

Relatório Trabalho Prático

Tecnologias e arquiteturas de computadores 2 semestre 1 lei - isec

Diogo Rafael Soares dos Santos – 2020131432

jOão miguel vicente ventura - 2022119090

Índice

[Introdução 2](#_Toc138024487)

[Descrição genérica das funções 2](#_Toc138024488)

[CalcBoardOffset 2](#_Toc138024489)

[CalcMoveOffset 4](#_Toc138024490)

[SetBxToCorner 5](#_Toc138024491)

[UpdateBoardWithMove 6](#_Toc138024492)

[FinalBoardIsPositionOcupied 7](#_Toc138024493)

[CheckForVictory 7](#_Toc138024494)

[UpdateFinalBoard 8](#_Toc138024495)

[PaintBoardColour 9](#_Toc138024496)

[DeclareWinner 10](#_Toc138024497)

# Introdução

O presente relatório descreve o desenvolvimento do jogo "Ultimate Tic-Tac-Toe" em linguagem Assembly para o processador 8086. O jogo é uma variante do clássico "Jogo do Galo", no qual dois jogadores realizam jogadas alternadas até que um deles vença ou ocorra um empate.

O objetivo principal deste projeto foi implementar o jogo em Assembly, aproveitando as capacidades e recursos do processador 8086. O jogo foi projetado para ser executado em um ambiente emulado, utilizando a interface de texto para exibir o tabuleiro e as informações relevantes aos jogadores.

Durante o desenvolvimento do projeto, foi necessário aplicar conceitos de programação em Assembly e explorar as funcionalidades do processador 8086 para criar uma experiência interativa e envolvente para os jogadores.

# Descrição genérica das funções

## CalcBoardOffset

Uma imagem com texto, captura de ecrã, documento

Descrição gerada automaticamente

O objetivo deste procedimento é calcular o deslocamento (offset) de um ponto num tabuleiro com base nas suas coordenadas de posição.

Na linha 277 carrega o valor de um byte localizado no endereço de memória “XPosCorners + 2” para o registo “bl”.

Na linha 278 move o valor da variável “POSx” para o registo “al”. Esta variável contém a posição X do ponto que será calculado.

A partir daqui, o código entra num bloco de condicionais usando instruções de comparação e saltos condicionais (jb – salta se o segundo valor for menor que o primeiro).

Na linha 282 compara o valor em “al” (POSx) com o valor em “bl” (byte carregado de “XPosCorners + 2”). Se o valor em “al” for menor, significa que o ponto não está na coluna da direita, e o código salta para a etiqueta “MiddleColumn”.

No bloco “MiddleColumn”, o código realiza operações semelhantes ao início, mas agora carregando um byte diferente de “XPosCorners” (provavelmente o byte em “XPosCorners + 1”). Em seguida, compara novamente “al” com “bl”. Se o valor em “al” for menor, significa que o ponto não está na coluna do meio, e o código salta para a etiqueta “FirstCol”.

No bloco “FirstCol”, o código atribui o valor 0 à variável “boardOffsetX”, indicando que o ponto está na primeira coluna.

Após a etiqueta “FirstCol”, o código salta para a etiqueta “YOffset”.

Na linha 298 o código carrega um byte de “YPosCorners + 2” para “bl” e move o valor de “POSy” para “al”, seguindo uma lógica semelhante à usada para calcular “boardOffsetX”.

O código segue uma estrutura semelhante para calcular “boardOffsetY”, verificando se o ponto está na última linha (“LastRow”), na linha do meio (“MiddleRow”) ou na primeira linha (“FirstRow”).

Após a etiqueta “EndCalc”, o código retorna da função usando a instrução “ret”.

Em resumo, este código recebe as posições X e Y de um ponto e calcula o deslocamento desse ponto num tabuleiro, armazenando os valores calculados nas variáveis “boardOffsetX” e “boardOffsetY”.

## CalcMoveOffset

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamente

O objetivo deste procedimento é calcular o deslocamento em X e Y com base na posição atual (POSx e POSy) e em algumas variáveis relacionadas ao tabuleiro (XPosCorners, YPosCorners, boardOffsetX e boardOffsetY). Os deslocamentos calculados são armazenados nas variáveis moveoffsetX e moveoffsetY para uso posterior no programa.

Na linha 326 e 327 usamos o operador XOR com o registo AX e BX para o zerar a variável por segurança.

Na linha 328 o valor de um byte armazenado na memória, no endereço indicado por XPosCorners, é movido para o registo BL. XPosCorners contém a posição da parede esquerda do tabuleiro, na linha seguinte o valor 2 é adicionado ao registo BL que serve para apontar para a posição do quadro.

