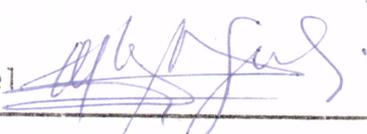




1. Classificação <i>INPE-COM. 3/NTE</i> <i>C.D.U. - 681.322</i>	2. Período	4. Critério de Distribuição:  interna <input type="checkbox"/> externa <input checked="" type="checkbox"/>
3. Palavras Chave (selecionadas pelo autor)		
5. Relatório nº <i>INPE-1028-NTE/086</i>	6. Data <i>Mai de 1977</i>	7. Revisado por - <i>Ricardo C.O. Martins</i>
8. Título e Sub-Título <i>Um Simulador Reconfigurável para o Microprocessador INTEL 8080</i>		9. Autorizado por - <i>Dr. Nelson Jesus Parada</i> Diretor
10. Setor <i>CCI</i>	Código <i>460</i>	11. Nº de cópias <i>10</i>
12. Autoria <i>Wilson Yamaguti</i> <i>Ricardo Correa de Oliveira Martins</i>		14. Nº de páginas <i>131</i>
13. Assinatura Responsável 		15. Preço
16. Sumário/Notas <p><i>O propósito deste trabalho é a implementação de um simulador reconfigurável para o microprocessador INTEL 8080, utilizando a linguagem ALGOL e o sistema B-6700 do INPE. A simulação é feita de maneira específica e a um nível de detalhe bem maior que o usual, nível este necessário principalmente em aplicações em tempo real, onde o projeto do sistema de atendimento de interrupções é uma das tarefas principais e de mais difíceis verificação.</i></p>		
17. Observações		

## ÍNDICE

LISTA DE FIGURAS . . . . .	vii
LISTA DE TABELAS . . . . .	xi
CAPÍTULO I	
INTRODUÇÃO . . . . .	01
CAPÍTULO II	
DESCRIÇÃO GERAL DO SISTEMA . . . . .	05
2.1 - Características Gerais . . . . .	05
2.2 - Analisador Léxico . . . . .	08
2.3 - Analisador Sintático . . . . .	11
CAPÍTULO III	
DESCRIÇÃO DO MICROPROCESSADOR 8080 E DA ROTINA QUE O SIMULA (SIM8080)	19
3.1 - Descrição da Unidade de Processamento Central (CPU) . . . . .	19
3.1.1 - Arquitetura do 8080 . . . . .	24
3.1.1.1 - Registradores . . . . .	25
3.1.1.2 - Unidade Lógica e Aritmética (ALU) . . . . .	26
3.1.1.3 - Registrador de Instruções e Controles . . . . .	27
3.1.1.4 - "Buffer" do Barramento de Dados . . . . .	27
3.1.2 - O Ciclo do Processador . . . . .	28
3.1.3 - Ciclos de Máquina . . . . .	29

3.1.4 - A Sequência dos Estados de Transição. . . . .	33
3.1.5 - A Sequência da Interrupção. . . . .	37
3.1.6 - A Sequência de "HOLD". . . . .	39
3.1.7 - A Sequência de "HALT" . . . . .	39
3.1.8 - Formato dos Dados e das Instruções . . . . .	42
3.1.9 - Comentários Adicionais. . . . .	43
3.2 - SIM8080 - A Rotina de Simulação . . . . .	44
3.2.1 - Tipos de Ciclos de Máquina . . . . .	48
3.2.2 - TABELAIR. . . . .	50
3.2.3 - MICROPROGRAMA . . . . .	56
3.2.4 - As Microoperações . . . . .	61
3.2.5 - Simulação dos Eventos READY, HOLD, INTERRUPÇÃO e Estado HALT	68
3.2.5 - Diagrama de Blocos do Simulador SIM8080. . . . .	76

#### CAPÍTULO IV

A LINGUAGEM DE CONTRÔLE DE SIMULAÇÃO. . . . .	83
4.1 - Componentes da Linguagens. . . . .	83
4.1.1 - Símbolos Básicos e Palavras Reservadas. . . . .	83
4.1.2 - Constantes. . . . .	85
4.1.3 - Comentários . . . . .	86
4.1.4 - Programa de Contrôle de Simulação . . . . .	86
4.2 - Os Comandos de Contrôle de Simulação. . . . .	87
4.2.1 - Os Comandos de inicialização . . . . .	87
4.2.2 - Comandos de Monitoramento. . . . .	91
4.2.3 - Comandos de Geração de Eventos Externos . . . . .	93
4.2.4 - Comandos de Finalização. . . . .	94

CAPÍTULO V

COMENTÁRIOS E CONCLUSÕES FINAIS. . . . . 95

AGRADECIMENTOS. . . . . 97

BIBLIOGRAFIA. . . . . 99

APÊNDICE A

LISTAGEM DE MONTAMEMORIA (MONTA) E TABELAIR

APÊNDICE B

MICROPROGRAMA

APÊNDICE C

EXEMPLO DE SIMULAÇÃO

## LISTA DE FIGURAS

Figura I.1	- Sistema de Simulação Implementado para o Microprocessador INTEL 8080. . . . .	03
Figura II.1	- Diagrama do Sistema de Simulação . . . . .	07
Figura II.2	- Diagrama de Estados do "Scanner" . . . . .	10
Figura II.3	- Tabela de Símbolos. . . . .	12
Figura II.4	- Diagrama de Estados do Analisador Sintático. . . . .	13
Figura II.5	- Mensagens de Erro de Sintaxe . . . . .	15
Figura II.6	- Sinalização dos Erros de Sintaxe num Programa de Simulação, . . . . .	16
Figura III.1	- Configuração dos Terminais da CPU 8080 . . . . .	20
Figura III.2	- Diagrama de Blocos Funcional da CPU. . . . .	24
Figura III.3	- Pulsos de Relógios $\phi_1$ e $\phi_2$ . . . . .	29
Figura III.4	- Diagrama de Transição de Estados da CPU . . . . .	34
Figura III.5	- Ciclo Básico do 8080 . . . . .	35
Figura III.6	- Diagrama de Tempo de uma Sequência de Interrupção. . . . .	38
Figura III.7	- Diagrama de Tempo da Sequência HOLD. . . . .	40
Figura III.8	- Relação entre HOLD e INT no Estado HALT. . . . .	41
Figura III.9	- Diagrama de Blocos da Sequência HALT. . . . .	42
Figura III.10	- Formato dos Dados. . . . .	42
Figura III.11	- Formato de Instruções. . . . .	43
Figura III.12	- Diagrama de Estados Simplificado . . . . .	47
Figura III.13	- a) Representação de um nó de TABELAIR. . . . .	54
	b) Estrutura TABELAIR. . . . .	54
Figura III.14	- a) Nó de MICROPROGRAMA . . . . .	57
	b) Estrutura de MICROPROGRAMA e sua relação com TABELAIR. . . . .	57

Figura III.15	- a) Instrução SHLD. . . . .	59
	b) TABELAIR e MICROPROGRAMA em SHLD. . . . .	60
Figura III.16	- Testes de READY, HOLD e HLTA. . . . .	69
Figura III.17	- Interrupção com Prioridade. . . . .	70
Figura III.18	- Diagrama de Blocos de INCRINST . . . . .	72
Figura III.19	- a) Ativação do RESET . . . . .	73
	b) Atendimento de WRITE. . . . .	73
Figura III.20	- Atendimento de HOLD. . . . .	74
Figura III.21	- Atendimento de Interrupção . . . . .	75
Figura III.22	- Diagrama de Blocos do Simulador SIM8080. . . . .	77

## LISTA DE TABELAS

Tabela III.1	- Nomenclatura e Função dos Terminais do 8080. . . .	21
Tabela III.2	- Informação de Estado dos Diversos Tipos de Ciclos de Máquina. . . . .	31
Tabela III.3	- Nomenclatura dos Bits da Informação de Estado . . .	32
Tabela III.4	- Atividades Associadas aos Estados de um Ciclo de Máquina. . . . .	36
Tabela III.5	- Tipos de Ciclos de Máquina de Subrotinas dos Estados Comuns . . . . .	49
Tabela III.6	- MICROOPERAÇÕES realizadas nas Subrotinas dos Estados T1 e T2. . . . .	51

## CAPÍTULO I

### INTRODUÇÃO

O advento dos microprocessadores reduziu sensivelmente os custos envolvidos no desenvolvimento do "hardware" de sistemas digitais. Porém, devido principalmente ao seu conjunto de instruções bastante primitivo, todo projeto que o utilize, necessita de um "software" de suporte. Este "software" de suporte se faz necessário tanto na fase de desenvolvimento do sistema como também na fase de programação do sistema final (o "microcomputador").

Essas ferramentas de suporte são desenvolvidas principalmente na forma de:

- "cross-assemblers" e "cross-compilers" - programas montadores que aceitam uma linguagem simbólica e geram código de máquina correspondente.
- "cross-simulators" - programas que aceitam os códigos de máquina gerados pelos "cross-assemblers" e "cross-compilers" e simulam sua execução no computador hospedeiro.

A parte de geração de código de máquina já foi abordada no INPE através do desenvolvimento e implementação de uma linguagem de alto nível para programação de microcomputadores. O "cross-compiler" ALGOL M, desenvolvido, permite a utilização de linguagem que apresenta uma série de facilidades na programação do microprocessadores atualmente existentes.

A utilização de simuladores possibilita certos diagnóstico

cos que não são obtidos com o próprio microcomputador em ação. Além de permitirem certas manipulações e listagens de memória e registradores da CPU, os simuladores usualmente possuem capacidade de:

- "break points", onde o processamento pode ser parado num determinado endereço de programa ou quando o programa lê ou escreve numa posição especificada da memória.

- "tracing", em que o simulador imprime os conteúdos dos registradores e da memória, num intervalo de endereços especificado, à medida que as instruções são executadas.

- Informação de tempo, tais como número de instruções executadas, número de ciclos de máquina, número de pulsos de relógio, etc.

É indiscutível o poder desses simuladores como ferramentas de suporte, porém não se pode substituir completamente os testes de programa no próprio microcomputador, dado que o tempo específico e as condições externas de circuitação real nunca podem ser totalmente simuladas.

A simulação pode ser abordada de duas maneiras:

a) através da utilização de linguagem de Descrição de "Hardware" (C.H.D.L.).

b) ou através de desenvolvimento de um simulador específico.

O presente simulador, desenvolvido e implementado de forma específica para o microprocessador INTEL 8080, apesar de se destinar à simulação de um único microprocessador, oferece uma série de vantagens tanto no desenvolvimento de sistemas que utilizam o INTEL 8080 como também na verificação de programa de aplicação para o sistema final.

Em termos gerais, o sistema de simulação desenvolvido pode ser esquematizado como na Figura I.1 abaixo:

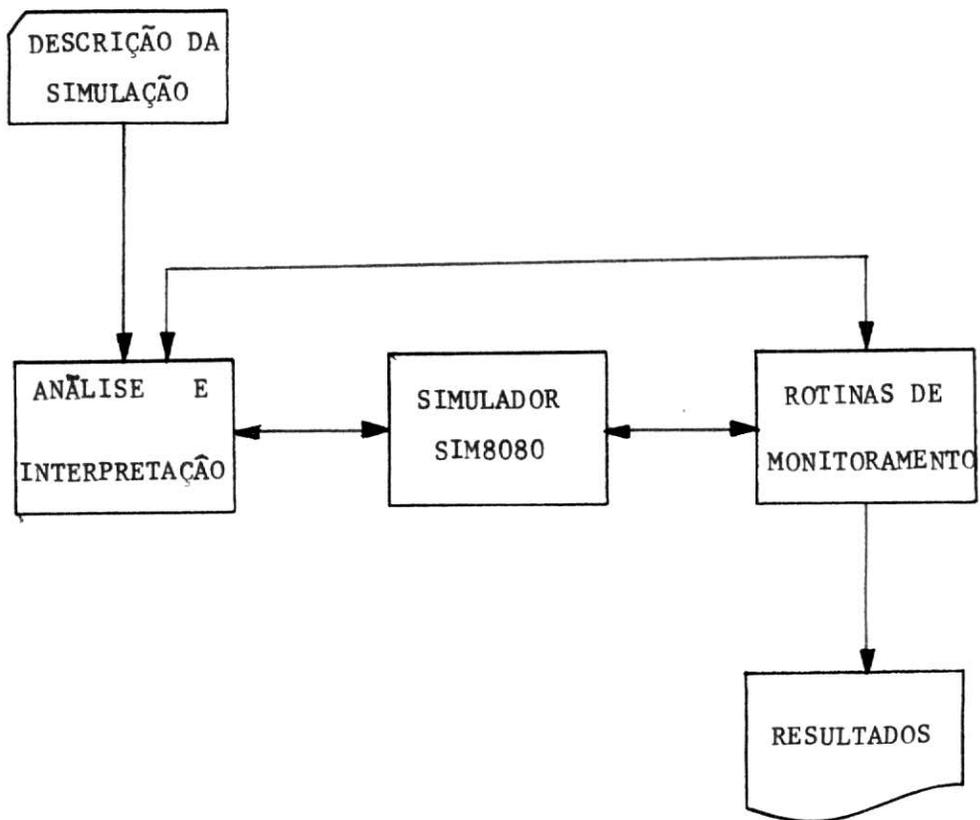


Fig. I.1 - Sistema de Simulação Implementado para o Microprocessador INTEL8080.

As características gerais do sistema de simulação, bem como a descrição dos analisadores sintáticos e lêxicos, são apresentadas no Capítulo II.

No Capítulo III, é introduzida a descrição da unidade de processamento central do 8080 de acordo com os dados fornecidos no manual do fabricante, enquanto que a linguagem de controle de simulação é descrita no Capítulo IV. Os comentários finais, bem como a bibliografia utilizada são apresentados no Capítulo V.

## CAPÍTULO II

### DESCRIÇÃO GERAL DO SISTEMA

A simulação de um sistema deve, em princípio, se aproximar tanto quanto possível da realidade do simulado, a fim de que resultados úteis possam surgir estabelecendo, desta forma, a capacidade do simulador e suas características.

#### 2.1 - CARACTERÍSTICAS GERAIS

A simulação do microprocessador INTEL 8080, de maneira específica, possibilita maior aproximação da realidade do simulado, sem aumento em demasia do grau de complexidade da implementação.

Escrito em linguagem ALGOL e implementado no sistema B-6700, o simulador reconfigurável para o 8080 apresenta, em linhas gerais, as características seguintes:

a) Estrutura Modular - simulação é feita a níveis de relógio ("clock"), possibilitando numa estrutura modular em estados assumidos pelo processador. A sincronização é obtida através de 3 rotinas idênticas de incremento de tempo: pulsos de relógio (INRCLOCK); ciclos (INRCYCLE) e instrução (INCRINST). Dessa forma, a inclusão de subrotinas de simulação de interfaces de entrada/saída e periférico é facilitada.

b) Reconfigurável - estrutura do simulador organizada de modo que as instruções podem ser microprogramadas. A sequência de microoperações pode ser recarregada através de programa auxiliar.

c) Comandos de execução - o sistema de simulação apresenta um conjunto de "comandos de execução" que comandam e definem a simulação desejada.

d) "TRACING" - apresenta as facilidades de "tracing", especificados pelos comandos de execução, de posições de memória e de registros do INTEL 8080.

e) Interrupção, "HOLD", "RESET" permitem a simulação e ativação, através do comandos de execução, do sistema de interrupções com prioridades associadas, do sistema de requisição de "HOLD" (DMA) e do sistema de "RESET".

f) Código de máquina - o simulador aceita código de máquina gerado pelo compilador da linguagem ALGOL M.

g) Memória e CPU - simula em "software" a memória do sistema e a unidade central constituída pelo INTEL8080.

O sistema de simulação pode ser apresentado como mostra o diagrama de blocos da Figura II.1

Na simulação de um dado programa, os comandos de execução são fornecidos, através de cartão, ao sistema simulador, que se encarrega

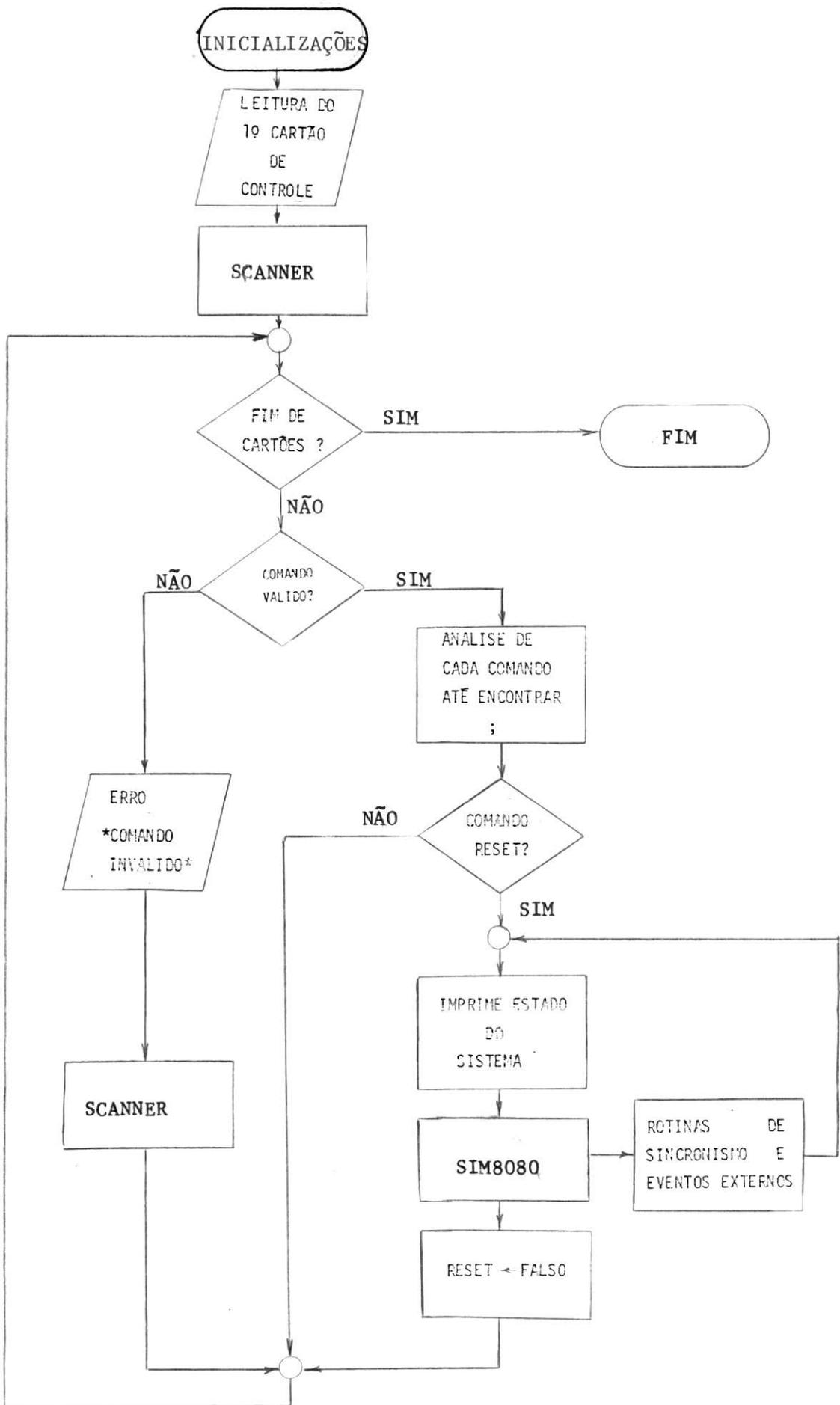


Fig. II.1 - Diagrama do Sistema de Simulação.

da análise léxica e sintática.

A análise léxica é feita pelo "SCANNER" (analisador léxico) que procura verificar se as entidades presentes num comando pertencem ao conjunto de símbolos da linguagem.

Caso ocorra erro de qualquer natureza, há indicação, na listagem do programa, onde o "SCANNER" parou, bem como as mensagens adequadas. O "SCANNER" é avançado até que o delimitador de fim de comando seja encontrado. Desta forma, a análise é feita até o último cartão.

A ativação do simulador propriamente dito (a subrotina SIM8080) é iniciada quando o comando "RESET" acaba de ser analisado. O estado inicial do sistema bem como os estados que assume ao longo da simulação são então impressos de acordo com os comandos de controle utilizados.

O simulador executa as instruções a partir do endereço de memória definido pelas condições iniciais do contador de programa ("PC") Esta simulação termina no tempo especificado pela instrução RESET encontrada, podendo iniciar uma nova simulação de acordo com um novo conjunto de comandos e por tempo determinado por um novo comando RESET.

## 2.2 - ANALISADOR LÉXICO ("SCANNER")

A análise léxica do programa é feita pela subrotina "SCANNER", que retorna nas variáveis globais:

- ▷ ENTIDADE, o "cordão" de caracteres identificado.
  
- ▷ SÍMBOLO, o número de classificação na tabela de símbolos. Caso se já encontrado erro, SÍMBOLO será igual a zero.

A Figura II.2, apresenta o diagrama de estados do "SCANNER". O Estado inicial (1) é mantido enquanto os caracteres de entrada forem brancos (GETNONBLANCK).

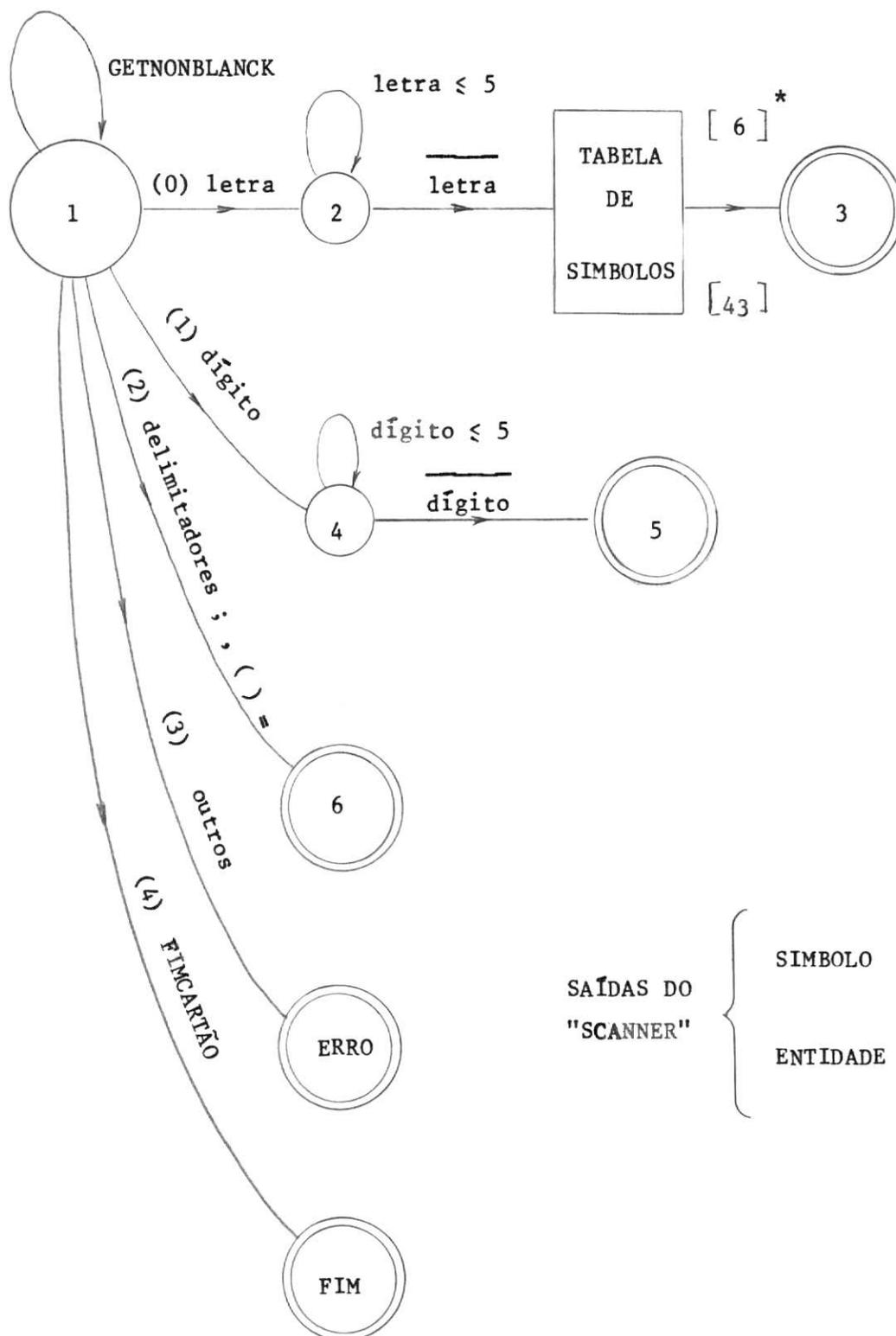
Os caracteres reconhecidos pelo "SCANNER" e que determinam a classe dos mesmos são os seguintes:

LETRAS:	A   B   C   D   . . . . . Z	Classe 0
DÍGITOS:	0   1   2   3   . . . . . 9	Classe 1
DELIMITADORES:	;   ,   (   )   =   %	Classe 2

Então em função da classe associada ao caracter, um novo estado é assumido.

