Faculdade de Ciências Naturais

Departamento de Ciências da computação

RELATÓRIO DE ARQUITECTURA DE COMPUTADORES

Dr.

_____

João Da Costa

Luanda 2025/2026

Integrantes do Grupo

- Luciano Manuel Ferreira
- Mena Miguel Tambilo
- Simão Kindanda Pedro
- Valdemiro dos Santos Quental

1- INTRODUÇÃO

O presente relatório, visa apresentar alguns pontos fundamentais do projecto de Arquitectura de Computadores, mencionadas durante o desenvolvimento do mesmo.

1.1-Contexto do Trabalho.

Neste trabalho que na qual foi nos dado para a criação de um jogo, este mesmo que consiste em um **Pac-Man,** que se locomovem no ecrã e tem por objectivo apanhar os quatros objectos que se encontram nos quatros cantos, e ao mesmo tempo não ser apanhado pelos fantasmas representados por uma estrela. O jogo termina quando é apanhado pelos fantasmas (perde o jogo) ou quando apanha os quatros objectos dos cantos (ganha o jogo). A pontuação final mostrada num dos displays de 7 segmentos, mede a qualidade do jogador e representa o tempo em segundos até apanhas os quatros objectos.

1.2-Objectivos

- Criar um Jogo Pac-Man que apanha quatro Objectos;
- Implementar os conceitos de programação em linguagem assembly, os periféricos e as interrupções, dados em Arquictetura de Computadores para a criação do jogo.

1.3- Restrições e Observações sobre aspectos importantes de projeto.

O nosso projecto tem algumas restrições concernente as funcionalidades das teclas do teclado, só as teclas:

0- Para movimentar-se para diagonal superior esquerdo;

1- Para movimentar-se para cima tecla ;

3- Para movimentar-se para diagonal superior direita;

4- Para movimentar-se para a esquerda;

6- Movimentar-se para a direita;

9- Movimentar-se para baixo;

8- Para movimentar-se para diagonal inferior esquerda;

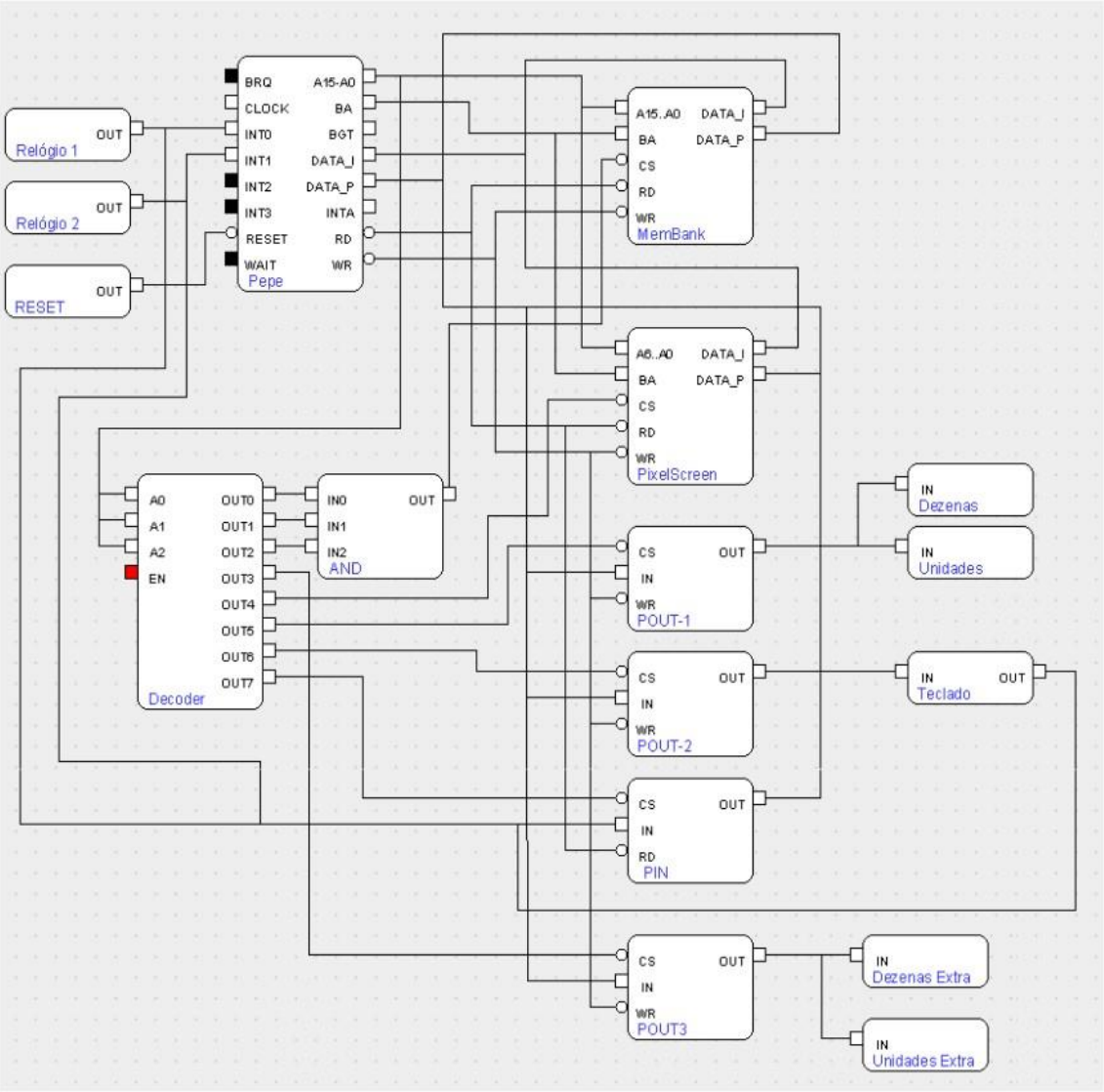
A- Para movimentar-se para diagonal inferior direita;

2- Para iniciar;

2- CONCEPÇÃO E IMPLEMENTAÇÃO.

