ARQUITETURA DE COMPUTADORES

RELATÓRIO DO PROJETO – BONECO SALTADOR

GRUPO 19:

Duarte Miguel Montes Do Nascimento - 87527

Gonçalo Nuno Carrilho Gomes dos Santos - 87533

Pedro André Ferreira Teixeira - 87555

IST-TAGUSPARK

1. INTRODUÇÃO

O Projeto proposto pela disciplina engloba todos os conhecimentos de programação em *assembly* adquiridos ao longo do semestre, nomeadamente:

- Leitura de periféricos;
- Manipulação da memória (leitura e escrita);
- Criação de rotinas;
- Uso de processos cooperativos;
- Uso de interrupções;

Conceptualmente consiste em criar um jogo de plataformas cujo objetivo é controlar um boneco pelo ecrã, apanhar os "objetos" que dão pontos e evitar os que retiram pontos. Estes podem estar no topo de plataformas sobre o qual o boneco deve conseguir andar livremente.

Na secção 2 será descrito de forma detalhada o funcionamento do trabalho desenvolvido. Embora não tenhamos conseguido concretizar o projeto na totalidade, os objetivos que cumprimos englobam um pouco de todas as competências, pelo que são feitos comentários e descritas algumas dificuldades nas secções 3 e 4.

O código do projeto encontra-se na secção 5.

2. FUNCIONAMENTO INTERNO

2.1. Estrutura Geral

Como já foi referido, o nosso projeto não foi concluído, porém das funcionalidades requeridas, foram concretizadas as seguintes:

- O boneco anda e salta. Para que este processo funcione, é necessária a leitura do teclado por varrimento, para o qual é fundamental compreender o funcionamento de periféricos de entrada/saída; a limpeza do ecrã e uma função responsável por desenhar o boneco.

- O boneco cai e reconhece o "chão". A concretização desta funcionalidade exigiu a compreensão geral do conceito de interrupção.

A implementação de ambos estes pontos dependeu do aperfeiçoamento das noções de processos cooperativos e geral interação entre a memória, o PEPE e os periféricos.

As rotinas são descritas em detalhe na secção 2.4.

2.1.1. Hardware

De uma forma geral, em termos de Hardware foi utilizado o modelo proposto pela disciplina:

