

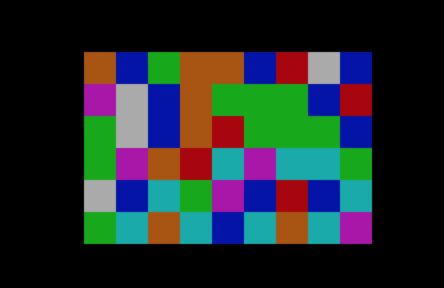
ToyBlast

Licenciatura

Engenharia Informática

TAC





Eduardo Barros – 21270614

Henrique Dias – 21260023

Introdução

Este trabalho foi realizado no âmbito da Unidade Curricular de Tecnologia e Arquitetura de Computadores. Refiro que o foco da disciplina bem como do trabalho é que os alunos se familiarizem com a linguagem Assembler no 8086 na vertente prática e que reforcem os conhecimentos da mesma.

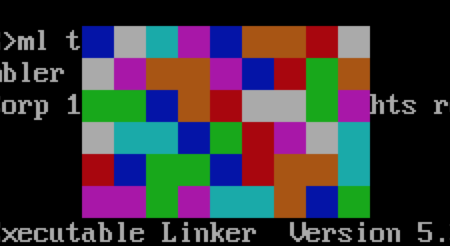
Faremos de inicio uma estruturação dos nossos objetivos para com este trabalho, dando assim um aspeto técnico do trabalho e a sua funcionalidade como programa.

Iremos apresentar uma breve explicação de cada tópico de avaliação desta atividade, bem como um pseudo algoritmo/opção de desenvolvimento para cada função que assim o exigiu. Será igualmente feito uma explicação/demonstração de algoritmo feito na linguagem com exemplos práticos

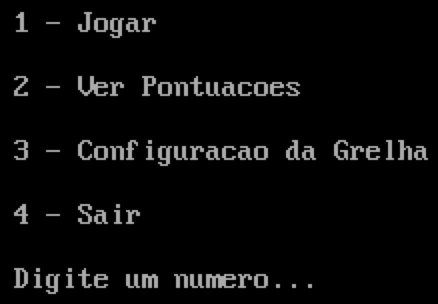
Por fim irá ser apresentada a conclusão deste trabalho, onde iremos falar das dificuldades encontradas, a nossa opinião sobre a realização desta atividade e outras considerações adjacentes e pertinentes.

Estruturação/Objetivos

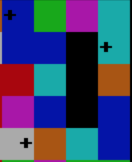
* Fazer o menu de jogo principal incluindo o ficheiro do tabuleiro com o ficheiro do cursor e o limite do cursor ao tabuleiro.

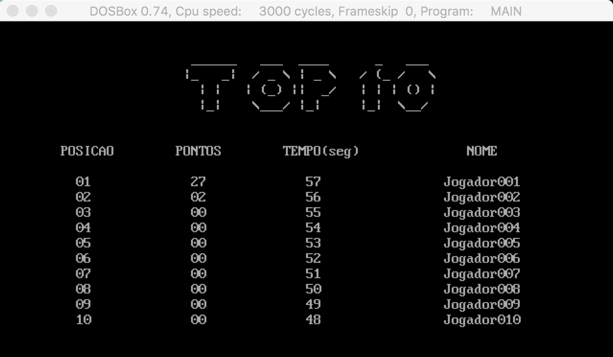


* Conhecendo desta forma a comunicação entre ficheiros e código adjacente.
* Criar menus de jogo.



* Conhecendo desta forma as interrupções (para imprimir mensagens, receber carateres, etc..) que possam vir a ser necessárias á criação deste jogo, as declarações de variáveis, a utilização de ascii art.
* Fazer na integra o algoritmo de funcionamento de jogo, bem como a configuração da grelha.



* Conhecendo desta forma a memória de vídeo associada a Assembler, a utilização de vetores, o salto de funções para funções, ciclos, comparações, a sintaxe na integra da linguagem.
* Guardar o top 10 ordenado, bem como a grelha previamente criada pelo utilizador.
* Dando uso a uma manipulação flexível de ficheiros em Assembler, os procedimentos necessários para tal.