2.1- Estrutura Geral

Parte hardware, que apresenta o diagrama de blocos.



2.1.1. Mapa de endereçamento escolhido

O mapa de endereços (em que os dispositivos podem ser acedidos pelo PEPE) é o

Dispositivo	Endereçamento
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

2.1.2. Comunicação entre processos

Foi usado processos cooperativos para suportar as diversas ações do jogo, aparentemente simultâneas.

Os processos são os seguintes:

- **Teclado** (varrimento e leitura das teclas);
- **Boneco** (para controlar os movimentos do boneco que é controlado pelo teclado);
- **Objetos** (para controlar as ações dos Objetos);
- **Controlo** (para tratar das teclas de começar e terminar)

2.1.3. Variáveis de Estado

Para cada processo, recomenda-se:

- Um estado 0, de inicialização. Assim, (re)começar um jogo é pôr todos os processos no estado 0, em que cada um inicializa as suas próprias variáveis. Fica mais modular;

- Planeie os estados (situações estáveis) em que cada processo poderá estar. O processamento (decisões a tomar, ações a executar) é feito ao transitar entre estados;

- Veja que variáveis são necessárias para manter a informação relativa a cada processo, entre invocações sucessivas (posição, direção, modo, etc.)

2.1.4. Interrupções

Interrupções utilizou-se para contar o tempo do contador.

2.1.5. Rotinas

- Rotinas de ecrã (desenhar/apagar um pixel numa dada linha e coluna, desenhar/apagar objetos/Boneco – represente os objetos pelas coordenadas de um determinado pixel (canto superior esquerdo, por exemplo, e desenhe-os relativamente às coordenadas desse pixel);

- Teclado (um varrimento de cada vez, inserido num ciclo principal onde as rotinas que implementam processos vão ser chamadas);

- Boneco (desenho do Boneco controlado pelo teclado, com deslocamentos de um pixel por cada tecla carregada no teclado);

- Objetos (desenho dos Objetos com movimento a 45 graus e que aparecem numa posição aleatória);

CONCLUSÃO

O objetivo inicial desse projeto é de criar um jogo usando todos os conceitos aprendidos na área de Arquitetura de Computador, no que se refere a linguagem Assembly.

Concluímos com uma satisfação de ter concretizado e alcançado tudo do que foi proposto inicialmente, acrescentando-se o método de movimentar na diagonal.

