return
c2:
        inc SI
        inc DI
        loop ciclo
```

A função verifica_carater compara o carater em que o avatar se encontra, primeiro compara se está dentro dos limites das letras a usar, seguidamente compara o nível em que está e aponta o offset do registo SI para a Palavra do nível atual.

Efetivamente, o ciclo compara a Palavra do nivel atual com uma igual auxiliar e coloca ‘_’ nessa posição da palavra e incrementa a variavel found.

```
VERIFICA_STRING PROC

;SE FOR 0 FAZ String nome se for 1 faz stringnome1 se for 2 faz string nome 2

        cmp Nivel_flag,0
        je  nivel1
        cmp Nivel_flag,1
        je  nivel2
        cmp Nivel_flag,2
        je  nivel3
nivel2:
       mov bl, String_nomeLen1
       jmp encontrou
nivel3:
       mov bl, String_nomeLen2
       jmp encontrou
nivel1:
        mov bl, String_nomeLen
encontrou:
       mov cl, found
       cmp bl, cl
       jne return
       call   PROX_NIVEL
return:
        ret
VERIFICA_STRING ENDP
```

A função verifica_string, verifica em que nível está, e usa a dimensão da palavra atual para comparar se já encontrou (found) todas as letras da string desse nivel.

4. Temporizador

```
MOV     ax,Tempo_j          ;ax auxiliar para  ajustar o tempo de jogo
add     Tempo_j,1           ;incrementa tempo de jogo
MOV     bl, 10              ;CONVERTE EM DECIMAL
div     bl
add     al, 30h             ; Caracter Correspondente às dezenas
add     ah, 30h             ; Caracter Correspondente às unidades
MOV     String_TJ[0],al         ;
MOV     String_TJ[1],ah
GOTO_XY 57,0
MOSTRA  String_TJ
```

```
                cmp     Nivel_flag,0    ;verifica nivel1
                je      nivel1
                cmp     Nivel_flag,1    ;verifica nivel2
                je      nivel2
                cmp     Nivel_flag,2    ;verifica nivel3
                je      nivel3