Admitindo que o caractere de entrada seja letra, tem-se o estado (2) e o mesmo é mantido enquanto os caracteres sucessivos pertencerem ao conjunto LETRAS. Haverá mensagem de erro, se o número desses caracteres for maior que 6.

Um caractere não pertencente a LETRAS, faz com que o "SCANNER", através de rotinas auxiliares, consulte sua tabela de símbolos



\* [ 6 ]  
indica o intervalo na tabela de símbolos  
[ 43 ]

Fig.II.2 - Diagrama de Estados do "SCANNER"

fornecendo sua classificação na mesma no estado terminal  $\textcircled{3}$ , bem como o "cordão" de caracteres identificado.

De modo análogo, os outros estados podem ser assumidos.

A Figura II.3, apresenta a tabela de símbolos das entidades reconhecidas e sua classificação.

### 2.3 - ANALISADOR SINTÁTICO

A análise sintática dos comandos de controle da simulação é feita pelo ANALISADOR SINTÁTICO, apresentado em forma de diagrama de estados da Figura II.4.

Os comandos aceitos são as constantes da tabela de símbolos, apresentada na Figura II.3, de 6 a 17, ressaltando os comandos INPUT, OUTPUT, COPY e SAVE que serão implementados futuramente.

Cabem, também, as observações seguintes, no diagrama de estados da Figura II.4:

- ▷  $\langle n^0 \rangle$  representa um número de no máximo 6 dígitos
- ▷ No comando LOAD,  $\langle id \rangle$  é um dos identificadores da tabela de símbolos de 26 a 43.
- ▷  $\langle id^1 \rangle$  é o conjunto associado a  $\langle id \rangle$  excetuando o identificador M.

%  
%  
%

TABELA DE SIMBOLOS  
VALUE ARRAY SIMBOLOS( "

"	"	%	1	
"	"	%	2	
"	"	%	3	
"	"	%	4	
"	"	%	5	
"MEMORY"	"	%	6	
"READY "	"	%	7	
"LOAD "	"	%	8	
"INTER "	"	%	9	
"HOLD "	"	%	10	
"RESET "	"	%	11	
"REFER "	"	%	12	
"WRITE "	"	%	13	
"INPUT "	"	%	14	
"OUTPUT"	"	%	15	
"COPY "	"	%	16	
"SAVE "	"	%	17	
"TD "	"	%	18	
"CYCLE "	"	%	19	
"CLOCK "	"	%	20	
"INST "	"	%	21	
"WITH "	"	%	22	
"DUMP "	"	%	23	
"STATUS"	"	%	24	
"AND "	"	%	25	
"PC "	"	%	26	
"SP "	"	%	27	
"A "	"	%	28	
"B "	"	%	29	
"C "	"	%	30	
"D "	"	%	31	
"E "	"	%	32	
"H "	"	%	33	
"L "	"	%	34	
"ZERO "	"	%	35	
"CARRY "	"	%	36	
"SIGN "	"	%	37	
"PARITY"	"	%	38	
"ACARRY"	"	%	39	
"M "	"	%	40	
"INT "	"	%	41	
"HOLD "	"	%	42	
"INTE "	"	%	43	
"	"	%	44	
"	"	%	45	
";	"	%	46	
"	"	%	47	
"( "	"	%	48	
)	"	%	49	
"= "	"	%	50	
"	"	%	51	<NUMERO>
"	"	%	52	ERRO
"	"	%	53	FIMCARTAO

%

Fig. II.3 - Tabela de Símbolos.

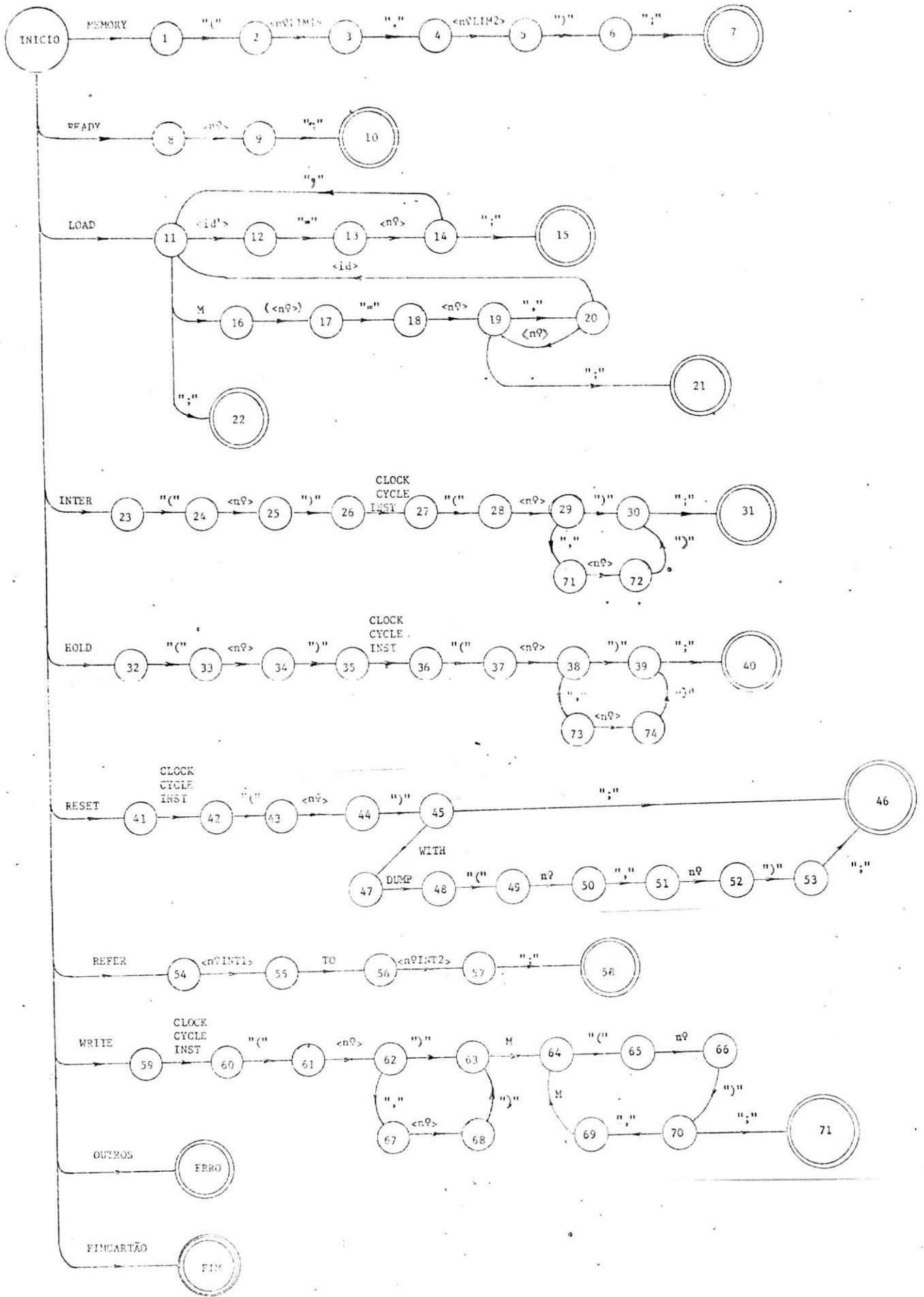


Fig.II.4 - Diagrama de Estados do Analisador Sintático

Os estados assinalados com duas circunferências concêntricas indicam fim de análise de cada comando, fazendo então o analisador retornar ao estado inicial. O estado assinalado em três circunferências indica estado final do analisador (fim do programa).

Os resultados da análise léxica determinam as transições do estado inicial a um dos estados terminais. Caso não ocorra o caracter esperado, em qualquer das transições, haverá mensagem de erro, bem como o sinalização da posição em que o "SCANNER" parou, no programa de simulação analisado.

A Figura II.5, apresenta as possíveis mensagens de erro de sintaxe implementadas até o momento e, a Figura II.6, um programa teste de simulação com erros de sintaxe.

```
%
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
%
  SWITCH FORMAT ERRADOS:=
%1
  (X*, "MAIS DE 6 CARACTERES EM CONST. OU ID"),
%2
  (X*, "IDENTIFICADOR INEXISTENTE"),
%3
  (X*, "CARACTER INVALIDO"),
%4
  (X*, "NUMERO EXCESSIVO DESTE COMANDO"),
%5
  (X*, "FALTOU O      ; "),
%6
  (X*, "FALTOU O ("),
%7
  (X*, "NUMERO INVALIDO"),
%8
  (X*, "FALTOU O )"),
%9
  (X*, "PAR DELIMITADOR INVALIDO"),
%10
  (X*, "COMANDO INVALIDO"),
%11
  (X*, "FALTOU O -TO- "),
%12
  (X*, "PERMITIDO APENAS-CYCLE, CLOCK OU INST"),
%13
  (X*, "NUMERO INVALIDO PARA INTERRUPTAO"),
%14
  (X*, "FALTOU O SINAL DE ATRIBUICAO"),
%15
  (X*, "CONSTANTE INVALIDA NESTA ATRIBUICAO"),
%16
  (X*, "ESPECIFICACAO DE MEMORIA INVALIDA"),
%17
  (X*, "MAIS DE 10 ELEMENTOS REFERENCIADOS"),
%18
  (X*, "ENDERECO INVALIDO");
.....
```

Fig. II.5 - Mensagens de Erro de Sintaxe.



```

      COMANDO INVALIDO
LOAD *('A)F 373* 8      EI
      311* 8          RE
LOAD *('A)F 373* 8      EI
      311* 8          RI
      8
RPSSET INST(100)

```

```

48
49
50
51
52
53

```

Fig. II.6 - Sinalização dos Erros de Sintaxe num Programa de Simulação.

## CAPÍTULO III

### DESCRIÇÃO DO MICROPROCESSADOR 8080 E DA ROTINA QUE O SIMULA (SIM 8080).

Nêste capítulo é apresentada, uma breve descrição do 8080, necessária à melhor compreensão da rotina de simulação SIM8080. Para a nomenclatura dos terminais do processador e de seus registros internos na simulação, utilizou-se aquela presente no diagrama funcional da Figura II.1 bem como para as variáveis da programação aquelas constantes no manual do Microprocessador INTEL 8080.

#### 3.1 - DESCRIÇÃO DA UNIDADE DE PROCESSAMENTO CENTRAL (CPU)

O 8080 é um microprocessador de 8 bits para uso geral, fabricado pela INTEL numa pastilha LSI, empregando o processo "n - channel silicon gate MOS".

Os seus terminais são constituídos pelo barramento de dados (8 bits), pelo barramento de endereço (16 bits), pelo barramento de controle e tempo e pelos pinos de alimentação.

O barramento de dados (D7D0) é bidirecional, por meio do qual há transferência de dados entre a CPU e o resto do sistema. A transferência de endereços da memória e periféricos é feita através do barramento de endereço (A15A0). A CPU possui condições de forçar os barramentos de dados e de endereço para o estado de alta impedância, permitindo assim o controle desses barramentos por outros dispositivos (por ex. acesso

direto à memória (DMA)).

SYNC, DBIN, WAIT, WR, HALDA e INTE constituem as saídas de controle e tempo enquanto que READY, HOLD, INT e RESET são as entradas.

Além dos terminais já mencionados, as entradas de alimentação +12V, +5V, -5V, Terra e duas de relógio  $\phi_1$  e  $\phi_2$  completam um rápido apanhado dos pinos do microprocessador 8080.

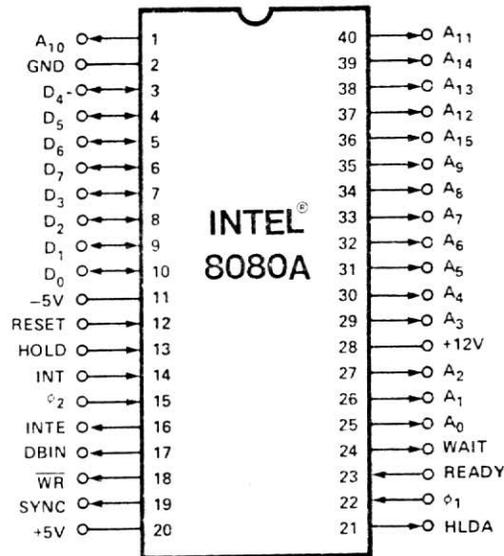


Fig. III.1 - Configuração dos Terminais da CPU 8080.

É apresentada em forma de tabela, a descrição funcional de cada terminal do 8080.

TABELA III.1

NOMENCLATURA E FUNÇÃO DOS TERMINAIS DO 8080

NOMENCLATURA	F U N Ç Ã O
A15A0	Barramento de endereço - constituído por 16 bits onde A0 é o menos significativos
D7D0	Barramento de dados - que provê comunicação bidirecional entre a CPU, memória e dispositivos de entrada e saída (I/O) para instruções e dados. D0 é bit menos significativo.
SYNC	Sinal de Sincronismo - fornecido pela CPU que indica o começo de um novo ciclo de máquina.
DBIN	Sinal que indica ao circuito externo o modo de operação do barramento de dados. Nível alto significa modo de entrada (ciclos de busca, leitura de memória, leitura de PILHA e interrupção).
READY	Sinal que indica à CPU que os dados estão disponíveis no barramento D7D0. Este sinal é utilizado para sincronizar a memória ou periféricos. Se depois de mandar um endereço, o 8080 não recebe o sinal READY, o 8080 entra em estado de espera enquanto READY estiver em nível baixo. O sinal READY também pode ser utilizado para controle da CPU por passos.

(continuação da Tabela III.1)

NOMENCLATURA	FUNÇÃO
WAIT	Sinal notificando que a CPU está no estado de espera, denominado estado WAIT.
$\overline{WR}$	Sinal utilizado para controle de escrita na memória ou I/O de periférico. Os dados estão estáveis no barramento D7D0 enquanto o sinal $\overline{WR}$ apresentar nível baixo (ciclos de escrita na memória, escrita na pilha e saída para periférico).
HOLD	Sinal para a CPU entrar em estado denominado de HOLD. Este estado possibilita o controle dos barramentos A15A0 e D7D0 por dispositivo externo, tão logo termine o ciclo de máquina corrente. A CPU anuncia que entrou em estado HOLD, pelo terminal HLDA. Para melhor compreensão vide o diagrama de estados, onde se pode observar que o sinal de HOLD é verificado no estado HALT e na entrada do estado T3.
HLDA	Sinal de resposta ao pedido de HOLD e indica que os barramentos entrarão em estado de alta impedância. O sinal HLDA poderá surgir em T3 no caso de leitura da memória ou dispositivo de entrada, ou no período seguinte ao estado T3 no caso de escrita em memória ou saída para periférico.
INTE	Indica o conteúdo de INTEFF que, quando em nível alto, habilita a CPU de receber interrupções.

(continuação da Tabela III.1)

NOMENCLATURA	F U N Ç Ã O
INT	Sinal de pedido de interrupção. No diagrama de estados fica evidente que a entrada INT é verificada no final da instrução corrente, ou enquanto a CPU está em estado HALT. Nota-se também que a CPU não aceitará o pedido de INT caso esteja em estado HOLD ou se o INTEFF estiver em nível zero.
RESET	Sinal que zera o contador de programa. Após o RESET, o programa começará a partir do endereço zero da memória, os "flip flops" INTE E HLDA são também levados ao nível zero. O Flag, o acumulador, o ponteiro da pilha, os <u>re</u> gistradores não são zerados.

### 3.1.1 - ARQUITETURA DO 8080

Pode-se dividir o microprocessador 8080 em cinco unidades funcionais:

- a) Conjuntos de registradores e lógica de endereçamento.
- b) Unidade Aritmética e Lógica (ALU).
- c) Registrador de Instruções e Seção de Controle.
- d) "Buffer" do barramento de dados.

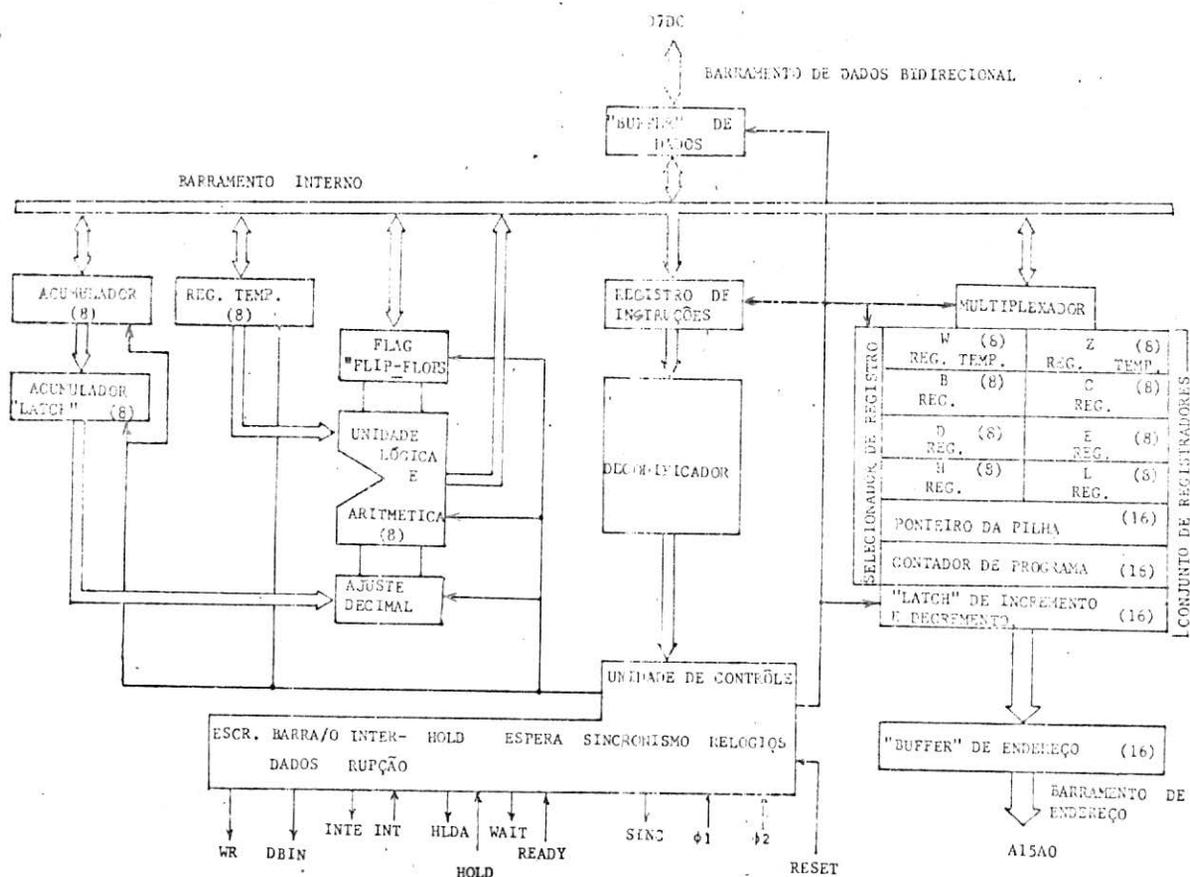


Fig. III.2 - Diagrama de Blocos Funcional da CPU.

### 3.1.1.1 - REGISTRADORES

Esta seção é constituída por conjunto de memórias estáti cas do tipo de acesso aleatório ("RAM"), organizadas em registradores de 16 bits.

Nomenclatura:

PC - Contador de Programa

SP - Ponteiro de PILHA

B, C, D, e H, L - Constituem os 6 registradores de proposito geral, os quais podem ser referidos em pares.

W, Z - Constituem o registro temporário.

O contador de programa mantém o endereço da próxima ins trução a ser executada. Isto é, no caso de ciclo de busca, o conteúdo de PC é colocado no barramento de endereço e, no período seguinte do relógio, é incrementado automaticamente.

Jã o registrador SP mantém o ponteiro da PILHA que poderã ser inicializado com qualquer valor, de modo a utilizar parte da memória principal do sistema como estruturada numa pilha. É necessário ressaltar que a PILHA cresce no sentido de locações decrescentes da memória.

Os registros temporários, W, Z não são endereçáveis por programação e são usados somente para a execução interna das instruções.

Os 8 bits de um registrador podem ser transferidos para o barramento de dados interno, como se pode observar no diagrama funcional a apresentado na Figura III.2, através do selecionador de registros. Já os 16 bits podem ser transferidos para o armazenador ("Latch") de endereço ou para o circuito de incremento/decremento. O armazenador de endereço recebe dados de qualquer dos pares de registradores e aciona o "buffer" de endereço, bem como o circuito de incremento/decremento. Este circuito recebe dados do armazenador de endereço e pode enviar seu conteúdo para um dos pares de registradores.

Os dados de 16 bits podem ser incrementados ou decrementados ou simplesmente transferidos entre os registradores.

### 3.1.1.2 -UNIDADE LÓGICA E ARITMÉTICA (ALU)

Esta unidade é constituída pelos seguintes registradores:

- A - Acumulador de 8 bits
- ACT - Acumulador Temporário de 8 bits.
- FLAG - Registrador de 6 bits, onde é guardada informação dos sinais: zero, transporte ("carry"), sinal, paridade e transporte auxiliar ("auxiliar carry").
- TMP - Registrador temporário de 8 bits.

As operações aritméticas, lógicas e operações de rotação são realizadas na ALU (Unidade Aritmética e Lógica). A ALU recebe de

TMP, de ACT e Transporte; e pode enviar os resultados de uma operação para o acumulador via barramento interno. No diagrama funcional são dadas as transferências possíveis, não havendo portanto necessidade de maiores comentários.

### 3.1.1.3 - REGISTRADOR DE INSTRUÇÕES E CONTRÔLES

Durante o ciclo de Busca ou Inicialização ("Fetch"), o primeiro byte da instrução código de operação ("OP-CODE") é transferido do "buffer" de dados, via barramento interno, para o registrador de Instrução IR. O código de operação fica então disponível ao circuito decodificador, que provê sinais, combinados com os de tempo, para coordenação da CPU. Além disso, os sinais do decodificador, mais os sinais de controle externo, alimentam a seção de tempos e contrôles que gera os sinais de tempo de estados e de ciclos.