ALGORITMOS/EXPLICAÇÕES

* Explosões:

1º Com a posição atual do cursor (quando foi ativado), calcular o índice para aceder ao array que tem as cores do tabuleiro

2º Colocar com o índice em AL, de forma a comparar com todas as posições adjacentes.

3º Calcular quais as posições adjacentes sendo que a posição da frente = (+2) e de baixo = (+18)

4º Caso seja igual, colocar a 0 (preto), e caso isto aconteça incrementar uma variável de pontuação

* Bónus:

1º Colocar com memória de vídeo dois carateres distintos em posições distintas

2º Incrementar uma variável bónus em cada explosão bem sucedida que é inicializada sempre a 0

3º Se a variável bónus for zero saltar para descer as peças, se não comparar em memória de vídeo a posição atual do cursor com o carater ascii de cada carater escolhido.

4º Comparar também todas as posições onde é possível haver uma explosão sendo que a posição da frente = (+2), e de baixo= (+160)

5º Comparar se a cor do carater é igual á cor explodida

5ª Caso seja o carater negativo, decrementar mais dois pontos do que é suposto

6º Caso seja o carater positivo, para aumentar o dobro adicionamos á pontuação o valor na variável bónus

* Descer peças:

1º Guardar a posição de y (numero de linha), e colocar em BX a ultima posição do tabuleiro de forma a percorrer o tabuleiro posição a posição para trás

2º Comparar posição com 0 (preto) e caso seja igual saltar para próximo passo, caso não seja, decrementar duas vezes de forma a passar para a posição anterior no array do tabuleiro

3º Fazer um ciclo a decrementar a posição de y, que acabe quando se encontrar a 0, se não for igual a 0, colocar o valor que se encontra em cima da posição atual (-18), na posição atual, decrementar 18 a BX de modo a ir para a posição de cima em cada fase do ciclo, se posição de y for igual a 0, saltar para próximo passo

4º Criar cor aleatória através do algoritmo já concebido, colocando a mesma na posição atual do BX

* Configurar a grelha:

1º Se for premida a tecla de configurar a grelha no menu, colocar uma variável editor a 1 e salta para o jogo normal

2º Se for premido a tecla space, apenas ativa a função de meter cor que calcula a posição atual do cursor e gera uma cor aleatória colocando-a no array tabuleiro com o índice BX

3º Se for premido a tecla S, usar os procedimentos já pré concebidos de ficheiros para guardar o array tabuleiro e coloca-lo mais na tarde na opção do menu “Carregar grelha”

* Limites do cursor: (trabalhando com array)

1º Fazer um goto para a posição inicial dentro tabuleiro

2º Compara al com 48h, se não for igual salta para passo seguinte

-Comparar posição da primeira linha(y=8) do tabuleiro com posição real da posição y, se for igual saltar para o ciclo do cursor, se não for igual decrementar duas vezes de forma a não passar o limite superior.

3º Compara al com 50h, se não for igual salta para passo seguinte

-Compara posição do limite inferior (+ 5 na posição de y do tabuleiro) com posição de y atual, se for igual saltar para o ciclo do cursor, se não for igual incrementar duas vezes a posição de y de forma a não passar o limite inferior.

4º Compara al com 4Bh, se não for igual salta para o passo seguinte

-Compara primeira posição da coluna(x=30) do tabuleiro com posição real da posição x, se for igual saltar para o ciclo do cursor, se não for igual, decrementar duas vezes de forma a não passar o limite á esquerda.

5º Compara posição do limite á direita (primeira posição da coluna + 16) com posição real da posição x, se for igual saltar para o ciclo do cursor, se não for igual, incrementar duas vezes.