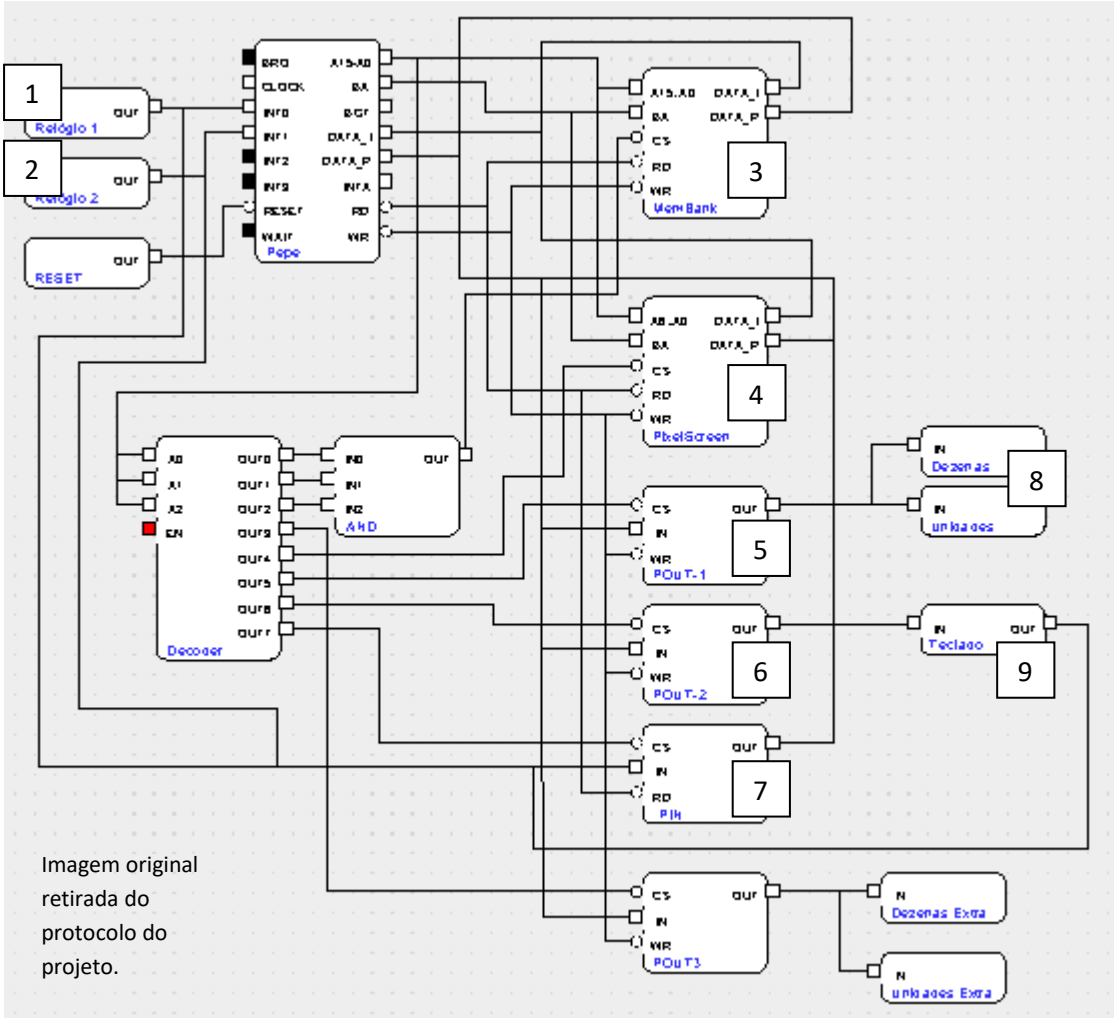


Imagem original retirada do protocolo do projeto.

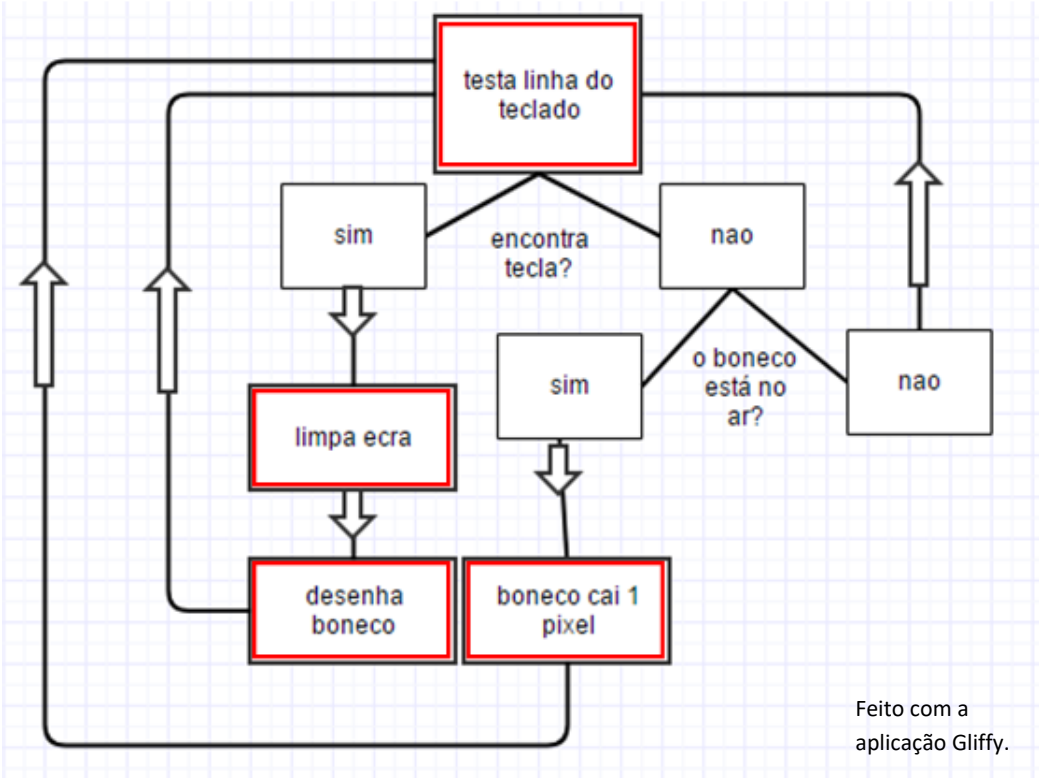
Onde se distinguem (da esquerda para a direita):

1. **Relógio 1**, ligado à rotina de interrupção 0 do PEPE, responsável por controlar a velocidade de descida do boneco;
2. **Relógio 2**, ligado à rotina de interrupção 1 do PEPE, responsável por controlar a velocidade de movimento das plataformas (não implementado);
3. **MemBank**, a memória;
4. **PixelScreen**, o ecrã de pixéis onde cada pixel corresponde a um bit, e cada linha corresponde a 4 bytes de memória, ocupando os endereços de 8000H a 807FH;
5. **POUT1**, o periférico de saída ao qual estão ligados os mostradores de pontos (8) (não implementado);
6. **POUT2**, o periférico de saída ao qual estão ligados os relógios que ligam às interrupções do PEPE (0 e 1) e os 4 bits de leitura de tecla do teclado (9);
7. **PIN**, o periférico de entrada que dita que linha é lida do teclado (9);

8. **Unidades** e **Dezenas**, dois Hexadisplays onde aparecem os pontos (não implementado);
9. **Teclado.**

2.1.2. Software e Processos Cooperativos

Como introdução a questões relacionadas com o Software, apresenta-se o seguinte fluxograma, onde cada caixa assinalada a vermelho é um processo levado a cabo por uma rotina principal, cada rotina será descrita com maior pormenor na secção 2.4.