Na linha 330 o valor de um byte armazenado na memória, no endereço indicado por XPosCorners, é movido para o registo BL.

De seguida o valor 10 é movido para o registo AH. Isto é usado posteriormente para multiplicar AL por 10. O valor de AL é adicionado ao valor de BL. É usado para calcular a diferença entre as posições dos cantos dos tabuleiros adjacentes em cada linha e voltamos a zerar o registo AX.

Nas linhas 338 até à 343 comparam o valor de AL com o valor de BL. Se AL for menor que BL, o código salta para a etiqueta IsFirstColumn. Se AL for maior que BL, o código salta para a etiqueta IsLastColumn.

Nas restantes verificações usamos uma lógica de jumps.

## SetBxToCorner

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamente

A macro recebe três parâmetros: "LargeBoard", "boardOffsetX" e "boardOffsetY". A macro calcula o endereço do canto do tabuleiro com base nos deslocamentos fornecidos. Os valores de "boardOffsetX" e "boardOffsetY" são multiplicados pelos fatores de diferença de localização e adicionados ao endereço inicial "LargeBoard". No final, o registo BX aponta para a posição do canto do tabuleiro na matriz de tabuleiros.

## UpdateBoardWithMove

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Este procedimento tem como objetivo atualizar a matriz do tabuleiro com uma jogada, baseando-se nos deslocamentos fornecidos e no jogador atual, para isso utiliza uma macro chamada "SetBxToCorner" para configurar o registo BX, apontando para a posição do canto do tabuleiro na matriz. Em seguida, os deslocamentos de movimento, denominados "moveOffsetX" e "moveOffsetY", são multiplicados pelos fatores de diferença de linhas e adicionados ao registo BX. Isso resulta no apontamento para a posição da jogada no tabuleiro.

O valor do registo AX é definido como zero e o byte do jogador atual é movido para o endereço de memória apontado por BX, atualizando assim o tabuleiro com a jogada efetuada. Por fim, a instrução "ret" é utilizada para retornar da chamada deste procedimento.

## FinalBoardIsPositionOcupied

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Esta macro verifica se uma posição específica do tabuleiro final está ocupada. Ela recebe as coordenadas "boardOffsetX" e "boardOffsetY". Através de cálculos de deslocamento, ela vai à posição correspondente no tabuleiro e verifica se está ocupada ou não, armazenando o resultado na variável "isOccupied".

## CheckForVictory

Uma imagem com texto, captura de ecrã, documento

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, documento

Descrição gerada automaticamente

O procedimento verifica se existe uma vitória num jogo de tabuleiro. Este procedimento percorre as linhas, colunas e diagonais da matriz "LargeBoard" à procura de sequências de símbolos iguais. Caso seja encontrada uma vitória, o procedimento chama a função "UpdateFinalBoard". Além disso, é também verificado se o tabuleiro está completamente preenchido. Este código verifica se houve uma vitória ou um empate num jogo de tabuleiro representado por uma matriz.

## UpdateFinalBoard

Uma imagem com texto, captura de ecrã, documento

Descrição gerada automaticamente

Procedimento que marca o fim de um jogo. Recebe como entrada o jogador vencedor e atualiza o tabuleiro final com essa informação. O código utiliza registos para manipular os endereços de memória e calcular as posições corretas no tabuleiro final. Além disso, define as coordenadas onde o jogador vencedor será impresso e realiza algumas verificações para determinar qual caráter deve ser escrito (X ou O) com base no jogador. Por fim, há uma chamada de função para alterar as cores do tabuleiro. No geral, o código atualiza o tabuleiro final com o jogador vencedor e realiza algumas operações de manipulação de memória e exibição de caracteres.

## PaintBoardColour

Uma imagem com texto, captura de ecrã, documento

Descrição gerada automaticamente

O procedimento recebe as coordenadas iniciais de pintura no tabuleiro e utiliza valores de variáveis para determinar a cor de preenchimento. Dependendo do valor da variável "\_player", a cor será definida como verde para vitória do jogador X, azul para vitória do jogador O ou amarelo para empate. Em seguida, o código utiliza um loop para preencher o tabuleiro com a cor definida. O procedimento termina após o loop.

## DeclareWinner

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamente

Este procedimento compara o valor da variável "\_player" com zero e um e, com base nessa comparação, imprime no ecrã o nome do jogador vencedor ou uma mensagem de empate. O código também inclui comandos para posicionar o cursor no ecrã e chamar uma função chamada "PaintFinalBoard" para pintar o tabuleiro final do jogo.