* Tempo:

1º Usar interrupção 2Ch, da int 21h para ir buscar as horas e colocar e guardar os segundos numa variável

2º Compara segundos reais com segundos da leitura anterior, e se for diferente coloca os reais nos anteriores, senão acaba o jogo

3º Decrementa a variável de segundos

4º Compara variável que guarda os segundos, e compara com 0 para saber se já acabou

5º Para mostrar dividimos os segundos por 10 de forma a obtermos separadamente as dezenas e acrescentamos 30h para contrariar o código ascii, colocamos esses produtos num array para poder imprimir no ecrã de forma correta, fazer um goto para imprimir na posição desejada

* Pontuação:

1º Quando uma explosão for bem sucedida, incrementar uma variável inicializada a 0

2º Para mostrar dividimos a pontuação por 10 de forma a obtermos separadamente as dezenas e acrescentamos 30h para contrariar o código ascii, colocamos esses produtos num array para poder imprimir no ecrã de forma correta, fazer um goto para imprimir na posição desejada

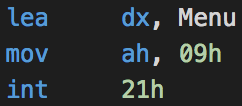
* Atualizar Top 10:

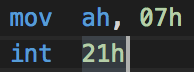
1º Correr um ciclo 10 vezes, para colocar na string previamente criada os pontos, tempo e os nomes em 3 vetores distintos

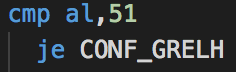
2º Guardar a pontuação feita pelo jogador bem como o seu tempo, comparando com a ultima posição do array que contem as pontuações todas

3º Se a pontuação temporária (ultima) for maior que a do ultimo colocado, colocar dados na ultima posição do array

4º Ir comparando com as pontuações de índice inferior e colocar caso seja maior que a ultima verificada

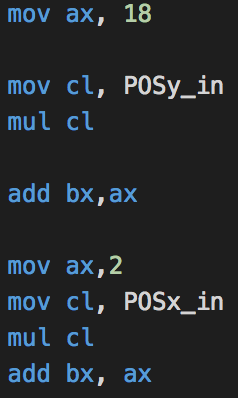
MENUS:

* Foram criados menus e vários submenus, usando strings (tamanho=byte), delimitadas por 10,13
* Imprimimos em ecrã através de um LEA (load efective adress) da string antes criada:
* Usando para isso a interrupção 09h (escreve string para STDOUT)
* Espera que o utilizador introduza um carater com interrupção 07h (leitura direta do STDIN)



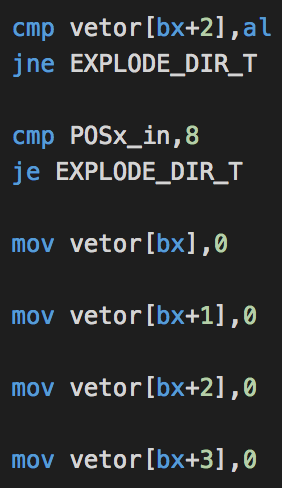
* Depois de introduzido, faz varias comparações para saber a qual submenu/instrução aceder (sendo que para o exemplo ‘51’ é o numero 3 em código ascii)

EXPLOSÕES:



* De linha para linha existe uma diferença de 18 (pois são 9 colunas sendo que cada uma constitui dois bytes) Então faremos uma multiplicação da posição Y do cursor por 18.
* De coluna para coluna existe uma diferença de 2 (pois cada coluna = 2 bytes) E faremos uma multiplicação da posição X do cursor por 2.
* Fazendo assim a conta bx = (posY\*18)+(posX\*2)

(Agora podemos saltar para cada posição que nos interessa, começando pela posição á frente)



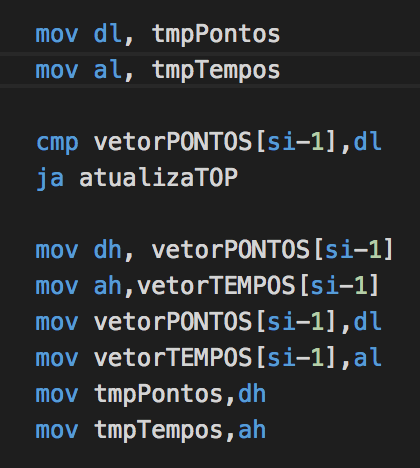
* Vetor[bx+2]=Posição 2 bytes á frente e AL = cor quando pressionado ENTER
* Se não for igual salta para a posição seguinte (direita-topo)
* Se estivermos na 8ºcoluna não explodimos á frente, dessa forma iria explodir a linha de baixo
* Colocamos o primeiro byte do meio a preto
* Colocamos o segundo byte do meio a preto
* Colocamos o primeiro byte da posição a frente a preto
* Colocamos o segundo byte da posição a frente a preto