nivel1:
                cmp     Tempo_j,101
                jb      continua
                call    DERROTA
                jmp     fim_horas
nivel2:
                MOV     String_TJ[4],39h
                MOV     String_TJ[5],35h
                MOV     String_TJ[6],20h
                cmp     Tempo_j,96
                jb      continua
                call    DERROTA
                jmp     fim_horas
nivel3:
                MOV     String_TJ[4],39h
                MOV     String_TJ[5],30h
                cmp     Tempo_j,91
                jb      continua
                call    DERROTA
                jmp     fim_horas
```

O temporizador é atualizado na variável Tempo_j sempre que passa 1 segundo tempo real.

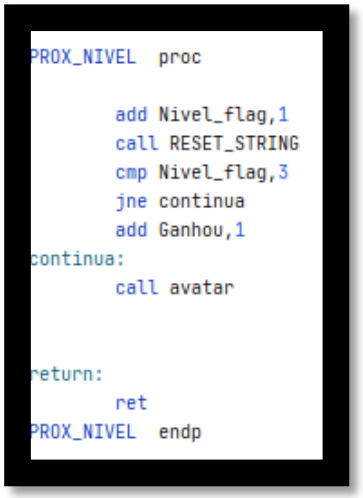
Com efeito, o Reg AX é usado como auxiliar para converter para decimal e mostrar o carater no ecrâ na String_tj

Além disto, compara o nivel em que está, atualiza o Timer na String_TJ e verifica se já passou o Tempo.

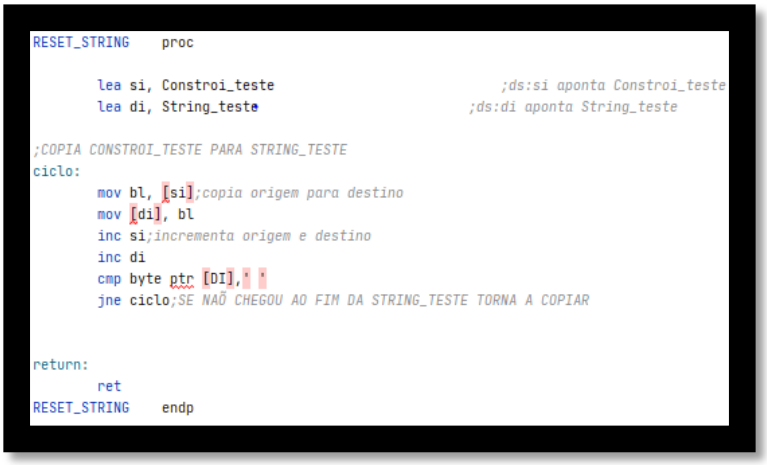
5. Passagem de Nível

A Função prox_nivel garante a passagem no nível com a incrementação da flag nível, resetando a string do Game e chamando o avatar. Por sua vez o avatar reseta o tempo, verifica o nivel , coloca o avatar na posição inicial (3,3).

Nota: A prox_nivel tambem incrementa a flag ganhou se já tiver completado o ultimo nivel.

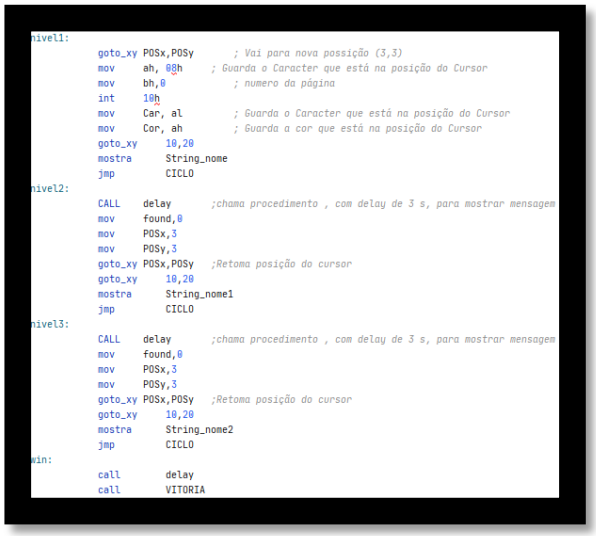


```
PROX_NIVEL  proc

        add Nivel_flag,1
        call RESET_STRING
        cmp Nivel_flag,3
        jne continua
        add Ganhou,1
continua:
        call avatar


return:
        ret
PROX_NIVEL  endp
```



```
RESET_STRING    proc

        lea si, Constroi_teste                  ;ds:si aponta Constroi_teste
        lea di, String_teste                    ;ds:di aponta String_teste

;COPIA CONSTROI_TESTE PARA STRING_TESTE
ciclo:
        mov bl, [si];copia origem para destino
        mov [di], bl
        inc si;incrementa origem e destino
        inc di
        cmp byte ptr [DI],' '
        jne ciclo;SE NAÕ CHEGOU AO FIM DA STRING_TESTE TORNA A COPIAR