Lista de rotinas principais e correspondência com os seus processos com base no fluxograma:

- **scan** – testa linha do teclado;
- **limpaecra** – limpa ecrã;
- **desenhaobjeto** – desenha boneco (esta rotina foi feita para poder desenhar qualquer objeto);
- **caiboneco** – boneco cai 1 pixel.

Através do fluxograma podemos perceber como comunicam os processos cooperativos, mediante uma flag que dita se foi premida alguma tecla. No entanto, não é apenas esta flag que coordena os processos, mas também a flag que deteta se o boneco está ou não no "chão".

2.2. Mapa de endereços

Para além dos endereços já definidos para os diferentes componentes de Hardware,

Dispositivo	Endereços
RAM (MemBank)	0000H a 5FFFH
POUT-3 (periférico de saída de 8 bits)	06000H
PixelScreen	8000H a 807FH
POUT-1 (periférico de saída de 8 bits)	0A000H
POUT-2 (periférico de saída de 8 bits)	0C000H
PIN (periférico de entrada de 8 bits)	0E000H

Retirado diretamente do protocolo do projeto.

na RAM (de 0000H a 5FFFH), o espaço de endereçamento foi gerido da seguinte forma:

- Instruções máquina –> **0000H – 0199H (0200H – onde guarda tecla, mas obviamente ajustável, caso alterações de código o exijam.)**;
- Stack para o SP efetuar o controlo de rotinas –> **1000H – 1200H**;
- Tabela de bytes, onde estão guardados cada byte a guardar para a correta impressão no ecrã de pixéis –> **1300H – … (< 1400H)**;
- Tabela de funções do teclado, onde estão tabeladas a influência de cada tecla nas coordenadas X e Y **–> 1400H – 143FH**;
- Tabela de pixéis para as figuras, onde estão as coordenadas iniciais ou finais de cada pixel de um dado objeto (cada tabela de figura e seu respetivo final são utilizados como argumentos pela rotina desenhaobjeto) –> **1440H – … (< 1700H)**;
- Tabela de exceções para rotinas de interrupção. –> **1700H – 1704H no máximo pois existem apenas 2 rotinas, 0 e 1**;
- Endereço da flag da rotina 0 (endereço_flag_rot0), Endereço da flag da rotina 1 (endereco_flag_rot1, não utilizada), Endereço da flag da tecla (endereço_flag_tcla) e Endereço onde é guardada a tecla clicada (endereço_tcla_clic), respetivamente -> **5000H, 5002H, 5004H, 0200H**.

2.3. Interrupções

Relembrando, no projeto existem 2 interrupções, a 0 e 1. A interrupção 0 está ligada ao relógio 0 que controla a velocidade de descida do boneco e a interrupção 1 ao relógio 1, que controla a velocidade de movimento das plataformas.

No nosso projeto apenas foi usada a interrupção 0, pois não concluímos o processo destinado à interrupção 1. Chamada de rot0, a interrupção 0 tem a simples função de detetar se o boneco está no ar, e se sim, ativar a flag guardada no endereço 5000H. Com esta flag ativa, o processo que faz o boneco cair 1 pixel depende agora só da flag da tecla.

2.4. Rotinas

O programa consiste num ciclo principal que corre uma vez para cada linha do teclado e contem todos os processos. De uma forma mais aprofundada passamos a descrever as quatro rotinas principais:

- **scan;**
- **limpaecra;**
- **desenhaobjeto;**
- **caiboneco;**

2.4.1. scan

Esta rotina tem como função ler uma das linhas do teclado e detetar se há alguma tecla a ser premida. Para isto importa a linha para o periférico de saída e exporta a coluna do periférico de entrada. Caso haja uma tecla premida, dado o facto de que a linha N e coluna M identificam-se respetivamente pela (N+1)-ésima potencia de 2 e (M+1)-ésima potencia de 2 (sendo 2^0 a primeira), a rotina leva a cabo um processo de conversão dos identificadores da linha/coluna para N e M ∈ {0,1,2,3} e converte-os para o hexadecimal correspondente à tecla premida através da formula :

$$\text{TECLA PREMIDA} = N \times 4M$$

Se nenhuma tecla for premida, passa á frente, saindo da rotina sem ativar a flag de tecla ou guardar a tecla em memória.