### 3.1.1.4 - "BUFFER" DO BARRAMENTO DE DADOS

O "buffer" bidirecional de 8 bits, associado ao barramento de dados, é utilizado para se isolar o barramento interno do externo (D7D0). No modo de operação "output", o conteúdo do barramento interno é carregado no armazenador ("latch") que, por sua vez, aciona o "buffer" de saída. O "buffer" de saída é desligado durante as operações de entrada e operações que não necessitam de transferência.

Durante o modo "input", os dados do barramento externo são transferidos para o interno. O barramento interno é recarregado no início de cada estado interno, exceto para o estado de transferência T3, que será descrito posteriormente na seção 3.1.4.

### 3.1.2 - O CICLO DO PROCESSADOR

A uma instrução é associada uma fase de busca, onde o código de operação é extraído da memória e depositado no registro de instruções IR. Numa outra fase, a de execução, o código de operação em IR é decodificado para fornecer os sinais que comandarão as atividades do processador.

Cada instrução poderá consistir de um a cinco ciclos de máquina. Estes ciclos são necessários a todo acesso à memória ou periféricos. Na fase de busca serão necessário tantos ciclos de máquina quantos forem os bytes que compõem a instrução, enquanto que, na fase de execução, o número do ciclos de máquina é ditado pela instrução obtida. Há instruções que não requerem nenhum ciclo de máquina adicional além dos necessários à busca.

Cada ciclo de máquina poderá consistir de 3, 4 ou 5 estados. Um estado é a menor unidade de atividade do processador e corresponde a duas transições positivas e sucessivas de  $\phi_1$  (relógio). O 8080 necessita de dois pulsos de relógio  $\phi_1$  e  $\phi_2$  não sobrepostos conforme mostra a Figura III.3. O início de cada ciclo de máquina é noticiada pelo 8080, através do sinal SYNC, exceto no segundo e terceiro ciclo de máquina da instrução DAD ("DOUBLE AND") onde o SYNC não ocorre, uma vez que estes ciclos de máquina são usados para operação interna de adição entre os pares de registros especificados.

Há três exceções, em que o estado tem duração indeterminada, isto é, no estado WAIT, no estado HOLD e no estado HALT, que depen

dem exclusivamente de eventos externos à CPU. Porém a duração indeterminada destes estados é sincronizada ao relógio, de modo que todos os estados são múltiplos do período de relógio.

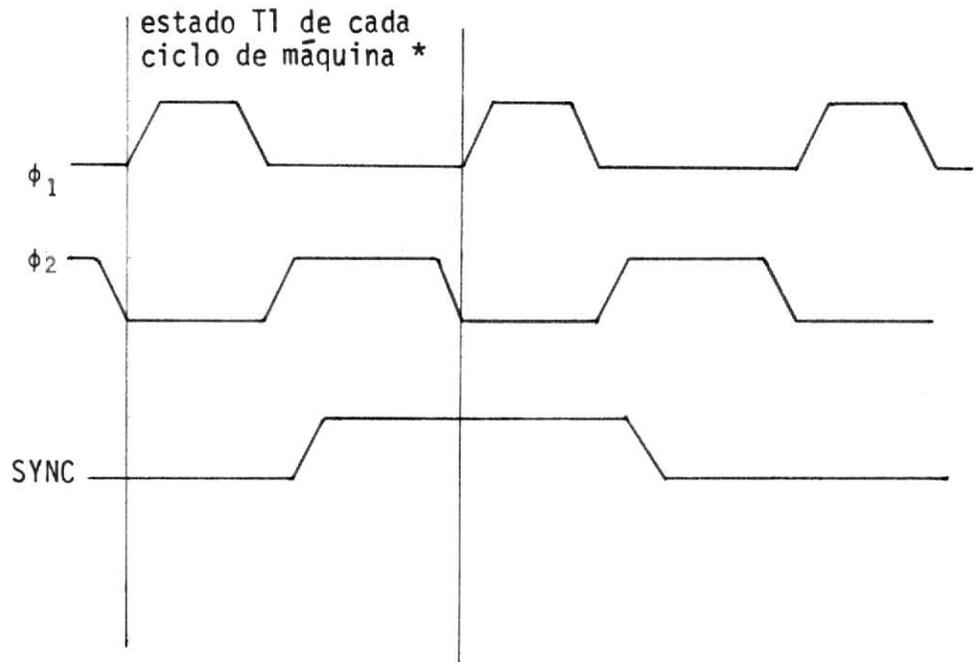


Fig. III.3 - Pulsos de Relógio  $\phi_1$  e  $\phi_2$ .

### 3.1.3 - CICLOS DE MÁQUINA

Com a exceção da instrução DAD, um dado ciclo de máquina poderá ser identificado pela informação de estado ("status"), presente no barramento de dados quando o sinal de SYNC ocorrer, isto é, no estado T1 do ciclo.

Dentre os ciclos de máquina possíveis no conjunto de instruções do 8080, podemos classificá-los em:

1. Busca ("fetch")
2. Leitura de memória ("memory read")
3. Escrita em memória ("memory write")
4. Leitura da Pilha ("stack read")
5. Escrita na Pilha ("stack write")
6. Entrada ("input")
7. Saída ("output")
8. Interrupção ("interrupt")
9. Parada ("halt")
10. Interrupção em "halt" ("interrupt while halt")
11. Operação entre registros

Cabe aqui uma observação de que o ciclo (11) foi adicionado para incluir os dois ciclos de máquina da instrução DAD, que não apresentam o sinal SYNC.

A seguir são apresentadas na forma de tabelas, as informações de estado, bem como as definições das nomenclaturas utilizadas e suas correspondências com os bits do barramento de dados.

TABELA III.2

INFORMAÇÕES DE ESTADO DOS DIVERSOS TIPOS DE CICLOS DE MÁQUINA

BITS DO BARRAMENTO DE DADOS	INFORMAÇÕES DE ESTADOS	BUSCA	LEITURA DA MEMORIA	ESCRITA EM MEMORIA	LEITURA DA PILHA	ESCRITA NA PILHA	ENTRADA	SAIDA	INTERRUPÇÃO	PARADA	INTERRUPÇÃO EM HALT	OPERAÇÃO ENTRE REGISTROS
		1	2	3	4	5	6	7	8	9	10	11
D0	INTA	0	0	0	0	0	0	0	1	0	1	NÃO APRESENTA
D1	$\bar{w} 0$	1	1	0	1	0	1	0	1	1	1	
D2	STACK	0	0	0	1	1	0	0	0	0	0	
D3	HLTA	0	0	0	0	0	0	0	0	1	1	
D4	OUT	0	0	0	0	0	0	1	0	0	0	
D5	M1	1	0	0	0	0	0	0	1	0	1	
D6	INP	0	0	0	0	0	1	0	0	0	0	
D7	MEMR	1	1	0	1	0	0	0	0	1	0	

TIPOS DE CICLOS DE MÁQUINA

TABELA III.3

NOMENCLATURA DOS BITS DE INFORMAÇÃO DE ESTADO

NOMEN- CLATURA	BIT DO BARRA MENTO DE DADOS	DEFINIÇÕES
INTA	D0	Reconhecimento de sinal de interrupção. Este sinal poderá ser utilizada para gatilhar a instrução RST no barramento de dados quando DBIN estiver ativo, isto é, DBIN=1 (nível alto)
$\bar{w}O$	D1	Indica que a operação no ciclo de máquina corrente será escrita em memória ou saída para periférico quando $\bar{w}O=0$ . Caso contrário, será uma leitura de memória ou entrada de periférico.
STACK	D2	Indica que o endereço em A15A0, é um endereço da pilha ("pushdown stack") dado pelo registrador SP, ponteiro da pilha.
HLTA	D3	Reconhecimento de instrução HALT.
OUT	D4	Indica que o endereço em A15A0 é de um dispositivo de saída e o barramento de dados conterá os dados de saída quando $\bar{w}R$ estiver ativo.
M1	D5	É um sinal indicando que a CPU está num ciclo de busca ("fetch") do primeiro byte de instrução.
INP	D6	Indica que A15A0 contém um endereço de dispositivo de entrada e que os dados de entrada poderão ser colocados no barramentos de dados quando DBIN for ativo.
MEMR	D7	Indica que é barramento de dados será utilizada para leitura de dados da memória.

### 3.1.4 - SEQUÊNCIA DOS ESTADOS DE TRANSIÇÃO

Cada ciclo de máquina poderá ter de três a cinco estados ativos, isto é, T1, T2, T3, T4, T5 e Tw. O número destes estados, num ciclo de máquina, dependerá da instrução particular que está sendo executada.

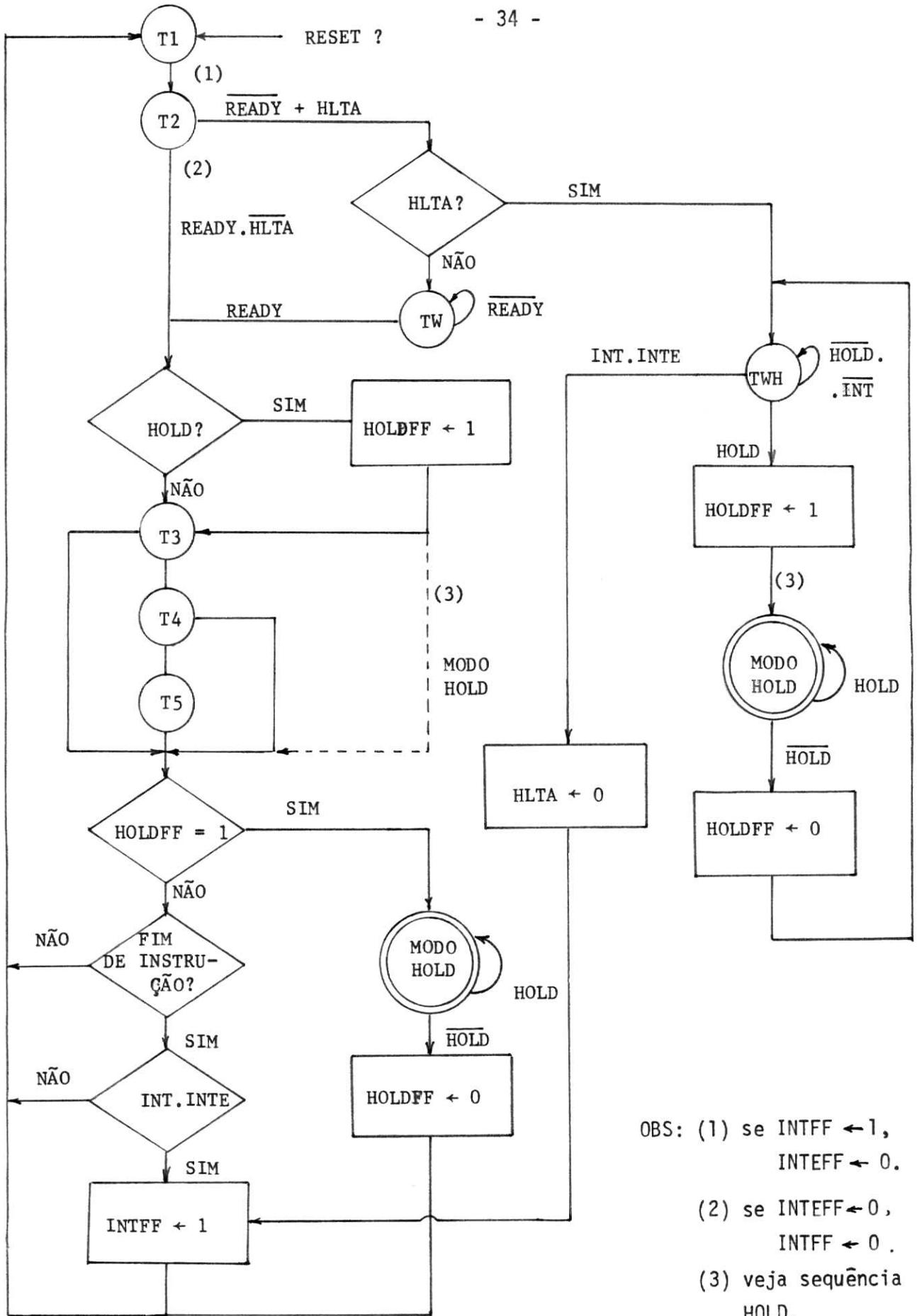
A diagrama da Figura III.4, apresenta onde os sinais de READY, HOLD e INT são amostrados e como eles influem na sequência básica dos estados. Posteriormente serão apresentadas as funções destes sinais.

Os estados mencionados anteriormente não são diretamente indicados pela CPU, porém são fornecidos os sinais de controle INTE, HLDA, DBIN,  $\overline{WR}$  e WAIT para uso do circuito externo.

O sinal de SYNC, já dito anteriormente, identifica o estado T1 de um ciclo de máquina (exceção feita à instrução DAD). Concomitantemente com o sinal de SYNC, é apresentado no barramento de dados D7D0, a informação de estado. Em T1 é fornecido, ao barramento A15A0, endereço de memória ou de algum periférico.

Em T2, o processador verifica os sinais READY e HOLD e a existência de instrução HALT.

No caso de o processador referenciar a memória ou periférico haverá oportunidade de pedido de espera (WAIT). Esta requisição é feita colocando-se o terminal READY em nível baixo.



OBS: (1) se  $INTFF \leftarrow 1$ ,  
 $INTEFF \leftarrow 0$ .  
 (2) se  $INTEFF \leftarrow 0$ ,  
 $INTFF \leftarrow 0$ .  
 (3) veja seqüência  
 HOLD.

Fig. III.4 - Diagrama de Transição de Estados da CPU.

O processador responde ao pedido de espera entrando no estado  $T_w$ , onde permanecerá ocioso por tempo indeterminado, porém múltiplo do período de relógio, até que o sinal de READY em nível alto apareça.

Na Figura III.5 é apresentado um ciclo básico do 8080.

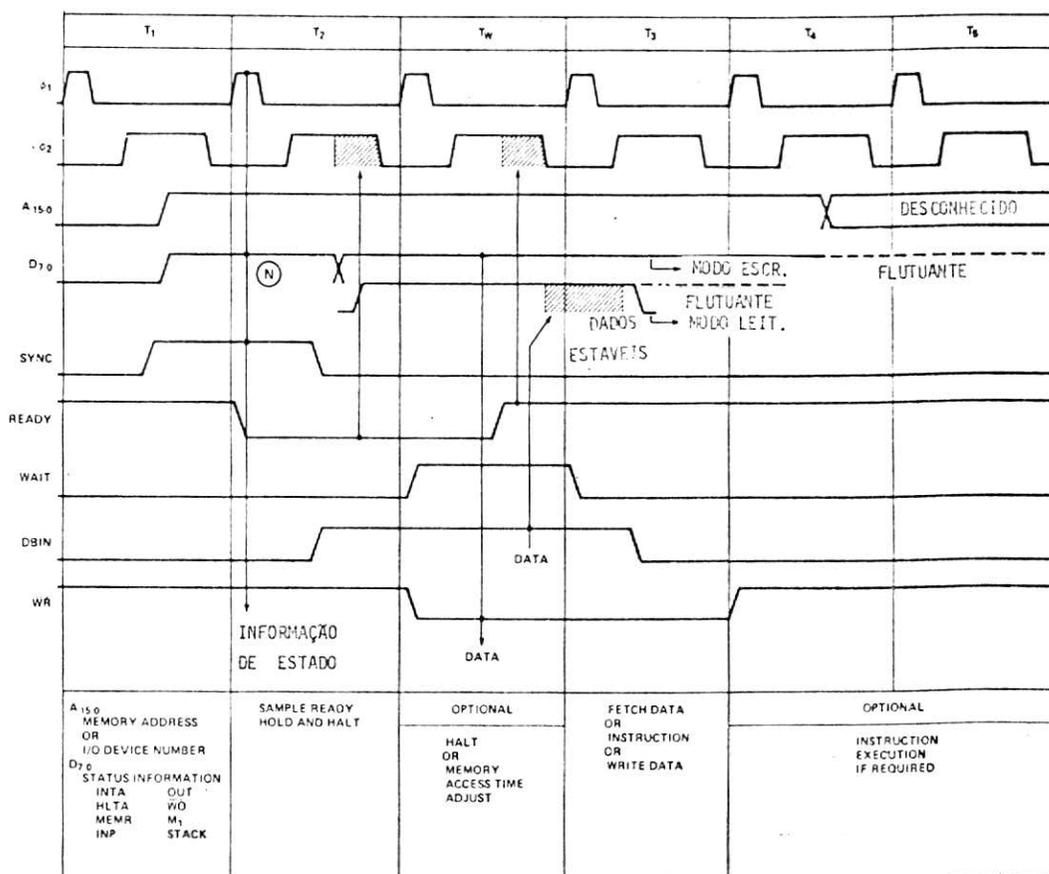


Fig. III.5 - Ciclo Básico do 8080.

Após o estado T3, fica muito difícil a generalização e todos os eventos que ocorrem, a partir de então, dependem do ciclo de máquina corrente. Sendo que os estados T4 e T5 são opcionais e somente utilizados em função da necessidade de cada instrução. Após o estado T5, o processador retorna diretamente ao estado T1 do ciclo de máquina se guinte.

TABELA III.4

ATIVIDADES ASSOCIADAS AOS ESTADOS DE UM CICLO DE MÁQUINA

ESTADO	ATIVIDADE ASSOCIADA
T1	É colocado no barramento de endereços A15A0, o endereço de alguma posição de memória ou de algum periférico. A informação de Estado é colocada no barramento de dados D7D0.
T2	Verificação das entradas READY e HOLD, e a existência de instrução HALT
T <sub>w</sub> (opcional)	O processador entra em estado de espera se READY tem nível baixo ou se ocorreu instrução de HALT
T3	Os dados entram no processador via barramento D7D0 nos ciclos de busca de leitura da memória, de leitura da pilha, de entrada ou de interrupção, enquanto que os dados são colocados como saída nos ciclos de escrita em memória, escrita na pilha ou de saída para periférico
T4 e T5 (opcionais)	Estes estados ficam a disposição das necessidades de cada instrução. Caso não sejam requisitadas estes estados podem ser saltados. São utilizadas somente para operações internas do processador

### 3.1.5 - A SEQUÊNCIA DA INTERRUPÇÃO

O 8080 tem capacidade inerente de manipular interrupções. O pedido de interrupção por um periférico é feito através da entrada INT.

A verificação do INT, como se pode observar no diagrama de estados na Figura III.4, é feita somente no final da instrução e só será aceita se INTE tem nível alto.

O ciclo de interrupção é muito semelhante ao ciclo de busca, na maioria dos aspectos. Na informação do estado ("status") é transmitido o bit MI de maneira usual, porém é acrescentado o bit INTA, que notifica o circuito externo da aceitação da interrupção. É jogado em T1, no barramento de endereços, o conteúdo do contador de programa, porém, durante o ciclo de interrupção, o contador de programa não é incrementado.

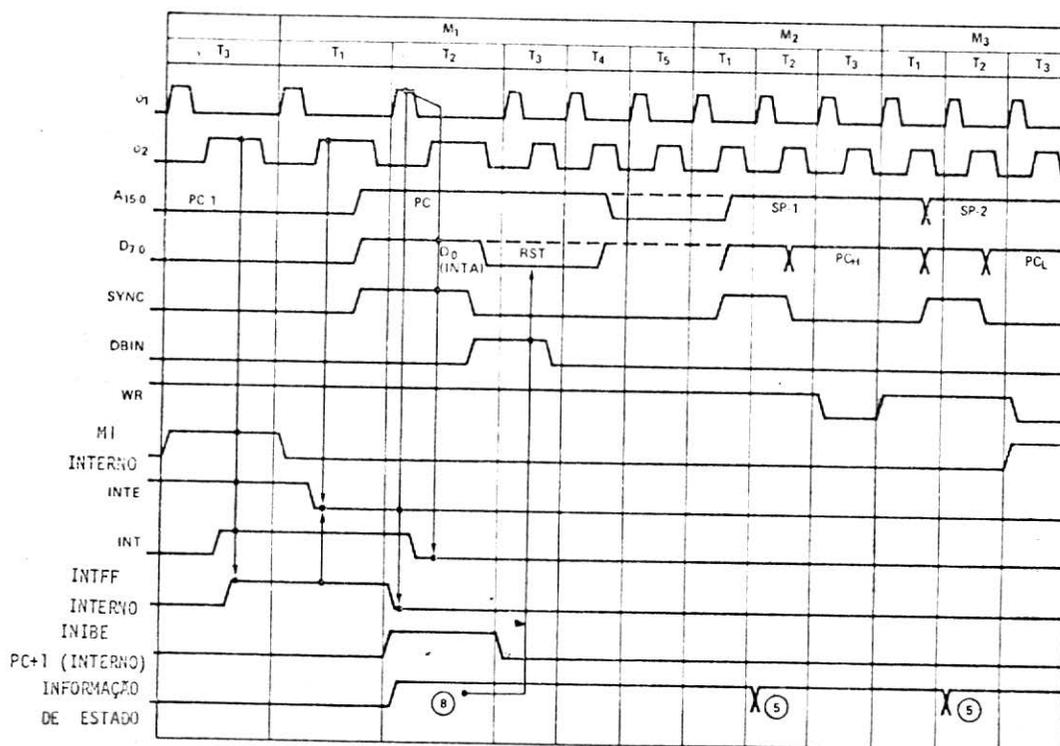
Deste modo o contador de programa é preservado, podendo ser utilizado posteriormente ao final do processamento da interrupção.

É função da lógica externa colocar, em T3, uma instrução de RST de máquina no barramento de dados. Isto, num sistema típico, significa que o barramento de dados será controlado pelo dispositivo que solicitou a interrupção.

No conjunto de instruções, RST possibilita a chamada de uma das oito posições fixas na memória, para alocar subrotinas de serviço de interrupções. Ainda nesta instrução RST, o valor anterior do contador de programa é armazenado na Bilha.

Os endereços dessas alocações em decimal são: 0, 8, 16,24, 32, 40, 48 e 56.

A Figura III.6 apresenta um pedido de interrupção enquanto INTE está em nível alto ("enable").



NOTA: (N) vide Tabela III.5.

Fig. III.6 - Diagrama de Tempo de uma Sequência de Interrupção.

