DEEC
DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES
TÉCNICO LISBOA

Arquitectura de Computadores

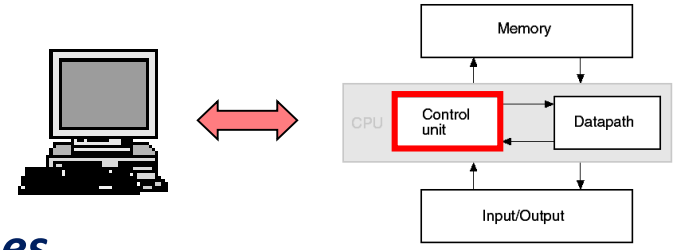
MEEC (2016/17 – 2º Sem.)

P3 – Processador e Periféricos

Prof. Nuno Horta

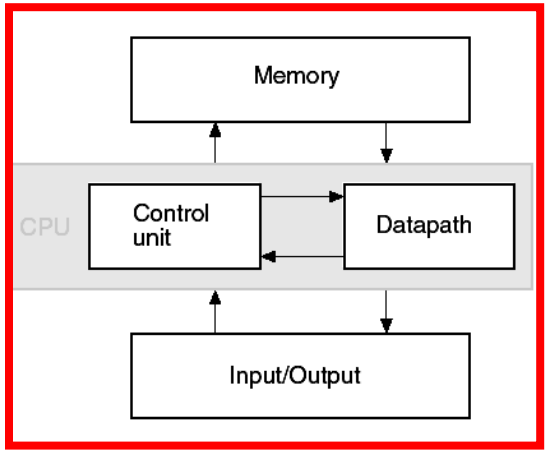
PLANEAMENTO

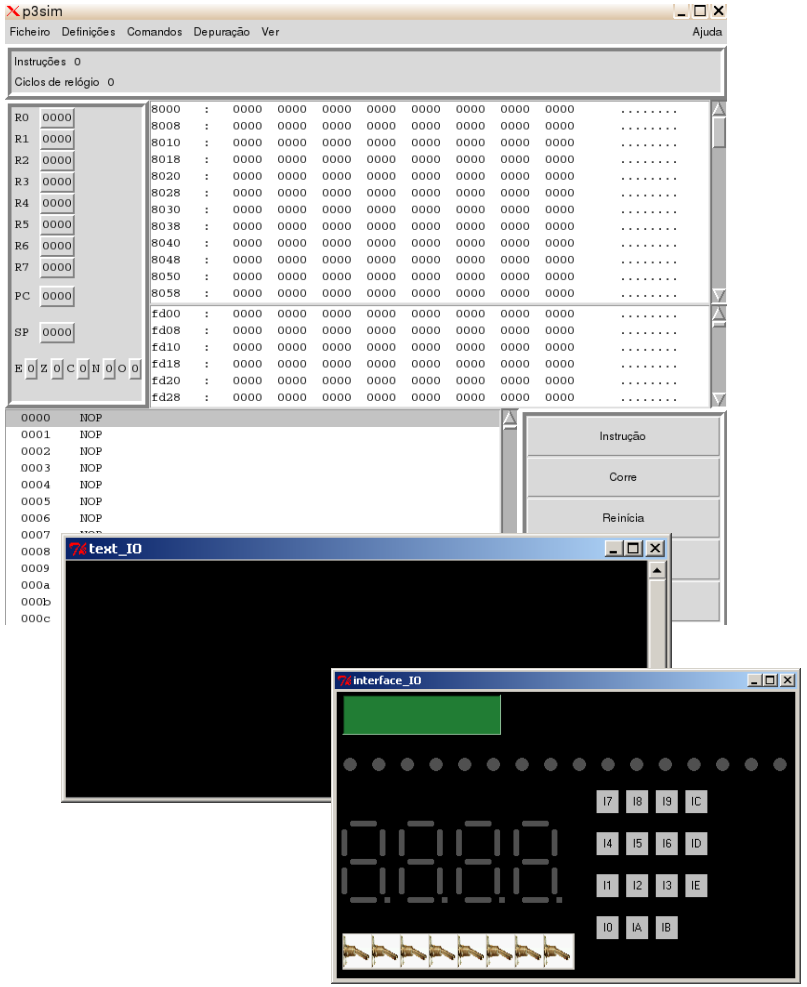
- *Introdução*
- *Unidade de Processamento*
- *Unidade de Controlo*
- *Arquitectura do Conjunto de Instruções*
- *Unidade Central de Processamento (CPU)*
- *P3 – Processador e Periféricos*
- *Unidade de Entrada/Saída (I/O)*
- *Unidade de Memória*



SUMÁRIO

- *P3 - Processador*
  - *Arquitectura do Processador P3*
  - *Desenvolvimento de Programas*
  - *Simulador*
  - *Periféricos*





P3 - PROCESSADOR

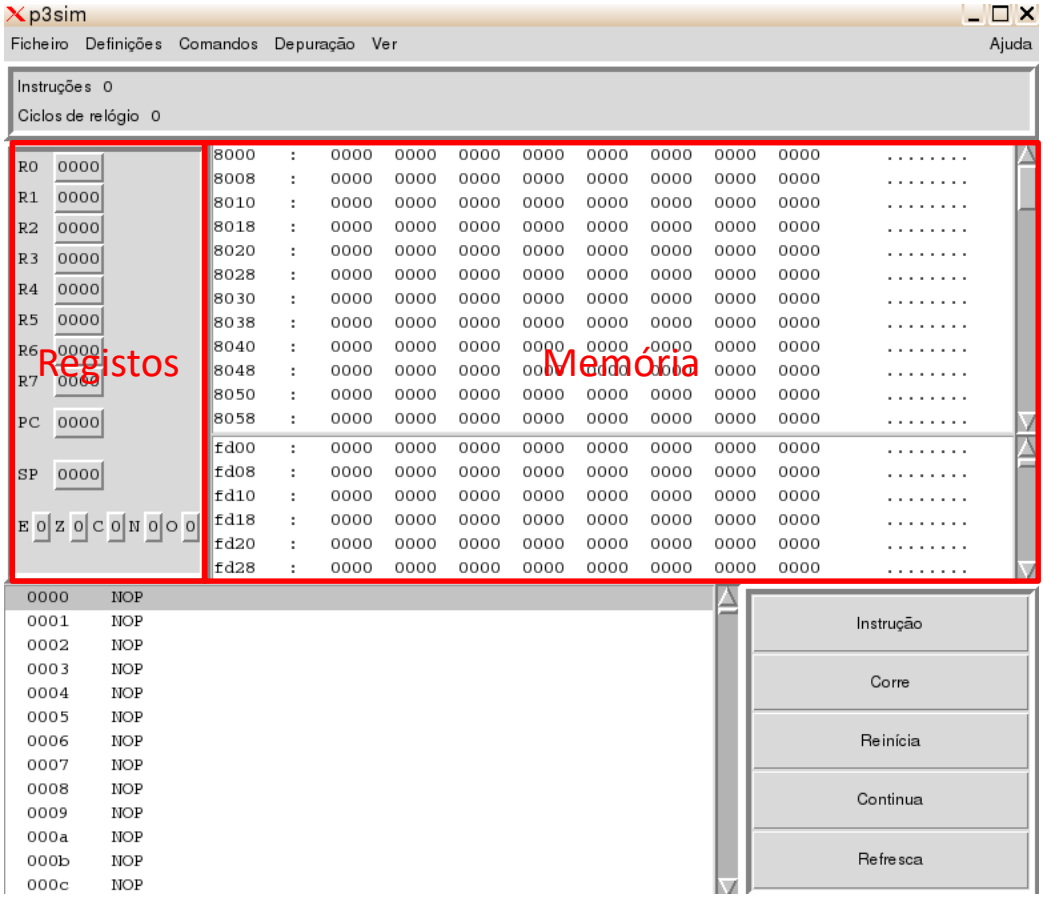
Arquitectura do Processador P3

1. *Registos*
   - *R0-R7 – Uso geral (R0 = 0)*
   - *PC – Program Counter*
   - *SP – Stack Pointer*
   - *RE – Registo de Estado (EZCNO)*

2. *Memória*
   - *Espaço de endereçamento: 64 KPalavras (2^16)*
   - *Dimensão das palavras: 16 bits*

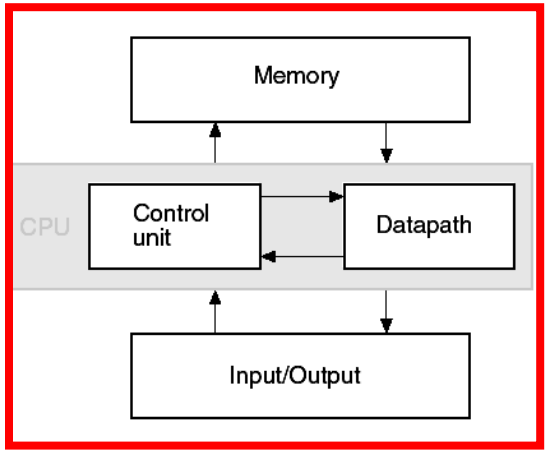
3. *Entradas/Saídas*
   - *Endereçamento de I/O: Memory Mapped I/O de FF00h em diante*

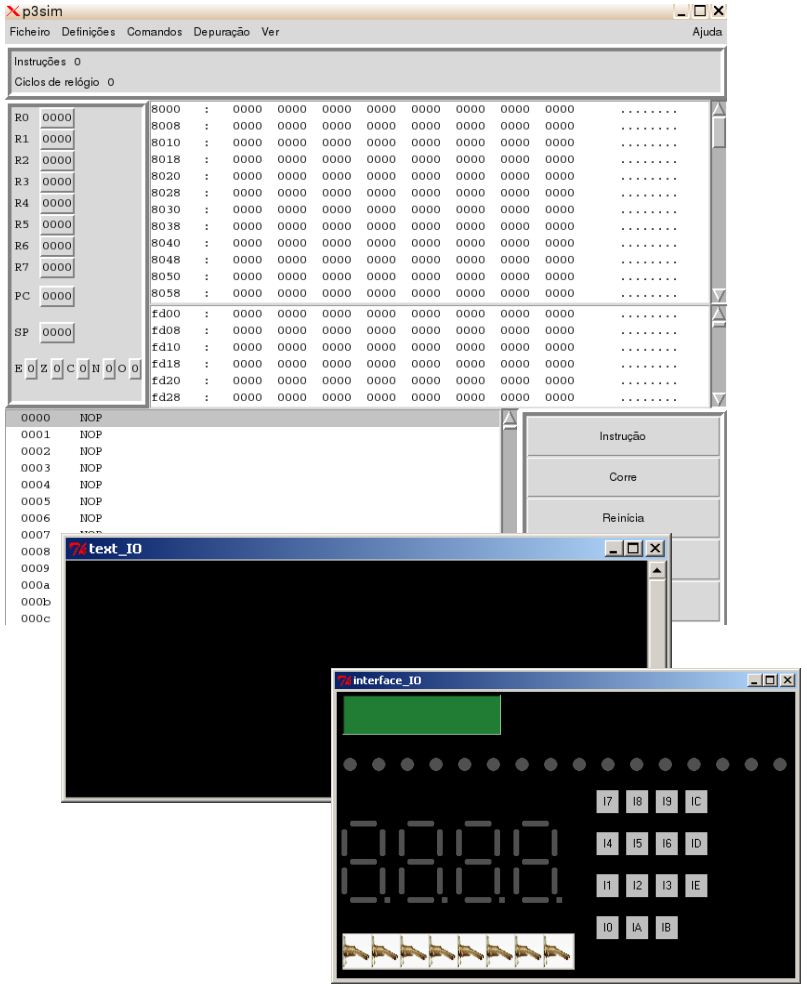
4. *Interrupções*
   - *Máscara de Interrupções: FFFAh*
   - *Tabela de Vectores de Interrupção: FE00h*



SUMÁRIO

- *P3 - Processador*
  - *Arquitectura do Processador P3*
  - *Desenvolvimento de Programas*
  - *Simulador*
  - *Periféricos*





Desenvolvimento de Programas

1. Desenvolvimento de Programas:

1.1. Pseudo-Instruções: Comandos reconhecidos pelo ASSEMBLER que não são instruções assembly, mas permitem dar ao ASSEMBLER um conjunto de informações necessárias para a sua correcta execução ou para simplificar a sua utilização.

ORIG, EQU, WORD, STR, TAB

Pseudo	Aritméticas	Lógicas	Deslocamento	Controlo	Transfer.	Genéricas
ORIG	NEG	COM	SHR	BR	MOV	NOP
EQU	INC	AND	SHL	BR. *cond*	MVBH	ENI
WORD	DEC	OR	SHRA	JMP	MVBL	DSI
STR	ADD	XOR	SHLA	JMP. *cond*	XCH	STC
TAB	ADDC	TEST	ROR	CALL	PUSH	CLC
	SUB		ROL	CALL. *cond*	POP	CMC
	SUBB		RORC	RET		
	CMP		ROLC	RETN		
	MUL			RTI		
	DIV			INT		

1.2. Comentários: Utiliza-se o ';' para indicar que todo o texto que se segue na linha deverá ser ignorado pelo ASSEMBLER.

```
;*********************************************************************************
; Programa BubbleSort.as
;
; Descricao: Implementação do Algoritmo de ordenação Bubble Sort para Assembly
do P3
;            (1) Inicialização de Dados em Memória
;            (2) Ordenação de dados por ordem crescente
;            (3) Ordenação de dados por ordem decrescente
;
;
; Autor: Nuno Horta
; Data: 04/2013                                    Ultima Alteracao:22/04/2013
;*********************************************************************************

;*********************************************************************************
; ZONA I: Definicao de constantes
;         Pseudo-instrucao : EQU
;
; STACK POINTER
SP_INICIAL      EQU     FDFFh

;
; ZONA II: Definicao de variaveis
;          Pseudo-instrucoes : WORD - palavra (16 bits)
;                              STR  - sequencia de caracteres.
;          Cada caracter ocupa 1 palavra
;---------------------------------------------------------------------------------
                ORIG    8000h
VDados          STR     20,1,3,5,7,9,2,4,6,8,10,12,14,16,18,20,11,13,15,17,19
```

Pseudo-instruções

Comentários: Devem permitir perceber as funcionalidades implementadas sem cair na redundância de descrever a execução de cada instrução.

P3 - PROCESSADOR

Desenvolvimento de Programas

1. Desenvolvimento de Programas (cont.):

1.3. Conjunto de Instruções

Pseudo	Aritméticas	Lógicas	Deslocamento	Controlo	Transfer.	Genéricas
ORIG	NEG	COM	SHR	BR	MOV	NOP
EQU	INC	AND	SHL	BR. *cond*	MVBH	ENI
WORD	DEC	OR	SHRA	JMP	MVBL	DSI
STR	ADD	XOR	SHLA	JMP. *cond*	XCH	STC
TAB	ADDC	TEST	ROR	CALL	PUSH	CLC
	SUB		ROL	CALL. *cond*	POP	CMC
	SUBB		RORC	RET		
	CMP		ROLC	RETN		
	MUL			RTI		
	DIV			INT		

1.4. Modos de Endereçamento

- *Registo* — *op = Rx*
- *Registo Indirecto* — *op = M[Rx]*
- *Imediato* — *op = W*
- *Directo* — *op = M[W]*
- *Indexado* — *op = M[Rx+W]*
- *Relativo* — *op = M[PC+W]*
- *Baseado* — *op = M[SP+W]*

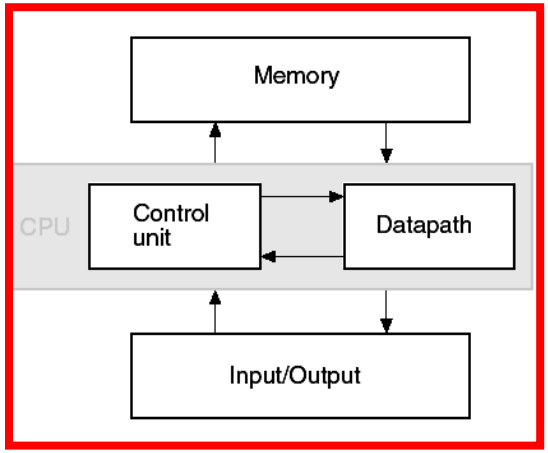
1.5. Etiquetas (Label):

