Relatório de Tecnologias e Arquiteturas de Computadores
"Ultimate Tic-Tac-Toe"

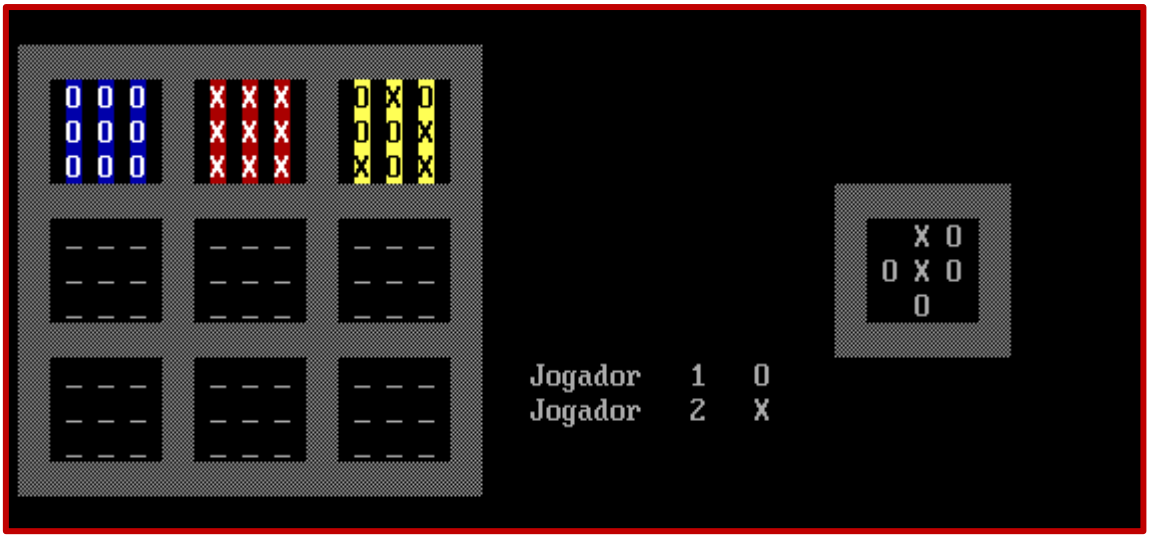

Imagem 1 – Interface do jogo Ultimate Tic Tac Toe

> *Resumo do Trabalho Prático*

No âmbito da cadeira de **Tecnologias e Arquiteturas de Computadores** foi-nos solicitada a realização de um trabalho prático que consiste num jogo "Ultimate Tic-Tac-Toe". Este jogo é composto por nove mini tabuleiros do jogo convencional e para além da vitória nos mini tabuleiros, o objetivo final é atingir a vitória no tabuleiro geral. Por fim, importa referir que o mesmo foi realizado com recurso à Linguagem Assembly e à aplicação utilizada nas aulas DOSBox.

> *Equipa de trabalho*

O trabalho prático foi realizado pelos seguintes alunos:

- Gonçalo Santiago_2022133003
- Mariana Maia_2022148315

> Objetivo do Projeto

Quando se inicia o jogo surge um menu composto por 2 opções: Jogar 1vs1 e Sair. Selecionando a primeira opção, aparece uma tela para escrever o nome dos jogadores, e ao selecionar a segunda opção, tal como o nome indica o jogo é fechado. Após inserir os nomes dos jogadores irá aparecer a tela do jogo.

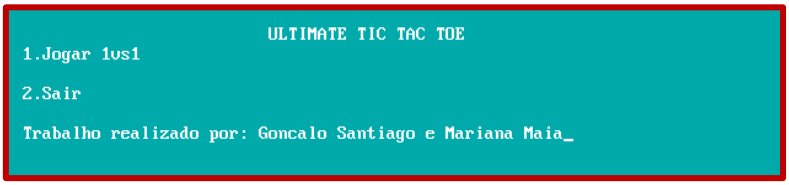

Imagem 2 – Menu do Jogo

Nesta tela são indicados os símbolos correspondentes a cada jogador, de quem é a próxima jogada e o vencedor do mini tabuleiro em que jogam (em caso de empate aparecerá que foi empate). Nos mini tabuleiros são marcadas as vitórias e empates com os símbolos correspondentes ao jogador vencedor e à cor do símbolo, sendo azul os O's e vermelho os X's, já no empate o tabuleiro fica com a cor amarela.

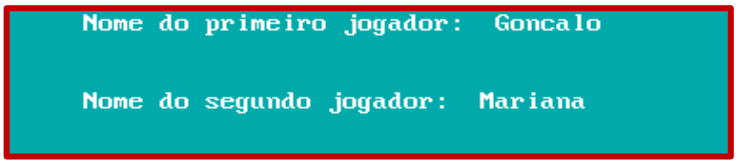

Imagem 3 – Tela para inserir nomes

É escolhida de forma aleatória o jogador que começa a jogar, bem como o símbolo que o mesmo tem atribuído. Por fim, depois de um dos jogadores ganharem no tabuleiro grande ou caso haja um empate final irá aparecer numa nova tela a mensagem correspondente.

Imagem 4 – Tela com jogador vencedor

> Funcionamento do Programa

A nível do funcionamento do programa, apenas é possível colocar "X" ou "O" onde é detetado um "_". Depois de se jogar são verificadas se existe combinação a nível de linha, coluna e diagonal.

```
ESTADO_JOGO proc
    xor si,si
    xor ax,ax
    xor cx,cx

    call VLINHA
    call VCOLUNA
    call VDIAGONAL
```

Imagem 5 – Funções de Verificação

Quando se verifica uma combinação, para além do mini tabuleiro ficar bloqueado, o mesmo muda de cor. Para tal as interrupções foram importantes para mudar a cor dos pixeis, bem como, uma bit color table.

```
    mov al,[di]+6;6
    cmp ah,al
    jne FIM
    je IGUAL

IGUAL:
    call MUDA_COR
    call ATUALIZA_ESTADO_JOGO
```

Imagem 6 – Quando se verifica uma combinação

HEX	BIN	COLOR
0	0000	black
1	0001	blue
2	0010	green
3	0011	cyan
4	0100	red
5	0101	magenta
6	0110	brown
7	0111	light gray
8	1000	dark gray
9	1001	light blue
A	1010	light green
B	1011	light cyan
C	1100	light red
D	1101	light magenta
E	1110	yellow
F	1111	white

Imagem 7 – Bit Color Table

Depois de se verificar vitória no jogo em geral é chamada uma função de mensagem final, neste caso o print abaixo, mostra quando ocorre uma vitória, é chamada a função “VenceJogo”.

```
MensagemFinal:
    call apaga_ecran
    mov cores,3fh
    call COR_ECRA
    goto_xy 28,12
    mov ah,09h
    lea dx, VenceJogo
    int 21h
```

```
VenceJogo       db 'Ganhou! Parabens Jogador $'
```

Imagem 8 – Mensagem Final

> *Conclusão*

Em suma, com a realização deste trabalho foi-nos possível expandir os nossos conhecimentos a nível lógico de Assembly, apesar da mesma ser uma linguagem de baixo nível bastante desafiadora.

Colocámos em prática tudo o que aprendemos durante o semestre e algo que foi bastante importante para o mesmo e que não foi muito abordado foram as interrupções para fazer certo tipo de execuções específicas, como por exemplo, mudar cor de pixel, colocar um cursor numa posição e muito mais.