### 3.1.6 - A SEQUÊNCIA DE HOLD

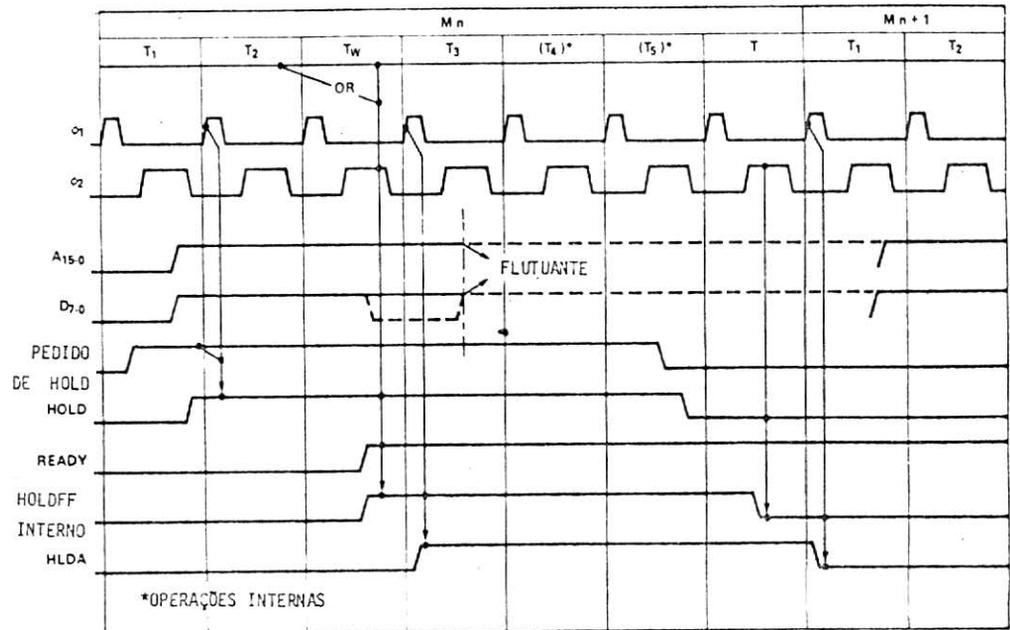
Aplicando-se nível alto em HOLD, um dispositivo externo pode suspender as atividades normais da CPU. O pedido dessa espécie é aceito pela CPU, que coloca os seus barramentos de dados e endereço em estado flutuante, de modo que o dispositivo requisitante de HOLD tenha total controle sobre os barramentos. A saída HLDA é colocada em nível alto possibilitando, destarte, o acesso direto a memória (DMA).

Aparentemente, o processador tem suas operações suspensas, uma vez que os barramentos de dados e endereços estão flutuantes. Porém nada impede a execução de operações internas a CPU. Esse processamento interno permite portanto um paralelismo com operações de transferência externa (DMA). A atividade da CPU não cessa enquanto não terminar de executar o ciclo de máquina corrente atingida pela HOLD.

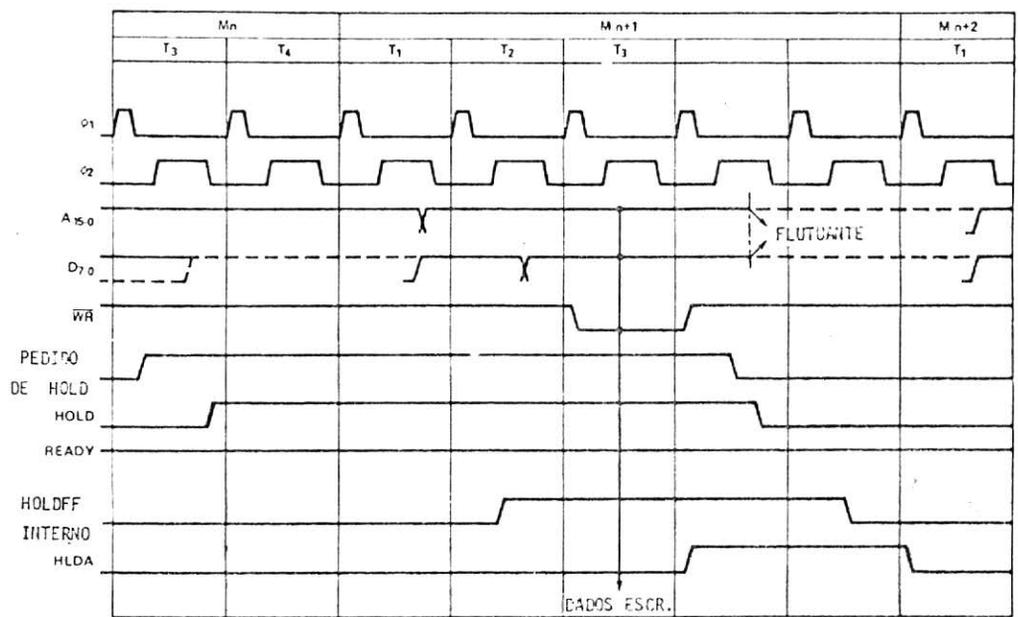
A saída do processador do estado HOLD é similar à maneira de sua entrada. O pedido de HOLD é terminado quando o dispositivo completou as transferências de dados. HLDA retorna então ao nível baixo e a CPU continua seu processamento normal, no próximo ciclo de máquina a ser executada.

### 3.1.7 - SEQUÊNCIA DE HALT

Se uma instrução de HALT for executada, o processador entra no estado HALT (TWH) após o estado T2 do próximo ciclo de máquina.



(a) - Sequência HOLD (MODO LEITURA)



(b) - Sequência HOLD (MODO ESCRITA)

Fig. III.7 - Diagramas de Tempo da Sequência HOLD.

Observando-se o diagrama de estados da Figura III.3.A, verifica-se que há somente três maneiras de sair deste estado de HALT:

1) Através de RESET, onde a CPU é jogada no estado T1, com o contador de programa zerado (vide Tabela III.1).

2) Através de HOLD, já descrito anteriormente. Após a saída do estado HOLD, o processador entra novamente em estado de HALT.

3) Através de Interrupção colocando-se INT em nível 1 enquanto INTE está habilitado, o que levará o processador ao estado T1, para execução da rotina de Interrupção. Note que ao entrar em estado de HALT, INTE deve estar no nível 1, caso contrário a saída de HALT, só será possível através de RESET.

A Figura III.8 relaciona as entradas HOLD, INT e HALT. E na Figura III.9 é apresentada um diagrama de blocos da sequência HALT. Verifica-se também, na Figura III.9, evidência de prioridade quanto à verificação dos sinais de RESET, HOLD e INT.

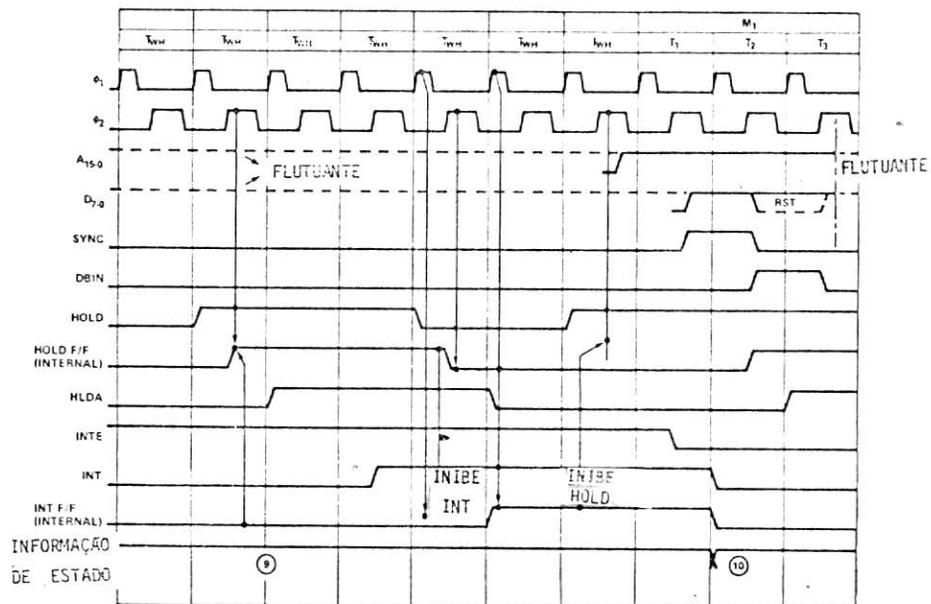


Fig.III.8 - Relação entre HOLD e INT no Estado HALT.

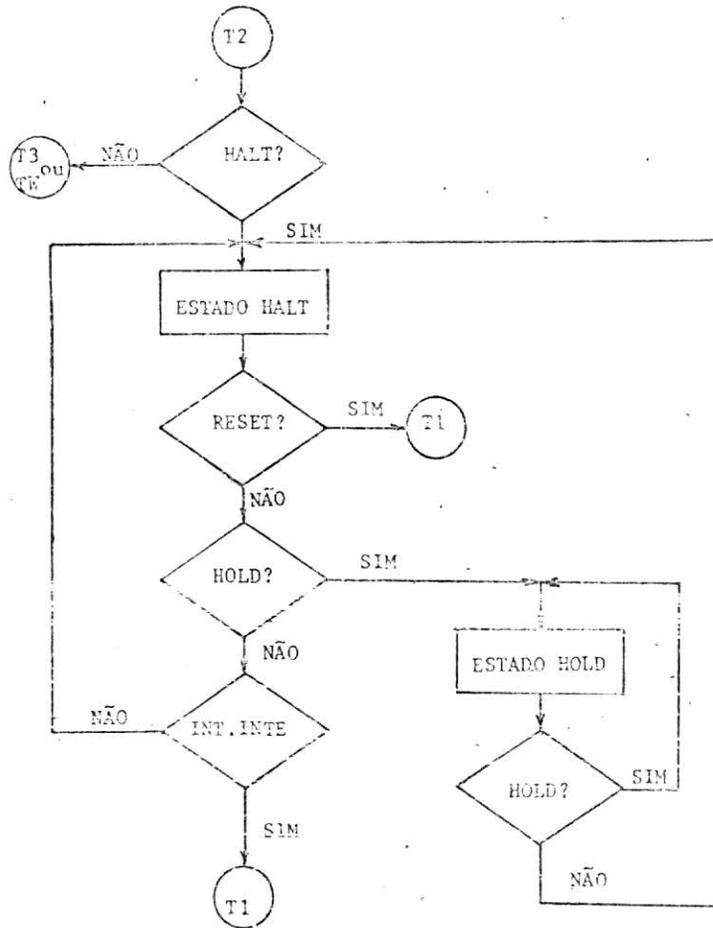


Fig. III.9 - Diagrama de Blocos da Sequência HALT.

### 3.1.8 - FORMATO DOS DADOS E DAS INSTRUÇÕES

Os dados na 8080 são armazenados em conjuntos de 8 bits conforme mostra a Figura III.10, onde o bit D7 é o mais significativo.

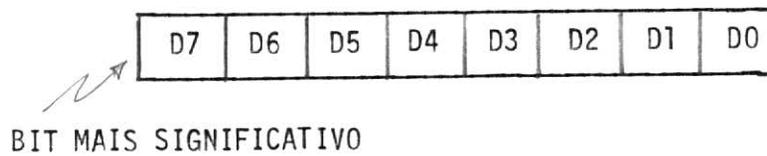
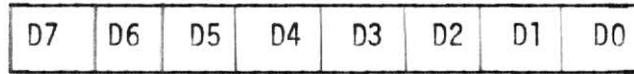


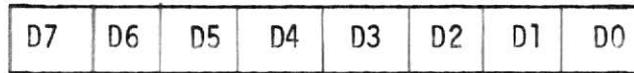
Fig. III.10 - Formato dos Dados.

Uma instrução pode ter 1, 2 ou 3 bytes de comprimento. Estes bytes são armazenados em posições sucessivas na memória, de acordo com o formato apresentado na Figura III.11.

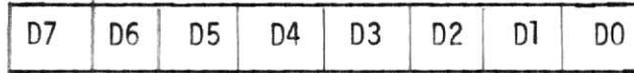
Instrução 1 byte



Instrução de 2 bytes

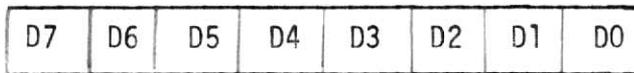


código de operação

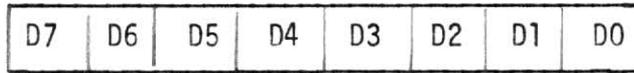


operando

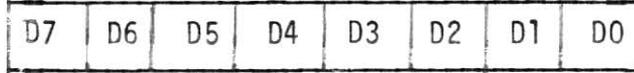
Instrução de 3 bytes



código de operação



{operando 1 (bits menos significativos)



{operando 2 (bits mais significativos)

Fig. III.11 - Formato das Instruções.

### 3.1.9 - COMENTÁRIOS ADICIONAIS

Quando o sistema 8080 é ligado, devido aos fatores aleatórios, os conteúdos dos registradores, contador de programa e Ponteiro de Pilha são ignorados. Desta forma, é conveniente ao se ligar o sistema começar com um RESET. O sinal de RESET zera o contador de programa e leva ainda os "flip-flops" INTE e HLDA ao nível zero. Porém os "flags" de estado ("status") ou qualquer outro registrador (acumulador, registradores e ponteiro de pilha) não são afetados; seus conteúdos permanecem inalterados até que sejam inicializadas por programação.

Hã sistemas que requerem o estado de espera para o processador, apõs o seu acionamento. Para tanto , nas duas primeiras posições de memõria devem ser colocadas as instruções respectivamente EI e HLT, uma vez que, dado o RESET, o programa ẽ iniciado pela posiçã zero de memõria, com INTE no estado 0. Assim, uma interrupçã manual ou automãtica poderã ser utilizada para o inĩcio de processamento.

### 3.2 - SIM8080 - A ROTINA DE SIMULAÇÃO

A rotina SIM8080 ẽ basicamente um implementaçã, em ALGOL, do diagrama de estados, em cada ciclo de mãquina do microprocessador, conforme o diagrama apresentado na Figura III.4.

Desta maneira, a simulaçã do microprocessador ẽ realizada a nĩvel de cada estado, ou seja, cada estado (dos 5 possĩveis: T1, T2, T3, T4 e T5) ẽ simulado como uma unidade indivisĩvel. Em consequẽcia cada instruçã de mãquina ẽ simulada atravẽs dos ciclos de mãquina que lhe estã associados sendo que estes, por sua vez, sã simulados atravẽs de uma sequẽncia de estados no tempo. Assim sendo, a simulaçã permite o monitoramento do microprocessador a um nĩvel de detalhe bem maior que o normal, nĩvel este necessãrio principalmente em aplicações em tempo real, onde o projeto do sistema de atendimento de interrupções ẽ uma das tarefas principais e de mais difĩcil verificaçã.

Assim, a partir deste enfoque, a rotina SIM8080 simula exatamente, estado a estado, o funcionamento do microprocessador INTEL 8080, de modo análogo a uma máquina de estados finita. As instruções de máquina são executadas através de uma sequência de ciclos no tempo, ciclos estes pertencentes a um conjunto de onze ciclos distintos. Por sua vez cada ciclo é executado através de uma sequência de estados, que varia entre um número mínimo de três estados a um número máximo de cinco estados. Cada um desses estados possíveis é "realizado" através de uma sequência de microoperações, sequência esta, em parte, função da instrução de máquina que está sendo realizada no momento.

Cada conjunto de microoperações, que "realiza" um estado da máquina, é executado num intervalo de tempo correspondente a um período do sinal de controle básico ("clock") da máquina. Assim sendo na rotina SIM8080, cada estado é simulado num intervalo de tempo lógico indivisível e as operações de sincronização com eventos externos (por ex. sinais de interrupção simulados) são realizadas no fim de cada estado.

Desta maneira, para cada instrução armazenou-se a sequência de ciclos correspondentes uma estrutura identificada por "TABELAIR".

Para os estados iniciais (T1 e T2) de cada ciclo de máquina, verificou-se que, com pequenas variações, apresentam características gerais comuns, possibilitando, desta forma, a simulação destes estados por meio de subrotinas específicas.

Contudo, os estados T3, T4 e T5, não apresentam tais características. As operações associadas a cada um desses estados são funções

exclusivas da instrução simulada. Sendo assim, a sequência de microoperações, envolvidas nos estados T3, T4 e T5, associados a cada ciclo de uma instrução, foi armazenada numa estrutura denominada MICROPROGRAMA.

E no caso de certas instruções, que apresentam processamento paralelo, no qual aquelas transferências entre registros internos da CPU, que são realizadas nos estados T1 e T2 da instrução seguinte, foram simuladas colocando-as no estado T1 da próxima instrução.

Desta maneira, para cada instrução, num dado ciclo, o simulador (imitando o microprocessador) percorre os estados apresentados no diagrama da Figura III.12, de maneira simplificada.

No estado T1, o microprocessador coloca no seu barramento de endereço A15A0, o endereço de alguma posição de memória ou de algum periférico e, no seu barramento de dados D7D0, a informação de ESTADO ("status") do ciclo corrente. Excetuando a instrução DAD, o sinal SYNC identifica o estado T1 do ciclo.

Como dito anteriormente, para os estados T1 e T2 foram implementadas subrotinas específicas, as quais são descritas na seção 3.22

Ao final de T2 são realizados testes sobre as entradas READY e HOLD e verificação de execução de instrução HLT.

Caso o ciclo seja de busca de instrução (CICLO = 0 ou CICLO = 11 ou CICLO = 13) é chamada a sub-rotina DECODIFICADOR, que procura

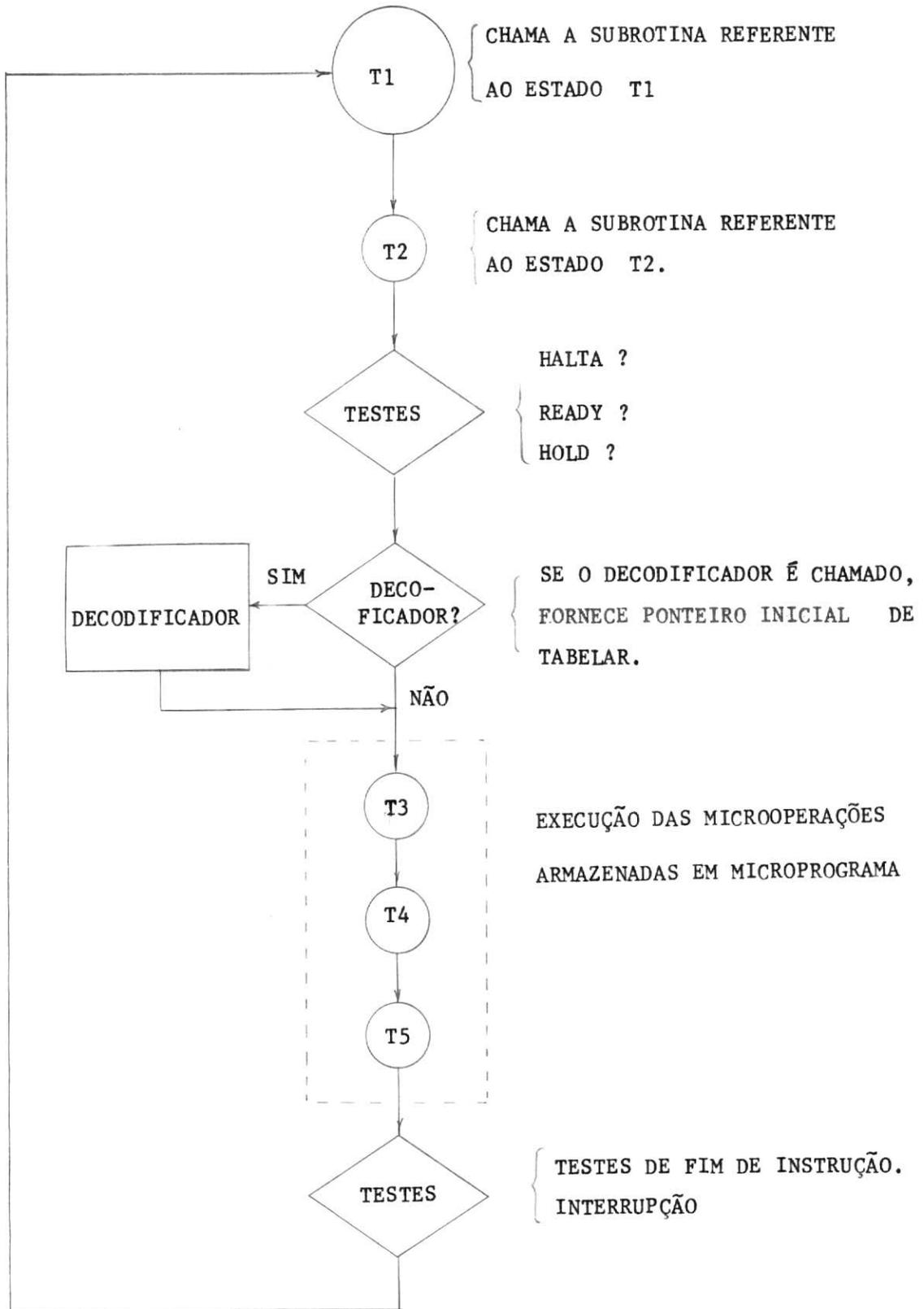


Fig.III.12 - Diagrama de Estados Simplificado.

ra identificar o código de operação colocado em IR. Se o DECODIFICADOR re conhece esta configuração de bits, o mesmo fornece um ponteiro inicial para a estrutura de armazenamento de ciclos TABELAIR.

Para cada ciclo, é fornecido um endereço inicial da se quência de estados armazenada em microprograma executando, desta forma, as operações referentes aos estados T3, T4 e T5.

A sincronização de todo o simulador é feita a partir de rotinas específicas, que atuam na transição de um estado para outro (INCRLOCK, INCRCYCLE e INCRINST). Nestas rotinas são realizados testes e verificação de pedido de interrupção, de pedido de "HOLD", etc.

Desta forma, através de transições sucessivas de estados e de ciclos, uma instrução é simulada.

### 3.2.1 - TIPOS DE CICLOS DE MÁQUINA

A numeração correspondente a cada ciclo de máquina, fornecida no manual do usuário, foi alterada para fins de simulação. Devido à possibilidade de se associar, aos estados T1 e T2 de um ciclo, subro tinas específicas, dadas as características comuns houve, para um tipo de ciclo, a necessidade de se implementar mais de uma subrotina, as sociada a cada um desses estados, e ativadas de acordo com a instrução de máquina a ser executada.

Na Tabela III.5 é apresentada a numeração associada a ca da conjunto T1 e T2, de um ciclo de máquina, utilizada na simulação.

TABELA III.5

TIPOS DE CICLOS DE MÁQUINA E SUBROTINAS DOS ESTADOS COMUNS

NUMERO DO CICLO	ESTADO	SUBROTINA ASSOCIADA	INFORMAÇÃO DE ESTADO	TIPO DE CICLO
0	T1	FETCHT1	D7 D6 D5 D4 D3 D2 D1 D0	Busca de instrução ("FETCH")
	T2	FETCHT2	0 1 0 0 0 1 0 1	
1	T1	MEMREADT1A	1 0 0 0 0 0 1 0	Leitura de Memória
	T2	MEMREADT2A		
2	T1	MEMREADT1B		
3	T1	MEMREADT1C		
2/3	T2	MEMREADT2BC		
4	T1	MEMREADT1A		
4	T1	MEMWRITET1A	0 0 0 0 0 0 0 0	Escrita na memória
	T1	MEMWRITET1B		
	T1	MEMWRITET1C		
	T2	MEMWRITET2ABC		
7	T1	STACKREADT1	1 0 0 0 0 1 1 0	Leitura da pilha
	T2	STACKREADT2		
8	T1	STACKWRITET1	0 0 0 0 0 1 0 0	Escrita na pilha
	T2	STACKWRITET2		
9	T1	INPUTREADT1	0 1 0 0 0 0 1 0	Leitura de periférico
	T2	INPUTREADT2		
10	T1	OUTPUTWRITET1	0 0 0 1 0 0 0 0	Escrita em periférico
	T2	OUTPUTWRITET2		
11	T1	INTERRUPTACKT1	0 0 1 0 0 0 1 1	Interrupção
	T2	INTERRUPTACKT2		
12	T1	HALTACKT1	1 0 0 0 1 0 1 0	Parada ("HALT")
	T2	HALTACKT2		
13	T1	INTERACKHALTT1	0 0 1 0 1 0 0 1	Interrupção no estado de parada
	T2	INTERACKHALTT2		
14	T1	REGISOPT1	não apresenta	Operação entre registros
	T2	REGISOPT2		

Na Tabela III.6 são apresentadas as microoperações realizadas nas subrotinas de simulação dos estados T1 e T2.