Link respositório github: https://github.com/Vquental/pacman/

4. Código assembly

```
; ==============================================================================
; 1. DEFINIÇÕES DE CONSTANTES E ENDEREÇOS
; ==============================================================================
BUFFER                EQU 4000H
PIN                   EQU 0E000H
POUT                  EQU 0C000H
ECRA_INICIO           EQU 8000H
DISPLAYS              EQU 0A000H

NUM_FANTASMAS         EQU 2
VELOCIDADE_FANTASMA   EQU 30
VELOCIDADE_PACMAN     EQU 10
MAX_PONTOS            EQU 4
TEMPO_MAXIMO          EQU 99
CICLOS_POR_SEGUNDO    EQU 1000

FANTASMA_SIZE         EQU 12
OBJETO_VAR_SIZE       EQU 6
TERMINADOR_SPRITE     EQU 0FFFEH

; ------------------------------------------------------------------------------
; 2. TABELAS DE PIXELS (SPRITES)
; ------------------------------------------------------------------------------
PLACE 2000H

tabela_pacman: ; **7 Pixels, Forma 'C' boca para a direita (CORRIGIDO)**
    ; Coordenadas Relativas (Linha, Coluna). Centro (0,0) e boca (0,1) não desenhados.
    WORD -1
    WORD -1     ; Cima Esquerda
    WORD -1
    WORD 0      ; Cima Meio
    WORD -1
    WORD 1      ; Cima Borda da boca
    WORD 0
    WORD -1     ; Meio Esquerda
    WORD 1
    WORD -1     ; Baixo Esquerda
    WORD 1
    WORD 0      ; Baixo Meio
```

```
    WORD 1
    WORD 1      ; Baixo Borda da boca
    WORD TERMINADOR_SPRITE

tabela_fantasma: ; 5 Pixels, Forma 'X'
    WORD 0
    WORD 0
    WORD -1
    WORD -1
    WORD -1
    WORD 1
    WORD 1
    WORD -1
    WORD 1
    WORD 1
    WORD TERMINADOR_SPRITE

tabela_objeto: ; 5 Pixels, Forma '+'
    WORD 0
    WORD 0
    WORD -1
    WORD 0
    WORD 1
    WORD 0
    WORD 0
    WORD -1
    WORD 0
    WORD 1
    WORD TERMINADOR_SPRITE

; --------------------------------------------------------------------------------
; 3. TABELA DE BITS E 7 SEGMENTOS
; --------------------------------------------------------------------------------
PLACE 2200H
tabela_bits:
    WORD 0001H
    WORD 0002H
    WORD 0004H
    WORD 0008H
    WORD 0010H
    WORD 0020H
```

```
    WORD 0040H
    WORD 0080H

tabela_7seg:
    WORD 003FH ; 0
    WORD 0006H ; 1
    WORD 005BH ; 2
    WORD 004FH ; 3
    WORD 0066H ; 4
    WORD 006DH ; 5
    WORD 007DH ; 6
    WORD 0007H ; 7
    WORD 007FH ; 8
    WORD 006FH ; 9

; ------------------------------------------------------------------------
; 4. VARIÁVEIS DO JOGO
; ------------------------------------------------------------------------
PLACE 3000H

estado_jogo:      WORD 0
pontuacao:      WORD 0
tempo_decorrido: WORD 0
tempo_contador:  WORD 0
contador_pacman:  WORD 0

objetos_cantos: ; Linha, Coluna, Estado (x4)
    WORD 2
    WORD 2
    WORD 0

    WORD 2
    WORD 29
    WORD 0

    WORD 29
    WORD 2
    WORD 0

    WORD 29
    WORD 29
```

```
    WORD 0

caixa_linha:      WORD 12 ; Usado para lógica de colisão, mas não para desenho
caixa_coluna:     WORD 12 ; Usado para lógica de colisão, mas não para desenho
caixa_largura:    WORD 7
caixa_altura:     WORD 7

linha_pacman:     WORD 25
coluna_pacman:    WORD 16
direcao_pacman:   WORD 0

fantasmas: ; Linha, Coluna, Direção, Estado, Contador (x2)
    WORD 15
    WORD 15
    WORD 0
    WORD 0
    WORD 0      ; Fantasma 1 (Na Caixa)

    WORD 7
    WORD 24
    WORD 0
    WORD 2
    WORD 0      ; Fantasma 2 (Livre)

tecla_anterior: WORD 0
tecla_atual:    WORD 0

; ------------------------------------------------------------------------
; 5. PILHA
; ------------------------------------------------------------------------
PLACE 4000H
pilha:    TABLE 200H
fim_pilha:

; ========================================================================
; 6. PROGRAMA PRINCIPAL
; ========================================================================
PLACE 0

inicio:
    MOV SP, fim_pilha
```

```
    CALL inicializar_tudo

menu_principal:
    CALL mostrar_menu
    CALL ler_teclado_menu
    CALL atualizar_displays

    MOV R0, estado_jogo
    MOV R0, [R0]
    MOV R1, 1
    CMP R0, R1
    JZ jogar_novo_jogo

    JMP menu_principal

jogar_novo_jogo:
    CALL inicializar_partida
    CALL desenhar_cenario
    CALL desenhar_pacman
    CALL desenhar_fantasmas

loop_jogo:
    CALL atualizar_tempo
    CALL ler_teclado_jogo
    CALL mover_pacman
    CALL mover_fantasmas
    CALL verificar_colisoes
    CALL verificar_vitoria
    CALL atualizar_displays

    MOV R0, estado_jogo
    MOV R0, [R0]
    MOV R1, 1
    CMP R0, R1
    JNZ fim_jogo_check

    JMP loop_jogo

fim_jogo_check:
    CALL mostrar_tela_final