2.4.2. limpaecra

Como todas as outras, a partir dos dados recolhidos pela rotina scan, esta verifica se deve ou não correr. Caso a flag de tecla indique que foi clicada uma tecla, a rotina mete, em ciclo, todos os endereços de 8000H a 807FH com o valor 0000H, limpando assim o ecrã por completo.

2.4.3. desenhaobjeto

A rotina desenhaobjeto é talvez a mais complexa. Pode-se dizer que esta recebe dois argumentos, a tabela com as coordenadas dos pixéis da figura a desenhar e o fim dessa tabela, sempre nos registos R10 e R0, respetivamente. Assim que verifica que uma tecla foi premida, a primeira operação é atualizar as variáveis globais de desvio das coordenadas X e Y originais do objeto (estas podem ser alteradas para 0 temporariamente antes da chamada da rotina sempre que se pretende "imprimir" um objeto que não mude de posição com o teclado) com uma sub-rotina chamada "ajustes1", que utiliza as tabelas de funções de cada tecla para cada coordenada para fazer os ajustes. Por exemplo, sabe-se que a tecla 2 desloca o boneco tanto para cima como para a direita (como provavelmente varia entre projetos, as funções de cada tecla são descritas na secção 3), logo decrementa a variável y e incrementa a x, tendo em conta que o referencial segue a seguinte disposição:

Após ter o valor correto dos desvios de x e y, que são os mesmos para todos os pixéis do objeto, entra num ciclo que se vai repetir segundo o número de pixéis tabelado. O boneco tem 11 pixéis, logo o ciclo repete-se 11 vezes para o imprimir. Neste ciclo, são somados os desvios à sua posição original do pixel, recorrendo a uma mini rotina auxiliar chamada "ajustaregistos", e finalmente é chamada uma outra sub-rotina "print", que calcula exatamente onde deve imprimir o pixel com base num algoritmo que relaciona as coordenadas x e y resultantes das operações anteriores com o endereço de byte e o bit deste a acender. Para isso, estão tabelados os bytes para cada caso na Tabelab. Após finalmente imprimir o pixel, procura na tabela de pixéis do boneco qual é o próximo pixel a imprimir e repete o ciclo. Termina quando detetar o final da tabela de pixéis do boneco.

2.4.4 caiboneco

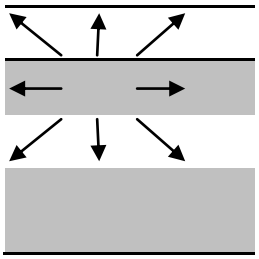
Para que esta rotina corra, são necessárias duas condições: Nenhuma tecla pode estar premida e a indicação de que o boneco não está no chão, dada pela rotina de interrupção 0. Se ambas as condições se verificarem, ativa a flag de tecla e coloca a tecla 9 no endereço que indica a tecla premida e chama a rotina limpaecra, seguida da rotina desenhaobjeto como se tivesse detetado a tecla 9 premida, que segundo a tabela de funções de tecla, incrementa a coordenada y, trazendo o boneco para baixo. No fim, repõe a flag de tecla para evitar bugs.

3. FUNCIONAMENTO EXTERNO – INSTRUÇÕES

3.1. Teclas e as suas funções

Para "jogar" o nosso jogo é preciso ter em atenção alguns detalhes, a começar pela função de cada tecla:

0	**1**	**2**	**3**
4	5	6	7
8	9	A	B
C	D	E	F



Numa versão experimental, criámos uma funcionalidade para a tecla 5 que imprimia um boneco com espada, mas tendo em conta que a limpeza de ecrã não passa despercebida, o resultado, embora funcional, não foi exatamente o que esperávamos e não foi incluído.

3.2. Alguns bugs:

O boneco "perde-se" se a tecla 9 for premida enquanto o boneco está no chão. Uma vez que se passa o limite de comparação, a rotina caiboneco vai correr sempre e o boneco nunca vai encontrar o chão. Se acontecer, basta carregar numa das teclas que leva o boneco para cima até este passar do "teto para o chão" outra vez. O mesmo bug verifica-se se o boneco andar contra a parede direita enquanto está no chão.

Se o boneco andar contra a parede esquerda ou contra o teto, desaparece para fora da tela, uma tentativa foi feita para tentar solucionar este bug e implementar mais uma funcionalidade, que consistia numa rotina extra que imprimia um sinal de "GAME OVER" se o boneco excedesse qualquer limite lateral, porem, não conseguimos que esta funcionasse até ao prazo de entrega do projeto pelo que foi excluída.

A rotina caiboneco corre periodicamente mesmo com a tecla premida. Para tentar solucionar este bug usámos breakpoints e simulámos o cenário. Com uma tecla premida em memória, a flag de tecla a 1 e a flag da rot0 a 1, verificamos linha a linha que a rotina caiboneco não corre assim que deteta que uma tecla foi premida, mesmo que a flag da rot0 esteja a 1, pelo que ficámos sem saber o porquê desta correr quando o programa é executado normalmente.