### 3.2.2 - TABELAIR

É uma estrutura, como mostra a Figura III.13, onde é armazenada a informação de sequência de ciclos para cada uma das instruções simuladas.

Uma vez obtido o código de operação de uma instrução, num ciclo de busca, o decodificador atua de modo a identificá-lo fornecendo o ponteiro PIR. Este ponteiro determina a posição inicial da sequência de ciclos da instrução reconhecida em TABELAIR.

TABELA III.6

MICROOPERAÇÕES REALIZADAS NAS SUBROTINAS DOS ESTADOS T1 E T2

CICLO	ESTADOT1	ESTADOT2
0	<p><u>FETCHT1</u></p> <p>1 : ADDRBUFFER + PC                      2 : A15A0 + ADDRBUFFER                      3 : D7D0 + 1"10100010"                      4 : ALUREG + 0                      *5 : MACRO1</p>	<p><u>FETCHT2</u></p> <p>1 : D7D0XXX - FLUTUANTE                      2 : PC + PC + 1                      3 : DBIN + 1                      4 : SYNC + 0</p>
1	<p><u>MEMREADT1A</u></p> <p>1 : ADDRBUFFER + PC                      2 : A15A0 + ADDRBUFFER                      3 : D7D0 + 1"10000010"                      4 : MACRO1</p>	<p><u>MEMREADT2A</u></p> <p>1 : D7D0XXX-FLUTUANTE                      2 : PC + PC + 1                      3 : DBIN + 1                      4 : SYNC + 0</p>
2	<p><u>MEMREADT1B</u></p> <p>***1 : SE IRS = 0 ENTÃO                      SE IRS                      00 : ADDRBUFFER + BC                      01 : ADDRBUFFER + DE                      ***10 : ERROPROC                      11 : ERROPROC                      CASO CONTRARIO                      ADDRBUFFER + WZ                      2 : A15A0 + ADDRBUFFER                      3 : D7D0 + 1"10000010"                      4 : MACRO1</p>	<p><u>MEMREADT2B</u></p> <p>****1 : SE IRS = 1 ENTÃO SE HÁ PROXIMO CICLO :                      WZ + WZ + 1                      2 : D7D0XXX - FLUTUANTE                      3 : DBIN + 1                      4 : SYNC + 0</p>
3	<p><u>MEMREADT1C</u></p> <p>1 : ADDRBUFFER + HL                      2 : A15A0 + ADDRBUFFER                      3 : D7D0 + 1"10000010"                      4 : MACRO1</p>	
4	<p><u>MEMWRITET1A</u></p> <p>1 : ADDRBUFFER + HL                      2 : A15A0 + ADDRBUFFER                      3 : D7D0 + 1"00000000"                      4 : MACRO1</p>	<p><u>MEMWRITET2</u></p> <p>1 : SE HÁ PROXIMO CICLO ENTÃO WZ + WZ + 1                      2 : D7D0XXX - FLUTUANTE                      3 : SYNC + 0</p>
5	<p><u>MEMWRITET1B</u></p> <p>1 : ADDRBUFFER + WZ                      2 : A15A0 + ADDRBUFFER                      3 : WZ + 1                      4 : MACRO1</p>	

continuação de Tabela III.6

CICLO	ESTADOT1	ESTADOT2
6	<p><u>MEMWRITET1C</u></p> <p>1 : CASO IR54            00 : ADDRBUFFER + BC            01 : ADDRBUFFER + DE            **10 : ERROPROC            **11 : ERROPROC</p> <p>2 : A15A0 + ADDRBUFFER            3 : D7D0 + 1"00000000"            4 : MACRO1</p>	
7	<p><u>STACKREADT1</u></p> <p>1 : ADDRBUFFER + SP            2 : A15A0 + ADDRBUFFER            3 : D700 + 1"10000110"            4 : MACRO1</p>	<p><u>STACKREADT2</u></p> <p>1 : D700XXX - FLUTUANTE            2 : SE IR543210 NEQ 1"100011"            ENTÃO SP + SP + 1            3 : SYNC + 0            4 : DBIN + 1</p>
8	<p><u>STACKWRITET1</u></p> <p>1 : ADDRBUFFER + SP            2 : A15A0 + ADDRBUFFER            3 : D700 + 1"00000100"            4 : MACRO1</p>	<p><u>STACKWRITET2</u></p> <p>1 : SE HA PROXIMO CICLO ENTÃO SP + SP - 1            2 : D7D0XXX - FLUTUANTE            3 : SYNC + 0</p>
9	<p><u>INPUTREADT1</u></p> <p>1 : ADDRBUFFER + WZ            2 : A15A0 + ADDRBUFFER            3 : D7D0 + 1"01000010"            4 : MACRO1</p>	<p><u>INPUTREADT2</u></p> <p>1 : DBIN + 1            2 : SYNC + 0</p>
10	<p><u>OUTPUTWRITET1</u></p> <p>1 : ADDRBUFFER + WZ            2 : A15A0 + ADDRBUFFER            3 : D7D0 + 1"00010000"            4 : MACRO1</p>	<p><u>OUTPUTWRITET2</u></p> <p>1 : DBIN + 1            2 : SYNC + 0</p>
11	<p><u>INTERRUPTACKT1</u></p> <p>1 : ADDRBUFFER + PC            2 : A15A0 + ADDRBUFFER            3 : D7D0 + 1"11000100"            4 : SE INTEFF = 1 ENTÃO INTEFF + 0            CASO CONTRARIO ERRO            5 : MACRO1</p>	<p><u>INTERRUPTACKT2</u></p> <p>1 : DBIN + 1            2 : SYNC + 0            3 : SE INTEFF = 0 ENTÃO INTEFF + 0            4 : D7D0XXX - FLUTUANTE</p>
12	<p><u>HALTACKT1</u></p> <p>1 : ADDRBUFFER + PC            2 : A15A0 + ADDRBUFFER            3 : D7D0 + 1"10001010"            4 : MACRO1            5 : HALTA + 1</p>	<p><u>HALTACKT2</u></p> <p>1 : SYNC + 0            2 : D7D0XXX - FLUTUANTE</p>

continuação da Tabela III.6

CICLO	ESTADOT1	ESTADOT2
13	<p style="text-align: center;"><u>INTERACKHALT1</u></p> <p>1 : ADDRBUFFER + PC                  2 : A15A0 + ADDRBUFFER                  3 : D7D0 + 1"10010100"                  4 : SE INTEFF = 0 ENTÃO INTEFF + 0                  CASO CONTRARIO : ERRO + SIMULAÇÃO                  SUSPENSA</p>	<p style="text-align: center;"><u>INTERACKHALT2</u></p> <p>1 : DBIN + TRUE                  2 : SE INTEFF = 0 ENTÃO INTEFF + 0                  3 : SYNC + 0                  4 : D7D0XXX - FLUTUANTE</p>
14	<p style="text-align: center;"><u>REGISOPT1</u></p> <p>1 : SE HÁ PROXIMO CICLO                  ENTÃO                  IR54                  00 : ACT + C                  01 : ACT + E                  10 : ACT + L                  11 : ACT + SPL                  CASO CONTRARIO                  IR54                  00 : ACT + B                  01 : ACT + D                  10 : ACT + H                  11 : ACT + SPH</p>	<p style="text-align: center;"><u>REGISOPT2</u></p> <p>1 : SE HA PROXIMO CICLO                  ENTÃO                  TMP + L                  ALUAUX + ACT + TMP                  ALUREG + ALUAUX                  CASO CONTRARIO                  TMP + H                  ALUAUX + ACT + TMP + CY                  ALUREG + ALUAUX</p>

NOTA

\* MACRO1 é definida como um conjunto de operações comuns

MACRO1 {  
 WRBARRA ← 1  
 DBIN ← 0  
 HLDA ← 0  
 WAIT ← 0  
 SYNC ← 1

\*\* IRS - Representa o bit numero 5 (D5) do IR

\*\*\* ERROPROC mensagem de erro na microprogramação da TABELAIR

A verificação da existência de próximo ciclo é feita pela subrotina PROXCIC (HACIC), onde o parametro formal HACIC é uma booleana. A subrotina consulta TABELAIR e, caso seja encontrado o próximo ciclo, HACIC é verdadeiro.



Na Figura III.13(a), é apresentado esquematicamente um no de TABELAIR, na qual o campo CIC armazena o numero de um dos ciclos constante na Tabela III.5. O campo ENDER fornece um ponteiro para a sequência de estados em MICROPROGRAMA, descrita na seção 3.2.3.

Terminada a simulação de um ciclo do processador, o ponteiro PIR é incrementado, de modo que PIR aponta para o proximo ciclo a ser simulado. Desta maneira, a sequência de ciclos é percorrida e simulada.

A instrução termina quando o no de sinalização de fim de instrução for apontado por PIR:

Como a estrutura de TABELAIR é sequencial, na implementação ao final da sequência de ciclos, deixou-se uma área de reserva constituída de alguns nos. Isto permite rapída alteração de TABELAIR, sem que seja necessária a modificação dos valores iniciais de PIR fornecida pelo Decodificador. A Figura III.13(b) mostra em TABELAIR essa área de reserva, bem como um no de sinalização de fim de instrução.

A implementação de TABELAIR foi num "VALUE ARRAY", declaração da linguagem ALGOL(B-6700/B-7700 - ALGOL Language Reference Manual - Burroughs) que simula uma memória so de leitura ("ROM"). Além disso, como a palavra do B-6700 é de 48 bits, utilizou-se palavras parciais para constituir o no de TABELAIR. O campo CIC é constituída de 8 bits, enqanto que o campo ENDER é de 16 bits. Deste modo, dois nos de TABELAIR foram acomodados numa palavra do B-6700.

Como o "VALUE ARRAY" requer uma forma adequada para o armazenamento desses dados, foi utilizado um programa auxiliar denominado MONTA, que, a partir de um formato mais livre dos cartões de dados, converte-o na forma adequada, gerando a tabela na forma compactada apresentada acima.

No Apêndice A, é apresentada a listagem dos cartões de entrada de MONTA, e o formato gerado para TABELAIR.

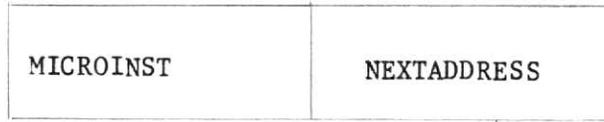
Assim, qualquer modificação ou acréscimo nas sequências de ciclos poderá ser introduzida no sistema, bastando, para tanto, utilizar a rotina MONTA e recompilar o programa de simulação.

### 3.2.3 - MICROPROGRAMA

Microprograma é conjunto de listas associadas à TABELAIR, de modo que cada lista armazena a sequência de microoperações correspondentes a cada estado T3, T4 e T5 de um ciclo de máquina.

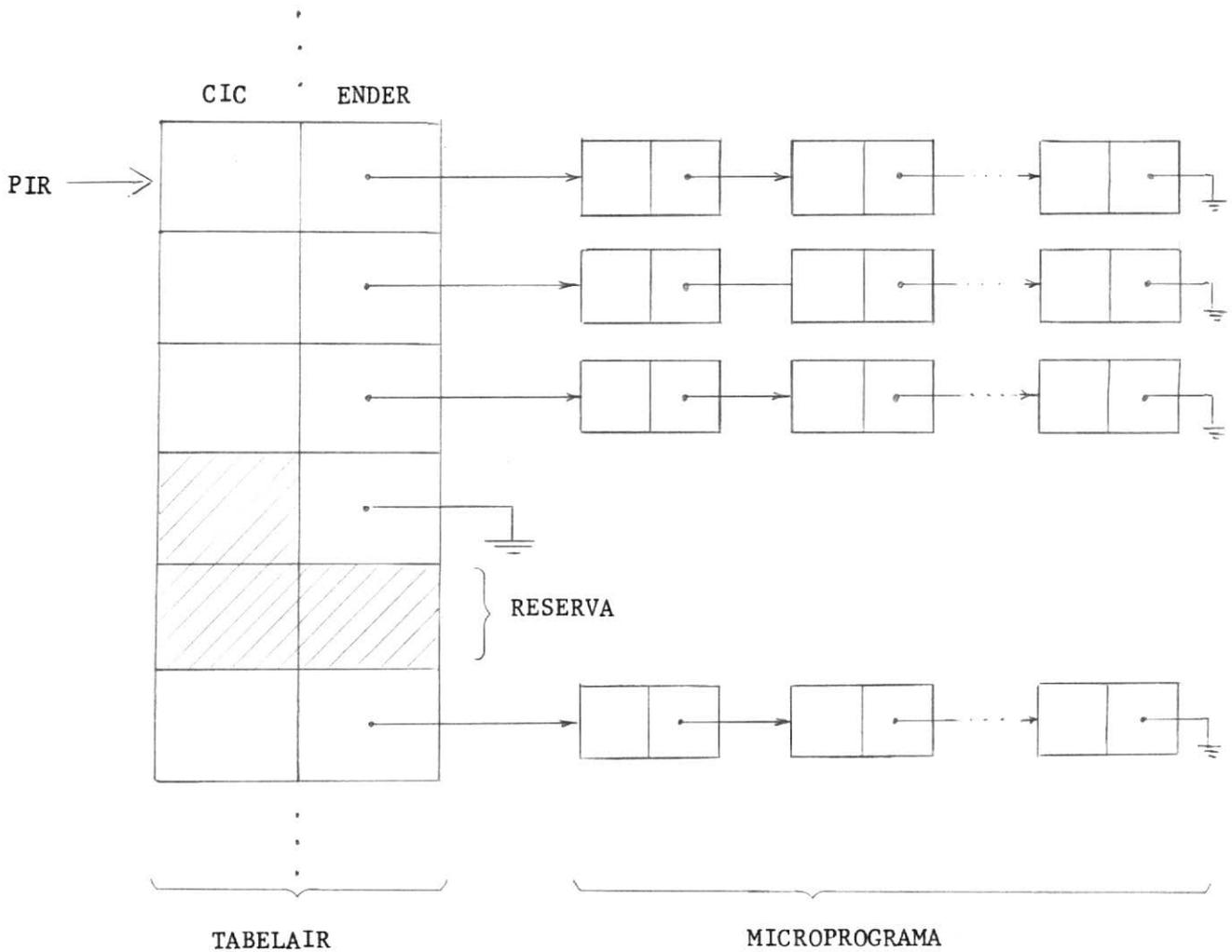
Uma vez determinada a sequência de ciclos de máquina referente a uma dada instrução, para cada ciclo corresponde um n° de TABELAIR, cujo campo ENDER, contém o ponteiro para o início da sequência de microoperações a serem realizadas no ciclo.

A Figura III.14(a) representa um n° de microprograma, com a designação de seus campos. O campo MICROINSTRUÇÃO contém o número de uma das microoperações apresentadas na tabela. O campo NEXTADDRESS, guarda o ponteiro para o n°, em microprograma que contém a próxima microope



numero correspondente a uma das microoperações (8)      endereço da próxima microoperação (16 bits)

(a) - N<sup>o</sup> de Microprograma



(b) - Estrutura Microprograma e sua Relação com TABELAIR

Fig. III.14 - Microprograma

ração. O ciclo é simulado através do percorrimto da lista até que se encontre o nó de sinalização de final da sequência de microoperações. A sinalização é feita através do campo NEXTDDRESS que, quando for nulo, indica o fim do ciclo. Então o PIR é incrementado a um novo ciclo se inicia.

Em cada ciclo os estados são delimitados através de nós especiais que ativam microoperações responsáveis pela chamada das rotinas de sincronização.

A Figura III.14 apresenta a estruturação de microprograma e seu relacionamento com TABELAIR.

A Figura III.15 mostra a instrução SHLD, com suas sequências de ciclos e estados utilizadas para a sua simulação, bem como trecho de TABELAIR e MICROPROGRAMA em SHLD.

Procedimentos análogos à SHLD foram utilizados em cada uma das instruções, de modo a obter as estruturas finais de TABELAIR e MICROPROGRAMA, apresentadas nos Apêndice A e B respectivamente.

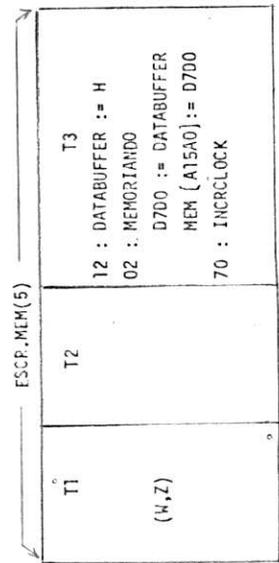
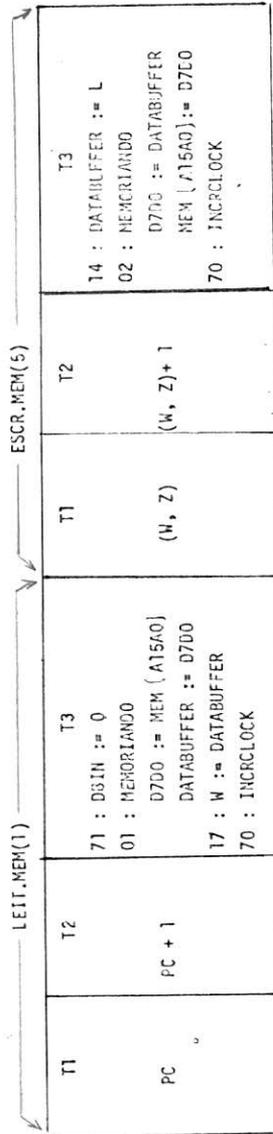
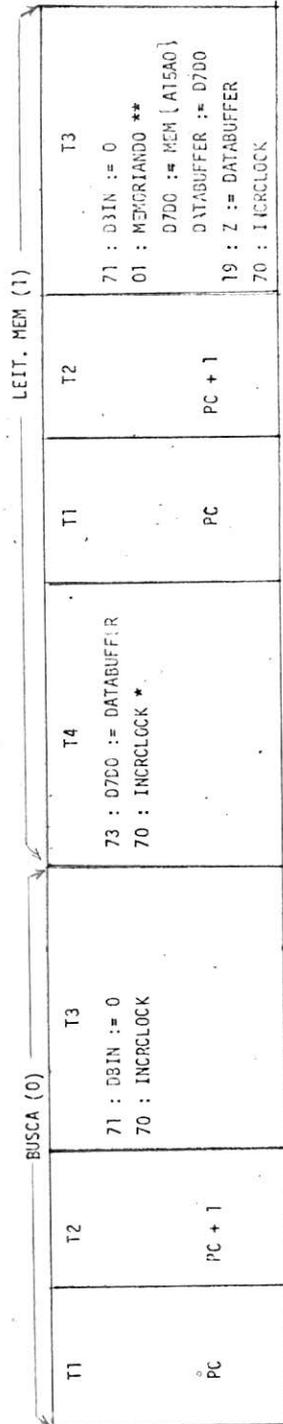
De maneira análoga a TABELAIR, MICROPROGRAMA foi armazenado num "VALUE ARRAY" e ainda, através de palavras parciais, dois nós de MICROPROGRAMA podem ser colocados numa palavra do computador B-6700.

Valem, também, as mesmas considerações, quanto à rotina MONTA mencionada em 3.2.2.

Instrução SHLD <B<sub>2</sub>> <B<sub>3</sub>> Armazena o conteúdo dos registradores H e L nas posições de memória endereçadas pelos bytes <B<sub>2</sub>> e <B<sub>3</sub>> da instrução. |<B<sub>3</sub>><B<sub>2</sub>>| ← (L), |<B<sub>3</sub>><B<sub>2</sub>>+ 1| ← (H)

Código de Operação: 00 100 010

PIR inicial dado pelo decodificador := 27



#### NOTAS

\* INCRLOCK :sub-rotina de sincronização e incremento de tempo

\*\* MEMORIANDO: teste de limites de memória declarado no comando de simulação MEMORY (vide Cap. IV - Descrição da Linguagem).

Fig. III.15 (a) - Instrução SHLD.

EXEMPLO SHLD

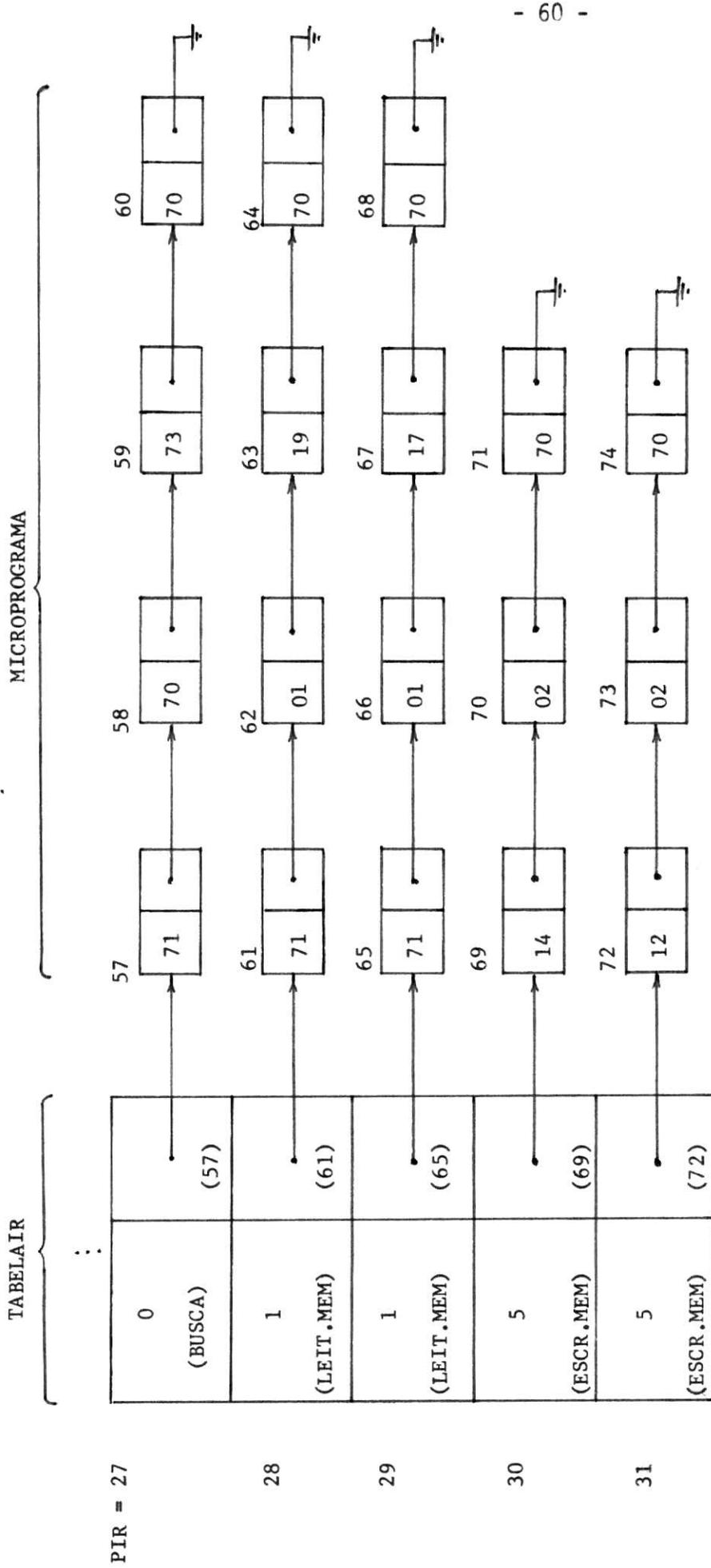


Fig. III.15(b) - TABELAIR e MICROPROGRAMA em SHLD

No Apêndice B, é apresentada a listagem correspondente a MICROPROGRAMA. MICROINST e NEXTADDRESS, correspondem respectivamente ao primeiro e segundo campo do cartão de entrada.