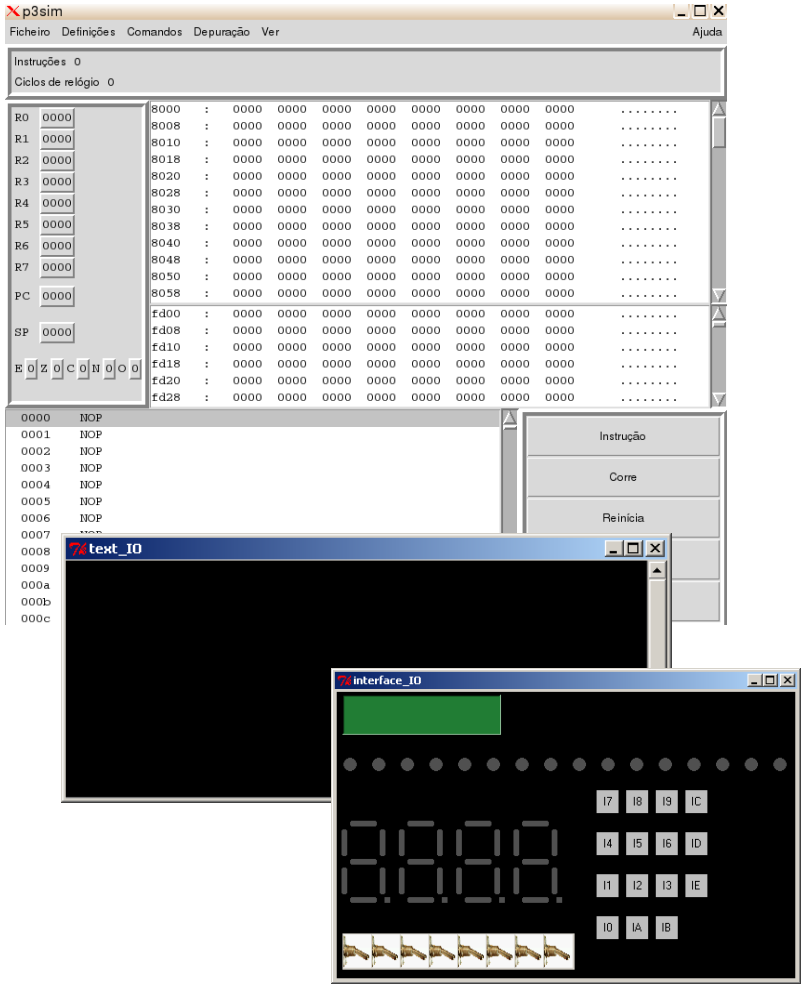
Nome seguido de ':' utilizado para referenciar uma dada posição de memória.

```
;=================================================================================
; ZONA III: Codigo
;           conjunto de instrucoes Assembly, ordenadas de forma a realizar
;           as funcoes pretendidas
;=================================================================================
                ORIG    0000h
                JMP     inicio

;=================================================================================
; BubbleSort: Implementa algoritmo de ordenação "Bubble sort"
;               Entradas: Sentido da Ordenação
;               Saidas: ---
;               Efeitos: Ordena vector dr dados em memória
;=================================================================================
BubbleSort:     MOV     R6, M[SP+2]  ; Sentido da ordenação
                MOV     R2, M[VDados]; Comprimento do Vector de Dados
                ADD     R2, VDados   ; Último Elemento
SortLoop:       MOV     R3, 0        ; Indicador de troca (0 - não houve
troca, 1  - houve troca)
                MOV     R1, VDados   ; Indice do vector de dados
                ADD     R1, 2
SwapLoop:       MOV     R4, M[R1-1]  ; Inicia loop de trocas e le dados
                MOV     R5, M[R1]
                CMP     R6, 1        ; Verifica tipo de ordenação
                BR.Z    IncOrder
DecOrder:       CMP     R4, R5
                BR.N    Swap
                BR      NoSwap
IncOrder:       CMP     R4, R5
                BR.P    Swap
                BR      NoSwap
Swap:           XCH     R4, R5       ; Executa troca
                MOV     M[R1-1], R4
                MOV     M[R1], R5
                MOV     R3, 1
NoSwap:         INC     R1
                CMP     R1, R2       ; Verifica se chegou ao fim do loop de
trocas
                BR.NP   SwapLoop
                CMP     R3, 1        ; Verifica se vector está ordenado
                BR.Z    SortLoop
                RETN    1

;=================================================================================
;                                Programa prinicipal
;=================================================================================
inicio:         MOV     R1, SP_INICIAL
                MOV     SP, R1
Ordena:         PUSH    1               ; Ordenação crescente
                CALL    BubbleSort
                PUSH    -1              ; Ordenação decrescente
                CALL    BubbleSort
                BR      Ordena
```

SUMÁRIO

- *P3 - Processador*
  - *Arquitectura do Processador P3*
  - *Desenvolvimento de Programas*
  - *Simulador*
  - *Periféricos*





P3 - PROCESSADOR

Simulador

2. Assemblador

2.1. Entrada

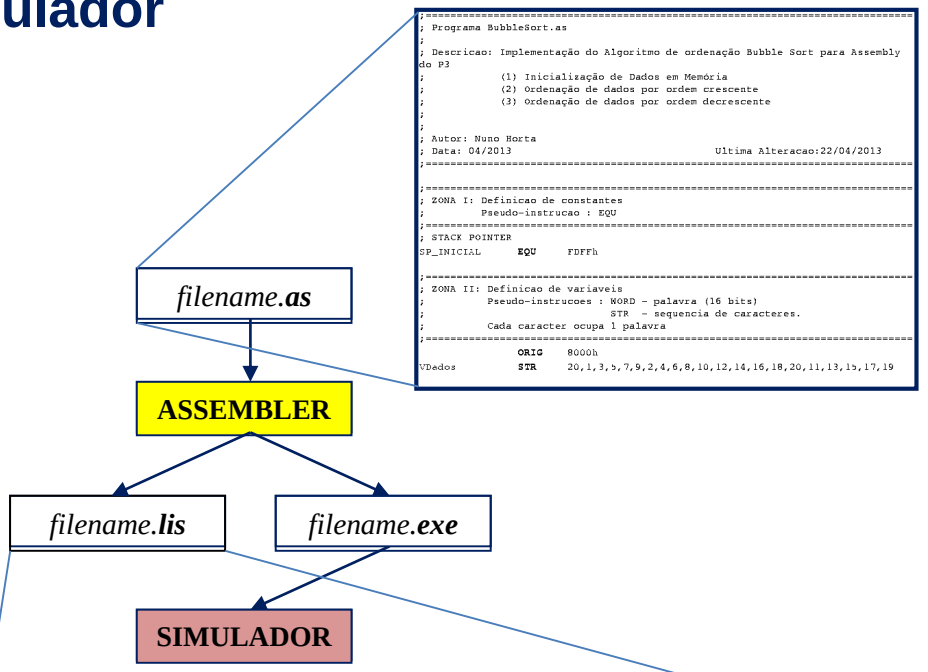
- *Ficheiro com código Assembly – filename.as.*

2.2. Comando

- *>p3as filename.as*

2.3. Saídas

- *Ficheiro filename.lis contém o valor atribuído às referências usadas no programa assembly.*
- *Ficheiro filename.exe contém o código binário pronto a ser executado pelo simulador p3sim.*




```
;=====================================================================================
; Programa BubbleSort.as
;
; Descricao: Implementação do Algoritmo de ordenação Bubble Sort para Assembly
do P3
;            (1) Inicialização de Dados em Memória
;            (2) Ordenação de dados por ordem crescente
;            (3) Ordenação de dados por ordem decrescente
;
;
; Autor: Nuno Horta
; Data: 04/2013                                   Ultima Alteracao:22/04/2013
;=====================================================================================

;=====================================================================================
; ZONA I: Definicao de constantes
;         Pseudo-instrucao : EQU
;=====================================================================================
; STACK POINTER
SP_INICIAL      EQU     FDFFh

;=====================================================================================
; ZONA II: Definicao de variaveis
;         Pseudo-instrucoes : WORD - palavra (16 bits)
;                             STR  - sequencia de caracteres.
;         Cada caracter ocupa 1 palavra
;=====================================================================================
                ORIG    8000h
VDados          STR     20,1,3,5,7,9,2,4,6,8,10,12,14,16,18,20,11,13,15,17,19
```

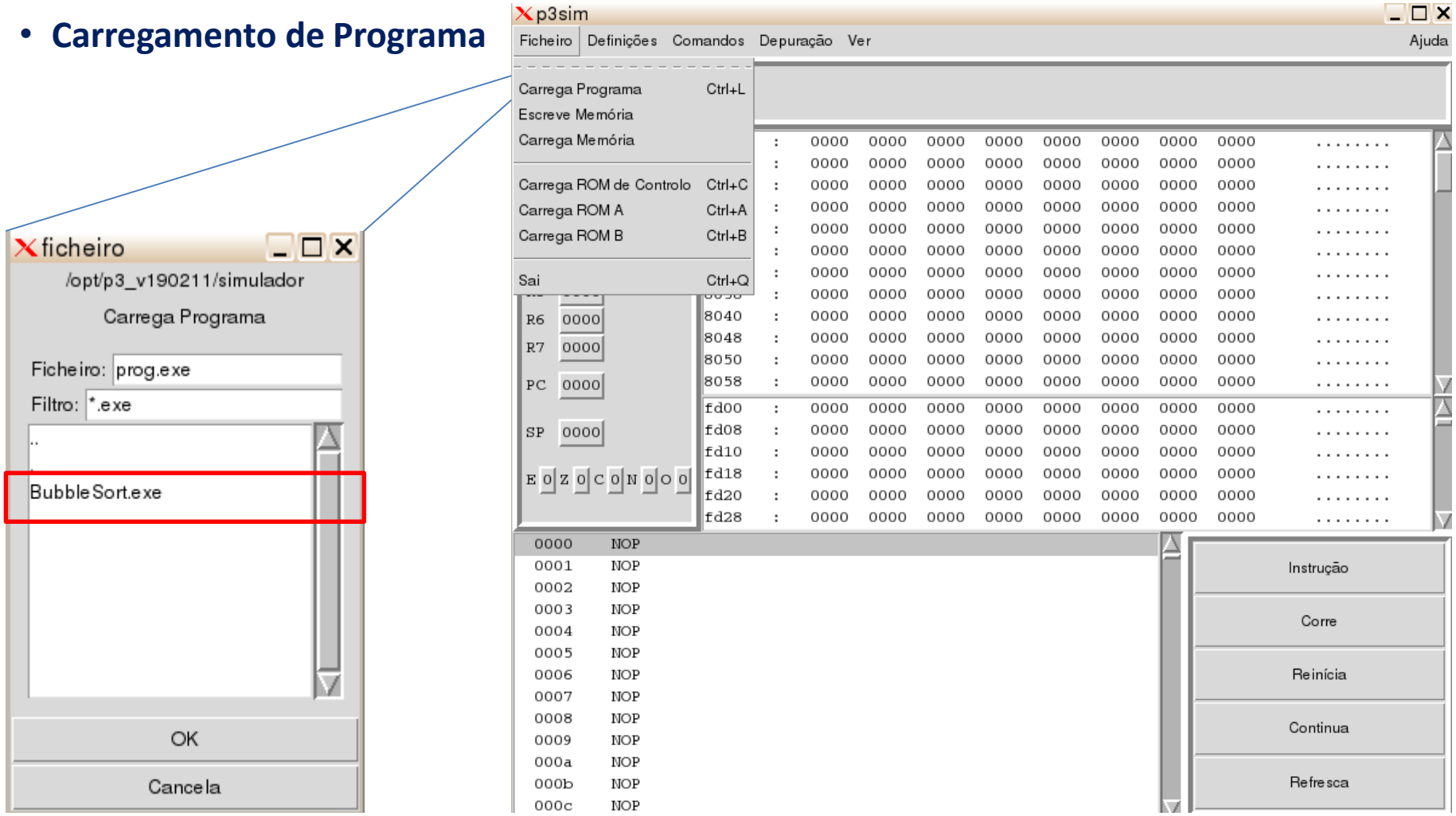
BubbleSort.lis

G L O B A L R E F E R E N C E S

Name	Value	Type
SP_INICIAL	FDFF	CONSTANT
VDados	8000	STRING
BubbleSort	0002	LABEL
SortLoop	0008	LABEL
SwapLoop	000E	LABEL
DecOrder	0014	LABEL
IncOrder	0017	LABEL
Swap	001A	LABEL
NoSwap	0020	LABEL
inicio	0027	LABEL
Ordena	002A	LABEL

P3 - PROCESSADOR Simulador

- **Carregamento de Programa**



P3 - PROCESSADOR

Simulador

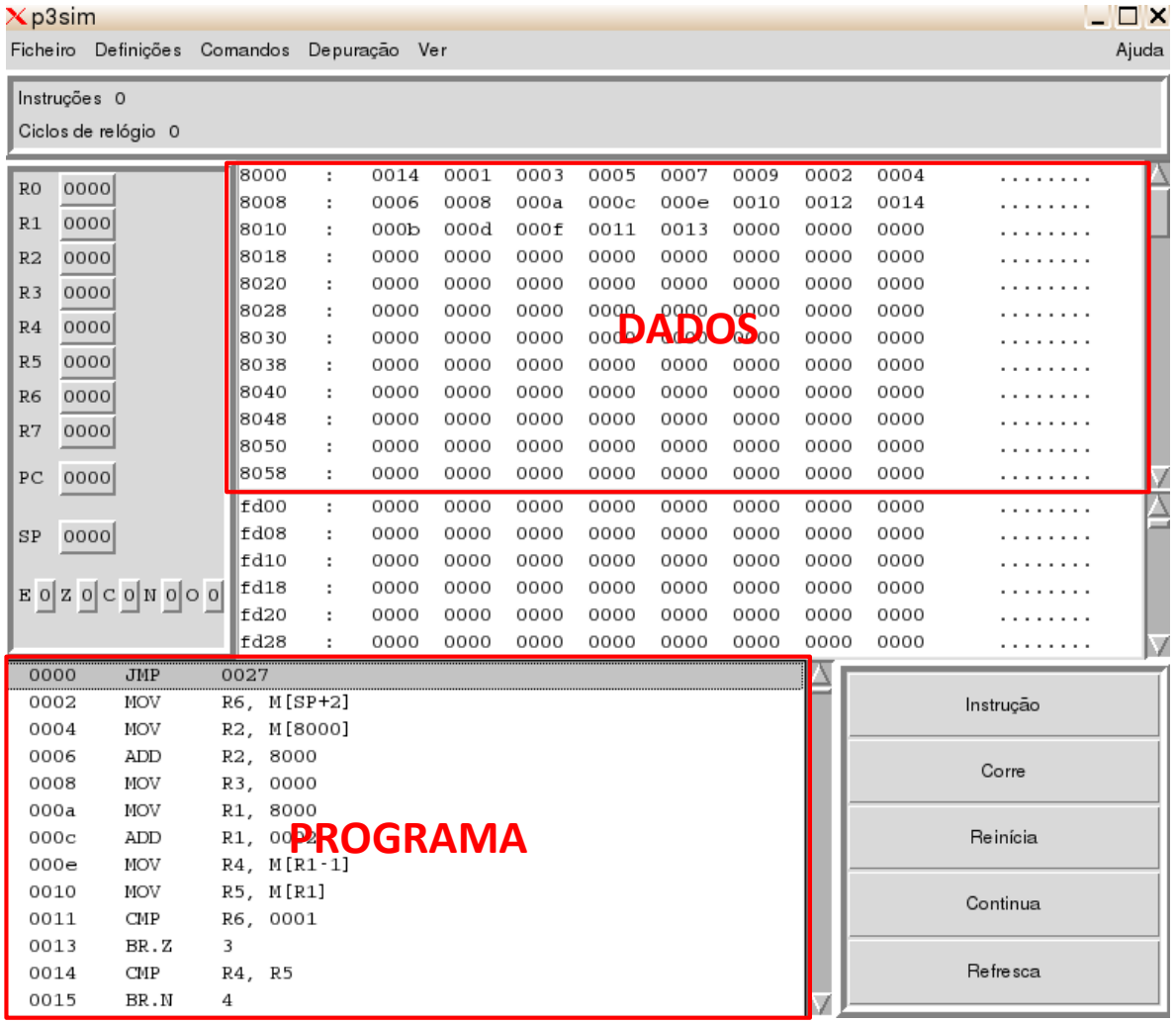
- **Carregamento de Programa**

```
;=====================================================================
; Programa BubbleSort.as
;
; Descricao: Implementação do Algoritmo de ordenação Bubble Sort para Assembly
do P3
;            (1) Inicialização de Dados em Memória
;            (2) Ordenação de dados por ordem crescente
;            (3) Ordenação de dados por ordem decrescente
;
;
; Autor: Nuno Horta
; Data: 04/2013                                  Ultima Alteracao:22/04/2013
;=====================================================================

;=====================================================================
; ZONA I: Definicao de constantes
;         Pseudo-instrucao : EQU
;=====================================================================
; STACK POINTER
SP_INICIAL      EQU     FDFFh

;=====================================================================
; ZONA II: Definicao de variaveis
;         Pseudo-instrucoes : WORD - palavra (16 bits)
;                             STR  - sequencia de caracteres.
;         Cada caracter ocupa 1 palavra
;=====================================================================
                ORIG    8000h
VDados          STR     20,1,3,5,7,9,2,4,6,8,10,12,14,16,18,20,11,13,15,17,19
```

DADOS

```
;=====================================================================
; ZONA III: Codigo
;           conjunto de instrucoes Assembly, ordenadas de forma a realizar
;           as funcoes pretendidas
;=====================================================================
                ORIG    0000h
                JMP     inicio

;=====================================================================
; BubbleSort: Implementa algoritmo de ordenação "Bubble sort"
;             Entradas: Sentido da Ordenação
;             Saidas: ---
;             Efeitos: Ordena vector dr dados em memória
;=====================================================================
BubbleSort:     MOV     R6, M[SP+2]  ; Sentido da ordenação
                MOV     R1, VDados   ; Endereço do vector de Dados
                ADD     R2, VDados   ; Ultimo Elemento
SortLoop:       MOV     R3, 0        ; Indicador de troca (0 - não houve troca,
1  - houve troca)
                MOV     R1, VDados   ; Indice do vector de dados
                ADD     R1, 2
SwapLoop:       MOV     R4, M[R1-1]  ; Inicia loop de trocas e le dados
                MOV     R5, M[R1]
                CMP     R6, 1        ; Verifica tipo de ordenação
                BR.Z    IncOrder
DecOrder:       CMP     R4, R5
                BR.N    Swap
                BR      NoSwap
```

PROGRAMA



P3 - PROCESSADOR
Simulador

- O código carregado no simulador substitui todas as constantes e etiquetas pelos respetivos valores.

```
                JMP     inicio
BubbleSort:     MOV     R6, M[SP+2]
                MOV     R2, M[VDados]
                ADD     R2, VDados
SortLoop:       MOV     R3, 0
                MOV     R1, VDados
                ADD     R1, 2
SwapLoop:       MOV     R4, M[R1-1]
                MOV     R5, M[R1]
                CMP     R6, 1
                BR.Z    IncOrder
DecOrder:       CMP     R4, R5
                BR.N    Swap
                BR      NoSwap
IncOrder:       CMP     R4, R5
                BR.P    Swap
                BR      NoSwap
Swap:           XCH     R4, R5
                MOV     M[R1-1], R4
                MOV     M[R1], R5
                MOV     R3, 1
NoSwap:         INC     R1
                CMP     R1, R2
                BR.NP   SwapLoop
                CMP     R3, 1
                BR.Z    SortLoop
                RETN    1
inicio:         MOV     R1, SP_INICIAL
                MOV     SP, R1
Ordena:         PUSH    1
                CALL    BubbleSort
                PUSH    -1
                CALL    BubbleSort
                BR      Ordena
```