4. O FIM

4.1. Comentários e notas relevantes

Até bem perto da data de entrega, o nosso código não dependia da coordenação entre processos. Embora funcionasse e fosse bastante rápido, não era propriamente um "bom" código, pelo que sacrificámos alguma performance e alterámo-lo com vista a torná-lo mais organizado. Posteriormente uma tentativa de refazer o código de raiz com mais prática foi feita, com a falta de tempo, a ideia teve de ser abandonada. Porém, em detrimento da ideia anterior procuramos tentar resolver uma *feature* que ainda não tínhamos, as plataformas. Para isso criámos uma rotina que trabalha em parceria com a interrupção 1 que tem a simples função de imprimir a plataforma, movimentá-la e detetar quando esta deve inverter a direção do movimento (quando "bate" numa parede).No entando, devido à desorganização do código, não a conseguimos integrar com o resto do projeto a tempo, pelo que enviamos um ficheiro de "teste" (teste.asm) que pode ser de interesse por motivos de avaliação. Apesar de não apresentar o jogo funcional, é possível verificar o funcionamento da rotina de movimento da plataforma e um exemplo onde a rotina desenhaobjeto é usada para imprimir outro objeto noutra posição do ecrã (a plataforma).

4.2. Conclusão

Achamos que no geral o projeto foi bastante interessante e o seu desenvolvimento de certa forma divertido. No início tínhamos a sensação de que era talvez muito ambicioso, mas com a experiência apercebemo-nos que não só era acessível como até dava lugar à nossa criatividade. A forma como o projeto estava orientado permitiu uma gradual compreensão tanto da vertente de programação, como dos diferentes componentes de Hardware envolvidos e foi um testemunho direto da proximidade da linguagem *assembly* à máquina.

Infelizmente, por motivos exteriores e a um certo grau de desorganização não conseguimos completar o projeto. Achamos que o nosso principal erro foi termos perdido demasiado tempo com experimentalismos (i.e. tentativa de refazer o código e "GAME OVER") e detalhes menos importantes em vez de nos focarmos diretamente nos objetivos pedidos. No entanto, por um lado perdemos tempo mas por outro ganhámos competências extras que podem ser úteis para o futuro. Porque no fundo, um engenheiro não se cinge a águas navegadas, pois "com os erros é que se aprende".