    CALL esperar_reinicio
```

```
    JMP inicio

; ========================================================================
; 7. ROTINAS DE E/S DO SISTEMA (TECLADO, DISPLAY)
; ========================================================================

inicializar_tudo:
    CALL limpar_ecra
    CALL inicializar_variaveis
    RET

inicializar_variaveis:
    PUSH R0
    PUSH R1

    MOV R0, 0
    MOV R1, estado_jogo
    MOV [R1], R0
    MOV R1, pontuacao
    MOV [R1], R0
    MOV R1, tempo_decorrido
    MOV [R1], R0
    MOV R1, tempo_contador
    MOV [R1], R0
    MOV R1, tecla_anterior
    MOV [R1], R0
    MOV R1, tecla_atual
    MOV [R1], R0

    MOV R1, contador_pacman
    MOV [R1], R0

    POP R1
    POP R0
    RET

ler_teclado_menu:
    PUSH R0
    PUSH R1
    PUSH R2
    PUSH R3
```

```
    MOV R1, 1
    MOV R2, POUT
    MOV R3, PIN

    MOVB [R2], R1
    MOVB R0, [R3]

    CMP R0, 0
    JZ fim_ler_menu

    MOV R1, 4
    CMP R0, R1
    JNZ fim_ler_menu

    MOV R0, 1
    MOV R1, estado_jogo
    MOV [R1], R0

fim_ler_menu:
    POP R3
    POP R2
    POP R1
    POP R0
    RET

ler_teclado_jogo:
    PUSH R0
    PUSH R1
    PUSH R2
    PUSH R3
    PUSH R4
    PUSH R6

    MOV R2, POUT
    MOV R3, PIN
    MOV R4, 0

varrer_teclado:
    CMP R4, 4
    JGE fim_loop_varredura
```

```
    MOV R6, tabela_bits
    ADD R6, R4
    ADD R6, R4
    MOV R1, [R6]

    MOVB [R2], R1
    MOVB R0, [R3]

    CMP R0, 0
    JNZ tecla_pressionada_jogo

    ADD R4, 1
    JMP varrer_teclado

fim_loop_varredura:
    MOV R0, 0
    MOV R1, tecla_atual
    MOV [R1], R0
    JMP fim_ler_jogo

tecla_pressionada_jogo:
    MOV R2, 1
    CMP R0, R2
    JZ tecla_coluna_0
    MOV R2, 2
    CMP R0, R2
    JZ tecla_coluna_1
    MOV R2, 4
    CMP R0, R2
    JZ tecla_coluna_2
    MOV R2, 8
    CMP R0, R2
    JZ tecla_coluna_3
    JMP fim_ler_jogo

tecla_coluna_0: MOV R0, 0
    JMP calcular_tecla
tecla_coluna_1: MOV R0, 1
    JMP calcular_tecla
tecla_coluna_2: MOV R0, 2
```

```
    JMP calcular_tecla
tecla_coluna_3: MOV R0, 3

calcular_tecla:
    PUSH R6
    ADD R4, R4
    ADD R4, R4
    POP R6
    ADD R0, R4

    MOV R1, 1
    CMP R0, R1
    JZ tecla_cima
    MOV R1, 4
    CMP R0, R1
    JZ tecla_esq
    MOV R1, 6
    CMP R0, R1
    JZ tecla_dir
    MOV R1, 9
    CMP R0, R1
    JZ tecla_baixo

    JMP fim_ler_jogo

tecla_cima: MOV R0, 1
    JMP definir_direcao
tecla_baixo: MOV R0, 2
    JMP definir_direcao
tecla_esq: MOV R0, 3
    JMP definir_direcao
tecla_dir: MOV R0, 4

definir_direcao:
    MOV R1, direcao_pacman
    MOV [R1], R0
    MOV R1, tecla_atual
    MOV [R1], R0

fim_ler_jogo:
    POP R6
```

```
    POP R4
    POP R3
    POP R2
    POP R1
    POP R0
    RET

converter_para_7seg:
    PUSH R1

    MOV R1, tabela_7seg
    ADD R0, R0
    ADD R1, R0
    MOV R0, [R1]

    POP R1
    RET

atualizar_displays:
    PUSH R0
    PUSH R1
    PUSH R2
    PUSH R3
    PUSH R6

    MOV R0, tempo_decorrido
    MOV R0, [R0]

    MOV R1, 10
    DIV R0, R1

    MOV R2, R1
    MOV R3, R0

    MOV R0, R2
    CALL converter_para_7seg
    MOV R1, DISPLAYS
    MOVB [R1], R0

    MOV R0, R3
    CALL converter_para_7seg
```

```
    MOV R6, R0
    ADD R6, R6
    ADD R6, R6
    ADD R6, R6
    ADD R6, R6

    MOV R1, DISPLAYS
    MOVB R2, [R1]

    MOV R3, 000FH
    AND R2, R3

    OR R6, R2

    MOVB [R1], R6

    POP R6
    POP R3
    POP R2
    POP R1
    POP R0
    RET

; ================================================================================
; 8. ROTINAS DE MOVIMENTO E LÓGICA DO JOGO
; ================================================================================

inicializar_partida:
    PUSH R0
    PUSH R1
    PUSH R2
    PUSH R3
    CALL limpar_ecra

    MOV R0, 1
    MOV R1, estado_jogo
    MOV [R1], R0

    MOV R0, 0
    MOV R1, pontuacao
```

```
    MOV [R1], R0
    MOV R1, tempo_decorrido
    MOV [R1], R0
    MOV R1, tempo_contador
    MOV [R1], R0
    MOV R1, direcao_pacman
    MOV [R1], R0

    MOV R0, 25
    MOV R1, linha_pacman
    MOV [R1], R0
    MOV R0, 16
    MOV R1, coluna_pacman
    MOV [R1], R0

    MOV R1, fantasmas
    MOV R2, 0
reset_fantasmas:
    CMP R2, NUM_FANTASMAS
    JZ fantasmas_resetados

    ; --- Fantasma 1 (R2 = 0): DENTRO DA CAIXA (L15, C15), ESTADO 0 (Na Caixa) ---
    CMP R2, 0
    JNZ fantasma2_init_config

    MOV R0, 15        ; Linha inicial: L15 (Centro da Caixa)
    MOV [R1], R0
    MOV R0, 15        ; Coluna inicial: C15 (Centro da Caixa)
    MOV [R1+2], R0
    MOV R3, 0         ; Direção: 0
    MOV [R1+4], R3
    MOV R3, 0         ; **ESTADO 0: Na Caixa**
    MOV [R1+6], R3