return:
        ret
RESET_STRING    endp
```

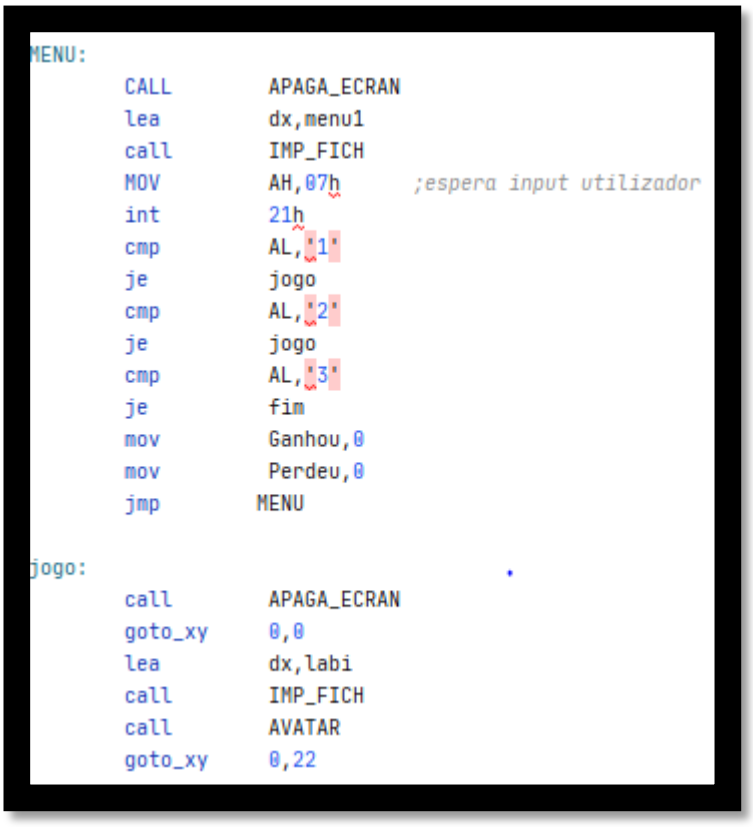
```
AVATAR  PROC
            mov     ax,0B800h
            mov     es,ax

            mov ax,Tempo_init
            mov Tempo_j,ax      ;RESET AO TIME

            call Trata_Horas

            cmp         Ganhou,1
            jne          nivel

nivel:
            cmp     Nivel_flag,0    ;verifica nivel1
            je      nivel1
            cmp     Nivel_flag,1    ;verifica nivel2
            je      nivel2
            cmp     Nivel_flag,2    ;verifica nivel3
            je      nivel3
            cmp     Nivel_flag,3    ;verifica vitoria
            je      win
```



```
nivel1:
            goto_xy POSx,POSy       ; Vai para nova possição (3,3)
            mov     ah, 08h     ; Guarda o Caracter que está na posição do Cursor
            mov     bh,0            ; numero da página
            int     10h
            mov     Car, al         ; Guarda o Caracter que está na posição do Cursor
            mov     Cor, ah         ; Guarda a cor que está na posição do Cursor
            goto_xy     10,20
            mostra      String_nome
            jmp         CICLO
nivel2:
            CALL    delay       ;chama procedimento , com delay de 3 s, para mostrar mensagem
            mov     found,0
            mov     POSx,3
            mov     POSy,3
            goto_xy POSx,POSy   ;Retoma posição do cursor
            goto_xy     10,20
            mostra      String_nome1
            jmp         CICLO
nivel3:
            CALL    delay       ;chama procedimento , com delay de 3 s, para mostrar mensagem
            mov     found,0
            mov     POSx,3
            mov     POSy,3
            goto_xy POSx,POSy   ;Retoma posição do cursor
            goto_xy     10,20
            mostra      String_nome2
            jmp         CICLO
win:
            call        delay
            call        VITORIA
```

6. Estrutura

Por último concluí-se com a estrutura Main do Trabalho que remete à impressão de um menu com 3 opções, nomeadamente 1.jogar , 2.Top10 e 3.sair. Deste modo, o programa espera pelo input do utilizador e impríme o Labirinto chamando o avatar para começar o jogo.



```
MENU:
        CALL        APAGA_ECRAN
        lea         dx,menu1
        call        IMP_FICH
        MOV         AH,07h      ;espera input utilizador
        int         21h
        cmp         AL,'1'
        je          jogo
        cmp         AL,'2'
        je          jogo
        cmp         AL,'3'
        je          fim
        mov         Ganhou,0
        mov         Perdeu,0
        jmp        MENU

jogo:
        call        APAGA_ECRAN
        goto_xy     0,0
        lea         dx,labi
        call        IMP_FICH
        call        AVATAR
        goto_xy     0,22
```