5. CÓDIGO *ASSEMBLY*

Para leitura do ficheiro projeto.asm sugere-se o uso do Notepad++ por questões de indentação.

```
; CONSTANTES SIMBOLICAS

PIN                   EQU 0E000H     ; endereco de periferico de entrada
POUT                  EQU 0C000H     ; endereco de periferico de saida
screen_ini            EQU 8000H      ; primeira celula do ecra de pixeis
screen_out_fin        EQU 8080H      ; supremo dos enderecos do ecra
endereco_flag_rot0    EQU 5000H      ; endereco da flag da rotina 0
endereco_frag_rot1    EQU 5002H      ; endereco da flag da rotina 1
endereco_flag_tcla    EQU 5004H      ; endereco da flag indicadora de tecla premida
endereco_tcla_clic    EQU 0200H      ; endereco onde se guarda a tecla clicada
y_no_chao             EQU 001BH      ; desvio máximo de y (y no chao)

;*********************************************************************************

; STACK PARA SP

PLACE 1000H                          ; inicio da pilha
pilha:                TABLE 100H     ; criar a pilha
fim_pilha:                           ; fim da pilha

; TABELA DE BYTES - PARA FACILITAR A IMPRESSAO DE PIXEIS

PLACE 1300H                          ; inicio da tabela
Tabelab:              STRING 80H, 40H, 20H, 10H, 8H, 4H, 2H, 1H   ; tabela

; TABELAS DE FUNCOES DAS TECLAS (O que cada tecla faz a cada pixel do boneco)

PLACE 1400H
TabelaX:              WORD -1        ; 1400H
                      WORD 0         ; 1402H
                      WORD 1         ; 1404H
                      WORD 0         ; 1406H
                      WORD -1        ; 1408H
                      WORD 0         ; 140AH
                      WORD 1         ; 140CH
                      WORD 0         ; 140EH
                      WORD -1        ; 1410H
                      WORD 0         ; 1412H
                      WORD 1         ; 1414H
                      WORD 0         ; 1416H
                      WORD 0         ; 1418H
                      WORD 0         ; 141AH
                      WORD 0         ; 141CH
                      WORD 0         ; 141EH
TabelaY:              WORD -1        ; 1420H
                      WORD -1        ; 1422H
                      WORD -1        ; 1424H
                      WORD 0         ; 1426H
                      WORD 0         ; 1428H
                      WORD 0         ; 142AH
                      WORD 0         ; 142CH
                      WORD 0         ; 142EH
                      WORD 1         ; 1430H
                      WORD 1         ; 1432H
                      WORD 1         ; 1434H
                      WORD 0         ; 1436H
                      WORD 0         ; 1438H
                      WORD 0         ; 143AH
                      WORD 0         ; 143CH
                      WORD 0         ; 143EH
```

```
; TABELAS DE PIXEIS PARA AS FIGURAS (Endereços Pares - Coordenadas Y, Endereços Impares
- Coordenadas X)

PLACE 1440H                         ; inicio da tabela de boneco
Boneco:             STRING 00, 02, 01, 00, 01, 01, 01, 02, 01, 03, 01, 04, 02, 02, 03,
01, 03, 03, 04, 00, 04, 03          ; tabela de boneco
fim_boneco:                         ; fim da tabela de boneco

; TABELA DE ROTINAS

PLACE 1700H                                     ; inicio da tabela
tab:                WORD    rot0                ; rotina de interrupcao 0

;*********************************************************************************
; PROGRAMA

PLACE 0                                         ; inicio das instrucoes maquina
inicio:             MOV BTE, tab                ; iniciar a Tabela de excecoes
                    MOV SP, fim_pilha           ; iniciar o Stack Pointer
                    EI0                         ; permitir rotina de interrupcao 0
                    EI                          ; permitir rotinas de interrupcao

reset:              MOV R3, 8                   ; testar linha 8

ciclo1:             CALL scan                   ; chamar rotina scan
                    CALL limpaecra              ; chamar rotina limpaecra
                    MOV R10, Boneco             ; o que vai imprimir?
                    MOV R0, fim_boneco          ; o fim da tabela da figura
                    CALL desenhaobjeto          ; chamar rotina desenhaobjeto
                    CALL caiboneco              ; chamar rotina cai boneco
                    SHR R3, 1                   ; testar linha anterior
                    AND R3, R3                  ; ativar flag 0
                    JZ reset                    ; evitar testar linha que não existe
                    JMP ciclo1                  ; repetir para a linha anterior

;*********************************************************************************

scan:               PUSH R1                     ; salvar valor de R1
                    PUSH R2                     ; salvar valor de R2
                    PUSH R3                     ; salvar valor de R3
                    PUSH R4                     ; salvar valor de R4
                    PUSH R5                     ; salvar valor de R5
                    PUSH R6                     ; salvar valor de R6
                    PUSH R7                     ; salvar valor de R7
                    PUSH R8                     ; salvar valor de R8

                    MOV R0, endereco_tcla_clic  ; R0 – endereco para guardar tecla
                    MOV R1, PIN                 ; R1 – endereco do periferico in
                    MOV R2, POUT                ; R2 – endereco do periferico out

                    MOV R4, 0FH                 ; R4 - mascara
                    MOVB [R2], R3               ; R3 – linha, testar linha
                    MOVB R2, [R1]               ; R2 – coluna, receber coluna
                    AND R2, R4                  ; verificar se foi premida tecla
                    JZ retest_no_tcla           ; se nao, salta para o fim

gravar_linha:       SHR R3, 1                   ; transforma linha em 0, 1, 2 ou 3
                    AND R3, R3
                    JZ gravar_coluna
                    ADD R5, 1
                    JMP gravar_linha

gravar_coluna:      SHR R2, 1                   ; transforma coluna em 0, 1, 2 ou 3
                    AND R2, R2
                    JZ gravar_tecla
                    ADD R6, 1
                    JMP gravar_coluna
```

```
gravar_tecla:         SHL R5, 2                    ; transforma linha e coluna em tecla
                      ADD R6, R5
                      MOV [R0], R6

                      MOV R7, endereco_flag_tcla   ; R7 - endereco da flag de tecla