    JMP end_fantasma_init

    ; --- Fantasma 2 (R2 = 1): L7, C24, ESTADO 2 (Livre) ---
fantasma2_init_config:
    MOV R0, 7         ; Linha inicial: L7
    MOV [R1], R0
    MOV R0, 24        ; Coluna inicial: C24
    MOV [R1+2], R0
```

```
    MOV R3, 0          ; Direção: 0
    MOV [R1+4], R3
    MOV R3, 2          ; **ESTADO 2: Livre**
    MOV [R1+6], R3

end_fantasma_init:
    MOV R3, 0
    MOV [R1+8], R3 ; Contador (Offset 8)

    PUSH R3
    MOV R3, FANTASMA_SIZE
    ADD R1, R3
    POP R3

    ADD R2, 1
    JMP reset_fantasmas

fantasmas_resetados:
    MOV R1, objetos_cantos
    MOV R2, 0
reset_objetos:
    CMP R2, 4
    JZ objetos_resetados

    MOV R0, 0
    MOV [R1+4], R0 ; Reinicia Estado (0 = ativo)

    PUSH R3
    MOV R3, OBJETO_VAR_SIZE
    ADD R1, R3
    POP R3

    ADD R2, 1
    JMP reset_objetos

objetos_resetados:
    POP R3
    POP R2
    POP R1
    POP R0
    RET
```

```
atualizar_tempo:
    PUSH R0
    PUSH R1
    PUSH R2

    MOV R0, tempo_contador
    MOV R1, [R0]
    ADD R1, 1
    MOV [R0], R1

    MOV R2, CICLOS_POR_SEGUNDO
    CMP R1, R2
    JLE fim_atualizar_tempo

    MOV R1, 0
    MOV [R0], R1

    MOV R0, tempo_decorrido
    MOV R1, [R0]
    ADD R1, 1
    MOV [R0], R1

    MOV R2, TEMPO_MAXIMO
    CMP R1, R2
    JLE fim_atualizar_tempo

    MOV R1, TEMPO_MAXIMO
    MOV [R0], R1

fim_atualizar_tempo:
    POP R2
    POP R1
    POP R0
    RET

mover_pacman:
    PUSH R0
    PUSH R1
    PUSH R2
    PUSH R3
```

```
PUSH R4
PUSH R5
PUSH R6
PUSH R7

; --- 1. CHECK DE VELOCIDADE (DELAY) ---
MOV R0, contador_pacman
MOV R1, [R0]
ADD R1, 1
MOV [R0], R1

MOV R2, VELOCIDADE_PACMAN
CMP R1, R2
JLT fim_mover_pac_skip

MOV R1, 0
MOV [R0], R1
; --- FIM CHECK DE VELOCIDADE ---

MOV R0, direcao_pacman
MOV R0, [R0]
MOV R1, 0
CMP R0, R1
JZ fim_mover_pac_skip

; Variáveis
MOV R1, linha_pacman
MOV R2, coluna_pacman
MOV R3, [R1] ; Linha Atual (L_antiga)
MOV R4, [R2] ; Coluna Atual (C_antiga)

MOV R5, R3 ; R5 = Linha Nova
MOV R6, R4 ; R6 = Coluna Nova

; Tenta calcular a nova posição (R5, R6)
MOV R0, direcao_pacman
MOV R0, [R0]

MOV R7, 1
CMP R0, R7
JZ tentar_cima
```

```
    MOV R7, 2
    CMP R0, R7
    JZ tentar_baixo

    MOV R7, 3
    CMP R0, R7
    JZ tentar_esq

    MOV R7, 4
    CMP R0, R7
    JZ tentar_dir

    JMP fim_mover_pac_skip

tentar_cima:
    SUB R5, 1
    JMP verificar_movimento
tentar_baixo:
    ADD R5, 1
    JMP verificar_movimento
tentar_esq:
    SUB R6, 1
    JMP verificar_movimento
tentar_dir:
    ADD R6, 1
    JMP verificar_movimento

verificar_movimento:
    ; R5 = Nova Linha, R6 = Nova Coluna

    ; 1. Verificação das Bordas da Tela (0 e 31)
    MOV R7, 0
    CMP R5, R7
    JLT movimento_invalido
    CMP R6, R7
    JLT movimento_invalido
    MOV R7, 31
    CMP R5, R7
    JGT movimento_invalido
    CMP R6, R7
```

```
    JGT movimento_invalido

    ; 2. Verificação da Caixa (L12-L18, C12-C18)

    ; L_min = 12, L_max = 18
    MOV R7, 12
    CMP R5, R7
    JLT pos_fora_da_caixa
    MOV R7, 18
    CMP R5, R7
    JGT pos_fora_da_caixa

    ; R5 está em [12, 18]. Checar C_min = 12, C_max = 18
    MOV R7, 12
    CMP R6, R7
    JLT pos_fora_da_caixa
    MOV R7, 18
    CMP R6, R7
    JGT pos_fora_da_caixa

    ; Posição (R5, R6) está DENTRO ou na BORDA da caixa [12-18] x [12-18].

    ; Checar se é a PORTA (L=12, C=15)
    MOV R7, 12
    CMP R5, R7
    JNZ movimento_invalido

    MOV R7, 15
    CMP R6, R7
    JNZ movimento_invalido

    ; É a porta (L12, C15). Movimento Válido.
    JMP movimento_valido_check

pos_fora_da_caixa:
    ; Posição está fora do limite da caixa. Movimento Válido.
    JMP movimento_valido_check

movimento_invalido:
    JMP fim_mover_pac_skip
```

```
movimento_valido_check:
    ; O movimento é VÁLIDO.

    ; 1. APAGA O SPRITE ANTIGO
    CALL apagar_pacman

    ; 2. Atualiza a posição na memória
    MOV [R1], R5
    MOV [R2], R6

    ; 3. Desenha no novo local
    CALL verificar_coleta
    CALL desenhar_pacman

fim_mover_pac_skip:
    POP R7
    POP R6
    POP R5
    POP R4
    POP R3
    POP R2
    POP R1
    POP R0
    RET

mover_fantasmas:
    PUSH R0
    PUSH R1
    PUSH R2
    PUSH R3
    PUSH R4
    PUSH R5
    PUSH R6
    PUSH R7

    MOV R4, fantasmas
    MOV R0, 0

mover_fant_loop:
    CMP R0, NUM_FANTASMAS
    JGE fim_mover_fantasmas
```

```
; --- Check de Velocidade ---
MOV R1, [R4+8]
ADD R1, 1
MOV [R4+8], R1
MOV R2, VELOCIDADE_FANTASMA
CMP R1, R2
JLT proximo_fantasma_move_skip

MOV R1, 0
MOV [R4+8], R1

CALL apagar_fantasma

; --- Lógica de Movimento ---
MOV R1, [R4+6]

MOV R7, 0
CMP R1, R7
JZ fantasma_na_caixa