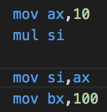
* Saltamos no fim para a posição seguinte (direita baixo)
* Sendo que a logica será a mesma para todas as posições desejadas (variando essas posições conforme a lógica anterior; Coluna->2 ; Linha->18)
* É de notar que nestes procedimentos incrementamos uma variável para ser mais tarde usada na pontuação, e uma para o bónus

TOP10:

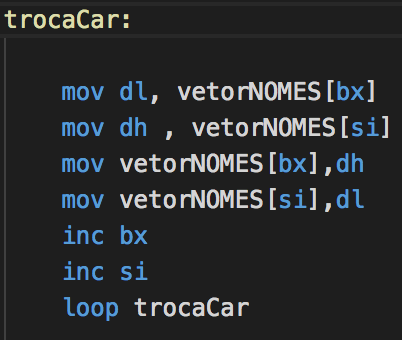
Atualiza TOP:



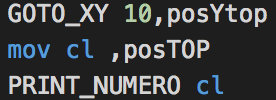
* Guardar a pontuação/tempo da ultima jogada
* Comparar com a ultima posição do array que contem as pontuações todas, se for maior=atualizaTOP (incrementa vetor e compara com posição acima)
* Se a pontuação temporária for menor que a do ultimo colocado colocar nome na ultima posição do array



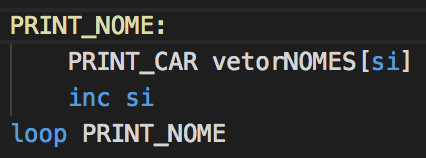
* Como vetorNOMES=100 bytes, sendo que cada nome terá 10 bytes, fazer MUL para aceder a cada nome
* Colocar BX=100 de forma a aceder ao ultimo nome do array



* Caso seja introduzida pontuação no vetorPONTOS, valores do vetorNOMES, fazem a troca do mesmo modo
* Este ciclo corre numero de vezes que tiver o CX

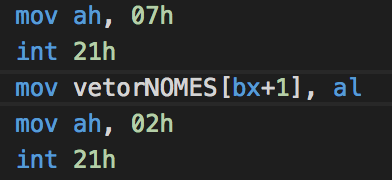


* Vai para a posição especifica da pontuação e usa macro de imprimir a números

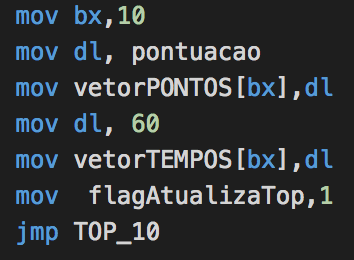


* Faz o mesmo com tempos e nomes sendo que neste caso usa a macro de imprimir caratéres

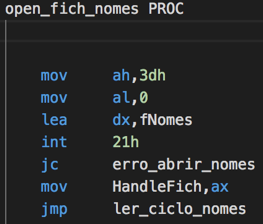
Recebe nome:

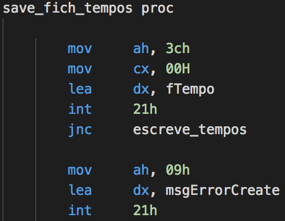


* Usar interrupção 07h para receber, guardar carater num array de dimensões definidas de forma ao nome só ter 10 bytes
* Usar interrupção 02h, para utilizador saber o que está a escrever

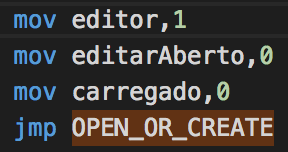


* Mal seja introduzido o nome, colocar a sua pontuação na posição vetorNOMES[10] = posição 11 (não aparece em ficheiro)