PROGRAMA

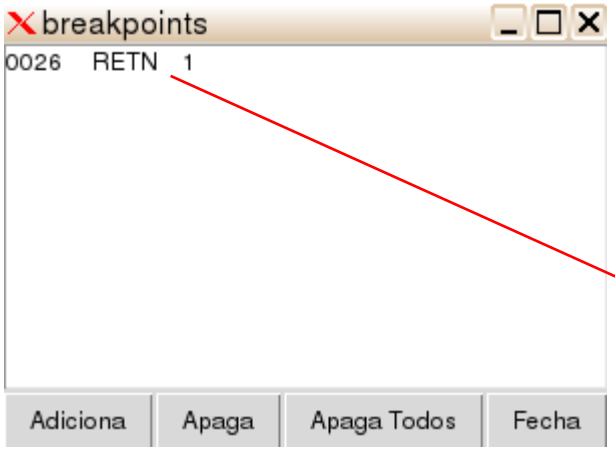
```
0000    JMP     0027
0002    MOV     R6, M[SP+2]
0004    MOV     R2, M[8000]
0006    ADD     R2, 8000
0008    MOV     R3, 0000
000a    MOV     R1, 8000
000c    ADD     R1, 0002
000e    MOV     R4, M[R1-1]
0010    MOV     R5, M[R1]
0011    CMP     R6, 0001
0013    BR.Z    3
0014    CMP     R4, R5
0015    BR.N    4
0016    BR      9
0017    CMP     R4, R5
0018    BR.P    1
0019    BR      8
001a    XCH     R4, R5
001b    MOV     M[R1-1], R4
001d    MOV     M[R1], R5
001e    MOV     R3, 0001
0020    INC     R1
0021    CMP     R1, R2
0022    BR.NP   -21
0023    CMP     R3, 0001
0025    BR.Z    -30
0026    RETN    1
0027    MOV     R1, fdff
0029    MOV     SP, R1
002a    PUSH    0001
002c    CALL    0002
002e    PUSH    ffff
0030    CALL    0002
0032    BR      -9
0033    NOP
```

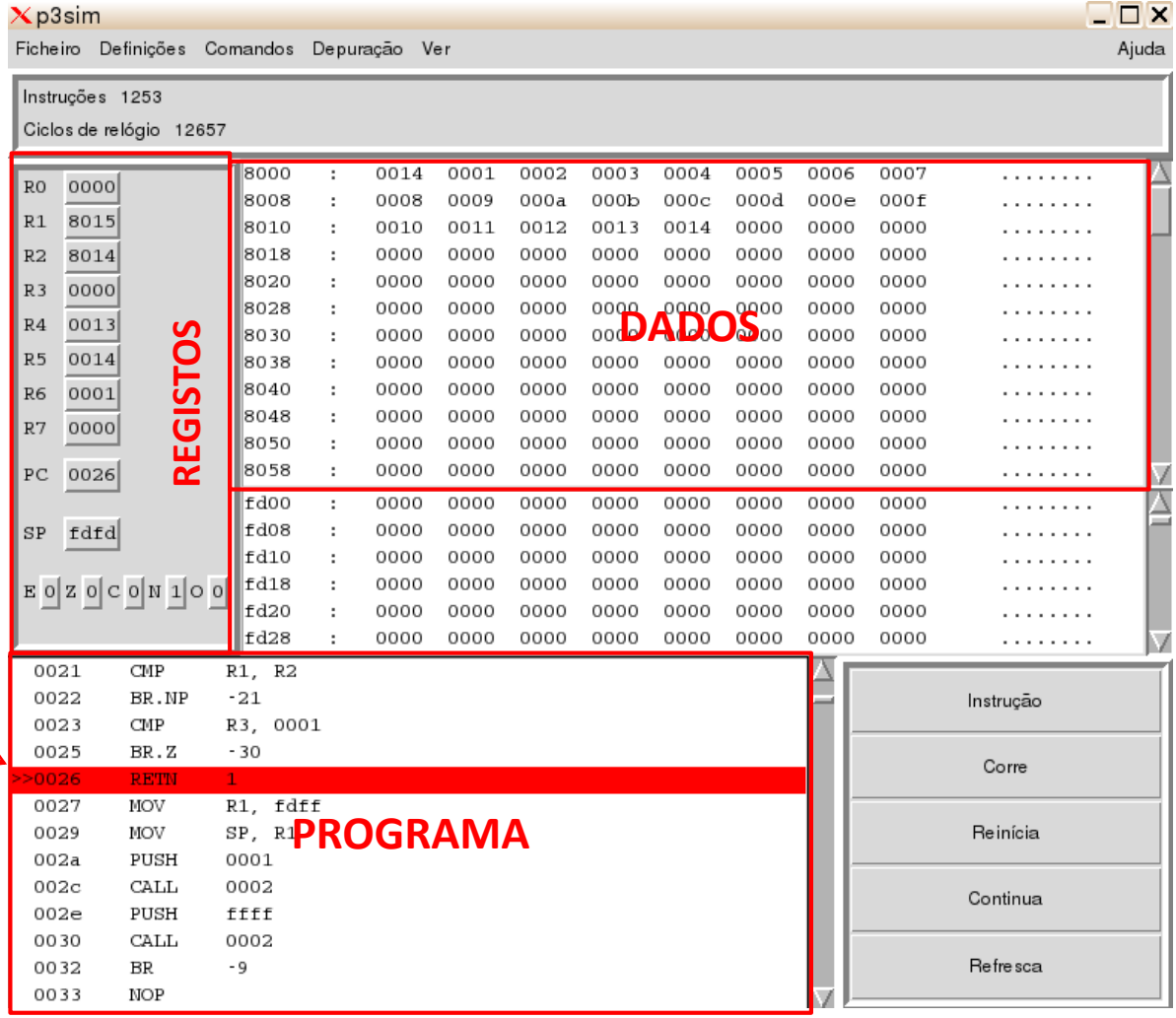
CÓDIGO CARREGADO NO P3

P3 - PROCESSADOR Simulador

- **Execução e Debug do Programa**
  Execução Completa
  Execução Instrução-a-Instrução
  Pontos de Paragem

Exemplo:
ordenação ascendente





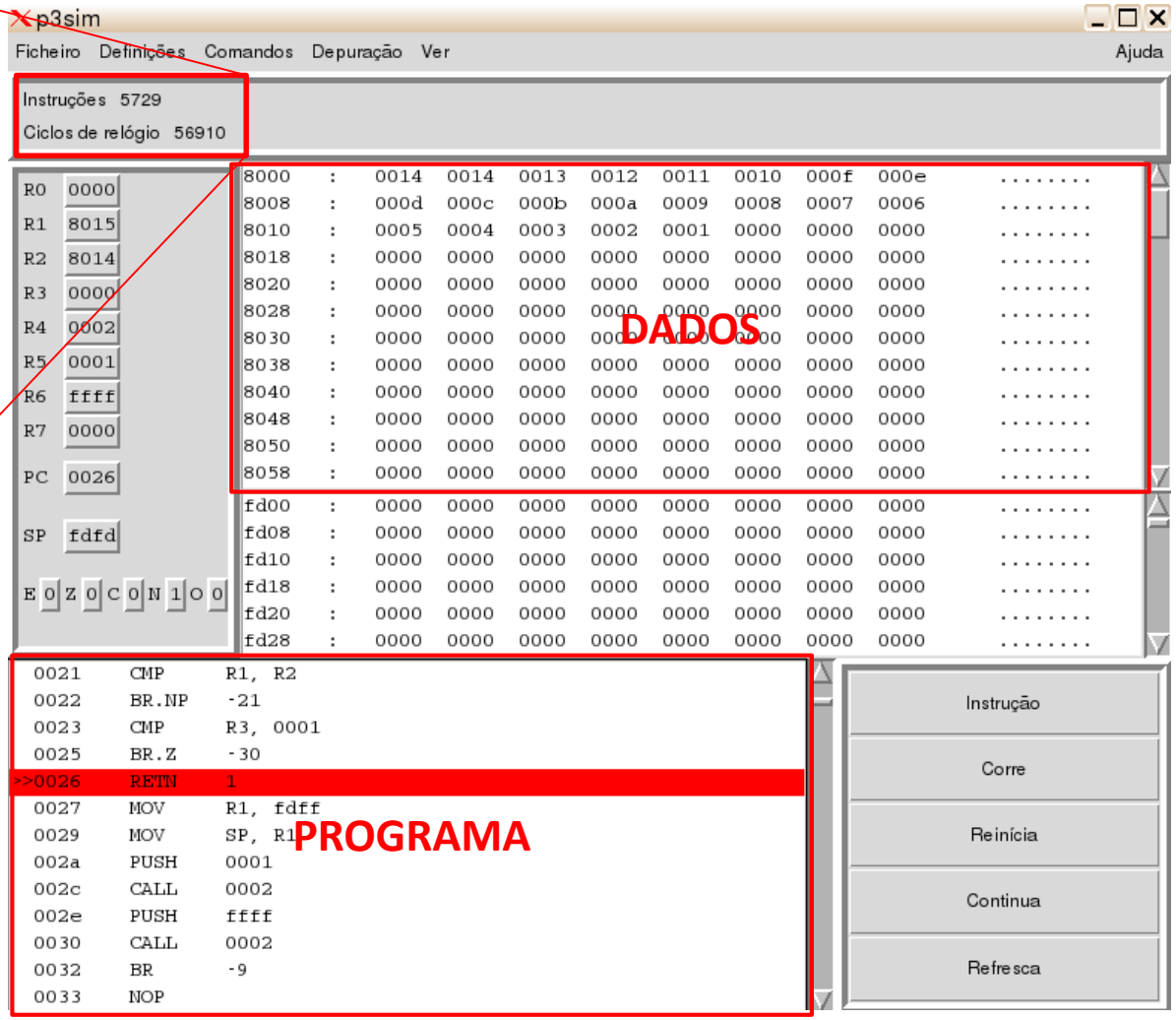
P3 - PROCESSADOR Simulador

Número de Instruções Executadas

Número de Ciclos de Relógio durante a execução do Programa (Arquitectura de Ciclo Múltiplo, porquê?)

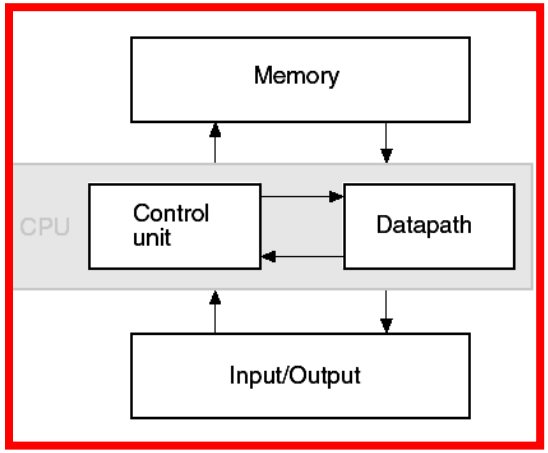
Exemplo:

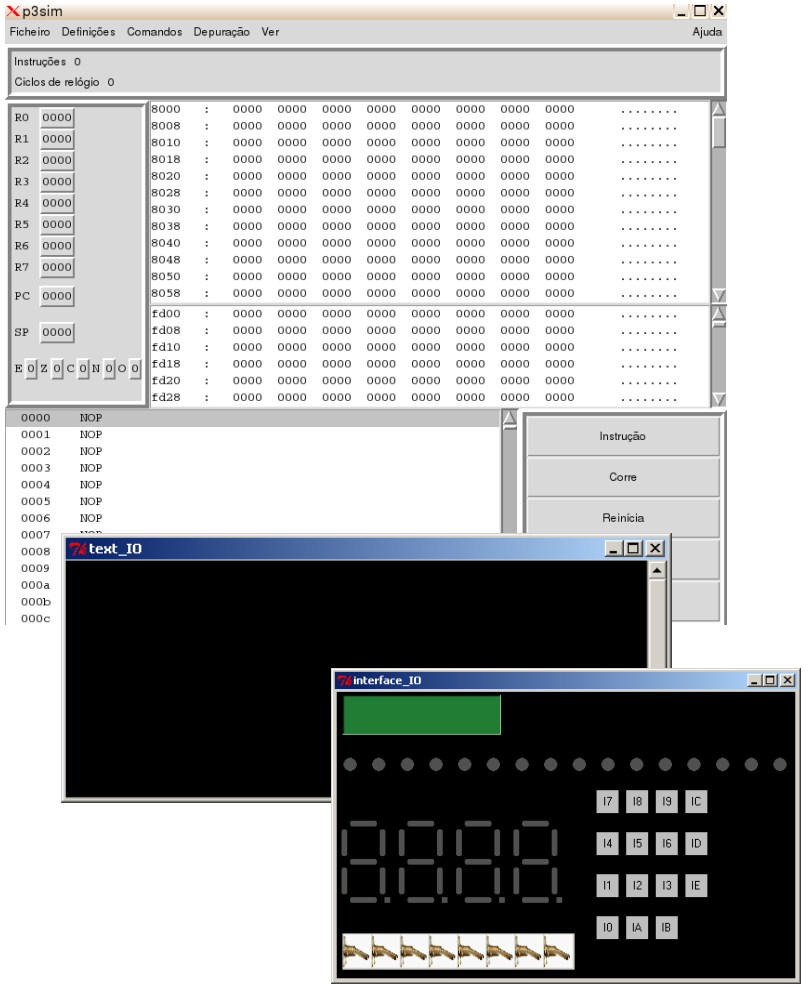
ordenação descendente



SUMÁRIO

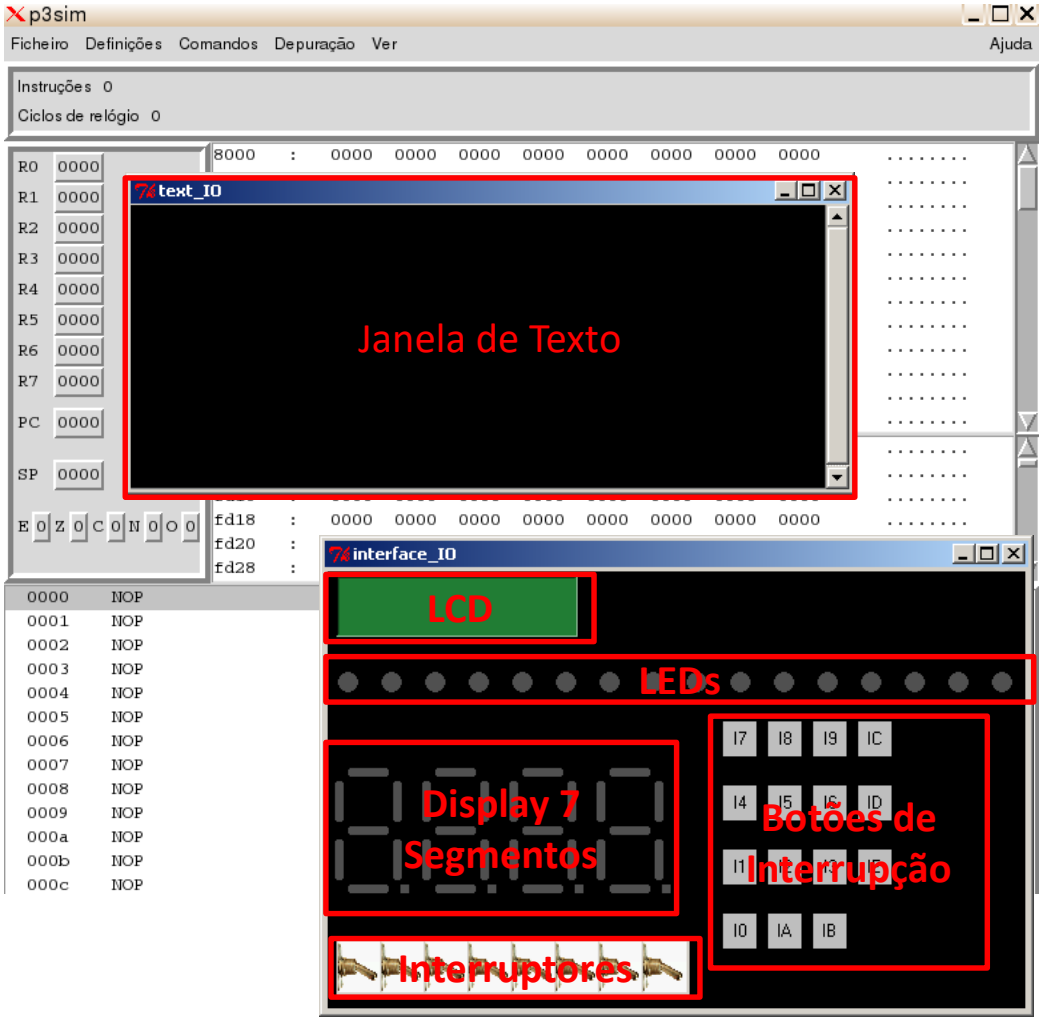
- *P3 - Processador*
  - *Arquitectura do Processador P3*
  - *Desenvolvimento de Programas*
  - *Simulador*
  - *Periféricos*