MOV R7, 1
CMP R1, R7
JZ fantasma_saindo

; MOVIMENTO LIVRE (Estado 2)
MOV R1, linha_pacman
MOV R1, [R1]
MOV R2, coluna_pacman
MOV R2, [R2]

MOV R3, [R4]
MOV R5, [R4+2]

; Lógica de perseguição simplificada atual (Prioriza Vertical, depois Horizontal)
MOV R7, 16
AND R7, R3

CMP R7, 0
JLE mover_horizontal_livre

CMP R3, R1
```

```
    JLT fantasma_abaixo
    JZ mover_horizontal_livre
    JMP fantasma_acima

fantasma_acima:
    SUB R3, 1
    JMP actualizar_pos_fantasma_check

fantasma_abaixo:
    ADD R3, 1
    JMP actualizar_pos_fantasma_check

mover_horizontal_livre:
    CMP R5, R2
    JLT fantasma_direita
    JZ actualizar_pos_fantasma_check
    JMP fantasma_esquerda

fantasma_esquerda:
    SUB R5, 1
    JMP actualizar_pos_fantasma_check

fantasma_direita:
    ADD R5, 1
    JMP actualizar_pos_fantasma_check

    ; --------------------------------------------------------------------

fantasma_na_caixa:
    MOV R3, [R4]
    MOV R5, [R4+2]

    MOV R6, 12
    CMP R3, R6
    JLE fantasma_saindo_inicio

    ; 1. Move para a Coluna Central (15)
    MOV R6, 15
    CMP R5, R6
    JNZ fantasma_na_caixa_move_H
```

```
    ; 2. Se estiver no centro, move 1 passo para cima
    SUB R3, 1
    JMP actualizar_pos_fantasma_check

fantasma_na_caixa_move_H:
    MOV R6, 15
    CMP R5, R6
    JGT fantasma_na_caixa_move_esq
    JLT fantasma_na_caixa_move_dir
    JMP actualizar_pos_fantasma_check

fantasma_na_caixa_move_esq:
    SUB R5, 1
    JMP actualizar_pos_fantasma_check

fantasma_na_caixa_move_dir:
    ADD R5, 1
    JMP actualizar_pos_fantasma_check

fantasma_saindo_inicio:
    MOV R1, 1
    MOV [R4+6], R1
    JMP redesenhar_fantasma

fantasma_saindo:
    MOV R3, [R4]
    MOV R5, [R4+2]

    MOV R6, 11
    CMP R3, R6
    JLE fantasma_livre_inicio

    ; Move 1 passo para cima (do 12 -> 11)
    SUB R3, 1
    JMP actualizar_pos_fantasma_check

fantasma_livre_inicio:
    MOV R1, 2
    MOV [R4+6], R1
    JMP redesenhar_fantasma
```

```
actualizar_pos_fantasma_check:
    MOV [R4], R3
    MOV [R4+2], R5

redesenhar_fantasma:
    PUSH R4
    MOV R1, [R4]
    MOV R2, [R4+2]
    MOV R3, tabela_fantasma
    CALL desenhar_objeto
    POP R4

proximo_fantasma_move_skip:
    ; ...

proximo_fantasma:
    MOV R6, FANTASMA_SIZE
    ADD R4, R6
    ADD R0, 1
    JMP mover_fant_loop

fim_mover_fantasmas:
    POP R7
    POP R6
    POP R5
    POP R4
    POP R3
    POP R2
    POP R1
    POP R0
    RET

; ================================================================================
; 9. ROTINAS DE COLISÃO
; ================================================================================

verificar_coleta:
    PUSH R0
    PUSH R1
    PUSH R2
    PUSH R3
```

```
    PUSH R4
    PUSH R5

    MOV R1, linha_pacman
    MOV R1, [R1]
    MOV R2, coluna_pacman
    MOV R2, [R2]

    MOV R3, objetos_cantos
    MOV R0, 0

verificar_objs:
    CMP R0, MAX_PONTOS
    JGE fim_verificar_coleta

    MOV R4, [R3+4]
    CMP R4, 0
    JNZ proximo_obj_verificar

    MOV R4, [R3]
    CMP R1, R4
    JNZ proximo_obj_verificar

    MOV R4, [R3+2]
    CMP R2, R4
    JNZ proximo_obj_verificar

    ; COLISÃO DETETADA
    MOV R4, 1
    MOV [R3+4], R4

    MOV R4, pontuacao
    MOV R1, [R4]
    ADD R1, 1
    MOV [R4], R1

    ; APAGA O OBJETO COLETADO
    PUSH R0
    PUSH R1
    PUSH R2
    PUSH R3
```

```
    MOV R1, [R3]
    MOV R2, [R3+2]
    MOV R3, tabela_objeto
    CALL apagar_objeto
    POP R3
    POP R2
    POP R1
    POP R0

proximo_obj_verificar:
    MOV R5, OBJETO_VAR_SIZE
    ADD R3, R5
    ADD R0, 1
    JMP verificar_objs

fim_verificar_coleta:
    POP R5
    POP R4
    POP R3
    POP R2
    POP R1
    POP R0
    RET

verificar_colisoes:
    PUSH R0
    PUSH R1
    PUSH R2
    PUSH R3
    PUSH R4
    PUSH R5
    PUSH R6

    MOV R1, linha_pacman
    MOV R1, [R1]
    MOV R2, coluna_pacman
    MOV R2, [R2]

    MOV R4, fantasmas
    MOV R0, 0
```

```
verificar_col_fant:
    CMP R0, NUM_FANTASMAS
    JGE fim_verificar_colisoes

    MOV R3, [R4+6]
    MOV R6, 2
    CMP R3, R6
    JNZ proximo_fant_col

    MOV R3, [R4]
    MOV R5, [R4+2]

    CMP R1, R3
    JNZ proximo_fant_col
    CMP R2, R5
    JNZ proximo_fant_col

    MOV R0, 2
    MOV R1, estado_jogo
    MOV [R1], R0
    JMP fim_verificar_colisoes

proximo_fant_col:
    MOV R6, FANTASMA_SIZE
    ADD R4, R6
    ADD R0, 1
    JMP verificar_col_fant

fim_verificar_colisoes:
    POP R6
    POP R5
    POP R4
    POP R3
    POP R2
    POP R1
    POP R0
    RET

verificar_vitoria:
    PUSH R0
    PUSH R1
```

```
    MOV R0, pontuacao
    MOV R0, [R0]
    MOV R1, MAX_PONTOS