A separação entre os estados T3, T4 e T5 é feita através da microinstrução MICRO70, que chama a rotina INCRLOCK, que incrementa o relógio da simulação e verifica a ocorrência de algum evento associado a interrupções, "HOLD" e "RESET".

#### 3.2.4 - AS MICROOPERAÇÕES

Segue-se, uma lista das microoperações implementadas e a numeração correspondente para a simulação de cada instrução do repertório do 8080.

====> DEFINE F = MICROOPERACOES

MICROOPERACOES UTILIZADAS NA EXECUCAO DA INSTRUCAO DE ACORDO COM O DIAGRAMA DE BLOCOS FUNCIONAL DO CPU 8080

```
MICRO0 = #
MICRO1 = BEGIN
    MEMORIANDO;
    D7D0:=MEM[A15A0];
    DATABUFFER:=D7D0;
    END#
MICRO2 = BEGIN
    MEMORIANDO;
    D7D0:=DATABUFFER;
    MEM[A15A0]:=D7D0;
    END#
MICRO3 = BEGIN
    D7D0:=DISP[BUS.[31:8]];
    DATABUFFER:=D7D0;
    END#
MICRO4 = BEGIN
    D7D0:=DATABUFFER;
    DISP[BUS.[31:8]]:=D7D0;
    END#
MICRO5 = Z:=8*IR543#
MICRO6 = BEGIN
    D7D0XXX;
    A15A0XXX;
    END#
MICRO7 = IR:=DATABUFFER #
MICRO8 = DATABUFFER:=PCL #
MICRO9 = PCL:=DATABUFFER #
MICRO10 = DATABUFFER:=PCH #
MICRO11 = PCH:=DATABUFFER #
MICRO12 = DATABUFFER:=H #
MICRO13 = H:=DATABUFFER #
MICRO14 = DATABUFFER:=L #
MICRO15 = L:=DATABUFFER #
MICRO16 = DATABUFFER:=W #
MICRO17 = W:=DATABUFFER #
MICRO18 = DATABUFFER:=Z #
MICRO19 = Z:=DATABUFFER #
MICRO20 = DATABUFFER:=A #
MICRO21 = A:=DATABUFFER #
MICRO22 = BEGIN REG3.[45:1]:=0; REG3.[43:1]:=0;
    REG3.[41:1]:=1;
    DATABUFFER:=FLAG; END #
MICRO23 = FLAG:=DATABUFFER #
MICRO24 = DATABUFFER:=CASE IR210 OF (B,C,D,E,H,L,M,A) #
```

```
MICRO25 = CASE IR543 OF
    BEGIN
        B:=DATABUFFER;
        C:=DATABUFFER;
        D:=DATABUFFER;
        E:=DATABUFFER;
        H:=DATABUFFER;
        L:=DATABUFFER;
        ERROM;
        A:=DATABUFFER;
    END #,
%MICRO26 EFETUA A TRANSFERENCIA DA PARTE MENOS SIG.
%NIFICATIVA DO PAR DE REGISTROS ESPECIFICADO PELO
%IR54 PARA O DATABUFFER
MICRO26 = DATABUFFER:=CASE IR54 OF (C,E,L,SPL) #,
%
%MICRO27 EFETUA A TRANSFERENCIA DA PARTE MAIS SIGNI
%FICATIVA DO PAR DE REGISTROS ESPECIFICADO PELOS
%BITS 5 E 4 DO IR PARA O DATABUFFER
MICRO27 = DATABUFFER:=CASE IR54 OF (B,D,H,SPH) #,
%
%MICRO28 EFETUA A TRANSFERENCIA DO CONTEUDO DO DATA
%BUFFER PARA A PARTE MENOS SIGNIFICATIVA DO PAR DE
%REGISTROS ESPECIFICADO POR IR54
MICRO28 = CASE IR54 OF
    BEGIN
        C:=DATABUFFER;
        E:=DATABUFFER;
        L:=DATABUFFER;
        SPL:=DATABUFFER;
    END #,
% MICRO29 IDEM A MICRO28 COM A PARTE MAIS SIGNIFICA
% TIVA DO PAR DE REGISTROS
MICRO29 = CASE IR54 OF
    BEGIN
        B:= DATABUFFER;
        D:= DATABUFFER;
        H:= DATABUFFER;
        SPH:= DATABUFFER;
    END #,
%
MICRO30 = DATABUFFER:=TMP #,
%
MICRO31 = TMP:=DATABUFFER #,
%
MICRO32 = DATABUFFER:=ALUREG #,
%
MICRO33 = A:=ALUREG #,
%
%MICRO34 EFETUA TRANSFERENCIA DO ALUREG PARA UM DOS
%REGISTROS ESPECIFICADOS POR IR543
MICRO34 = CASE IR543 OF
    BEGIN
        B:=ALUREG;
        C:=ALUREG;
        D:=ALUREG;
        E:=ALUREG;
        H:=ALUREG;
        L:=ALUREG;
        ERROM;
        A:=ALUREG;
    END #,
```

```
%
%MICRO35 ESPECIFICA O CONTEUDO DO FLAG EM FUNCAO DO
%RESULTADO EM ALUREG
MICRO35 = BEGIN
    S:=ALUAUX7;
    BEGIN
        REAL PTEST;
        PTEST:=K:=0;
        THRU 8 DO
            BEGIN
                PTEST:=**+BIT(K);
                K:=**+1;
            END;
        P:=REAL(NOT BOOLEAN(PTEST));
        ZER:=IF PTEST EQL 0 THEN 1 ELSE 0;
    END;
END #
%
MICRO36 = BEGIN
    ALUAUX:=TMP+1;
    AC:=IF TMP4 NEQ BIT(4) THEN 1 ELSE 0;
END #
MICRO37 = BEGIN
    ALUAUX:=TMP+4"FF";
    AC:=IF TMP4 EQL BIT(4) THEN 1 ELSE 0;
END #
%
MICRO38 = #
MICRO39 = CASE IR543 OF
    BEGIN
        TMP:=B;
        TMP:=C;
        TMP:=D;
        TMP:=E;
        TMP:=H;
        TMP:=L;
        ERR0M;
        TMP:=A;
    END #
%
%MICRO40 INCREMENTA OU DECREMENTA O PAR DE REGISTRO
%ESPECIFICADO POR IR54
%SE BIT 3 DO IR FOR 0 INCREMENTA
%      1 DECREMENTA
MICRO40 = INCDEC:=CASE IR543 OF (
    BC+1,
    BC+4"FF",
    DE+1,
    DE+4"FF",
    HL+1,
    HL+4"FF",
    SP+1,
    SP+4"FF") #
%
MICRO41 = CASE IR54 OF
    BEGIN
        BC:=INCDEC;
        DE:=INCDEC;
        HL:=INCDEC;
        SP:=INCDEC;
    END #
```

```
%
MICRO42 = BEGIN
    ALUAUX:=ACT+TMP ;
    CY:=MSBALUAUX;
    PY:=REAL(BOOLEAN(TMP4) EQV BOOLEAN(ACT4));
    AC:=IF BIT(4) EQL PY
        THEN 1 ELSE 0;
    END #
MICRO43 = BEGIN
    ALUAUX:=TMP+CY;
    AC:=IF TMP4 NEQ BIT(4) THEN 1 ELSE 0;
    TMP:=ALUAUX;
    CY:=MSBALUAUX;
    ALUAUX:=ACT+TMP;
    IF CY=0 THEN CY:=MSBALUAUX;
    IF AC=0 THEN
        PY:=REAL(BOOLEAN(TMP4) EQV BOOLEAN(ACT4));
        AC:=IF BIT(4) EQL PY
            THEN 1 ELSE 0;
    END #
MICRO44 = BEGIN
    ALUREG:=REAL(NOT BOOLEAN(TMP));
    TMP:=ALUREG+1;
    ALUAUX:=ACT+TMP;
    PY:=REAL(BOOLEAN(TMP4) EQV BOOLEAN(ACT4));
    AC:=IF BIT(4) EQL PY
        THEN 1 ELSE 0;
    CY:=IF MSBALUAUX=0 THEN 1 ELSE 0;
    END #
MICRO45 = BEGIN
    ALUAUX:=TMP+CY;
    TMP:=ALUAUX;
    TMP:=REAL(NOT BOOLEAN(TMP));
    TMP:=TMP+1;
    ALUAUX:=ACT+TMP;
    PY:=REAL(BOOLEAN(TMP4) EQV BOOLEAN(ACT4));
    AC:=IF BIT(4) EQL PY
        THEN 1 ELSE 0;
    CY:=IF MSBALUAUX=0 THEN 1 ELSE 0;
    END #
MICRO46 = BEGIN
    ALUREG:=REAL(BOOLEAN(ACT) AND BOOLEAN(TMP));
    CY:=0;
    END #
MICRO47 = BEGIN
    ALUREG:=REAL(NOT(BOOLEAN(ACT) EQV BOOLEAN(TMP)));
    CY:=0;
    END #
MICRO48 = BEGIN
    ALUREG:=REAL(BOOLEAN(ACT) OR BOOLEAN(TMP)) ;
    CY:=0;
    END #
%
MICRO49 = ACT:=A #
%
%MICRO50 EFETUA TRANSFERENCIA DO TMP PARA UM DOS
%REGISTROS ESPECIFICADOS PELU IR543
```

```
MICRO50 = CASE IR543 OF
      BEGIN
          B:=TMP;
          C:=TMP;
          D:=TMP;
          E:=TMP;
          H:=TMP;
          L:=TMP;
          ERRDM;
          A:=TMP;
      END #,
%
%MICRO51 EFETUA TRANSFERENCIA DO CONTEUDO DO REGIS-
%TRO ESPECIFICADO POR IR210 PARA O TMP
MICRO51 = CASE IR210 OF
      BEGIN
          TMP:=B;
          TMP:=C;
          TMP:=D;
          TMP:=E;
          TMP:=H;
          TMP:=L;
          ERRDM;
          TMP:=A;
      END #,
%
%MICRO52 COMPLEMENTA TODOS OS BITS DO ACUMULADOR
%
MICRO52 = A:=REAL (NOT BOOLEAN(A)) #,
MICRO53 = CY:=1"1" #,
%
MICRO54 = CY:=REAL (NOT BOOLEAN(CY)) #,
%
MICRO55 = PC:=HL #,
%
MICRO56 = PC:=WZ #,
MICRO57 = CONDICAO:=CASE IR543 OF
      (
          IF ZFR=0 THEN TRUE ELSE FALSE, % NZ
          IF ZFR=1 THEN TRUE ELSE FALSE, % Z
          IF CY=0 THEN TRUE ELSE FALSE, % NC
          IF CY=1 THEN TRUE ELSE FALSE, % C
          IF P=0 THEN TRUE ELSE FALSE, % PD
          IF P=1 THEN TRUE ELSE FALSE, % PE
          IF S=0 THEN TRUE ELSE FALSE, % P
          IF S=1 THEN TRUE ELSE FALSE % M
      )#,
MICRO58 = SP:=HL #,
MICRO59 = WZ:=HL #,
MICRO60 = HL:=WZ #,
MICRO62 = DE:=WZ #,
MICRO63 = HL:=INCDEC#,
MICRO61 = HL:=DE #,
%
%MICRO64 EFETUA A SOMA DO PAR DE REGISTROS HL COM
%O ESPECIFICADO POR IR54
MICRO64 = WZ:=CASE IR54 OF
      (HL+BC,
       HL+DE,
       HL+HL,
       HL+SP) #,
```

```
MICRO65 = BEGIN
    TMP:=A;
    ALUAUX:=4;
    IF (Y GEQ 10) OR (AC=1)
        THEN BEGIN ALUAUX:= TMP+6;
                    AC:=IF BOOLEAN(TMP4) EQV
                        BOOLEAN(BIT(4)) THEN 0 ELSE 1;
                END;
    IF (X GEQ 10) OR (CY=1)
        THEN BEGIN
            ALUAUX:=ALUAUX+4"60";
            CY:=MSBALUAUX;
        END;
END #
MICRO66 = BEGIN
    ALUAUX:=2*A;
    CY:=MSBALUAUX;
    ALUAUX0:=CY;
END #
MICRO67 = BEGIN
    ALUAUX:=A;
    CY:=ALUAUX0;
    ALUAUX:=ALUAUX/2;
    ALUAUX7:=CY;
END #
MICRO68 = BEGIN
    ALUAUX:=2*A;
    ALUAUX0:=CY;
    CY:=MSBALUAUX;
END #
MICRO69 = BEGIN
    ALUAUX:=A;
    MSBALUAUX:=CY;
    CY:=ALUAUX0;
    ALUAUX:=ALUAUX/2;
END #
MICRO70 = BEGIN
    INCRLOCK;
END #
MICRO71 =
    BEGIN
        DBIN:=FALSE;
    END #
MICRO72 =
    BEGIN
        PROG:=FALSE;
    END #
MICRO73 = D7D0:=DATABUFFER #
MICRO74 = CY:=MSBALUAUX #
MICRO75 = L:=ALUREG #
MICRO76 = H:=ALUREG #
MICRO77 = WAIT:=TRUE #
MICRO78 = IF NOT CONDICA0 THEN BEGIN PIR:=PIR+2;
                                         PC:=PC+2; END #

MICRO79 = WRBARRA:=FALSE #
MICRO80 = INTEFF:=TRUE #
MICRO81 = INTEFF:=FALSE #
MICRO82 = IF CONDICA0 THEN SP:=SP-1 #
MICRO83 = SP:=SP-1 #
MICRO84 = W:=0 #
```

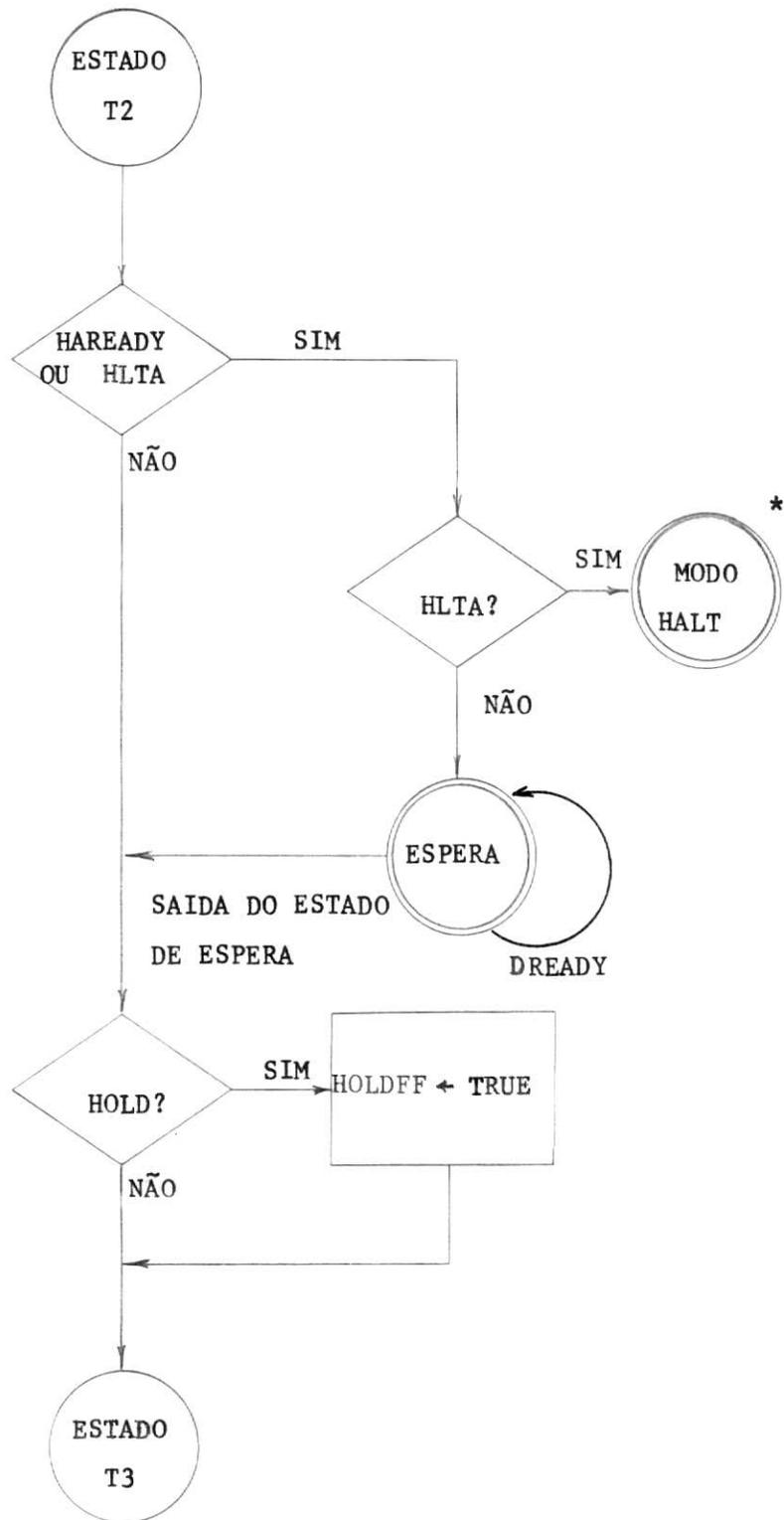
### 3.2.5 - SIMULAÇÃO DOS EVENTOS READY, HOLD, INTERRUPÇÃO E ESTADO HALT

Os eventos READY, HOLD e INTERRUPÇÃO são estabelecidos através dos comandos de controle de simulação correspondentes, enquanto que o estado HALT provém da execução da instrução HLT, que ativa HLTA.

O comando de controle de simulação "READY", fornece ao simulador, na variável global "DREADY", o número de estados de espera e, na variável global booleana "HAREADY", a informação da existência de pedido de espera.

O comando de controle de simulação "HOLD" fornece a duração do evento HOLD na variável global "DURHOLD". O tempo e a frequência do pedido de HOLD também são dados pelo comando. A variável INSTHOLD determina o tempo do pedido em função das opções oferecidas CLOCK, CYCLE e INST (vide Cap. IV) e DHOLD; o período dos pedidos de HOLD. HAHOLD é uma variável booleana que indica a existência do evento. Os testes dos eventos READY e HOLD são dados na Figura III.16.

As interrupções são simuladas de modo análogo ao evento HOLD. Ao invés de se fornecer a duração de eventos, como no caso anterior, as prioridades associadas às interrupções são estabelecidas. Há 8 níveis de prioridades possíveis: 0, 1, 2, ..., 7. Como, também, as opções CLOCK, CYCLE e INST são disponíveis no comando INTER, foram criadas indicadores correspondentes. A Figura III.17 apresenta os indicadores mencionados (INDCYCLE, INDCLOCK, INDINST) bem como os outros parâmetros.



\* Modo HALT é descrito na seção 3.2.5

Fig. III.16 - Testes de "READY", "HOLD" e "HLTA"

INDICADORES BOOLEANAS

MOSTRA	0	1	2	3	4	5	6	7
0	200	00	0	0	20	15	0	
0	0	225	0	0	0	25	15	
0	0	0	0	0	0	0	0	
0	0	25	0	0	0	5	0	
0	1	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	

último pedido de interrupção  
 prioridade  
 conteúdo: próximo instante de pedido de interrupção.  
 intervalo se = 0 é apenas uma interrupção é gerada

QUEM  
1

indica interrupção pendente de maior prioridade (Iniciado com 8).

AQUEM  
1

conteúdo anterior de QUEM

- THAINTER indica se há interrupção por "clock"
- CHAIINTER indica se há interrupção por "cycle"
- IHAINTER indica se há interrupção por "Inst".

Fig. III.17 - Interrupção com prioridade.

No diagrama de INCRLOCK:

- NROCKLOCK    é o contador do número de pulsos de relógio ("clock")  
ocorridos desde o início da simulação.
- HARESET      variável booleana que notifica a presença de RESET.
- TVARIOS      indicador booleano de existência de interrupção, de pe  
dido de HOLD ou comando WRITE.
- INST, INSTW, INSTHOLD e VETINT [ I ] estabelecidos pelos comandos  
RESET, WRITE, HOLD e INTER respectivamente, indicam o  
instante de ativação do evento correspondente.

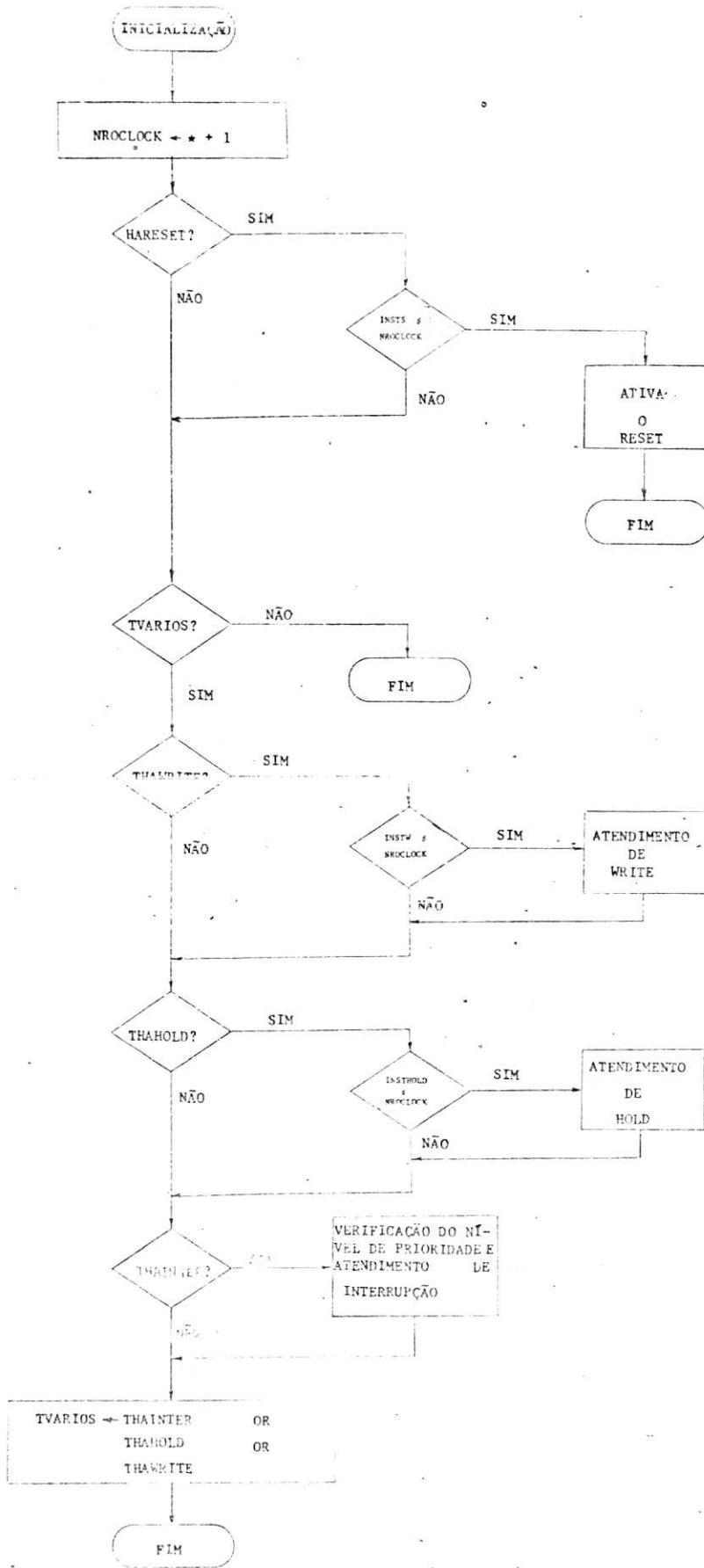
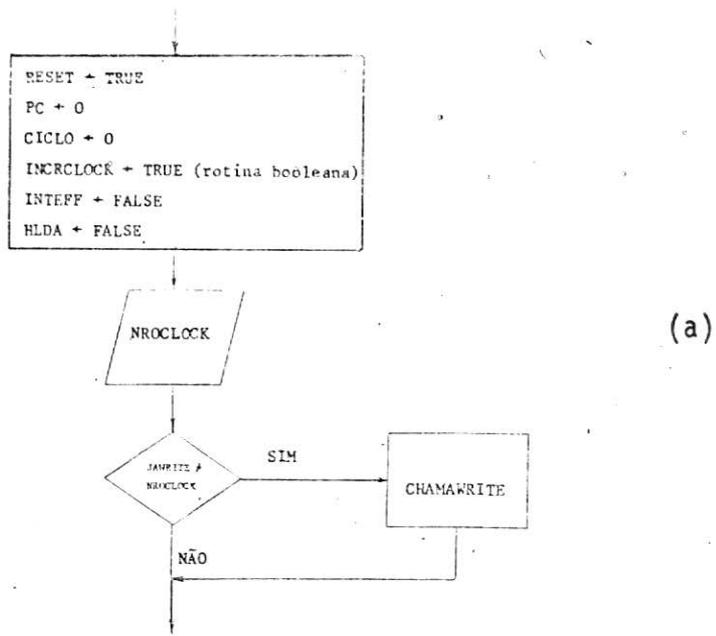
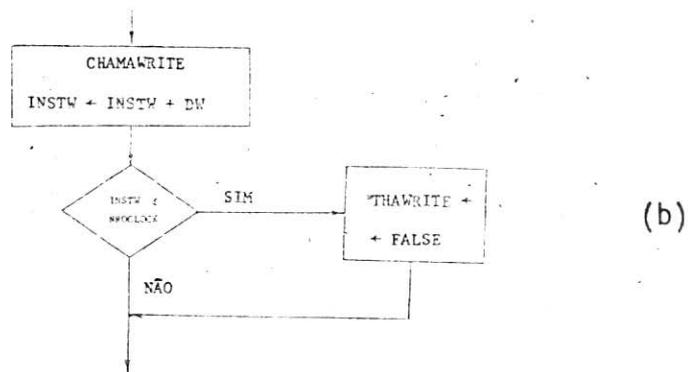


Fig. III.18 - Diagrama de Blocos de INCRINST.