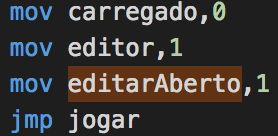


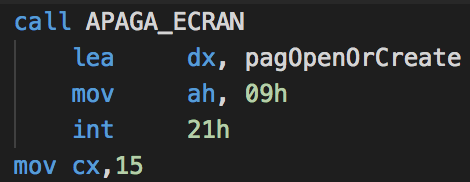
* Usar funções já pré concebidas para abrir, guardar e fechar pontuação/tempo/nome em 3 ficheiros de texto separados

CRIAR GRELHA:



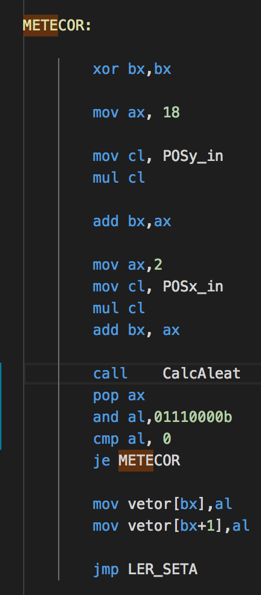
* Quando a configurar a grelha nova, editor=1, carregado=0
* Editor aberto a 0 porque não vamos editar um já editado



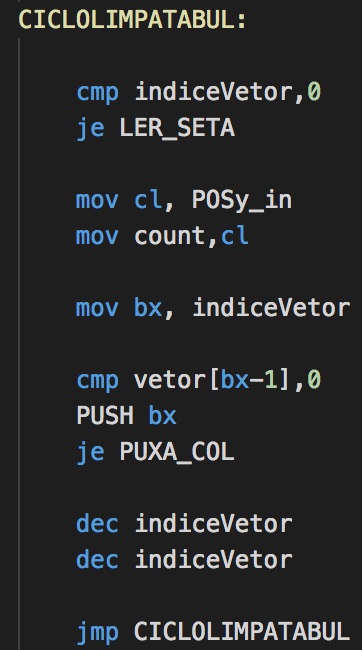
* Se for para editar um já editado, editarAberto=1
* Depois salta para menu jogar de forma a comparar qual das situações é
* Se for editarAberto -> OPEN\_OR\_CREATE
* Apresenta tabuleiro criado
* Se não faz código normal de edição



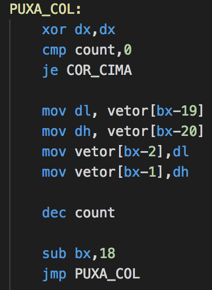
* No OPEN\_OR\_CREATE, lemos uma input com o procedimento, desta forma podemos guardar novo tabuleiro



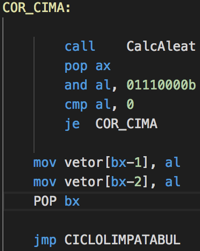
* Vai colocar o valor em BX, da atual posição do cursor, para dessa forma conseguirmos aceder á exata posição do array onde ele se encontra
* Colocando em bx = (posx\*2)+(posy\*18), como já averiguamos antes
* Chamamos a função de calculo aleatório para preencher a atual posição onde nos encontramos
* Verificamos se a cor gerada é preta para não a colocarmos
* Preenchemos o primeiro byte de atual posição
* Preenchemos o segundo byte de atual posição
* Voltamos ao ciclo

DESCER PEÇAS:

* IndíceVetor é inicializado com o valor final do tabuleiro, quando for 0 acaba o ciclo de percorrer o tabuleiro inteiro
* Guardar a posição atual de y (nº de linha) numa variável
* Comparamos a posição final com 0 (saber se é preta), se for saltar para passo seguinte \*PUSH BX, pois será usado mais á frente para índice de valores diferentes
* Decrementar duas vezes o índice para passar á posição atrás do vetor tabuleiro
* Se não for preto, repete o ciclo de forma a encontrar o preto