P3 - PROCESSADOR Periféricos

1. *Registos*
2. *Memória*
3. *Entradas/Saídas*
   - *Endereçamento de I/O: Memory Mapped I/O de FF00h em diante*
   - *Janela de Texto: FFFCh a FFFFh*
   - *Interruptores: FFF9h*
   - *LEDs: FFF8h*
   - *LCD: FFF4h a FFF5h*
   - *Display de 7 Segmentos: FFF0h a FFF3h*
   - *Temporizador: FFF6h a FFF7h*
4. *Interrupções*
   - *Máscara de Interrupções: FFFAh*
   - *Tabela de Vectores de Interrupção: FE00h a FE0Fh*



P3 - PROCESSADOR Periféricos – Janela de Texto

Janela de Texto: FFFCh a FFFFh

- ***FFFCh - Permite colocar o cursor numa dada posição da janela.***
- ***FFFDh - Permite testar se houve alguma tecla premida.***
- ***FFFEh - Permite escrever um caracter na janela.***
- ***FFFFh - Permite ler a última tecla premida.***

XY para escrita (Linha Coluna)

```
;=================================================================================
; Programa Demo1.as
;
; Descricao: Demonstracao da utilização da janela de texto
;
; Autor: Nuno Horta
; Data: 28/05/2013                                  Ultima Alteracao:28/05/2013
;=================================================================================

;=================================================================================
; ZONA I: Definicao de constantes
;         Pseudo-instrucao : EQU
;=================================================================================
; STACK POINTER
SP_INICIAL      EQU     FDFFh

; I/O a partir de FF00H
IO_CURSOR       EQU     FFFCh
IO_WRITE        EQU     FFFEh

LIMPAR_JANELA   EQU     FFFFh
XY_INICIAL      EQU     0614h
FIM_TEXTO       EQU     '@'

;=================================================================================
; ZONA II: Definicao de variaveis
;          Pseudo-instrucoes : WORD - palavra (16 bits)
;                              STR  - sequencia de caracteres.
;          Cada caracter ocupa 1 palavra
;=================================================================================
                ORIG    8000h
```

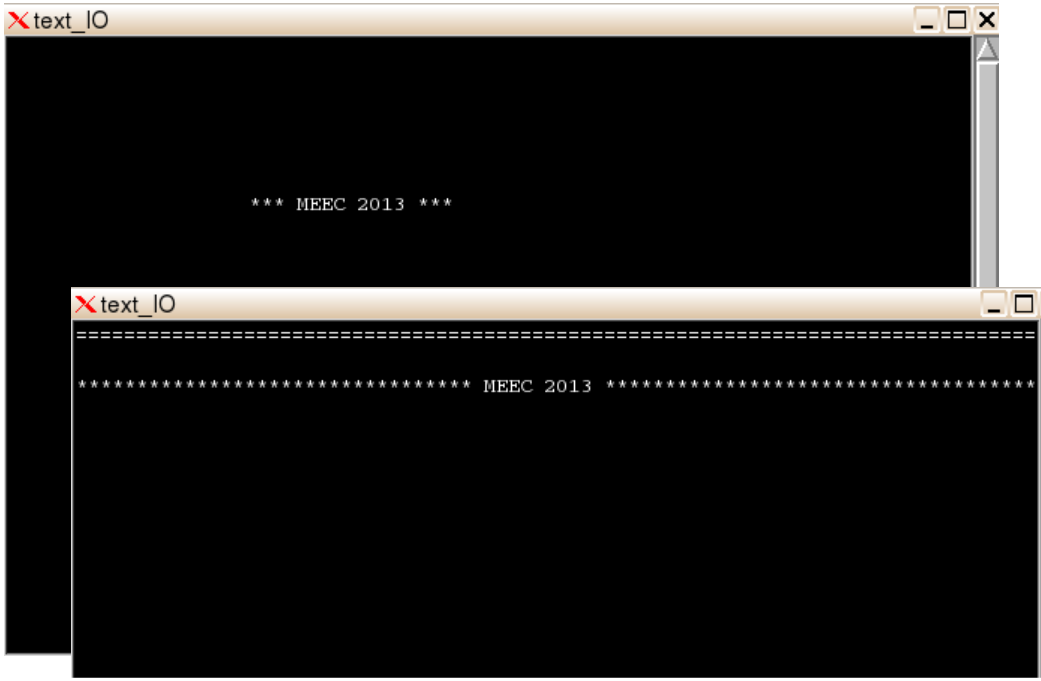
P3 - PROCESSADOR

Periféricos – Janela de Texto

Zona de Dados inicializada com texto a Enviar para a janela de texto.

- *Utilização de símbolo para definir fim da string.*
- *Etiquetas/Labels utilizados para identificar endereço de começo da string.*
- *Ilustração do output a visualizar durante a execução do programa de teste.*

```
;=================================================================================
; ZONA II: Definicao de variaveis
;          Pseudo-instrucoes : WORD - palavra (16 bits)
;                              STR  - sequencia de caracteres.
;          Cada caracter ocupa 1 palavra
;=================================================================================
                ORIG    8000h
VarTexto1       STR     '*** MEEC 2013 ***',FIM_TEXTO
VarTexto2       STR     '================================================
FIM_TEXTO
VarTexto3       STR     '********************************* MEEC 2013 **********
FIM_TEXTO
```



P3 - PROCESSADOR Periféricos – Janela de Texto

Zona de Código:

- *Rotina LimpaJanela utilizada para apagar todos os carateres da janela.*

- *Rotina EscCar para escrita do carater na posição da janela de texto onde se encontra o cursor.*

(Passagem de parâmetros por Registo)

(PUSHs e POPs para preservar e recuperar as variáveis alteradas durante a rotina)

```
;=================================================================================
; ZONA III: Codigo
;           conjunto de instrucoes Assembly, ordenadas de forma a realizar
;           as funcoes pretendidas
;=================================================================================
                ORIG    0000h
                JMP     inicio

;=================================================================================
; LimpaJanela: Rotina que limpa a janela de texto.
;               Entradas: --
;               Saidas: ---
;               Efeitos: ---
;=================================================================================
LimpaJanela:    PUSH R2
                MOV     R2, LIMPAR_JANELA
                MOV     M[IO_CURSOR], R2
                POP R2
                RET

;=================================================================================
; EscCar: Rotina que efectua a escrita de um caracter para o ecran.
;         O caracter pode ser visualizado na janela de texto.
;               Entradas: R1 - Caracter a escrever
;               Saidas: ---
;               Efeitos: alteracao da posicao de memoria M[IO]
;=================================================================================
EscCar:         MOV     M[IO_WRITE], R1
                RET

;=================================================================================
```

P3 - PROCESSADOR Periféricos – Janela de Texto

Zona de Código (cont.):

- *Rotina EscStr para escrita de uma string de carateres na posição da janela de texto onde se encontra o cursor.*

(Passagem de parâmetros pelo STACK)

(PUSHs e POPs para preservar e recuperar as variáveis alteradas durante a rotina)

```
;*********************************************************************************
; EscString: Rotina que efectua a escrita de uma cadeia de caracter, terminada
;            pelo caracter FIM_TEXTO, na janela de texto numa posicao
;            especificada. Pode-se definir como terminador qualquer caracter
;            ASCII.
;               Entradas: pilha - posicao para escrita do primeiro carater
;                         pilha - apontador para o inicio da "string"
;               Saidas: ---
;               Efeitos: ---
;*********************************************************************************
EscString:      PUSH    R1
                PUSH    R2
                PUSH    R3
                MOV     R2, M[SP+6]     ; Apontador para inicio da "string"
                MOV     R3, M[SP+5]     ; Localizacao do primeiro carater
Ciclo:          MOV     M[IO_CURSOR], R3
                MOV     R1, M[R2]
                CMP     R1, FIM_TEXTO
                BR.Z    FimEsc
                CALL    EscCar
                INC     R2
                INC     R3
                BR      Ciclo
FimEsc:         POP     R3
                POP     R2
                POP     R1
                RETN    2               ; Actualiza STACK

;*********************************************************************************
```

P3 - PROCESSADOR Periféricos – Janela de Texto

Programa Principal:

- ***Inicialização do STACK, obrigatória antes da sua utilização.***

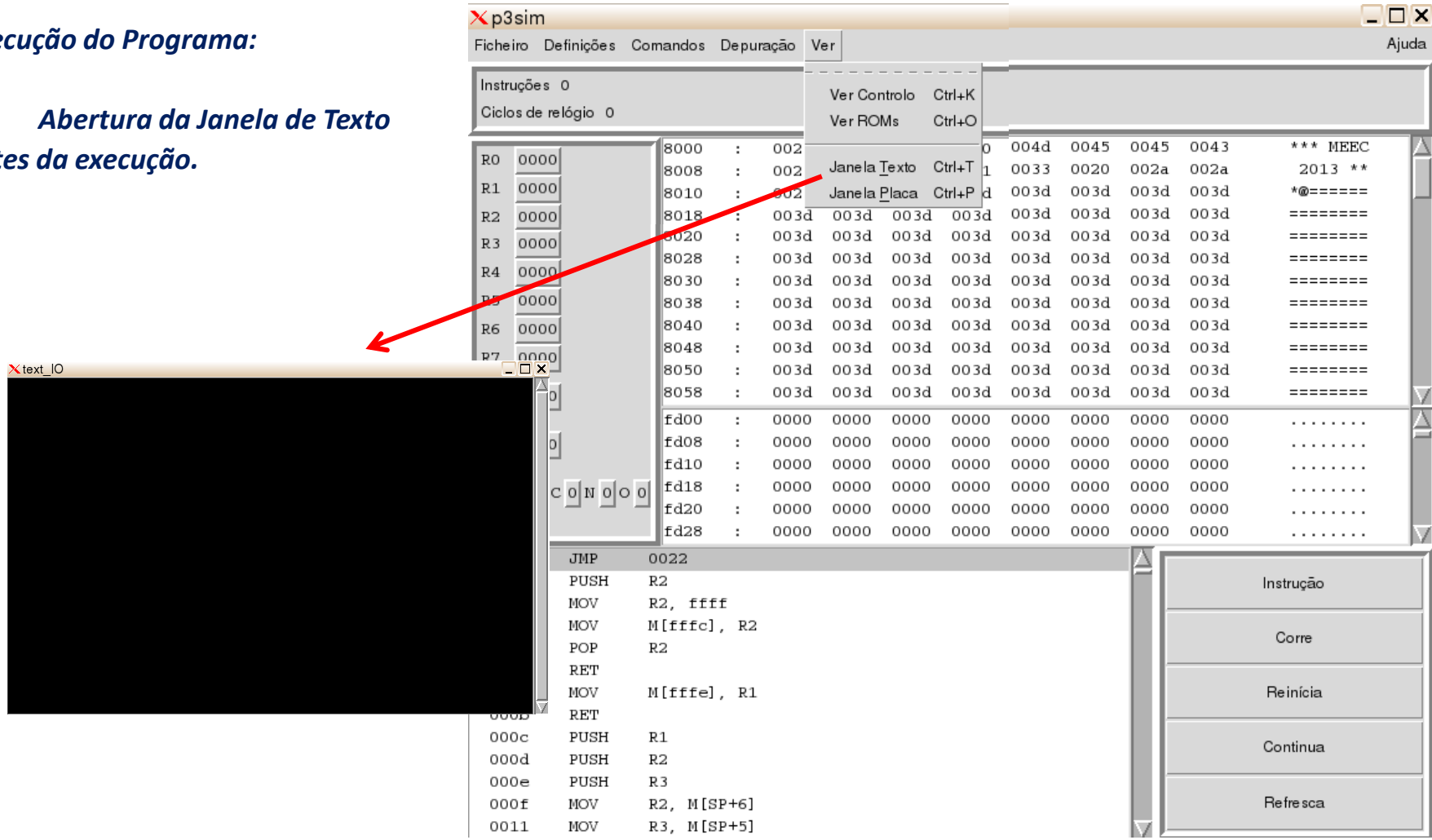
- ***Escreve 1 String > Limpa Janela > Escreve 2 Strings***

```
;==============================================================================
;                              Programa prinicipal
;==============================================================================
inicio:         MOV     R1, SP_INICIAL
                MOV     SP, R1
                CALL    LimpaJanela
                MOV     R1, XY_INICIAL
                PUSH    VarTexto1           ; Passagem de parametros pelo STACK
                PUSH    R1                  ; Passagem de parametros pelo STACK
                CALL    EscString
                CALL    LimpaJanela
                MOV     R1, 0000h
                PUSH    VarTexto2           ; Passagem de parametros pelo STACK
                PUSH    R1                  ; Passagem de parametros pelo STACK
                CALL    EscString
                ADD     R1, 0200h
                PUSH    VarTexto3           ; Passagem de parametros pelo STACK
                PUSH    R1                  ; Passagem de parametros pelo STACK
                CALL    EscString
Fim:            BR      Fim
;==============================================================================
```

P3 - PROCESSADOR Periféricos – Janela de Texto

Execução do Programa:

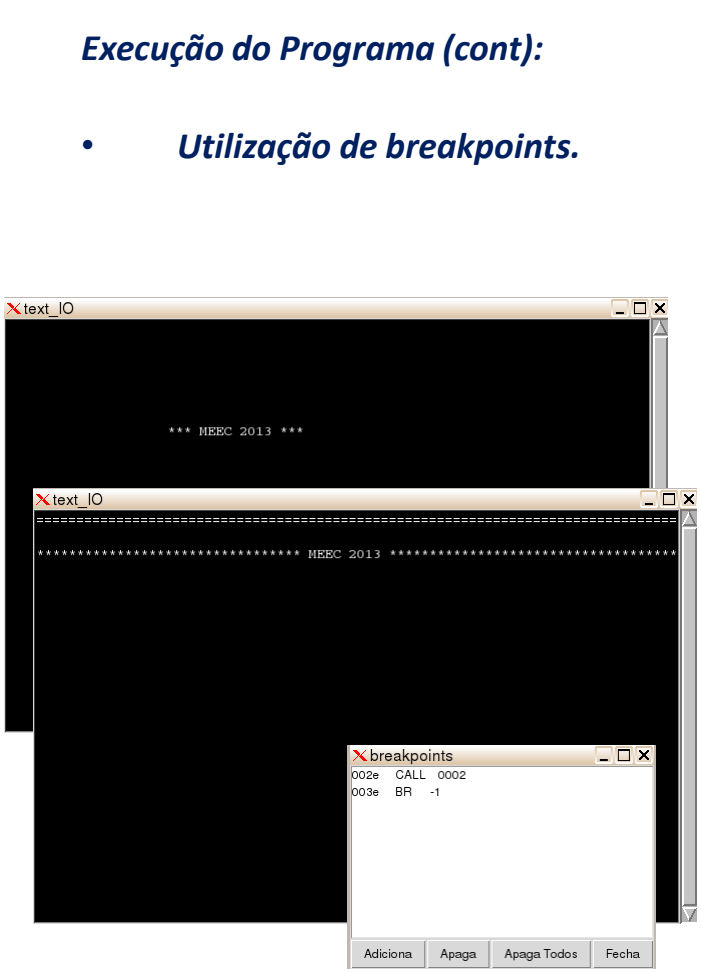
- ***Abertura da Janela de Texto antes da execução.***

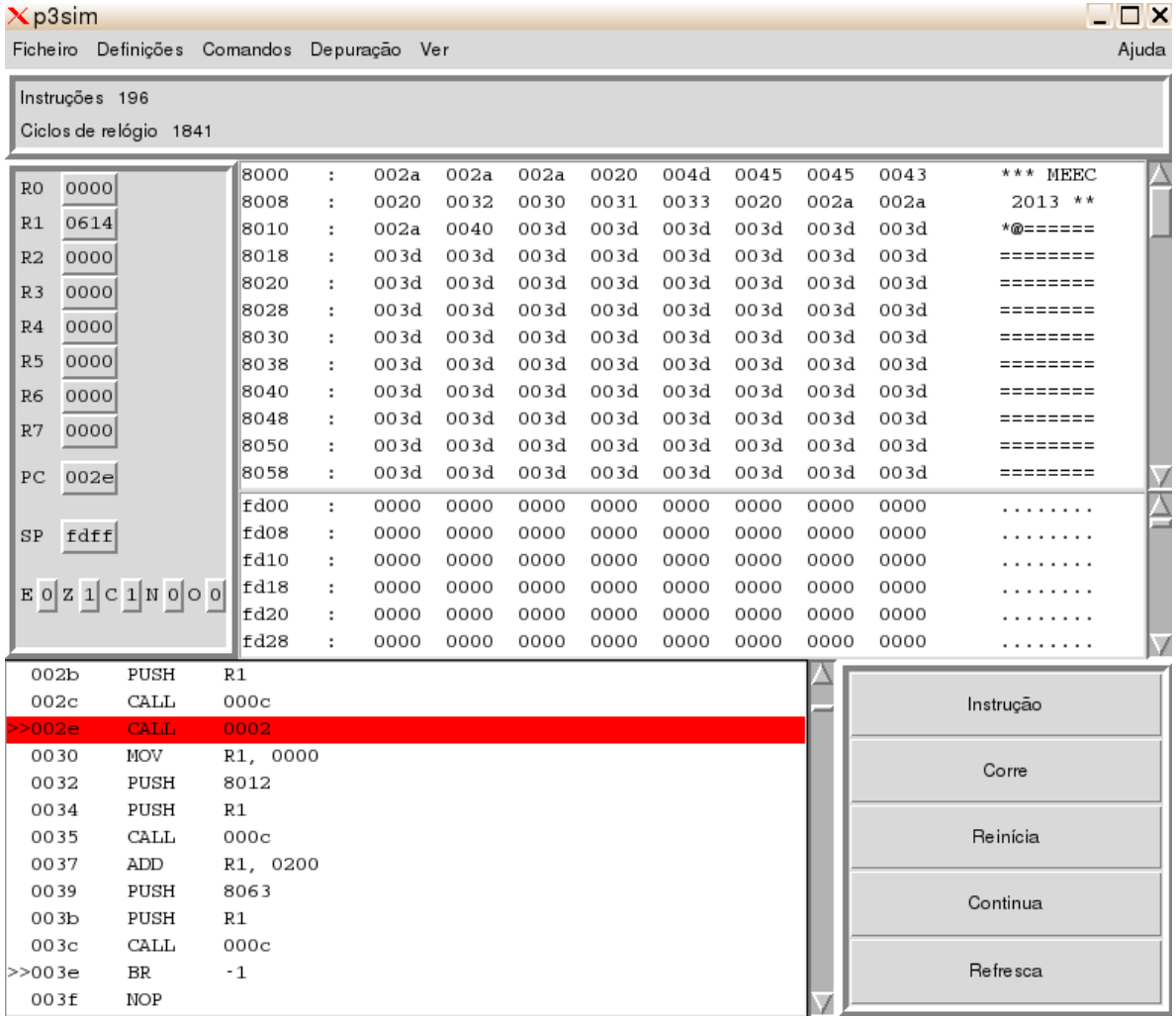


P3 - PROCESSADOR Periféricos – Janela de Texto

Execução do Programa (cont):

- ***Utilização de breakpoints.***





P3 - PROCESSADOR Periféricos – Display 7 Seg.