    CMP R0, R1
    JLT fim_verificar_vitoria

    MOV R0, 3
    MOV R1, estado_jogo
    MOV [R1], R0

fim_verificar_vitoria:
    POP R1
    POP R0
    RET

; ==============================================================================
; 10. ROTINAS DE ECRÃ E DESENHO
; ==============================================================================

limpar_ecra:
    PUSH R0
    PUSH R1
    PUSH R2

    MOV R1, ECRA_INICIO
    MOV R2, 128

limpar_loop_ecra:
    MOV R0, 0
    MOVB [R1], R0
    ADD R1, 1
    SUB R2, 1
    JNZ limpar_loop_ecra

    POP R2
    POP R1
    POP R0
    RET

pixel_xy:
```

```
    PUSH R3
    PUSH R4
    PUSH R5
    PUSH R6

    ; Clipping (Linha)
    MOV R3, 31
    CMP R1, R3
    JLE R1_not_gt_31_p
    JMP fim_pixel
R1_not_gt_31_p:
    MOV R3, 0
    CMP R1, R3
    JLT fim_pixel
    ; Clipping (Coluna)
    MOV R3, 31
    CMP R2, R3
    JLE R2_not_gt_31_p
    JMP fim_pixel
R2_not_gt_31_p:
    MOV R3, 0
    CMP R2, R3
    JLT fim_pixel

    ; Endereço do Byte
    MOV R3, R1
    ADD R3, R3
    ADD R3, R3

    MOV R4, R2
    MOV R5, 8
    DIV R4, R5

    ADD R3, R4
    MOV R4, ECRA_INICIO
    ADD R3, R4

    ; Bitmask
    MOV R4, R2
    MOV R5, 8
    MOD R4, R5
```

```
    MOV R5, 7
    SUB R5, R4

    MOV R4, tabela_bits
    ADD R5, R5
    ADD R4, R5
    MOV R6, [R4]

    MOVB R5, [R3]
    OR R5, R6
    MOVB [R3], R5

fim_pixel:
    POP R6
    POP R5
    POP R4
    POP R3
    RET

apagar_pixel:
    PUSH R3
    PUSH R4
    PUSH R5
    PUSH R6

    ; Clipping (Linha)
    MOV R3, 31
    CMP R1, R3
    JLE R1_not_gt_31_a
    JMP fim_apagar
R1_not_gt_31_a:
    MOV R3, 0
    CMP R1, R3
    JLT fim_apagar
    ; Clipping (Coluna)
    MOV R3, 31
    CMP R2, R3
    JLE R2_not_gt_31_a
    JMP fim_apagar
R2_not_gt_31_a:
    MOV R3, 0
```

```
    CMP R2, R3
    JLT fim_apagar

    ; Endereço do Byte
    MOV R3, R1
    ADD R3, R3
    ADD R3, R3

    MOV R4, R2
    MOV R5, 8
    DIV R4, R5

    ADD R3, R4
    MOV R4, ECRA_INICIO
    ADD R3, R4

    ; Bitmask
    MOV R4, R2
    MOV R5, 8
    MOD R4, R5
    MOV R5, 7
    SUB R5, R4

    MOV R4, tabela_bits
    ADD R5, R5
    ADD R4, R5
    MOV R6, [R4]

    NOT R6
    MOVB R5, [R3]
    AND R5, R6
    MOVB [R3], R5

fim_apagar:
    POP R6
    POP R5
    POP R4
    POP R3
    RET

desenhar_objeto:
```

```
    PUSH R0
    PUSH R4
    PUSH R5
    PUSH R6
    PUSH R7

    MOV R4, R1
    MOV R5, R2

loop_desenho_obj:
    MOV R0, [R3]

    MOV R7, TERMINADOR_SPRITE
    CMP R0, R7
    JZ fim_desenho_obj

    MOV R7, 2
    ADD R3, R7
    MOV R6, [R3]

    PUSH R1
    PUSH R2
    ADD R1, R0
    ADD R2, R6

    CALL pixel_xy

    POP R2
    POP R1

    MOV R7, 2
    ADD R3, R7
    JMP loop_desenho_obj

fim_desenho_obj:
    POP R7
    POP R6
    POP R5
    POP R4
    POP R0
    RET
```

```
apagar_objeto:
    PUSH R0
    PUSH R4
    PUSH R5
    PUSH R6
    PUSH R7

    MOV R4, R1
    MOV R5, R2

loop_apagar_obj:
    MOV R0, [R3]

    MOV R7, TERMINADOR_SPRITE
    CMP R0, R7
    JZ fim_apagar_obj

    MOV R7, 2
    ADD R3, R7
    MOV R6, [R3]

    PUSH R1
    PUSH R2
    ADD R1, R0
    ADD R2, R6

    CALL apagar_pixel

    POP R2
    POP R1

    MOV R7, 2
    ADD R3, R7
    JMP loop_apagar_obj

fim_apagar_obj:
    POP R7
    POP R6
    POP R5
    POP R4
```

```
    POP R0
    RET

desenhar_pacman:
    PUSH R1
    PUSH R2
    PUSH R3

    MOV R1, linha_pacman
    MOV R1, [R1]
    MOV R2, coluna_pacman
    MOV R2, [R2]
    MOV R3, tabela_pacman

    CALL desenhar_objeto

    POP R3
    POP R2
    POP R1
    RET

apagar_pacman:
    PUSH R1
    PUSH R2
    PUSH R3

    MOV R1, linha_pacman
    MOV R1, [R1]
    MOV R2, coluna_pacman
    MOV R2, [R2]
    MOV R3, tabela_pacman

    CALL apagar_objeto

    POP R3
    POP R2
    POP R1
    RET

desenhar_fantasmas:
    PUSH R0
```

```
    PUSH R1
    PUSH R2
    PUSH R3
    PUSH R4
    PUSH R5

    MOV R4, fantasmas
    MOV R0, 0

desenhar_fant_loop:
    CMP R0, NUM_FANTASMAS
    JGE fim_desenhar_fantasmas

    MOV R1, [R4]
    MOV R2, [R4+2]

    PUSH R4
    MOV R3, tabela_fantasma
    CALL desenhar_objeto
    POP R4

    MOV R5, FANTASMA_SIZE
    ADD R4, R5
    ADD R0, 1
    JMP desenhar_fant_loop

fim_desenhar_fantasmas:
    POP R5
    POP R4
    POP R3
    POP R2
    POP R1
    POP R0
    RET

apagar_fantasma:
    PUSH R1
    PUSH R2
    PUSH R3

    MOV R1, [R4]
```

```
    MOV R2, [R4+2]
    MOV R3, tabela_fantasma

    CALL apagar_objeto

    POP R3
    POP R2
    POP R1
    RET

desenhar_cenario:
    PUSH R0
    PUSH R1
    PUSH R2
    PUSH R3

    CALL desenhar_paredes