(a)



(b)

Fig. III.19(a) - Ativação do RESET  
(b) - Atendimento de WRITE

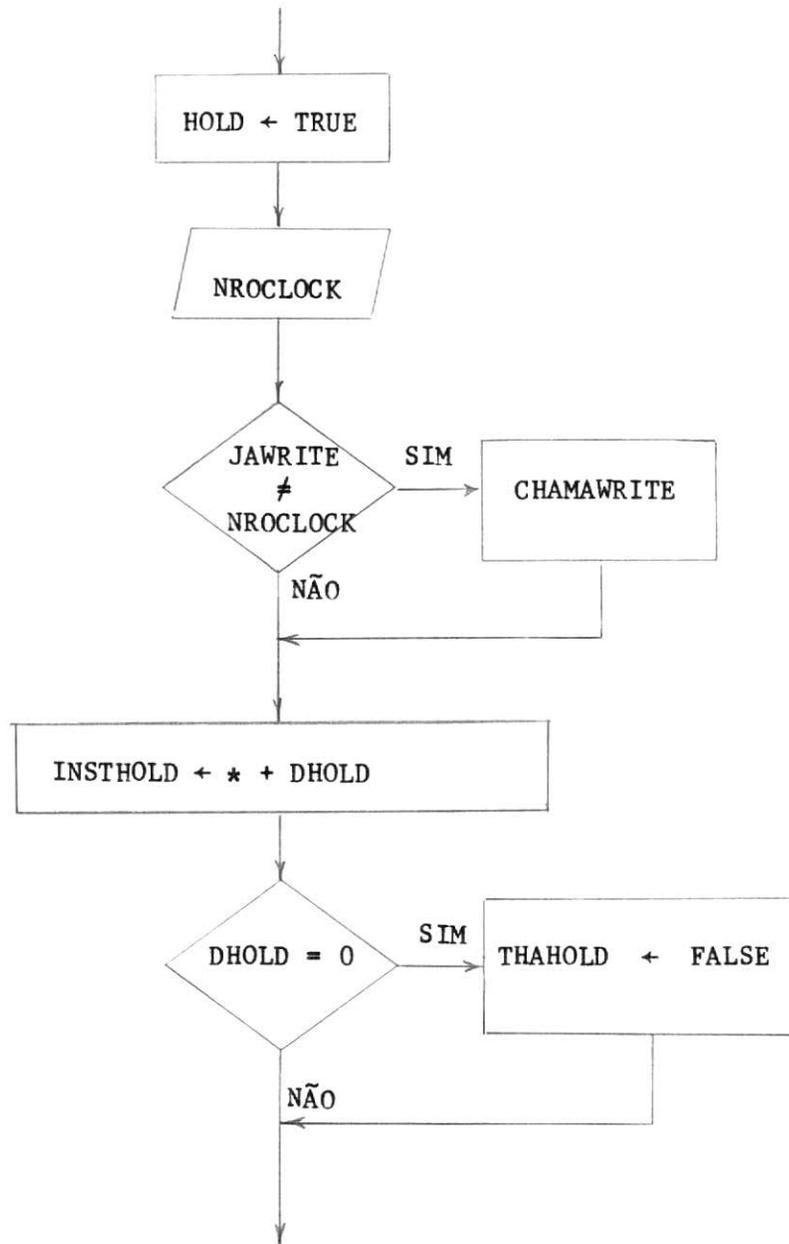


Fig. III.20 - Atendimento de HOLD

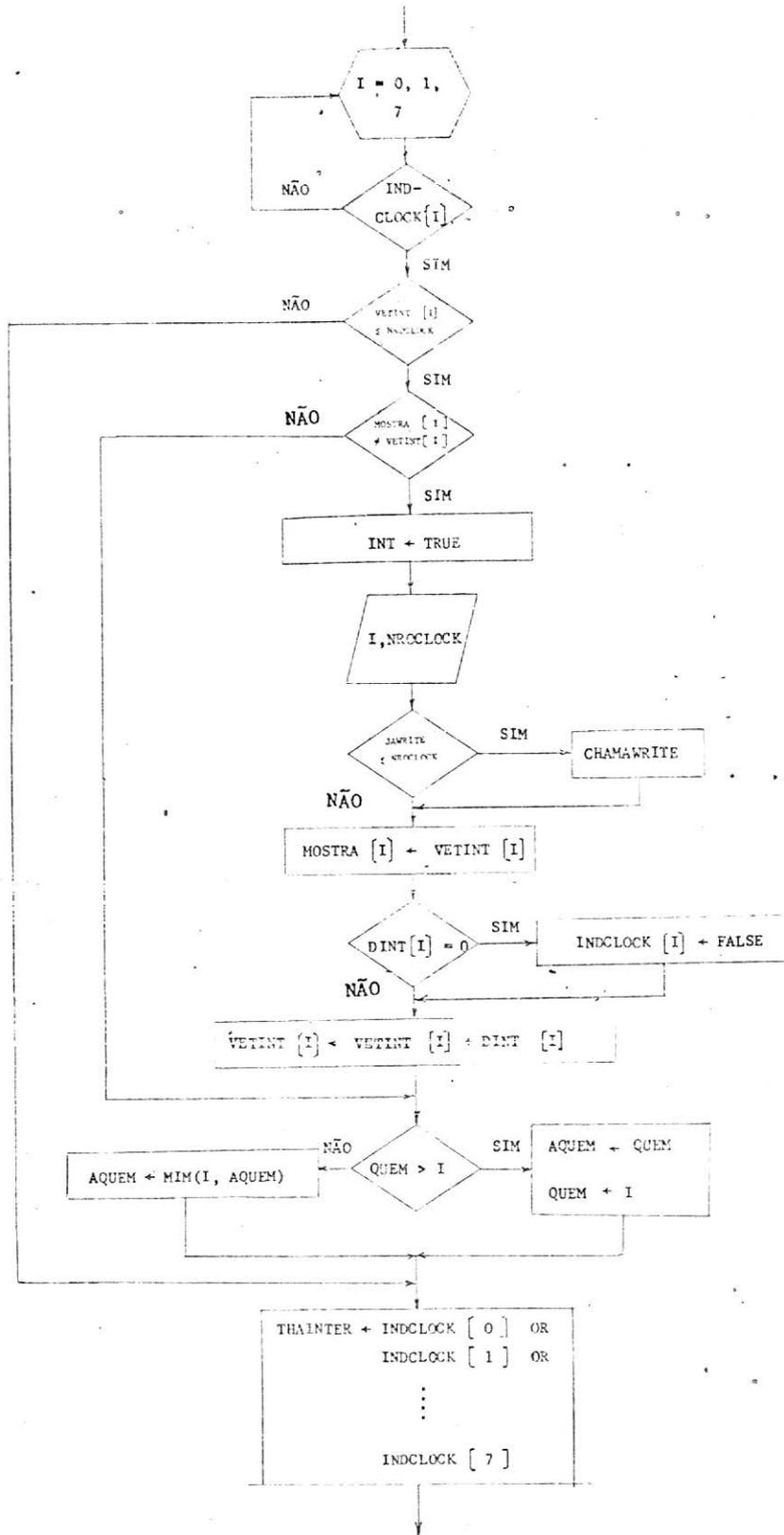


Fig. III.21 - Atendimento de Interrupção.

### 3.2.6 - DIAGRAMA DE BLOCOS DO SIMULADOR SIM8080

Nesta seção é apresentado o diagrama do blocos correspon  
dente à implementação da sub-rotina SIM8080.

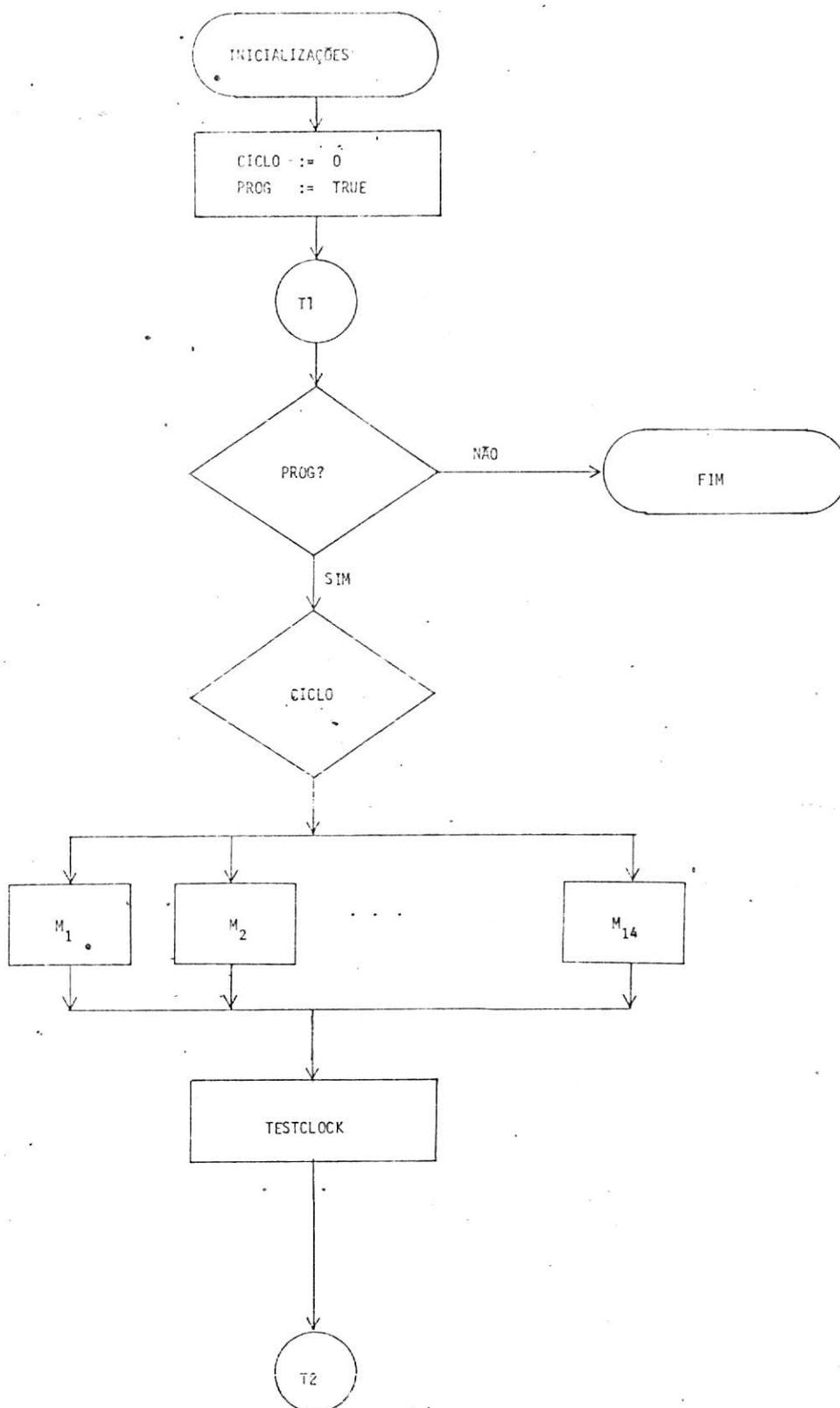


Fig. III.22 (a) - Simulação do Estado T1.

Fig. III.22 - Diagrama de Blocos de Simulador SIM 8080.

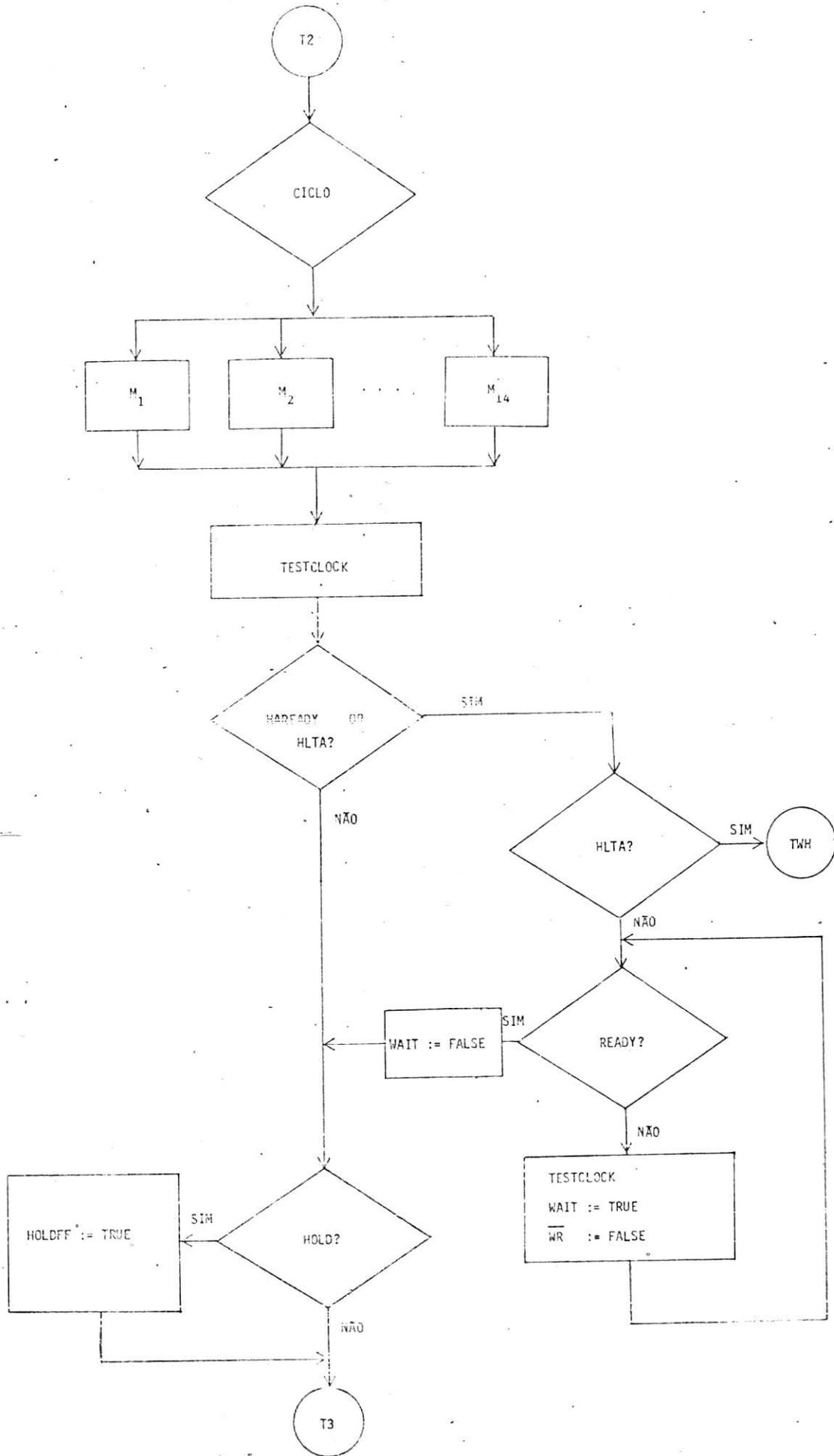


Fig. III.22 (b) - Simulação do Estado T2.

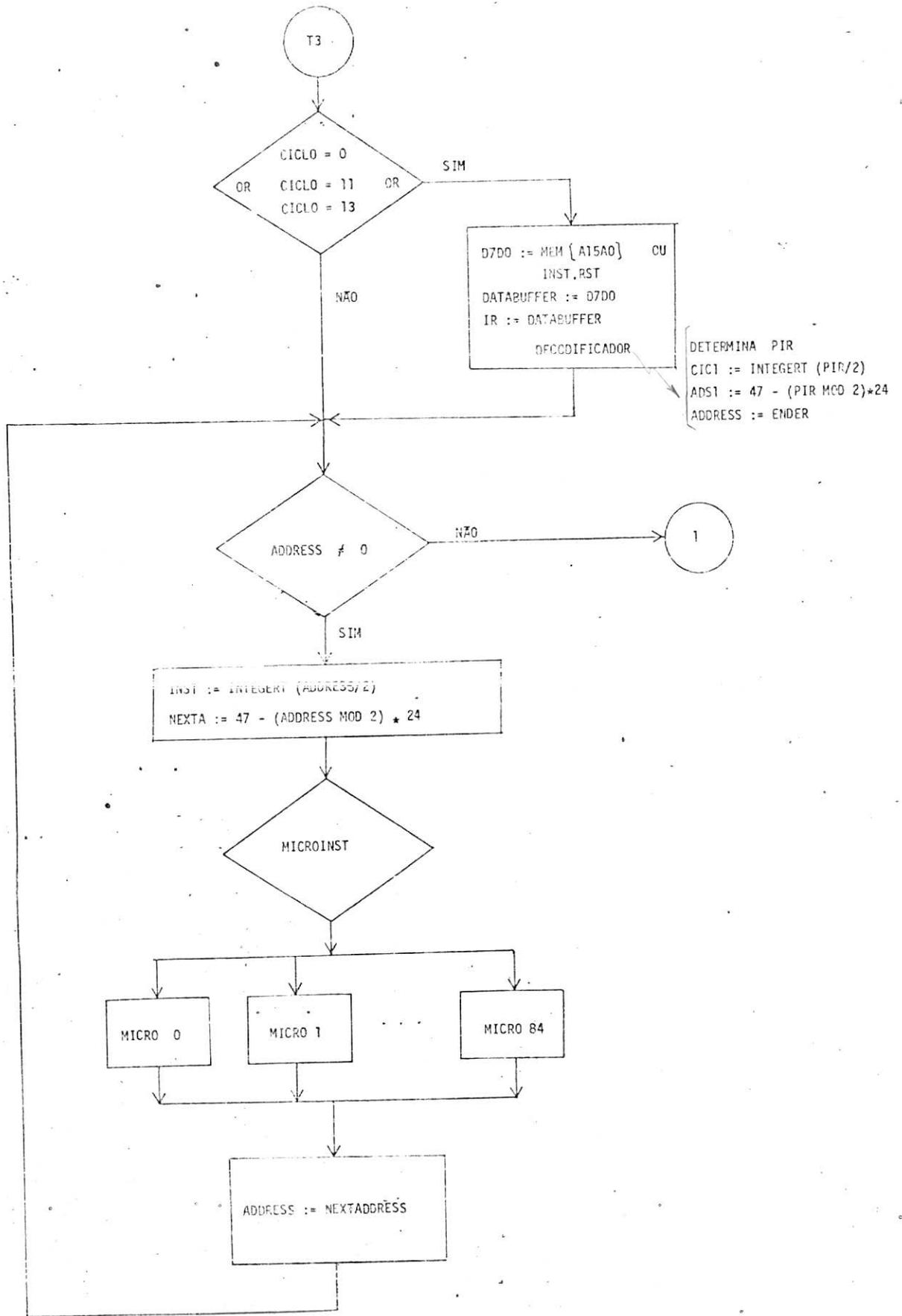


Fig. III.22 (c) - Simulação do Estado T3, T4 e T5.