Display de 7 Segmentos: FFF0h a FFF3h

- ***FFF0h – (Display 7 segmentos 0) Permite escrever no display de 7 segmentos mais à direita. Só são considerados os 4 bits menos significativos escritos no endereço.***
- ***FFF1h – (Display 7 segmentos 1)***
- ***FFF2h – (Display 7 segmentos 2)***
- ***FFF3h – (Display 7 segmentos 3)***

```
;*********************************************************************************
; Programa Display7seg.as
;
; Descricao: Demonstracao da utilizacao dos interruptores
;
; Autor: Nuno Horta
; Data: 28/05/2013                                  Ultima Alteracao:28/05/2013
;*********************************************************************************
;*********************************************************************************
; ZONA I: Definicao de constantes
;         Pseudo-instrucao : EQU
;*********************************************************************************

; TEMPORIZACAO
DELAYVALUE      EQU     0300h

; STACK POINTER
SP_INICIAL      EQU     FDFFh

; I/O a partir de FF00H
DISP7S1         EQU     FFF0h
DISP7S2         EQU     FFF1h
DISP7S3         EQU     FFF2h
LCD_WRITE       EQU     FFF5h
LCD_CURSOR      EQU     FFF4h
LEDS            EQU     FFF8h
INTERRUPTORES   EQU     FFF9h
```

P3 - PROCESSADOR

Periféricos – Display 7 Seg.

Zona de Código:

- ***Rotina EscDisplay utilizada para escrever no display selecionado.***
- ***Rotina Delay utilizada para provocar um atraso (ver interrupções para soluções Alternativas - Temporizador).***

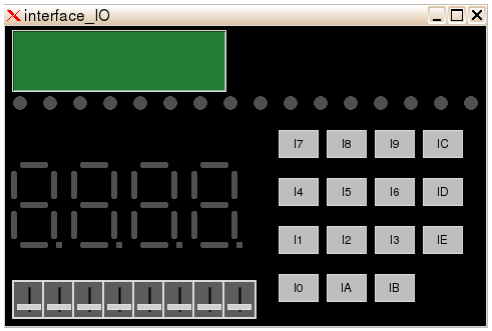
Programa Principal:

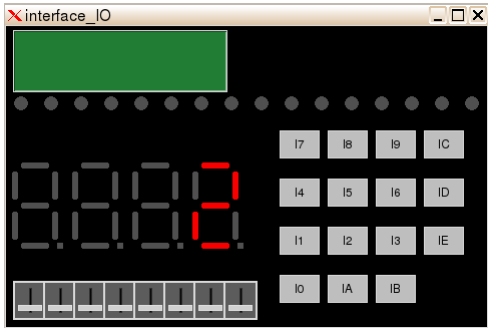
- ***Ciclo para escrita dos valores 1, 2, 4 e 8 no display da direita.***

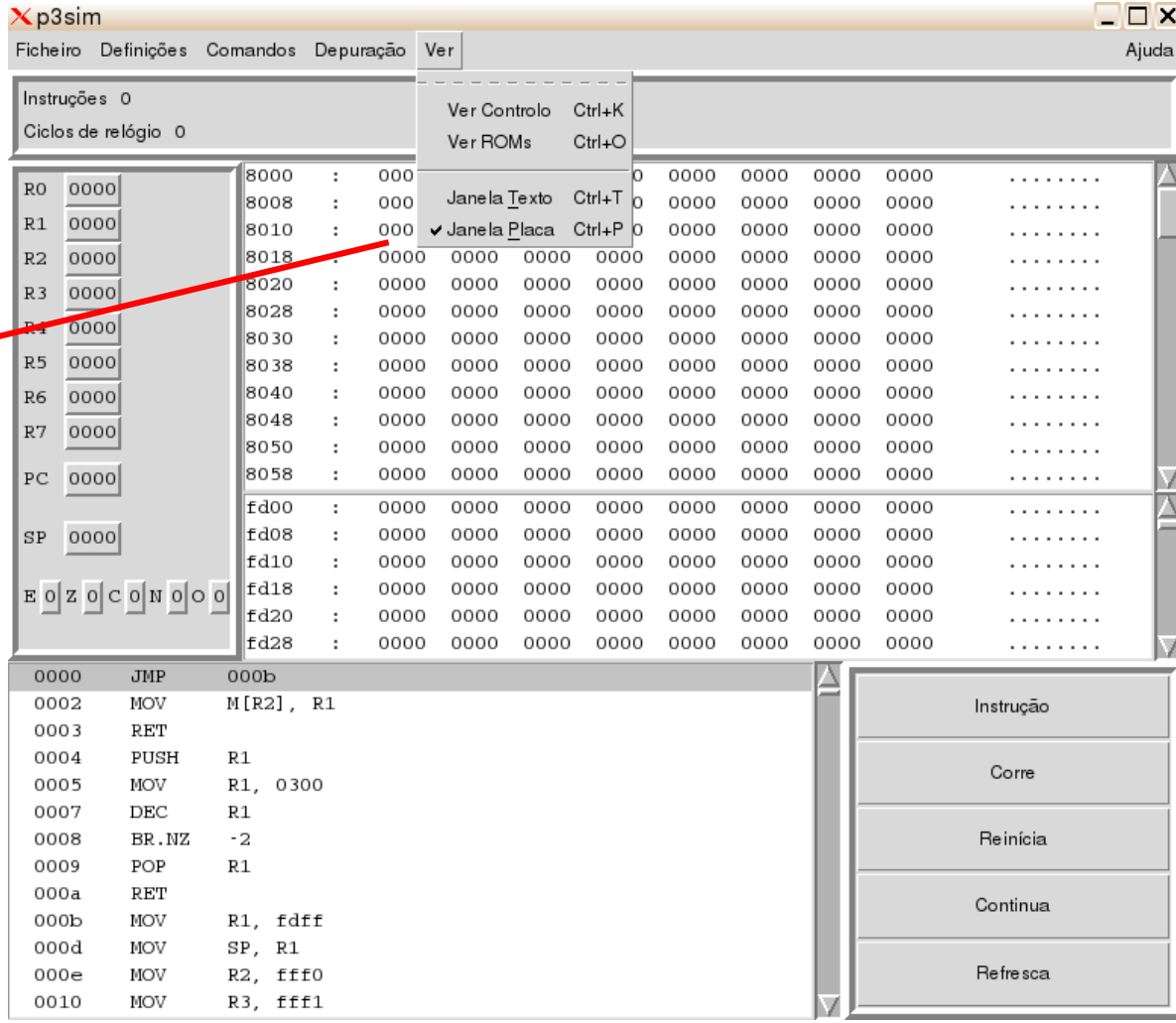
```
;=================================================================================
; ZONA III: Codigo
;           conjunto de instrucoes Assembly, ordenadas de forma a realizar
;           as funcoes pretendidas
;=================================================================================
                ORIG    0000h
                JMP     inicio

;=================================================================================
; EscDisplay: Rotina que efectua escrita no DISPLAY de 7 segmentos
;               Entradas: R1 - Valor a enviar para o porto do DISPLAY
;                         R2 - Porto do DISPLAY a utilizar
;               Saidas: ---
;               Efeitos: alteracao da posicao de memoria/porto M[R2]
;=================================================================================
EscDisplay:     MOV     M[R2], R1
                RET

;=================================================================================
; EscDisplay: Rotina que permite gerar um atraso
;               Entradas: ---
;               Saidas: ---
;               Efeitos: ---
;=================================================================================
Delay:          PUSH    R1
                MOV     R1, DELAYVALUE
DelayLoop:      DEC     R1
                BR.NZ   DelayLoop
                POP     R1
                RET
;=================================================================================
;                                   Programa prinicipal
;=================================================================================
inicio:         MOV     R1, SP_INICIAL
                MOV     SP, R1
                MOV     R2, DISP7S1
                MOV     R3, DISP7S2
ResetCont:      MOV     R1, 0001H
CicloCont:      CALL    EscDisplay          ; Passagem de parametros por registo
                CALL    Delay
                ROL     R1, 1
                TEST    R1, 0010H
                BR.Z    CicloCont
                BR      ResetCont
```

P3 - PROCESSADOR Periféricos – Display 7 Seg.

Execução do Programa







P3 - PROCESSADOR Periféricos – Interruptores

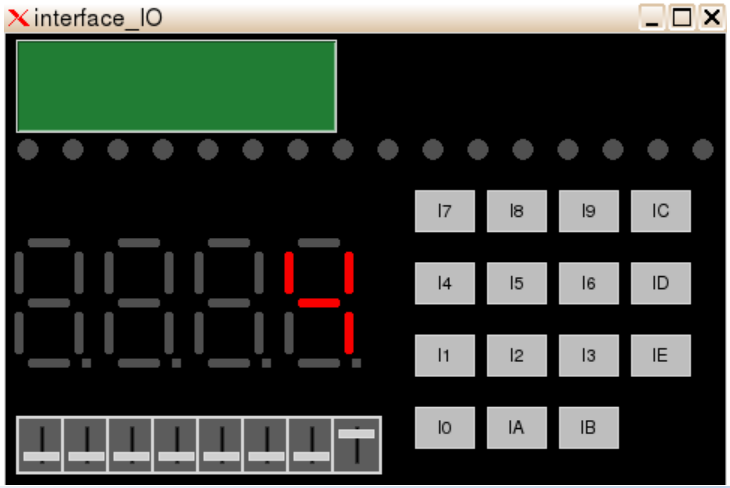
Interruptores: FFF9h

- *FFF9h - Permite ler, nos 8 bits menos significativos, o valor definido pela posição dos interruptores. O interruptor da direita corresponde ao bit menos significativo.*

Execução do Programa:

- *A ativação do interruptor da direita bloqueia a contagem no display de 7 segmentos.*

```
;================================================================================
;                                Programa prinicipal
;================================================================================
inicio:         MOV     R1, SP_INICIAL
                MOV     SP, R1
                MOV     R2, DISP7S1
                MOV     R3, DISP7S2
ResetCont:      MOV     R1, 0001H
CicloCont:      CALL    EscDisplay          ; Passagem de parametros por registo
                CALL    Delay
Stop:           MOV     R5, M[INTERRUPTORES]
                TEST    R5, 0001H
                BR.NZ   Stop
                ROL     R1, 1
                TEST    R1, 0010H
                BR.Z    CicloCont
                BR      ResetCont
;================================================================================
```



P3 - PROCESSADOR Periféricos – LEDs

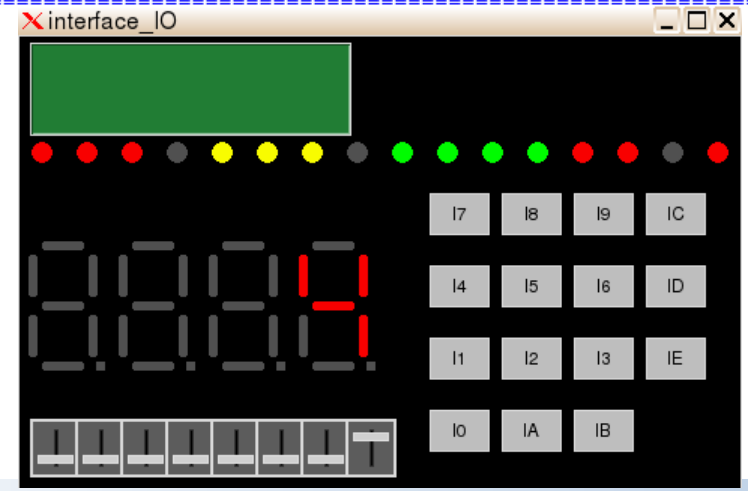
LEDs: FFF8h

- ***Permite acender os LEDs correspondentes ao valor em binário que se escreve no endereço. O LED da direita corresponde ao bit menos significativo.***

Execução do Programa:

- ***A contagem em R4 é representada nos LEDs***

```
;================================================================================
;                                Programa prinicipal
;================================================================================
inicio:         MOV     R1, SP_INICIAL
                MOV     SP, R1
                MOV     R2, DISP7S1
                MOV     R3, DISP7S2
                MOV     R4, 0001H
ResetCont:      MOV     R1, 0001H
CicloCont:      CALL    EscDisplay          ; Passagem de parametros por registo
                CALL    Delay
Stop:           MOV     R5, M[INTERRUPTORES]
                INC     R4
                MOV     M[LEDS], R4
                TEST    R5, 0001H
                BR.NZ   Stop
                ROL     R1, 1
                TEST    R1, 0010H
                BR.Z    CicloCont
                BR      ResetCont
;================================================================================
```



P3 - PROCESSADOR

Periféricos – LCD

LCD: FFF4h a FFF5h

- *FFF4h – Permite enviar sinais de controlo para o LCD. (Bit 15 ativa LCD, Bit 5 Limpa LCD, Bit 4 representa a linha, Bits 3-0 indicam a coluna.*

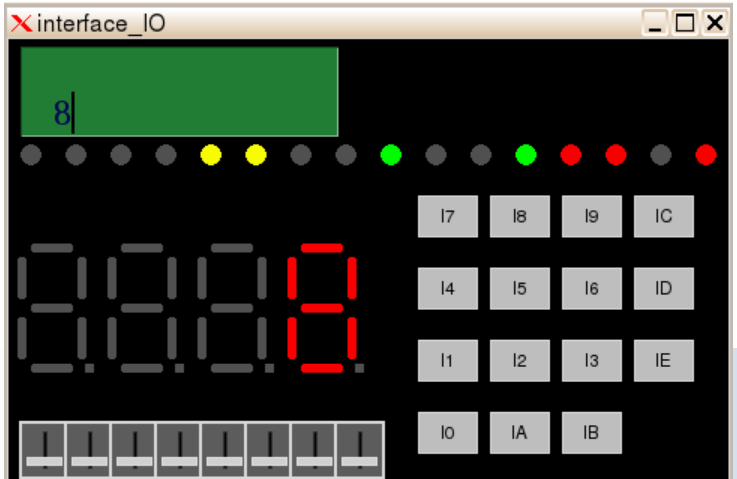
- *FFF5h – Permite escrever um caracter no LCD cujo código ASCII foi escrito no endereço.*

Zona de Código:

- *Rotina CLRLCD utilizada para limpar LCD*

- *Rotina EscLCD utilizada para escrever no LCD*

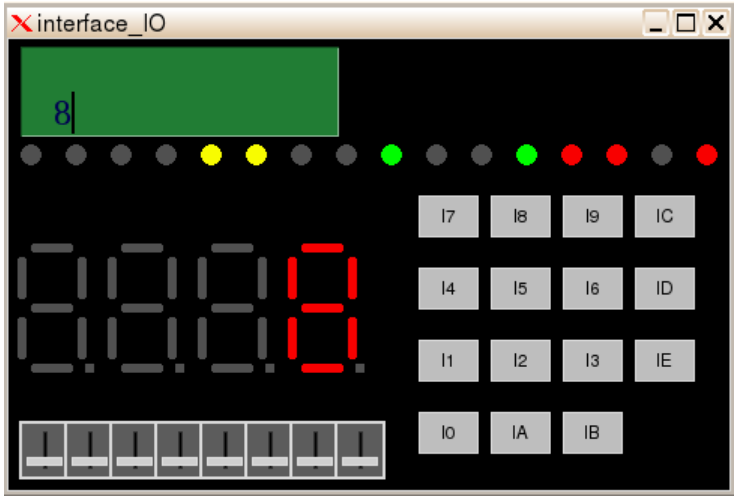
```
;*****************************************************************************
; CLRLCD: Rotina para limpar LCD
;               Entradas:
;               Saidas:
;               Efeitos:
;*****************************************************************************
CLRLCD:         PUSH    R1
                MOV     R1, 8020h
                MOV     M[LCD_CURSOR], R1
                POP     R1
                RET
;*****************************************************************************
; ESCLCD: Rotina para escrita no LCD
;               Entradas: R4, R1
;               Saidas:
;               Efeitos:
;*****************************************************************************
EscLCD:         PUSH    R1
                PUSH    R2
                MOV     R1, M[SP+4]
                MOV     R2, M[SP+5]
                MOV     M[LCD_CURSOR], R1
                ADD     R2, 0030h       ; Converte valor para ASCII
                MOV     M[LCD_WRITE], R2
                POP     R2
                POP     R1
                RETN    2
```