    CALL desenhar_caixa

    MOV R3, objetos_cantos
    MOV R0, 0

desenhar_objs_loop:
    CMP R0, MAX_PONTOS
    JGE fim_desenho_cenario

    MOV R1, [R3+4]
    CMP R1, 0
    JNZ proximo_obj_cenario

    MOV R1, [R3]
    MOV R2, [R3+2]
    PUSH R3
    MOV R3, tabela_objeto
    CALL desenhar_objeto
    POP R3

proximo_obj_cenario:
    PUSH R4
    MOV R4, OBJETO_VAR_SIZE
    ADD R3, R4
```

```
    POP R4

    ADD R0, 1
    JMP desenhar_objs_loop

fim_desenho_cenario:
    POP R3
    POP R2
    POP R1
    POP R0
    RET

desenhar_paredes:
    PUSH R1
    PUSH R2
    PUSH R3
    PUSH R7

    MOV R7, 32

    MOV R1, 0
    MOV R3, 0
loop_horizontal_topo:
    MOV R2, R3
    CALL pixel_xy
    ADD R3, 1
    CMP R3, R7
    JLT loop_horizontal_topo

    MOV R1, 31
    MOV R3, 0
loop_horizontal_fundo:
    MOV R2, R3
    CALL pixel_xy
    ADD R3, 1
    CMP R3, R7
    JLT loop_horizontal_fundo

    MOV R2, 0
    MOV R3, 0
loop_vertical_esq:
```

```
    MOV R1, R3
    CALL pixel_xy
    ADD R3, 1
    CMP R3, R7
    JLT loop_vertical_esq

    MOV R2, 31
    MOV R3, 0
loop_vertical_dir:
    MOV R1, R3
    CALL pixel_xy
    ADD R3, 1
    CMP R3, R7
    JLT loop_vertical_dir

    POP R7
    POP R3
    POP R2
    POP R1
    RET

desenhar_caixa:
    PUSH R0
    PUSH R1
    PUSH R2
    PUSH R3
    PUSH R4
    PUSH R5
    PUSH R6

    ; --- USO DE VALORES IMEDIATOS (CONSTANTES) PARA O DESENHO (Correção) ---
    MOV R1, 12              ; R1 = Linha inicial (12)
    MOV R2, 12              ; R2 = Coluna inicial (12)
    MOV R3, 7               ; R3 = Altura (7)
    MOV R4, 7               ; R4 = Largura (7)

    MOV R0, 0               ; R0 = Contador da linha (offset)

caixa_linhas:
    CMP R0, R3
    JGE abrir_porta_caixa
```

```
    MOV R5, 0               ; R5 = Contador da coluna (offset)

caixa_colunas:
    CMP R5, R4
    JGE prox_linha_caixa

    ; --- LÓGICA DE DESENHO DA BORDA SÓ ---

    ; (R0 == 0) - Borda Superior
    MOV R6, 0
    CMP R0, R6
    JZ desenhar_borda_nova

    ; (R0 == R3-1) - Borda Inferior (offset 6)
    MOV R6, R3
    SUB R6, 1
    CMP R0, R6
    JZ desenhar_borda_nova

    ; (R5 == 0) - Borda Esquerda
    MOV R6, 0
    CMP R5, R6
    JZ desenhar_borda_nova

    ; (R5 == R4-1) - Borda Direita (offset 6)
    MOV R6, R4
    SUB R6, 1
    CMP R5, R6
    JZ desenhar_borda_nova

    ; É o interior (não desenha)
    JMP prox_coluna_caixa

desenhar_borda_nova:
    ; R1 e R2 são a base (12, 12), R0 e R5 são o offset.
    PUSH R1
    PUSH R2
    ADD R1, R0 ; R1 = Linha absoluta (12 + R0)
    ADD R2, R5 ; R2 = Coluna absoluta (12 + R5)
    CALL pixel_xy
```

```
    POP R2      ; Restaura R2 para 12
    POP R1      ; Restaura R1 para 12

prox_coluna_caixa:
    ADD R5, 1
    JMP caixa_colunas

prox_linha_caixa:
    ADD R0, 1
    JMP caixa_linhas

abrir_porta_caixa:
    ; R1 e R2 ainda são 12 (Base da Caixa)

    ; Linha da porta (L12, ou R1 + 0)
    ; Coluna da porta: R2 = Coluna 12 + 3 = Coluna 15
    PUSH R0
    MOV R0, 3
    ADD R2, R0
    POP R0

    CALL apagar_pixel ; Apaga a porta (L12, C15)

fim_caixa_desenho:
    POP R6
    POP R5
    POP R4
    POP R3
    POP R2
    POP R1
    POP R0
    RET

; ================================================================================
; 11. ROTINAS DE ESTADO DO JOGO
; ================================================================================

mostrar_menu:
    PUSH R0
    PUSH R1
    PUSH R2
```

```
    PUSH R3
    CALL limpar_ecra

    MOV R1, 10
    MOV R2, 10
    CALL pixel_xy
    MOV R2, 11
    CALL pixel_xy

    MOV R1, 12
    MOV R2, 10
    CALL pixel_xy
    MOV R2, 11
    CALL pixel_xy

    MOV R1, 20
    MOV R2, 15
    CALL pixel_xy

    POP R3
    POP R2
    POP R1
    POP R0
    RET

mostrar_tela_final:
    PUSH R0
    PUSH R1
    PUSH R2
    CALL limpar_ecra

    MOV R0, estado_jogo
    MOV R0, [R0]

    MOV R1, 2
    CMP R0, R1
    JZ game_over

    MOV R1, 3
    CMP R0, R1
    JZ game_vitoria
```

```
game_over:
    MOV R1, 15
    MOV R2, 10
    CALL pixel_xy
    MOV R1, 15
    MOV R2, 11
    CALL pixel_xy
    JMP tela_final_saida

game_vitoria:
    MOV R1, 15
    MOV R2, 15
    CALL pixel_xy
    MOV R1, 15
    MOV R2, 16
    CALL pixel_xy

tela_final_saida:
    POP R2
    POP R1
    POP R0
    RET

esperar_reinicio:
    PUSH R0
    PUSH R1
    PUSH R2
    PUSH R3

    MOV R0, 0
    MOV R1, CICLOS_POR_SEGUNDO
wait_loop:
    ADD R0, 1
    CMP R0, R1
    JLT wait_loop

wait_key:
    MOV R1, 1
    MOV R2, POUT
    MOV R3, PIN
```

```
MOVB [R2], R1
MOVB R0, [R3]

CMP R0, 0
JZ wait_key

POP R3
POP R2
POP R1
POP R0
RET
```