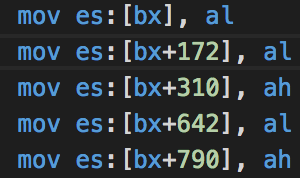
* Compara a variável do numero de linhas com 0, se for 0, salta para COR\_CIMA
* Coloca a cor que se encontra em cima (-19 sendo que começamos em -1) na cor em que se encontra vetor[bx-2]
* Repetimos duas vezes pois uma cor ocupa dois bytes
* Decrementa a variável do numero de linhas
* Subimos uma linha ao decrementar 18 a BX (índice do vetor)
* Repetimos o ciclo PUXA\_COL



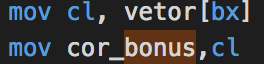
* Chamamos o procedimento CalcAleat
* Comparamos se a cor for preto, volto a repetir o procedimento
* Colocamos a cor nesse espaço do array tabuleiro
* Fazemos POP do BX de modo a obter o endereço

onde tínhamos ficado

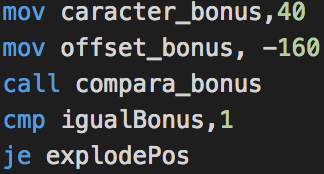
* Saltamos para a primeira fase

BÓNUS:

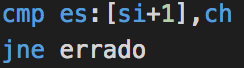
* Copiar string para AL/AH
* Colocar carater que desejarmos em posições especificas usando segmente ES para memória de vídeo



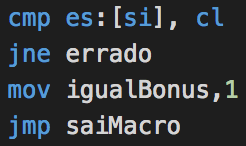
* Guarda cor explodida

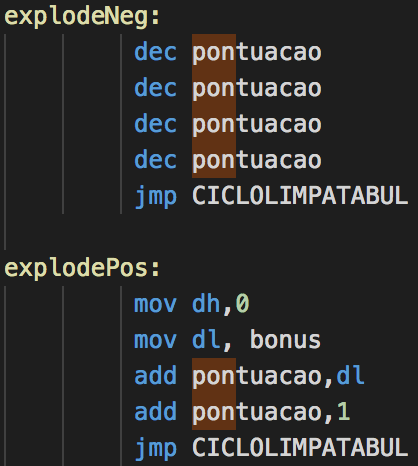


* Testa com os dois carateres(40/41)
* Bem como todas as posições desejadas (exemplo=posição em cima=-160)

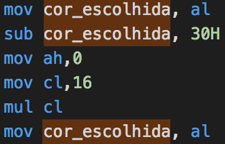


* Compara cor bónus (ch) com posição do carater
* Compara carater do bónus com sua posição e neste caso o bónus é acionado





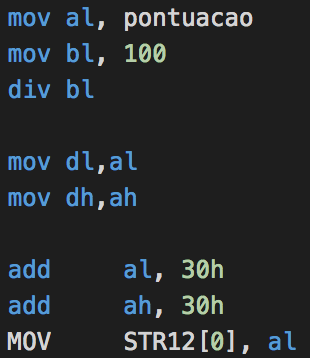
* Se for o carater negativo, decrementa 4 valores na pontuação supondo que incrementa originalmente na explosão
* Se for positivo, a variável bónus é adicionada á pontuação

****

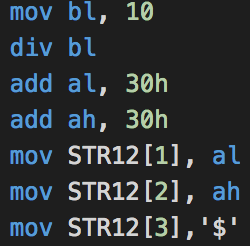
* Guarda a cor introduzida pelo utilizador
* Cor = valor decimal (menos 30h) \*16



* Guarda numero introduzido
* Usar macro para imprimir os valores

MOSTRAR PONTUAÇÃO:

* Dividir pontuação por 100, de forma a obtermos o algarismo das centenas separado
* Para colocar em código ascii da forma que queremos, devemos somar mais 30h ou 48, desta forma obtemos a pontuação em algarismos decimais
* Colocar numa string esses algarismos, sendo que o ultimo tem de terminar em $



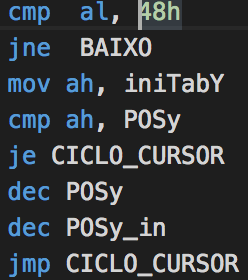
* Dividir pontuação por 10, de forma a obter algarismo das dezenas separado
* Usar mesmo método do exemplo anterior