P3 - PROCESSADOR Periféricos – LCD

Execução do Programa

- ***Escreve no LCD o mesmo valor que é escrito no display de 7 segmentos.***




```
;=============================================================================
;                              Programa prinicipal
;=============================================================================
inicio:         MOV     R1, SP_INICIAL
                MOV     SP, R1
                MOV     R2, DISP7S1
                MOV     R3, DISP7S2
                MOV     R4, 0001H
ResetCont:      MOV     R1, 0001H
CicloCont:      CALL    EscDisplay          ; Passagem de parametros por registo
                CALL    Delay
Stop:           MOV     R5, M[INTERRUPTORES]
                INC     R4
                MOV     M[LEDS], R4
                TEST    R5, 0001H
                BR.NZ   Stop
                PUSH    R1
                PUSH    8014h
                CALL    EscLCD
                ROL     R1, 1
                TEST    R1, 0010H
                BR.Z    CicloCont
                CALL    CLRLCD
                BR      ResetCont
;=============================================================================
```

P3 - PROCESSADOR Interrupções

Interrupções e Temporizador

- *As interrupções são simuladas através dos botões de pressão (interrupções de 0 a 14) e do temporizador (interrupção 15).*
- *A cada interrupção ativa deve corresponder uma rotina de atendimento da interrupção.*
- *A localização (endereço) das rotinas de interrupção deve ser guardado na Tabela de Vectores de Interrupção (TVI). A TVI corresponde às posições de memória FE00h a FE0Fh, correspondendo o endereço FE00h à posição de memória onde é guardado o endereço de atendimento da interrupção 0, e assim sucessivamente.*
- *A Máscara de Interrupções (FFFAh) permite selecionar as interrupções que se pretendem aceitar (cada interrupção é permitida colocando a 1 o respetivo bit da posição de memória indicada por este endereço)*
- *As interrupções só são aceites depois de se fazer o enable das interrupções, executando a instrução ENI.*

- *Temporizador: FFF6h a FFF7h*
  - *FFF6h - uma escrita para este endereço define o número de unidades de contagem, cada com a duração de 100ms. Por exemplo, para se ter um intervalo de 1s, deve ser escrito para endereço o valor 10. Uma leitura deste endereço permite obter o valor atual de contagem;*
  - *FFF7h - este porto permite dar início ou parar uma contagem por escrita, respetivamente, de um 1 ou um 0 no bit menos significativo (os restantes bits são ignorados). Uma leitura deste endereço indica, no bit menos significativo, o estado do temporizador, em contagem ou parado.*

P3 - PROCESSADOR Interrupções

Zona de Código:

- ***Inicialização da TVI*** – *Coloca os endereços das rotinas nas respetivas posições da TVI*

- ***Inicialização da Máscara de Interrupções*** – *Permite apenas as interrupções 15 (temporizador) e as interrupções 0 e 1 dos botões de pressão.*

- ***Inicialização do Temporizador*** – *inicializa o temporizador com 2s e inicia a contagem.*

- ***Enable das Interrupções*** – *só depois desta instrução é que são aceites interrupções.*

```
;==============================================================================
;InitInt: Inicializa TVI, Mascara de INT e Temporizador
;               Entradas: ---
;               Saidas: ---
;               Efeitos: ---
;==============================================================================
InitInt:        MOV     R1, RotinaInt0
                MOV     M[TAB_INT0], R1
                MOV     R1, RotinaInt1
                MOV     M[TAB_INT1], R1
                MOV     R1, RotinaIntTemp
                MOV     M[TAB_INTTemp], R1
                MOV     R1,8003h
                MOV     M[MASCARA_INT], R1
                MOV     R1,0020h
                MOV     M[TempValor], R1
                MOV     R1,0001h
                MOV     M[TempControlo], R1
                ENI
                RET
;==============================================================================
```

P3 - PROCESSADOR Interrupções

Zona de Código:

- ***RotinaInt0*** *– Troca o display de 7 segmentos onde se faz a escrita do valor da contagem.*
- ***RotinaInt1*** *– Acende todos os LEDs*
- ***RotinaIntTemp*** *– Acende os LEDs centrais e reprograma o temporizador.*

```
;=================================================================================
; RotinaInt0: Rotina de interrupcao 0
;               Entradas: R2, R3 - Portos dos Display de 7 Segmentos
;               Saidas:   R2, R3
;               Efeitos: (1) Comutacao de R2 e R3
;=================================================================================
RotinaInt0:     XCH     R2, R3
                RTI

;=================================================================================
; RotinaInt1: Rotina de interrupcao 1
;               Entradas: R4
;               Saidas:   R4
;               Efeitos: (1) Acende todos os LEDS
;=================================================================================
RotinaInt1:     MOV     R4, FFFFh
                MOV     M[LEDS], R4
                RTI
;=================================================================================
; RotinaIntTemp: Rotina de interrupcao do temporizador
;               Entradas: R4
;               Saidas:   R4
;               Efeitos: (1) Activa Leds com valor de R4
;                        (2) Reprograma temporizador
;=================================================================================
RotinaIntTemp:  PUSH    R1
                MOV     R4,0FF0h
                MOV     M[LEDS], R4
                MOV     R1,0020h
                MOV     M[TempValor], R1
                MOV     R1,0001h
                MOV     M[TempControlo], R1
                POP     R1
                RTI
;=================================================================================
```

P3 - PROCESSADOR Interrupções

Programa Principal

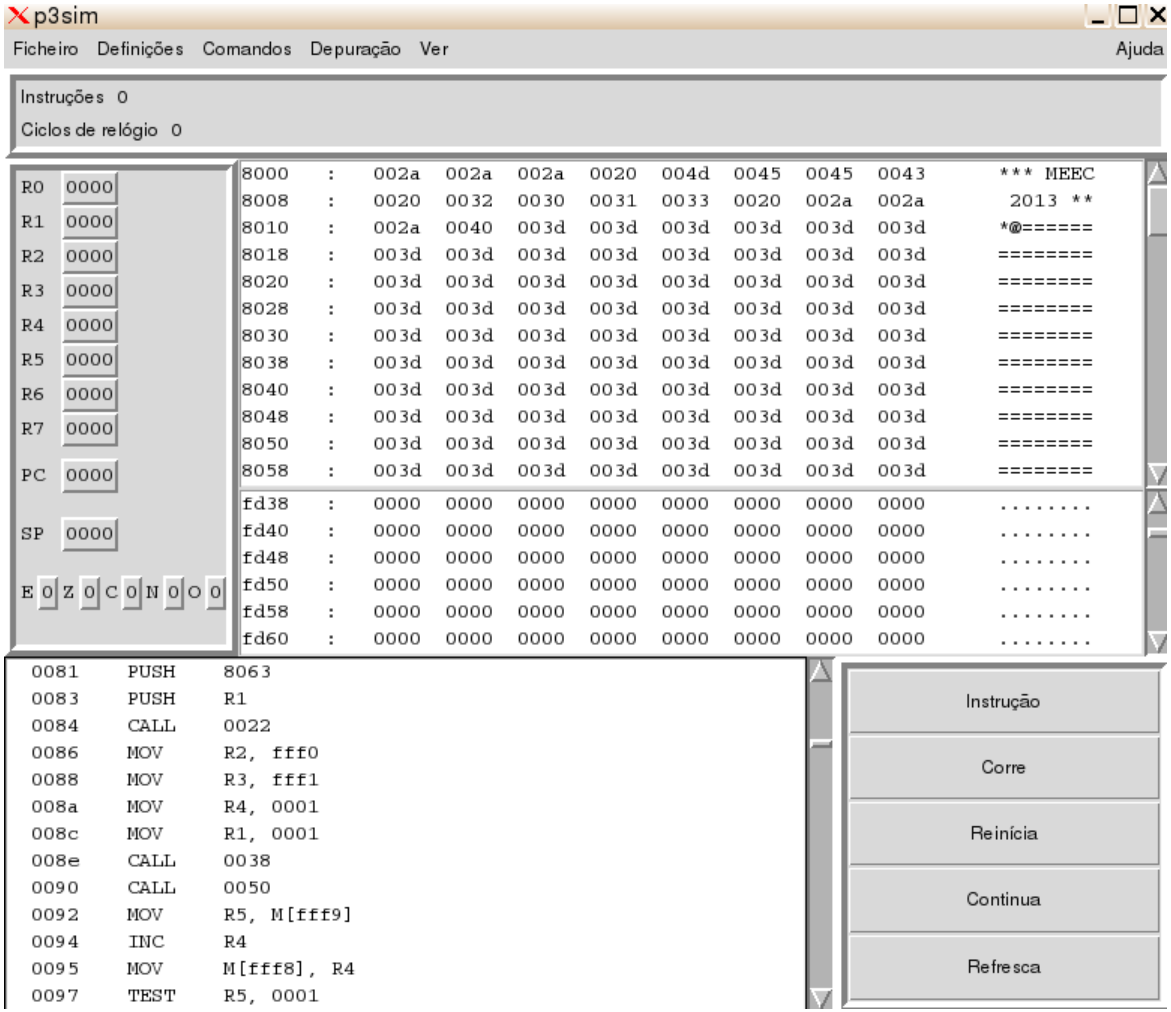
- *Inicialização do STACK*
- *Inicialização das interrupções*
- *Escrita na Janela de Texto*
- *Escrita no Display de 7 seg.*
- *Escrita no LCD*
- *Utilização de Interruptores*
- *Ativação dos LEDs*

```
;*********************************************************************************
;                                Programa prinicipal
;*********************************************************************************
inicio:         MOV     R1, SP_INICIAL
                MOV     SP, R1
                CALL    InitInt
                CALL    LimpaJanela
                MOV     R1, 0000h
                PUSH    VarTexto2           ; Passagem de parametros pelo STACK
                PUSH    R1                  ; Passagem de parametros pelo STACK
                CALL    EscString
                ADD     R1, 0200h
                PUSH    VarTexto3           ; Passagem de parametros pelo STACK
                PUSH    R1                  ; Passagem de parametros pelo STACK
                CALL    EscString
                MOV     R2, DISP7S1
                MOV     R3, DISP7S2
                MOV     R4, 0001H
ResetCont:      MOV     R1, 0001H
CicloCont:      CALL    EscDisplay          ; Passagem de parametros por registo
                CALL    Delay
Stop:           MOV     R5, M[INTERRUPTORES]
                INC     R4
                MOV     M[LEDS], R4
                TEST    R5, 0001H
                BR.NZ   Stop
                PUSH    R1
                PUSH    8014h
                CALL    EscLCD
                ROL     R1, 1
                TEST    R1, 0010H
                BR.Z    CicloCont
                CALL    CLRLCD
                BR      ResetCont
;*********************************************************************************
```

P3 - PROCESSADOR Interrupções

Execução do Programa:

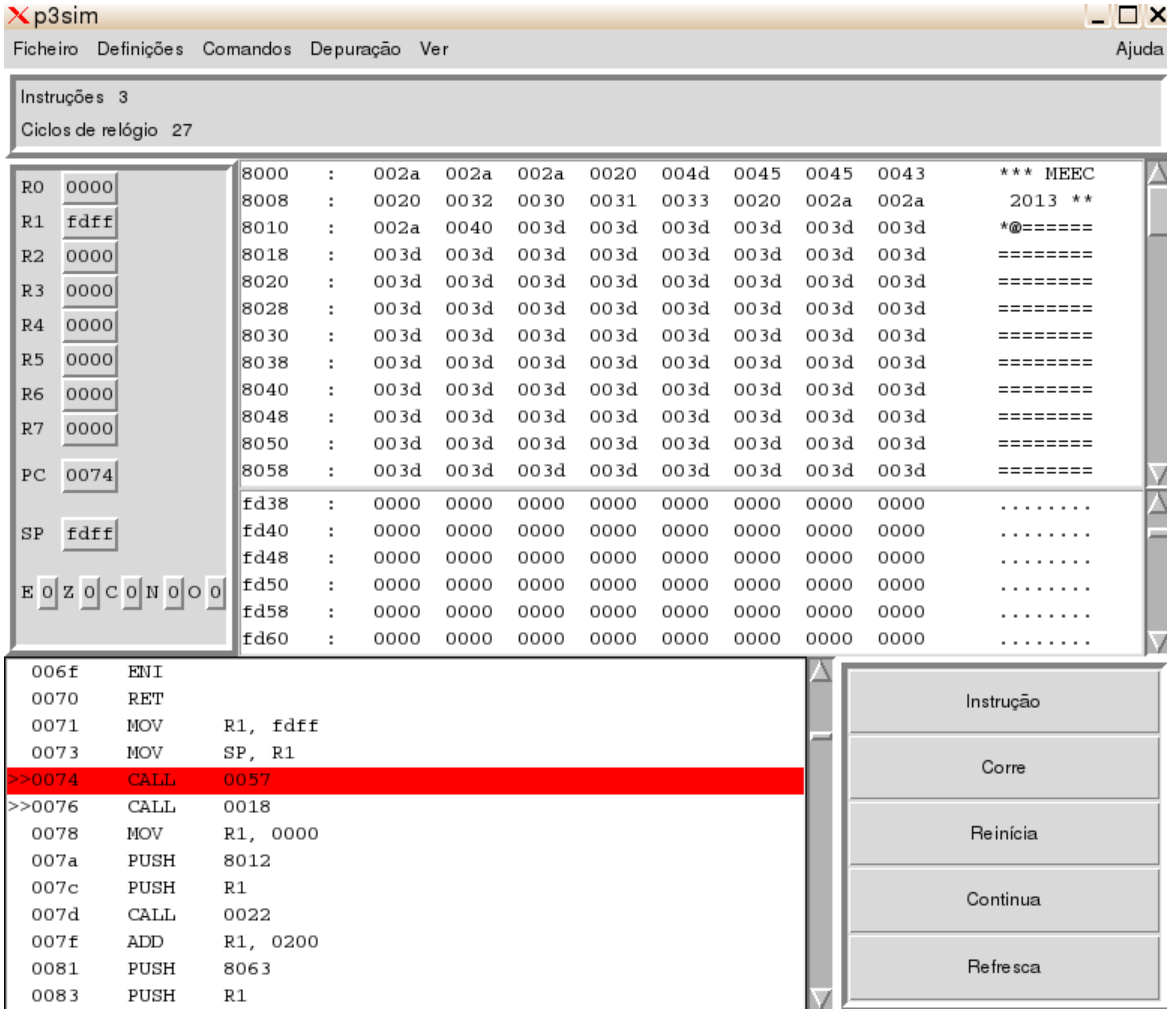
- ***Definição de Breakpoints***



P3 - PROCESSADOR Interrupções

Execução do Programa:

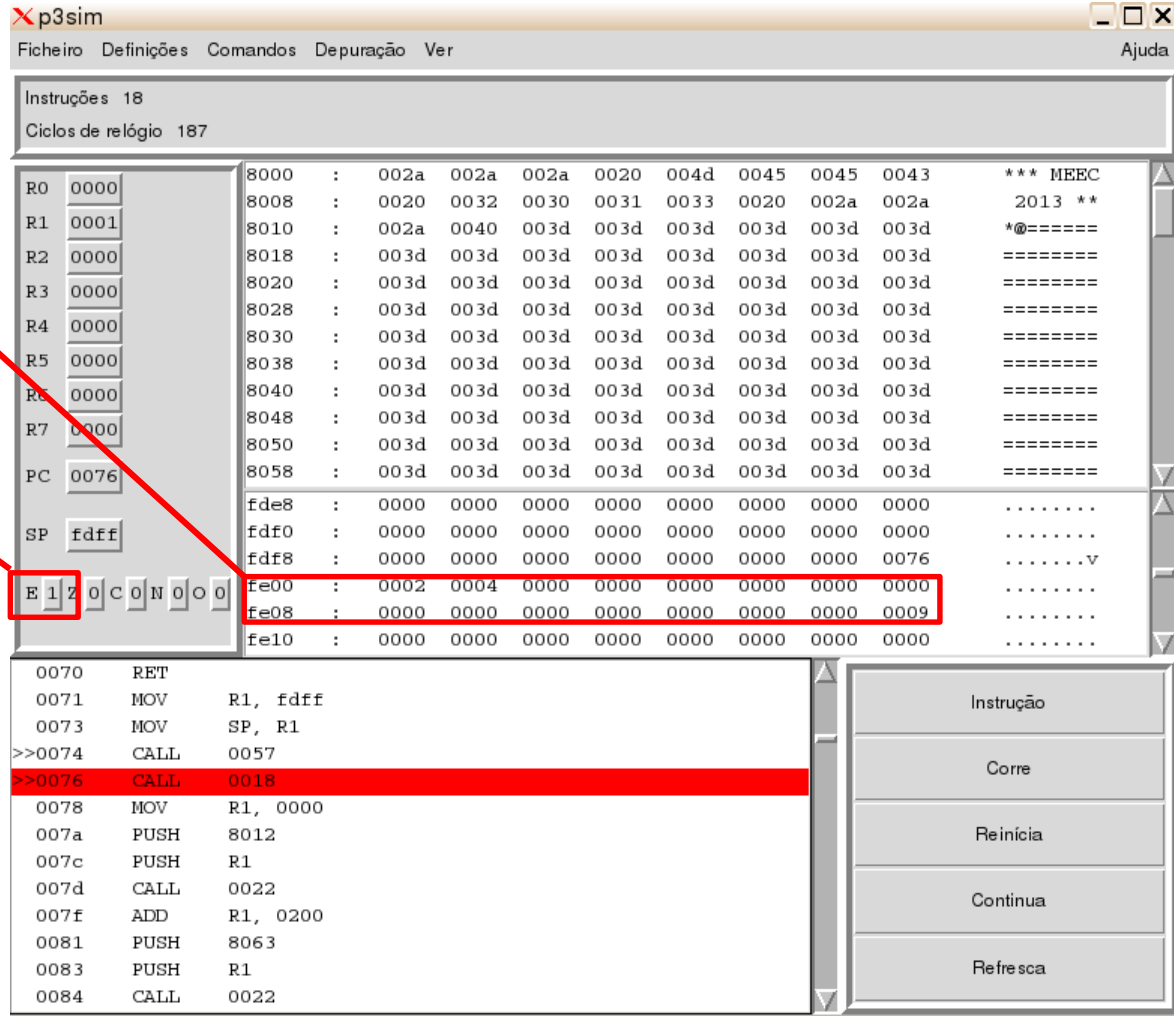
- ***Entrada na rotina de inicialização das interrupções***