                      MOV R8, 1                    ; R8 = 1
                      MOV [R7], R8                 ; ativa a flag de tecla com 1
                      JMP retest_com_tcla          ; salta para o fim

retest_no_tcla:       MOV R7, endereco_flag_tcla   ; R7 -endereco de flag de tecla
                      MOV R8, 0                    ; R8 = 0
                      MOV [R7], R8                 ; ativa a flag de tecla com 1

retest_com_tcla:      POP R8                       ; recuperar valor de R8
                      POP R7                       ; recuperar valor de R7
                      POP R6                       ; recuperar valor de R6
                      POP R5                       ; recuperar valor de R5
                      POP R4                       ; recuperar valor de R4
                      POP R3                       ; recuperar valor de R3
                      POP R2                       ; recuperar valor de R2
                      POP R1                       ; recuperar valor de R1
                      RET                          ; retornar (fim de scan)

;*******************************************************************************

limpaecra:            PUSH R1                      ; salvar valor de R1
                      PUSH R2                      ; salvar valor de R2
                      PUSH R3                      ; salvar valor de R3

                      MOV R1, endereco_flag_tcla   ; R1 - endereco da flag de tecla
                      MOV R3, [R1]                 ; exportar valor
                      SUB R3, 1                    ; verificar se e 1 ou 0
                      JNZ fim_limpaecra            ; se for salta para o fim

                      MOV R3, 0H                   ; 0000H, o valor a colocar
                      MOV R1, screen_ini           ; do inicio do ecra
                      MOV R2, screen_out_fin       ; ao fim do ecra
ciclo2:               MOV [R1], R3                 ; colocar o valor na primeira
                      ADD R1, 2                    ; passar para a próxima celula
                      CMP R1, R2                   ; verificar se já chegou ao fim
                      JNZ ciclo2                   ; se nao, repetir o ciclo

fim_limpaecra:        POP R3                       ; recuperar valor de R3
                      POP R2                       ; recuperar valor de R2
                      POP R1                       ; recuperar valor de R1
                      RET                          ; retornar (fim de limpaecra)

;*******************************************************************************

desenhaobjeto:        PUSH R0                      ; salvar valor de R0
                      PUSH R1                      ; salvar valor de R1
                      PUSH R2                      ; salvar valor de R2
                      PUSH R3                      ; salvar valor de R3
                      PUSH R4                      ; salvar valor de R4
                      PUSH R5                      ; salvar valor de R5
                      PUSH R6                      ; salvar valor de R6
                      PUSH R7                      ; salvar valor de R7

                      MOV R2, endereco_flag_tcla   ; R2 - endereco da flag de tecla
                      MOV R2, [R2]                 ; exportar flag de tecla
                      SUB R2, 1                    ; verificar se esta a 0
                      JNZ fim_desenhaobjeto        ; se sim salta para o fim

                      MOV R2, screen_ini           ; R2 - inicio do ecra
                      MOV R7, Tabelab              ; R7 - inicio da tabela de bytes
```

```
                    CALL ajustes1               ; chamar rotina ajustes1
                    ADD R8, R6                  ; ajustar desvio de Y
                    ADD R9, R5                  ; ajustar desvio de X

                    MOV R3, R0                  ; detetar fim da figura
                    MOV R4, R10                 ; R4 - inicio da tabela de pixeis
                    MOV R5, R10                 ; R5 - inicio da tabela de pixeis
                    ADD R5, 1                   ; os Xs sao os elementos impares

ciclo3:             MOVB R0, [R4]               ; exportar coordenada Y do pixel
                    MOVB R1, [R5]               ; exportar coordenada X do pixel
                    CALL ajustaregistos         ; chamar rotina ajustregistos
                    CALL print                  ; chamar rotina print
                    ADD R4, 2                   ; passar para o proximo pixel (Y)
                    ADD R5, 2                   ; passar para o próximo pixel (X)
                    CMP R4, R3                  ; detetar se chegou ao ultimo pixel
                    JNZ ciclo3                  ; se nao, repete ciclo para proximo

fim_desenhaobjeto:  POP R7                      ; recuperar valor de R7
                    POP R6                      ; recuperar valor de R6
                    POP R5                      ; recuperar valor de R5
                    POP R4                      ; recuperar valor de R4
                    POP R3                      ; recuperar valor de R3
                    POP R2                      ; recuperar valor de R2
                    POP R1                      ; recuperar valor de R1
                    POP R0                      ; recuperar valor de R0
                    RET                         ; retornar (fim de desenhaboneco)

print:              PUSH R0                     ; salvar valor de R0
                    PUSH R1                     ; salvar valor de R1
                    PUSH R2                     ; salvar valor de R2
                    PUSH R3                     ; salvar valor de R3
                    PUSH R5                     ; salvar valor de R5
                    PUSH R6                     ; salvar valor de R6
                    PUSH R7                     ; salvar valor de R7

                    MOV R3, R1                  ; clonar R1
                    SHL R0, 2                   ; ajuste ao inicio do ecrã, com Y
                    ADD R2, R0                  ; ajustar endereco a Y
                    SHR R1, 3                   ; obter ajuste extra com base em X
                    ADD R2, R1                  ; ajustar endereco a X
                    SHL R1, 3                   ; X sem o resto da divisao por 8
                    SUB R3, R1                  ; obter o resto da divisao por 8