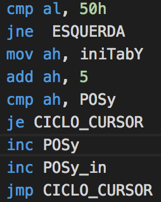
* Fazer um goto para a posição desejada e chamar a MACRO que usa a interrupção 09h para imprimir a string

LIMITES DO CURSOR:

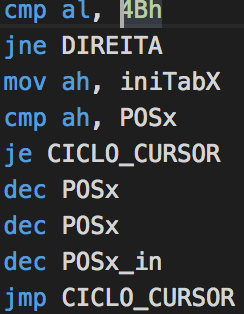
* Fazer um goto para a posição inicial dentro tabuleiro



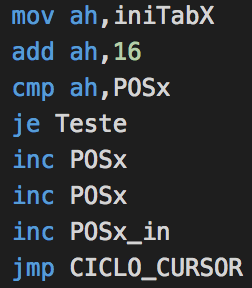
* Compara al(valor teclado) com seta, se não for igual passa para outro limite
* Compara (y=8) com posição real da posY
* Se for igual, continua sem ações a serem feitas
* Se não for igual decrementar de forma a não passar o limite superior.



* Compara al(valor teclado) com seta, se não for igual passa para outro limite
* Compara (y=13) com posição real da posY
* Se for igual, continua sem ações a serem feitas
* Se não for igual incrementa de forma a não passar o limite inferior.

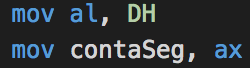


* Compara al(valor teclado) com seta, se não for igual passa para outro limite
* Compara (x=60) com posição real da posX
* Se for igual, continua sem ações a serem feitas
* Se não for igual decrementar de forma a não passar o limite esquerdo.

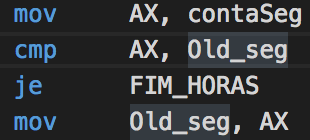


* Compara al(valor teclado) com seta, se não for igual passa para outro limite
* Compara (x=76) com posição real da posX
* Se for igual, continua sem ações a serem feitas
* Se não for igual incrementa de forma a não passar o limite direito.

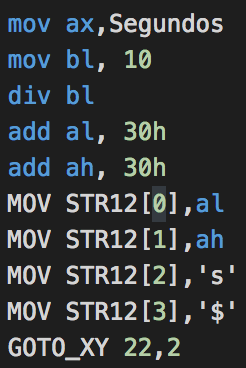
MOSTRAR TEMPO:

* Usa interrupção 2Ch, para obter horas atuais do sistema operativo
* Guarda os segundos dessa leitura na variável contaSeg





* Compara os segundos anteriores (Old\_seg), com os atuais (contaSeg)
* Caso sejam iguais acaba o tempo pois não alteram mais
* Se forem diferentes, atualiza Old\_seg



* Dividimos os segundos por 10 para obter as dezenas
* Acrescentamos 30h sair do código ascii,
* Colocamos num array STR12



* Imprimimos com macro previamente criada

Conclusão

Com a realização desta atividade deparámo-nos com a utilização rápida e cuidadosa que o Assembler nos pode trazer na vertente de desenvolvimento de aplicações primordiais, e a sua importância sendo um começo de enorme importância para o mundo de programação que existe hoje.

A programação nesta linguagem obrigou-nos a entender a baixo nível o funcionamento da programação, dando assim mais sentido á mesma.

Obtivemos algumas dificuldades relativamente á comparação recursiva, para concluir o primeiro bónus, em relação ao segundo, tivemos dificuldade em comparar a cor escolhida pelo utilizador com a cor explodida.

Existe também um bug, da qual não conseguimos identificar a sua origem, que consiste na aparição de meias peças a preto quando geradas aleatoriamente, este bug não segue nenhum tipo de padrão.

É de dar enfase também que a maior dificuldade tem como foco a compreensão inicial da sintaxe da linguagem, como o manejamento da mesma.

Em relação ao conhecimento da plataforma, melhoramos significativamente o traquejo da mesma, bem como algumas das suas funções e sintaxe.

Associado a esta aprendizagem damos enfase também á matéria de ‘sistemas de numeração’, pois sem ela seria impossível compreender a sintaxe de utilização desta linguagem.