P3 - PROCESSADOR Interrupções

Execução do Programa:

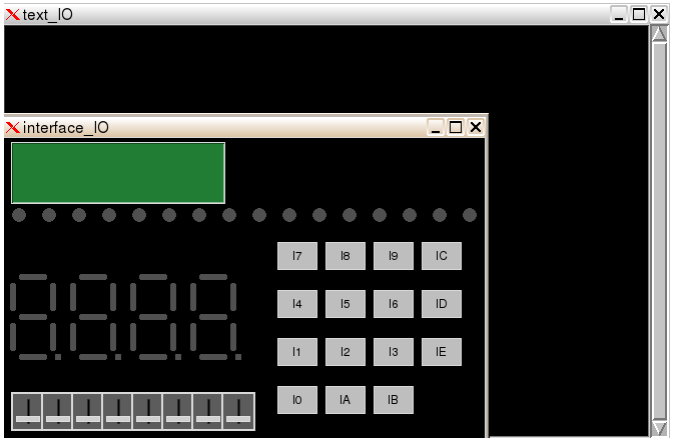
- ***TVI Programada na rotina de inicialização das interrupções.***
- ***FLAG E ativa***

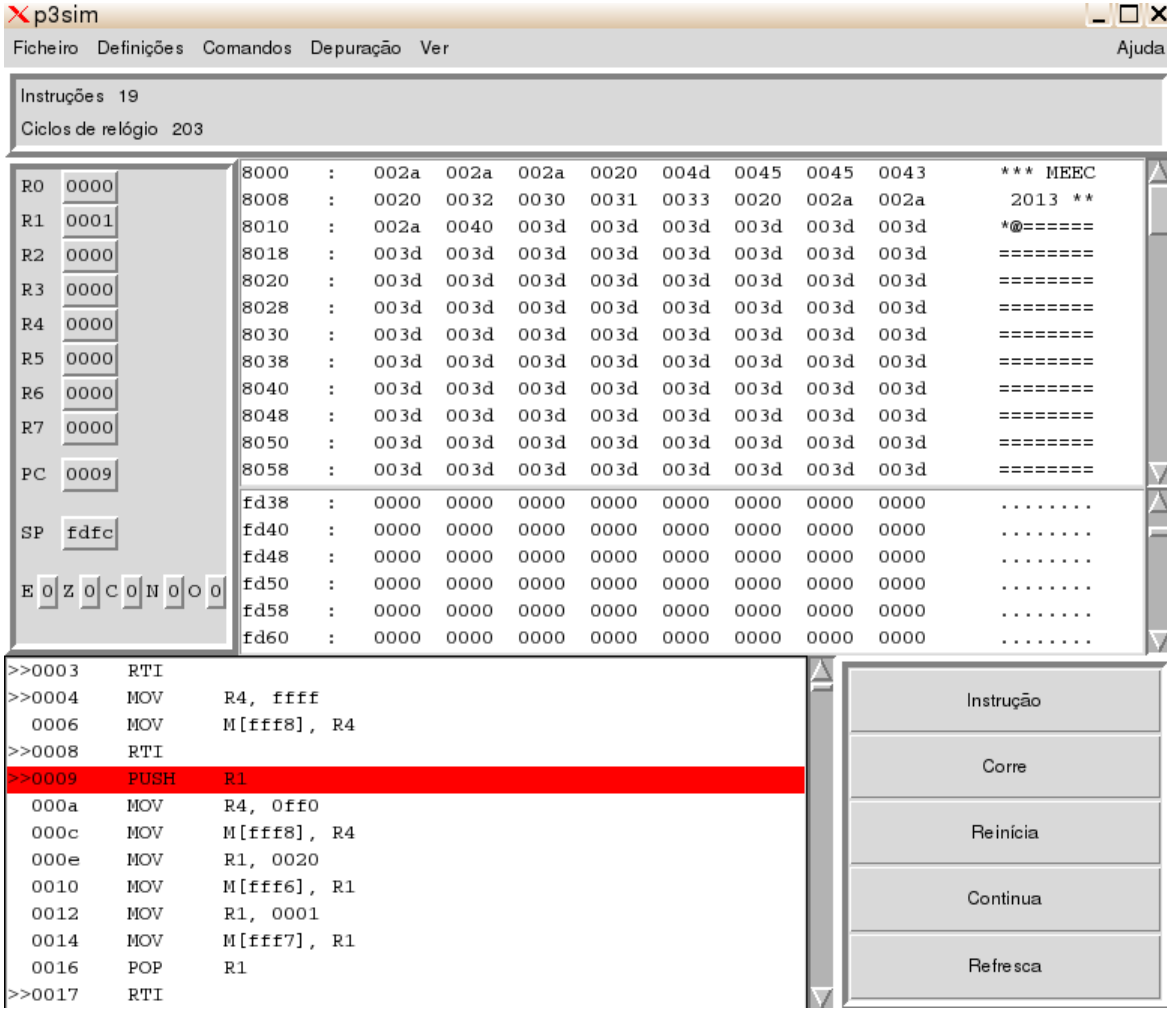


P3 - PROCESSADOR Interrupções

Execução do Programa:

- ***Entrada na rotina de atendimento do temporizador.***
- ***Flag E a 0, Interrupções desativadas enquanto executa rotina de atendimento a uma interrupção.***

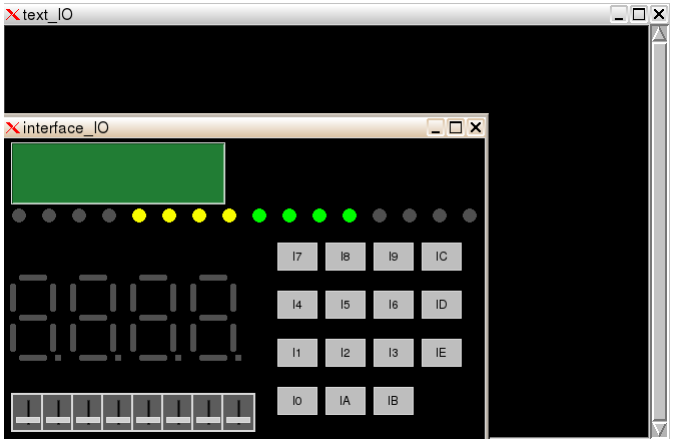


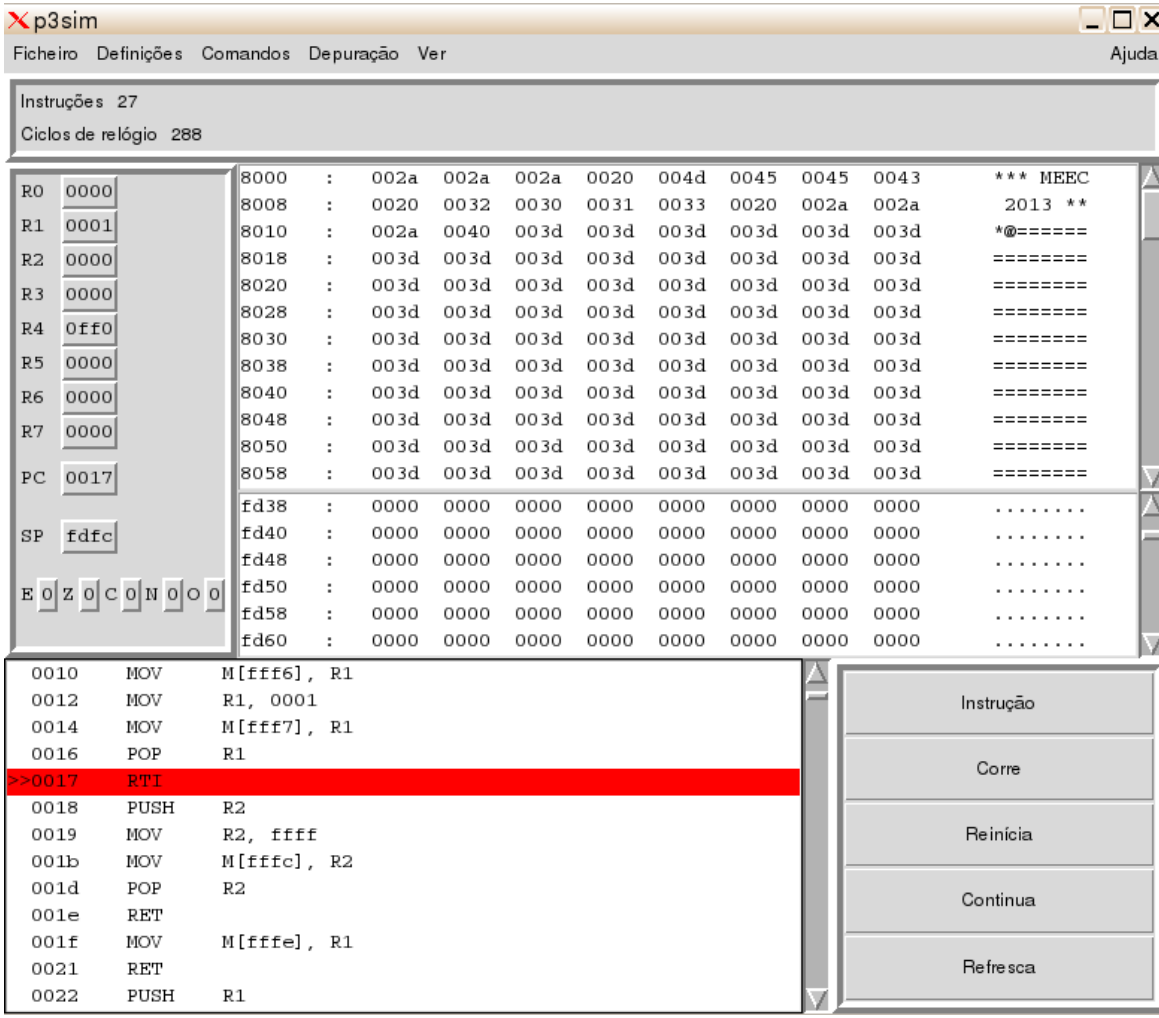


P3 - PROCESSADOR Interrupções

Execução do Programa:

- ***Ativação dos LEDS centrais e reprogramação do temporizador.***

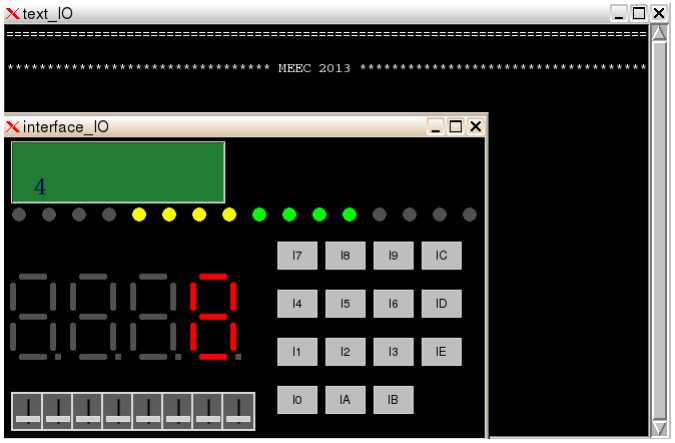


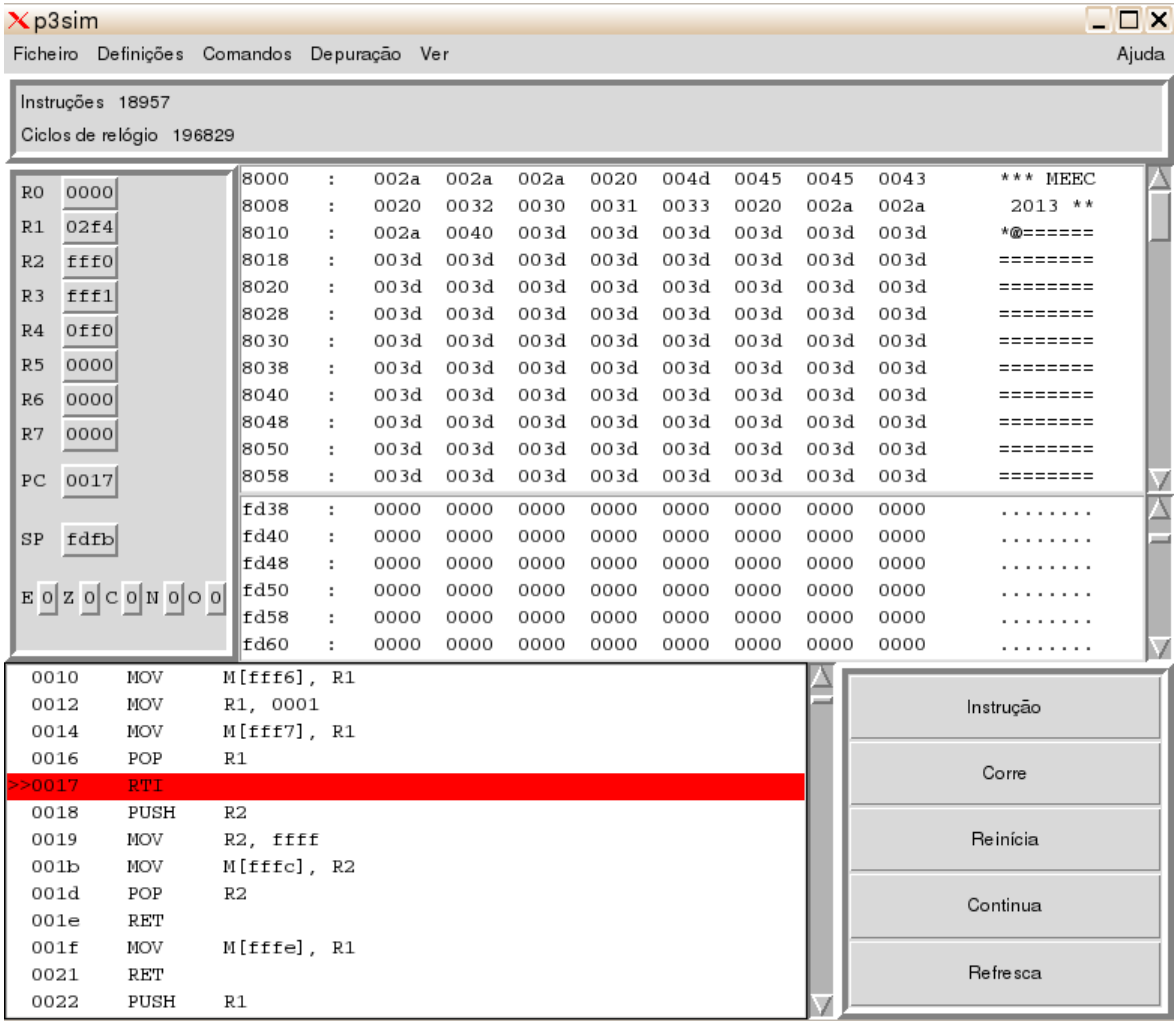


P3 - PROCESSADOR Interrupções

Execução do Programa:

- ***Entrada na rotina de atendimento do temporizador, após inicio da contagem (ver Display de 7 seg. e LCD).***

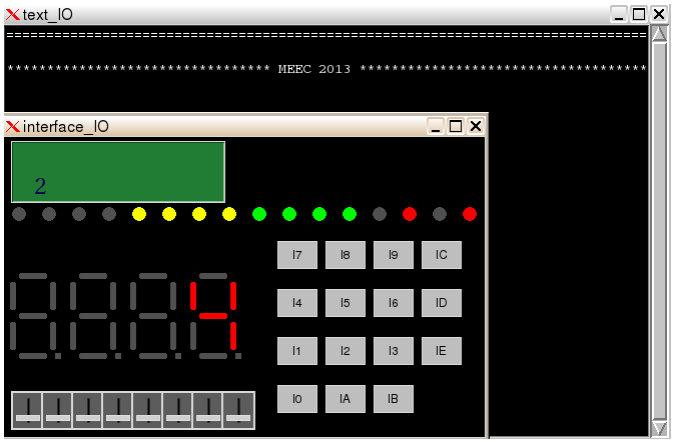


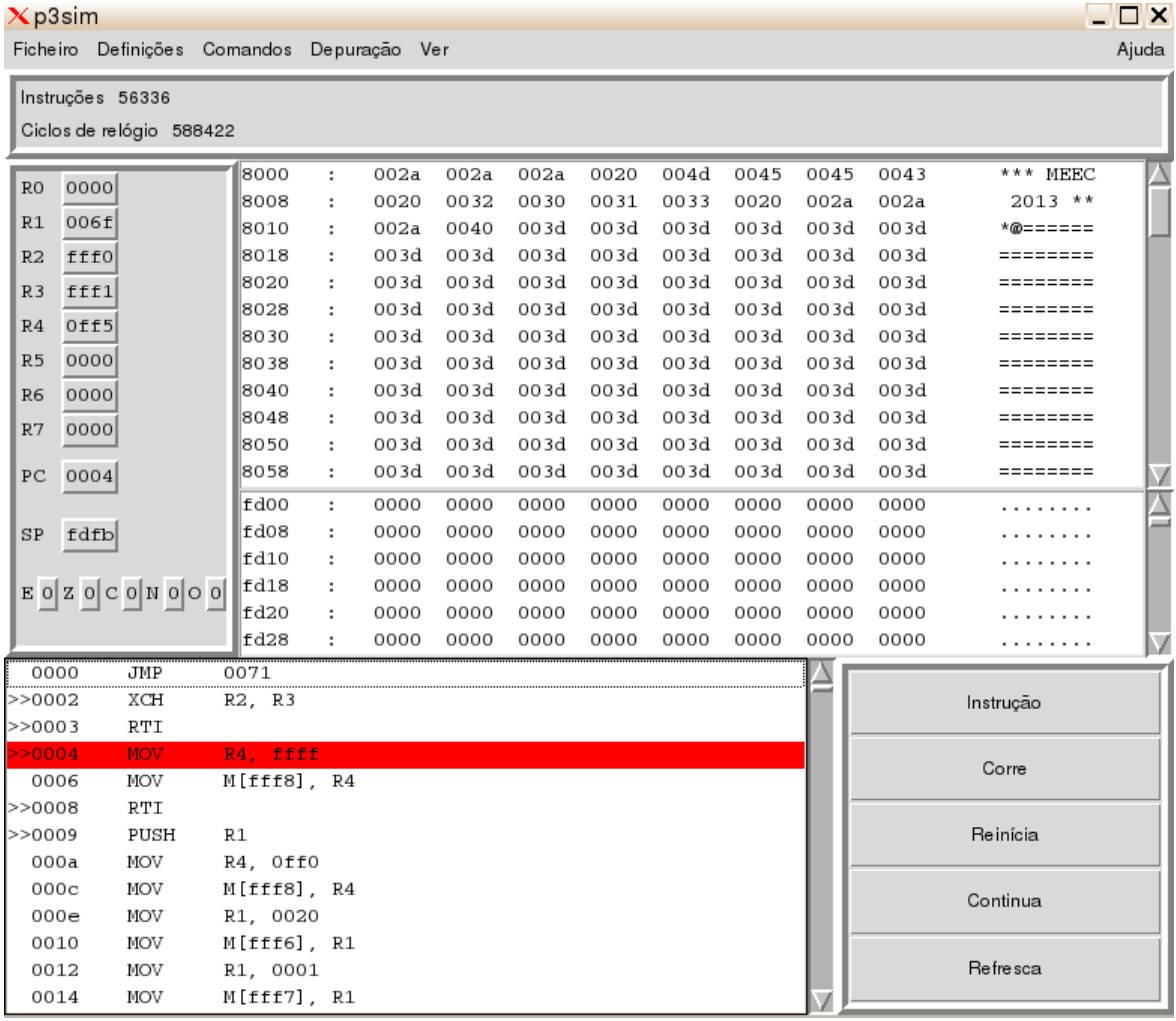


P3 - PROCESSADOR Interrupções

Execução do Programa:

- ***Entrada na rotina de atendimento à interrupção 1.***

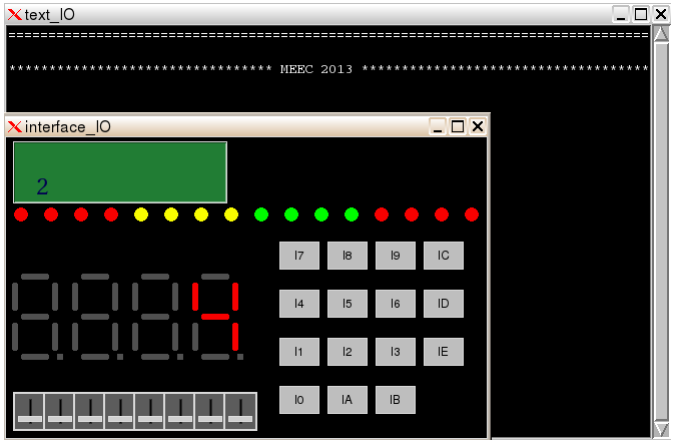


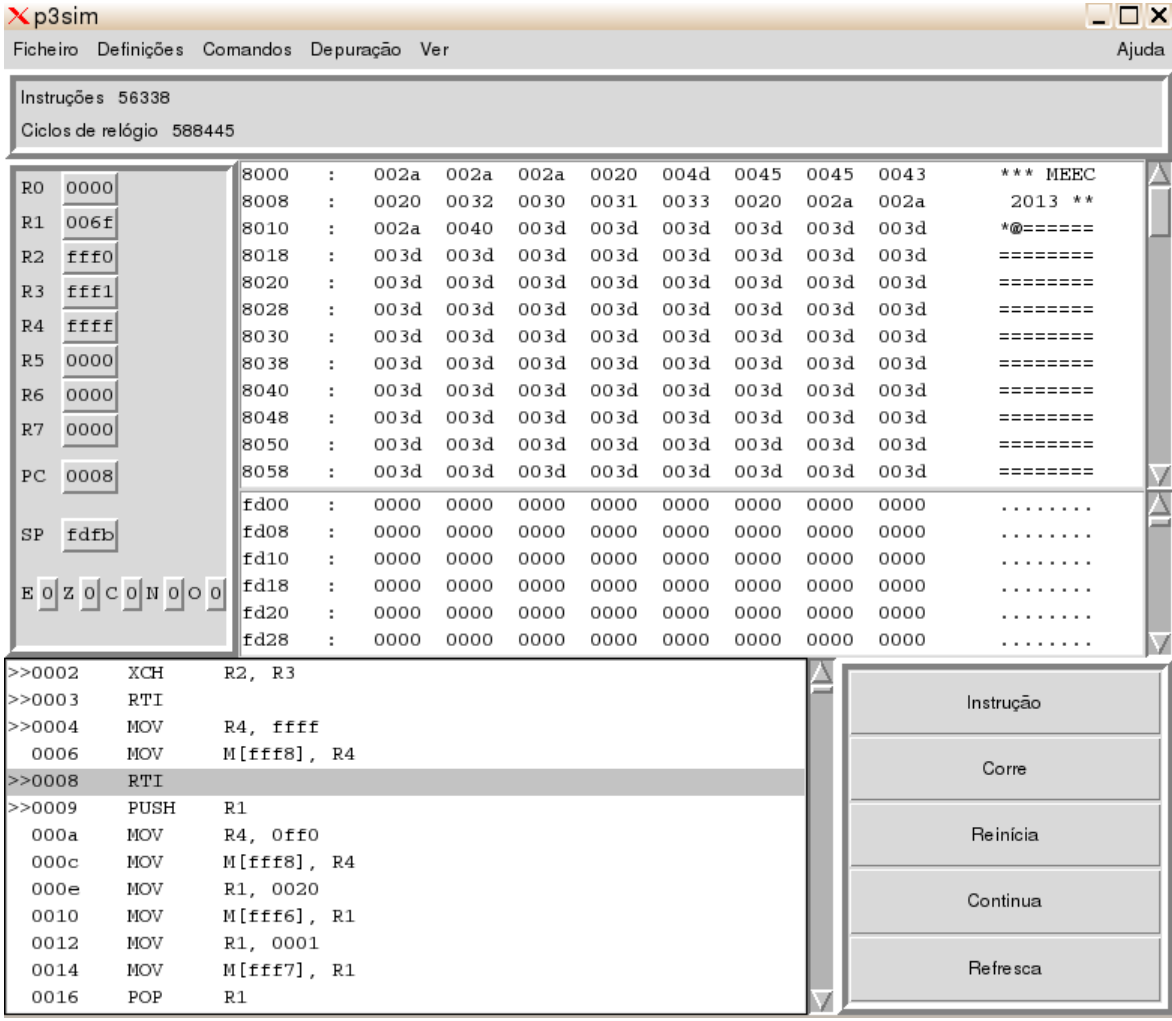


P3 - PROCESSADOR Interrupções

Execução do Programa:

- ***Ativação de todos os LEDS.***

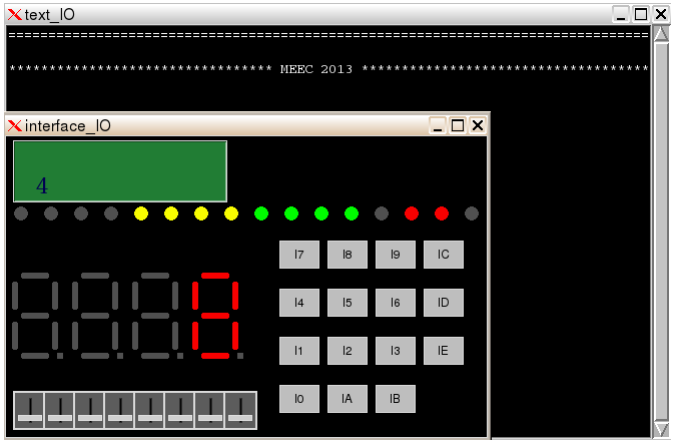


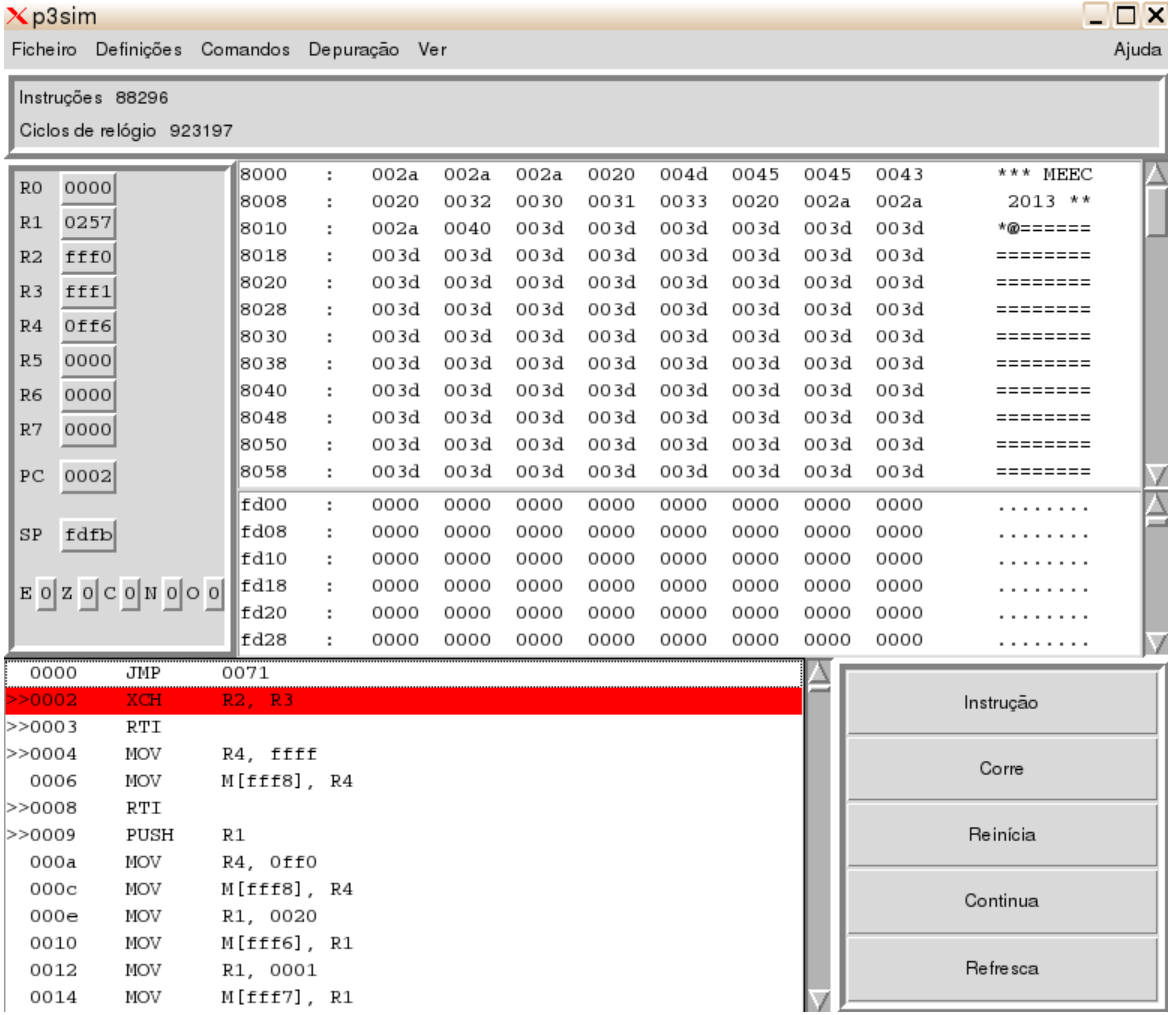


P3 - PROCESSADOR Interrupções

Execução do Programa:

- ***Entrada na rotina de atendimento à interrupção 0 para troca do Display de 7 seg.***

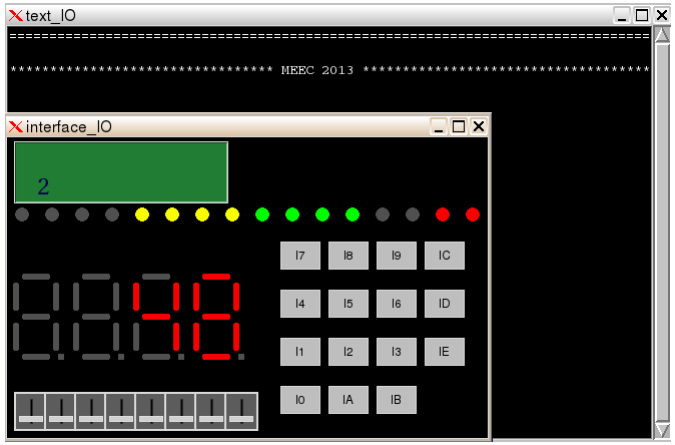


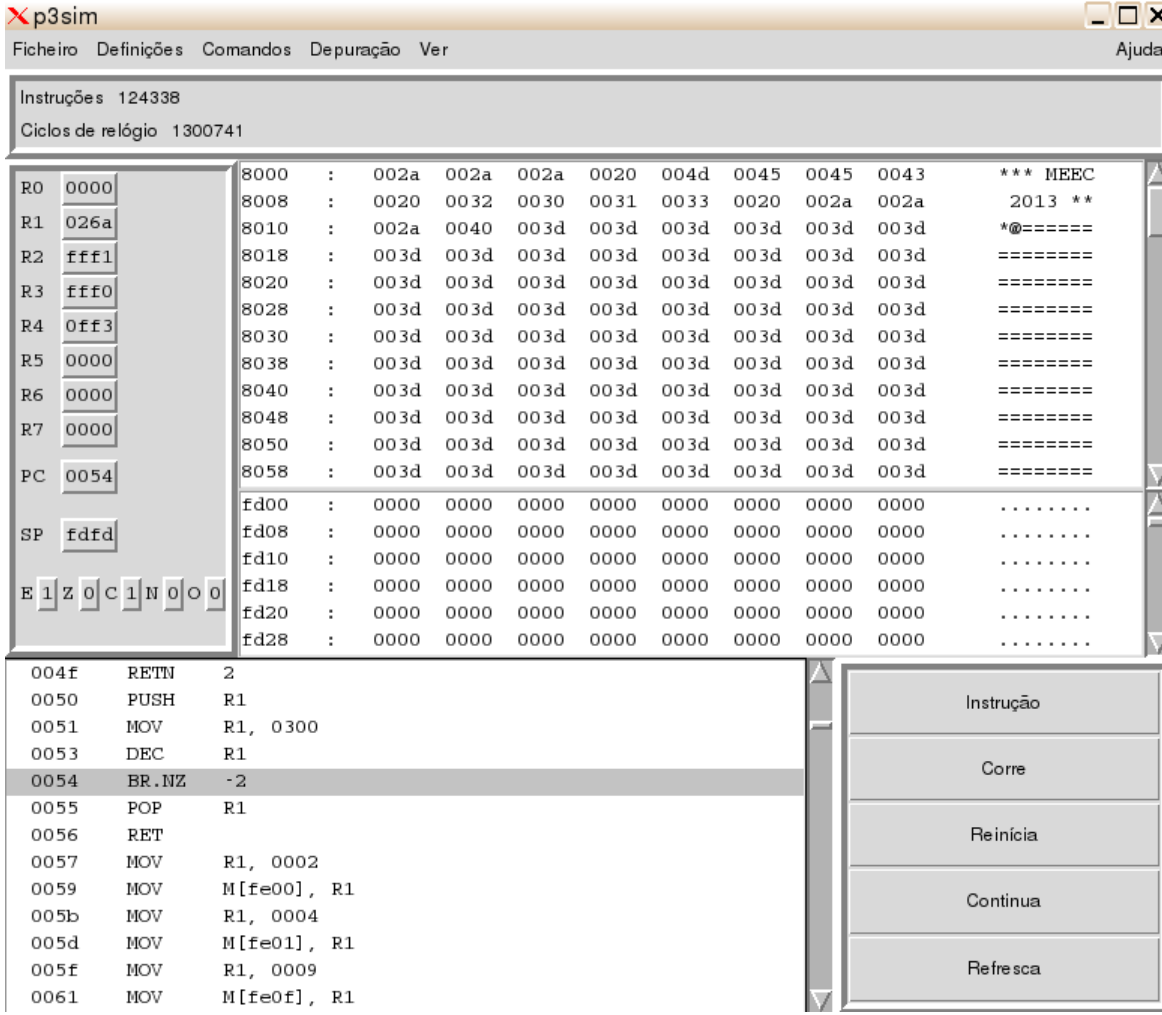


P3 - PROCESSADOR Interrupções

Execução do Programa:

- ***Escrita no 2º Display e FLAG E ativa fora da rotina de interrupção.***





P3 - PROCESSADOR BIBLIOGRAFIA

- **Bibliografia**

[1] M. Morris Mano, Charles R. Kime, “Logic and Computer Design Fundamentals”, Prentice-Hall International, Inc. (Capítulos 9, 10 e 11)

[2] N. Horta, “Arquitecturas de Computadores”, Aulas Teóricas, 2013.

[3] G.Arroz, J.C.Monteiro, A.Oliveira, “Manual do Simulador do P3”, IST, 2007

[4] G.Arroz, J.C.Monteiro, A.Oliveira, “Introdução aos Sistemas Digitais e Microprocessadores”, IST Press, 2007.