                    ADD R7, R3                  ; selecionar byte correto na Tabelab
                    MOVB R5, [R7]               ; exportar byte correto
                    MOVB R6, [R2]               ; R6 - byte original
                    OR R6, R5                   ; concatenar bytes
                    MOVB [R2], R6               ; produzir byte no ecra

                    POP R7                      ; recuperar valor de R7
                    POP R6                      ; recuperar valor de R6
                    POP R5                      ; recuperar valor de R5
                    POP R3                      ; recuperar valor de R3
                    POP R2                      ; recuperar valor de R2
                    POP R1                      ; recuperar valor de R1
                    POP R0                      ; recuperar valor de R0
                    RET                         ; retornar (fim de print)

ajustes1:           PUSH R4                     ; salvar valor de R4
                    PUSH R7                     ; salvar valor de R7
                    MOV R7, endereco_tcla_clic  ; R7 - endereco guardada tecla
                    MOV R4, [R7]                ; exportar tecla clicada
                    MOV R5, TabelaX             ; R5 - inicio da TabelaX
                    MOV R6, TabelaY             ; R6 - inicio da TabelaY
                    SHL R4, 1                   ; ajustat tecla para palavras
                    ADD R5, R4                  ; somar tecla ao inicio de TabelaX
                    ADD R6, R4                  ; somar tecla ao inicio de TabelaY
```

```
                    MOV R5, [R5]                ; exportar operação correta para X
                    MOV R6, [R6]                ; exportar operação correta para Y
                    POP R7                      ; recuperar valor de R7
                    POP R4                      ; recuperar valor de R4
                    RET                         ; retornar (fim de ajustes1)

ajustaregistos:     ADD R0, R8                  ; ultimo ajuste da coordenada X
                    ADD R1, R9                  ; ultimo ajuste da coordenada Y
                    RET                         ; retornar (fim ajusteregistos)

;*********************************************************************************

caiboneco:          PUSH R0                     ; salvar o valor de R0
                    PUSH R1                     ; salvar o valor de R1
                    PUSH R2                     ; salvar o valor de R2
                    PUSH R3                     ; salvar o valor de R3
                    PUSH R4                     ; salvar o valor de R4
                    PUSH R5                     ; salvar o valor de R5
                    PUSH R6                     ; salvar o valor de R6
                    PUSH R7                     ; salvar o valor de R7

test_tcla:          MOV R0, endereco_flag_tcla  ; R0 - endereco da flag de tecla
                    MOV R0, [R0]                ; exportar flag de tecla
                    SUB R0, 1                   ; verificar se esta a 1
                    JZ fim_caiboneco            ; se sim salta para o fim

test_rot0:          MOV R0, endereco_flag_rot0  ; R0 - endereco da flag de rotina 0
                    MOV R0, [R0]                ; exportar flag de rotina
                    SUB R0, 1                   ; verificar se esta 0 (esta no chao)
                    JNZ fim_caiboneco           ; se sim, salta para o fim

continue:           MOV R2, endereco_tcla_clic  ; R2 - endereco da tecla clicada
                    MOV R3, 9                   ; simular tecla 9
                    MOV [R2], R3                ; importar tecla 9
                    MOV R2, endereco_flag_tcla  ; R2 - endereco de flag da tecla
                    MOV R3, 1                   ; simular que uma tecla foi premida
                    MOV [R2], R3                ; importar 1 para a flag
                    MOV R10, Boneco             ; o que vai imprimir?
                    MOV R0, fim_boneco          ; o fim da tabela da figura
                    CALL limpaecra              ; chamar rotina limpaecra
                    CALL desenhaobjeto          ; chamar rotina desenhaobjeto
                    MOV R3, 0                   ; simular que não a tecla premida
                    MOV [R2], R3                ; exportar 0 para a flag
                    MOV R0, endereco_flag_rot0  ; R3 - endereco de flag de rotina 0
                    MOV R1, 0                   ; repor flag a 0
                    MOV [R0], R1                ; importar 0 para a flag

fim_caiboneco:      POP R7                      ; recuperar valor de R7
                    POP R6                      ; recuperar valor de R6
                    POP R5                      ; recuperar valor de R5
                    POP R4                      ; recuperar valor de R4
                    POP R3                      ; recuperar valor de R3
                    POP R2                      ; recuperar valor de R2
                    POP R1                      ; recuperar valor de R1
                    POP R0                      ; recuperar valor de R0
                    RET                         ; retornar (fim de caiboneco)

;*********************************************************************************

rot0:               PUSH R1                     ; salvar valor de R1
                    PUSH R2                     ; salvar valor de R2
                    PUSH R3                     ; salvar valor de R3
                    PUSH R4                     ; salvar valor de R4
```

```
                    MOV R3, y_no_chao           ; R3 - ajuste max de y (no chao)
                    CMP R8, R3                  ; o boneco esta no chao?
                    JZ close                    ; se sim, saltar para o fim

                    MOV R1, 1                   ; R1 com valor de flag
                    MOV R2, endereco_flag_rot0  ; R2 - endereco de flag de rotina 0
                    MOV [R2], R1                ; importar 1 para a flag de rotina 0


close:              POP R4                      ; recuperar valor de R4
                    POP R3                      ; recuperar valor de R3
                    POP R2                      ; recuperar valor de R2
                    POP R1                      ; recuperar valor de R1
                    RFE                         ; retornar (fim da interrupcao 0)
```