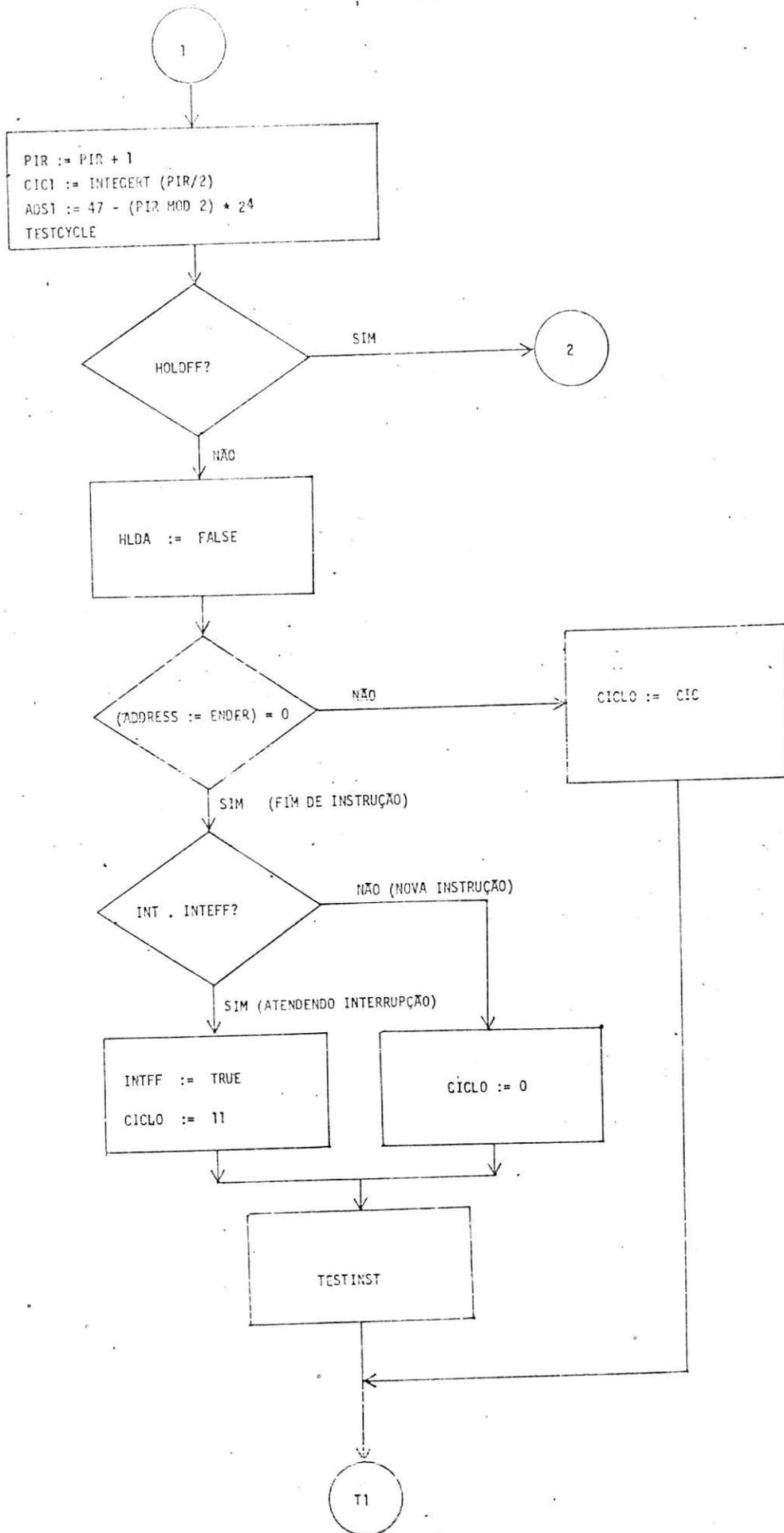


Fig. III.22 (d) - Teste de Fim de Instrução

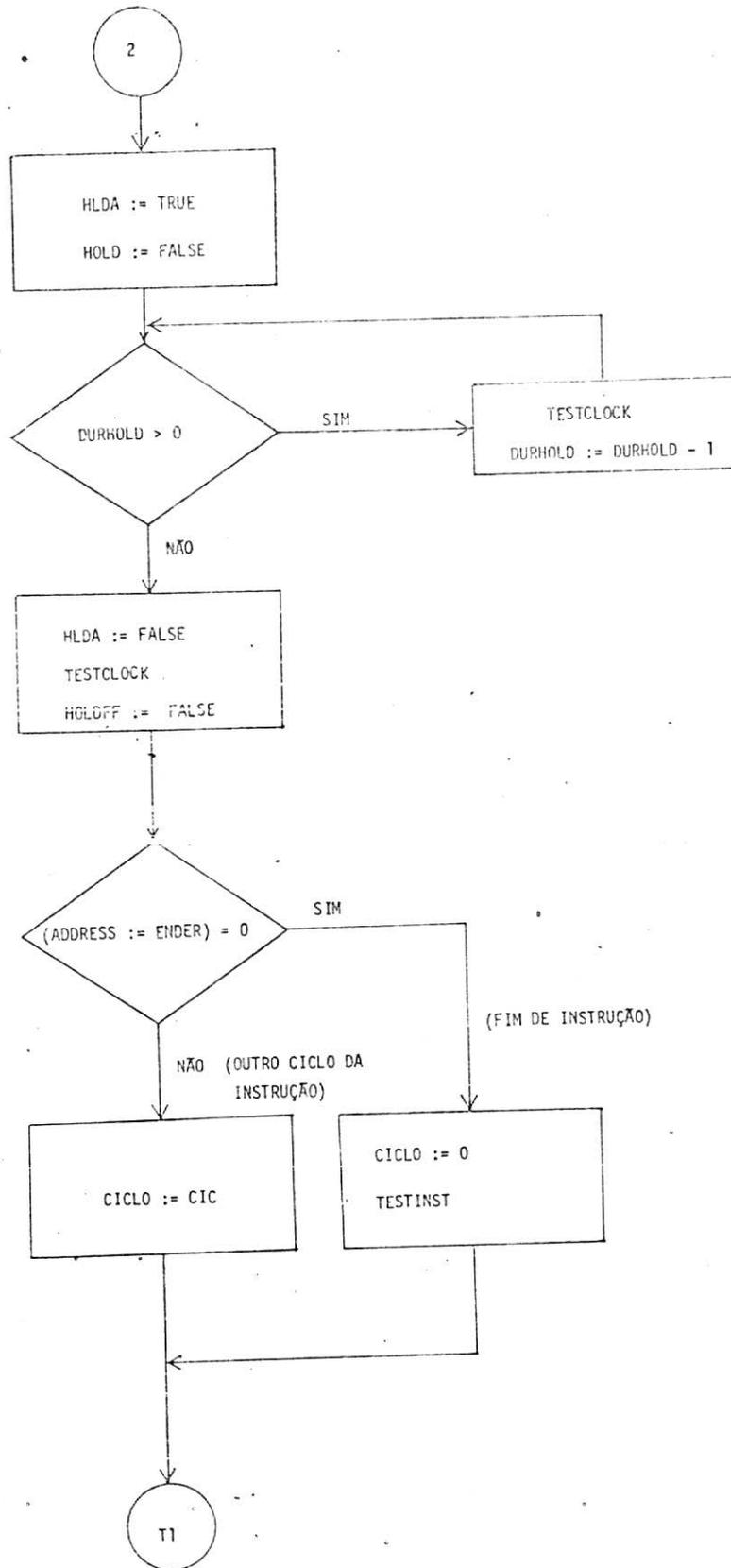


Fig. III.22 (e) - Atendimento de HOLD e Teste de Fim de Instrução.

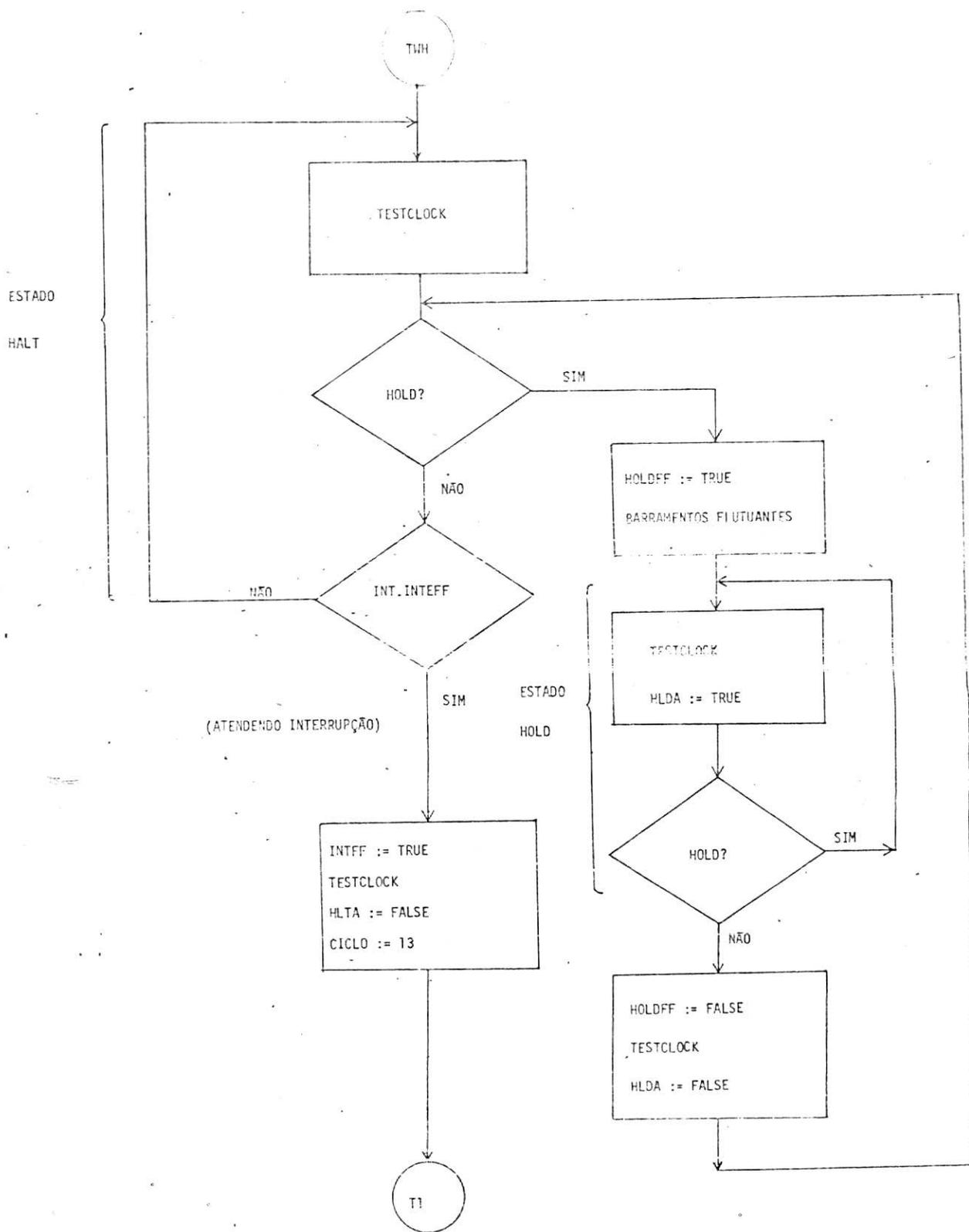


Fig. III.22 (f) - Simulação do Estado HALT. e HOLD

## CAPÍTULO IV

### A LINGUAGEM DE CONTROLE DE SIMULAÇÃO

A linguagem de controle de simulação do sistema especifica as condições iniciais do simulador, bem como as características do monitoramento desejado e os eventos externos que se deseja ativar. Como se trata de uma linguagem de controle específica de um simulador é constituída basicamente por um conjunto de comandos que controlam diretamente cada uma dessas condições, sendo totalmente condicionada pelas características do microprocessador.

#### 4.1 - COMPONENTES DA LINGUAGEM

Sintaxe:

```
<componentes da linguagem> ::= <simbolos básicos> |  
                                <palavras reservadas> |  
                                <constante> |  
                                <comentários> |  
                                <programa de controle>
```

##### 4.1.1 - SIMBOLOS BÁSICOS E PALAVRAS RESERVADAS

Sintaxe:

```
<simbolos básicos> ::= <letras> |
```

*<digito>* |

*<vazio>* |

*<caracteres especiais>*

*<letra>* ::= A | B | C | D | E | F | G | H | I | J | K | L | M | N |  
O | P | Q | R | S | T | U | V | W | X | Y | Z

*<digito>* ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

*<caracteres especiais>* ::= ( | ) | , | ; | = | %

*<vazio>* ::= {conjunto vazio de caracteres}

*<palavras reservadas>* ::= *<palavra chave>* |  
*<identificador especial>* |  
*<conectivo>*

*<palavra chave>* ::= MEMORY | READY | LOAD | INTER | HOLD | RESET |  
REFER | WRITE | INPUT | OUTPUT | COPY | SAVE

*<identificador especial>* ::= A | B | C | D | E | H | L | ZERO |  
CARRY | SIGN | PARITY | ACARRY | INT |  
HOLD | INTE

*<conectivo>* ::= *<indicador de tempo>* | TO | WITH | DUMP | STATUS |  
AND | M

*<indicador de tempo> ::= CLOCK | CYCLE | INST*

Os símbolos básicos e as palavras reservadas constituem o "alfabeto" sobre o qual a linguagem de controle de simulações é definida.

#### 4.1.2 - CONSTANTES

Sintaxe:

*<caracter octal> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7*

*<constante> ::= <constante octal> |  
                  <constante decimal>*

*<constante octal> ::= <caracter octal> |  
                  <constante octal><caracter octal>*

*<constante decimal> ::= <numero>*

*<numero> ::= <dígito> |  
                  <numero><dígito>*

Apenas dois tipos de constante podem ser usados na linguagem de controle, a *<constante octal>* e a *<constante decimal>*, sendo que o ponto do programa onde utilizar cada um deles será definido quando da definição dos comandos.

#### 4.1.3 - COMENTÁRIOS

O comentário é uma forma de documentar o programa, podendo ser inserido em qualquer ponto, sempre de acordo com a sintaxe:

$$\langle \text{comentário} \rangle ::= \% \{ \text{qualquer sequência de caracteres até o fim do cartão} \}$$

Os caracteres à direita do sinal "%" são, então, ignorados pelo analisador.

#### 4.1.4 - PROGRAMA DE CONTROLE DE SIMULAÇÃO:

A sintaxe de cada programa de controle de simulação tem a seguinte estrutura:

$$\langle \text{sequência de comandos} \rangle ::= \langle \text{comandos simples} \rangle | \langle \text{sequência de comandos} \rangle ; \langle \text{comando simples} \rangle$$
$$\langle \text{conjuntos de comandos} \rangle ::= \langle \text{sequência de comandos} \rangle ; \langle \text{comando de finalização} \rangle ;$$
$$\langle \text{programa de controle} \rangle ::= \langle \text{conjunto de comandos} \rangle | \langle \text{programa de controle} \rangle \langle \text{conjunto de comandos} \rangle$$

O programa de controle de simulação pode ser constituído por uma sequência de  $\langle \text{conjunto de comandos} \rangle$ , onde cada  $\langle \text{conjunto de co$

*mandos*> especifica as condições iniciais e as condições de monitoramento vigentes em cada ativação do simulador. Cada *<comando de finalização>*, ao ser analisado, provoca uma ativação do simulador. Desta maneira, em um mesmo programa de controle, poderemos especificar varias "simulações", sob condições totalmente distintas.

#### 4.2 - OS COMANDOS DE CONTROLE DE SIMULAÇÃO

Sintaxe:

*<comandos de controle> ::= <comandos simples> |  
<comando de finalização>*

*<comandos simples> ::= <comandos de inicialização> |  
<comandos de monitoramento> |  
<comandos de eventos>*

Os comandos de controle são divididos em duas classes gerais, sendo que a classe dos *<comandos simples>* é por sua vez dividida em 3 subclasses de acordo com as características funcionais dos comandos que as constituem.

##### 4.2.1 - OS COMANDOS DE INICIALIZAÇÃO:

Sintaxe:

*<comandos de inicialização> ::= <comando MEMORY> |*

<comando READY> |

<comando LOAD> |

<comando COPY>

<comando MEMORY> ::= MEMORY (<numero>, <numero>)

<comando READY> ::= READY <numero>

<comando LOAD> ::= LOAD <sequência de inicializações>

<sequência de inicializações> ::= <inicialização simples> |  
<sequência de inicializações> ,  
<inicialização simples>

<inicialização simples> ::= <identificador-especial>=<constante-  
octal> |  
M (<endereço>)= <constante octal> |  
M (<endereço>)= <sequência de constan\_  
tes>

<endereço> ::= <numero>

<sequência de constantes> ::= <constante-octal> |  
<sequência de constantes><constante  
octal> |

<comando SAVE> ::= COPY M(<numero>, <numero>) |  
COPY STATUS AND M (<numero>, <numero>)

## OBSERVAÇÕES

- O <comando MEMORY> especifica o espaço de memória a ser utilizado na simulação (o par de números indica os endereços limites). Deve ser utilizado sempre como o primeiro comando do primeiro <conjunto de comandos> do programa de controle. O <comando READY> especifica o intervalo de tempo (em múltiplos do ciclo do "clock" básico) que a memória utiliza antes de fornecer o sinal READY no pino correspondente do microprocessador.

- O <comando LOAD> inicializa os registros e as posições de memória especificados com os valores indicados pelas constantes octais utilizadas.

- O <comando COPY> inicializa os registros do microprocessador e/ou as posições de memória, especificadas com os valores armazenados anteriormente num arquivo em disco (de nome "SALVADOR"), através do <comando SAVE>.



O <comando REFER> significa que toda acesso à área de memória, por ele especificada, fará com que seja listado o estado de microprocessador e os conteúdos das posições de memória, especificados no <comando WRITE>.

O <comando WRITE> especifica a partir de que instante da simulação e com que frequência devem ser listados o estado do microprocessador e os conteúdos das posições de memória especificadas.

O <comando de E/S> indica, através da <sequência de números>, os endereços de periféricos que, quando referenciados na simulação de instruções de E/S do microprocessador, implicam em ações especiais por parte do simulador, a saber:

- os endereços especificados em "INPUT" indicam que, toda vez que for simulada uma instrução de entrada a eles se referindo, o simulador deve retirar sequencialmente de arquivos de cartão, fornecidos ao simulador (arquivos de nomes DADOS1, DADOS2,..... DADOS8), um carácter (codificado em ASCII (e colocá-lo na via de dados do microprocessador.

- os endereços especificados em "OUTPUT" indicam que, toda vez que for simulada uma instrução de saída a eles se referindo, o conteúdo da via de dados do microprocessador deve ser listado.

O <comando SAVE> é semelhante ao <comando COPY>, porém com o sentido oposto.

#### 4.2.3 - COMANDOS DE GERAÇÃO DE EVENTOS EXTERNOS

Sintaxe:

*<comandos de eventos> ::= <comando INTER> |  
<comando HOLD>*

*<comando INTER> ::= INTER (<nível de interrupção>) <indicador de  
tempo> (<ocasião>)*

*<nível de interrupção> ::= <numero>*

*<comando HOLD> ::= HOLD (<duração>) <indicador de tempo> (<ocasião> )*

*<duração> ::= <numero>*

O *<comando INTER>* controla a execução de interrupções externas no microprocessador. As interrupções serão atendidas de acordo com a prioridade indicada pelo *<nível de interrupção>* (de 0 a 7, onde 0 é a maior prioridade).

O *<comando HOLD>* indica em que instante o microprocessador deve ser colocado no estado Hold e qual a duração deste estado.



## CAPÍTULO V

### COMENTÁRIOS E CONCLUSÕES FINAIS

A performance do simulador pode ser considerada satisfatória, tendo em vista o nível de detalhe empregado ser bem maior que o usual. A simulação desce a níveis de pulsos de relógio ("clock"), para se obter maior flexibilidade, quanto a possíveis modificações e aumento de capacidade, e quanto a diminuição da complexidade, na simulação de dispositivos de E/S, bem como de eventos externos.

Uma vez que na transição de um estado para outro do processador, são ativadas rotinas de sincronização e verificação de eventos pendentes, tem-se, aí, um consumo relativamente grande do tempo de simulação. Principalmente no atendimento desses eventos (interrupções com prioridades, HOLD, WRITE e RESET).

Porém uma avaliação sistemática do simulador só poderá ser fornecida quando se fizer a comparação das instruções executadas, passo a passo, no simulador, com as mesmas instruções executadas no protótipo do microcomputador, que se encontra em fase de montagem.

## AGRADECIMENTOS

Ao Instituto de Pesquisas Espaciais (INPE) e ao Instituto Tecnológico de Aeronáutica (ITA) de onde se destacam pela atuação neste trabalho:

Eng. Ricardo Corrêa de Oliveira Martins, M.Sc., orientador do INPE, com quem praticamente dividi os encargos deste trabalho, e que possibilitou a realização do mesmo.

Eng. Jacinto da Encarnação Cavaco Mendes, M.Sc., orientador no ITA.

Doroti Akico Tiba, que com grande dedicação datilografou este trabalho.

José Benedito Soares Junior (Nino) que muito auxiliou na depuração de programas.

Os meus sinceros agradecimentos.

## BIBLIOGRAFIA

- BURROUGHS. *B.6700/B.7700 Algol Language: Reference Manual*. Detroit 1974.
- CHU, Y. *Introduction to Computer Organization*. New Jersey, Prentice Hall, 1970.
- ELECTRONICS. New York, V. 49, n° 8, Apr. 1976.
- GEAR, C.W. *Computer Organization and Programming*. New York, McGraw-Hill, 1969.
- GRIES, D. *Compiler Construction for Digital Computers*. New York, Wiley 1971. pp 50-83. Cap III.
- IEEE Transactions on Industrial Electronics and Control Instrumentation. New York, V. 22, n° 3, Aug. 1975.
- INTEL - 8080 Microprocessor User's Manual. Santa Clara, Ca., USA, 1975.
- INTEL - 8080 Assembly Language Programming Manual. Santa Clara, Ca. USA, 1974.
- INTEL - Intellec 8/Mod 80 Microcomputer Development System Reference Manual, Santa Clara, Ca., USA, 1974.

INTEL - Interp/80 User's Manual. Santa Clara, Ca., USA, 1975.

KNUTH, D.E. *The Art of Computer Programming*. 2 ed. Reading, Addison  
-Wesley, 1975, V.1

APÊNDICE A

LISTAGEM DE MONTAMEMORIA (MONTA)

TABELAIR







PROGRAMA FONTE

```
=====
1 <=== ACP
2
3 <=== LXI RP
4 H
5
6
7
8 <=== DAD RP
9 H
10
11
12
13 <=== STAX RP
14 H
15
16
17 <=== LDAX RP
18 H
19
20
21
22
23
24
25
26 <=== SHLD
27 H
28
29
30
31
32 <=== LHLD
33 H
34
35
36
37
38 <=== STA
39 H
40
41
42
43 <=== LDA
44 H
45
46
47
48
49
50
51
52 <=== -IAX RP / DCBX RP
53 H
54
55
56 <=== IARW
=====
```

57	003 0159	=	
58	034 0167		
59	000 0000		
60			
61			
62	000 0164	<=== INRR	
63	000 0000	=	
64			
65			
66	000 0174	<=== DCRH	
67	003 0173	=	
68	000 0143		
69	000 0000		
70			
71			
72	000 0190	<=== DCHR	
73	000 0000	=	
74			
75			
76	000 0200	<=== MVIR	
77	001 0201	=	
78	004 0203		
79	000 0000		
80			
81			
82	000 0215	<=== MVIR	
83	001 0215	=	
84			
85			
86	000 0224	<=== RLC	
87	000 0000	=	
88			
89			
90	000 0234	<=== RRC	
91	000 0000	=	
92			
93			
94	000 0242	<=== RAL	
95	000 0000	=	
96			
97			
98	000 0250	<=== RAR	
99	000 0000	=	
100			
101	000 0258	<=== DAA	
102	000 0000	=	
103			
104			
105			
106	000 0266	<=== CMA	
107	000 0000	=	
108			
109			
110	000 0273	<=== STC	
111	000 0000	=	
112			
113			
114	000 0280	<=== CMC	
115	000 0000	=	
116			
117			

000 0219	118	<===	HALT
000 0000	119	=	
	120		
000 0226	121	<===	MOVEM
003 0300	122	=	
000 0000	123		
	124		
	125		
000 0307	126	<===	MOVEM
004 0311	127	=	
000 0000	128		
	129		
	130		
000 0317	131	<===	MOVEM2
000 0000	132	=	
000 0000	133		
	134		
000 0324	135	<===	ADDR
000 0000	136	=	
	137		
	138		
000 0317	139	<===	ADDR
000 0000	140	=	
	141		
	142		
000 0308	143	<===	SUPR
000 0000	144	=	
	145		
	146		
000 0350	147	<===	SUPR
000 0000	148	=	
	149		
	150		
	151		
000 0370	152	<===	ANAR
000 0000	153	=	
	154		
000 0381	155	<===	YRAN
000 0000	156	=	
	157		
	158		
000 0392	159	<===	CRAR
000 0000	160	=	
	161		
	162		
000 0403	163	<===	CPPR
000 0100	164	=	
	165		
	166		
003 0413	167	<===	ADDR
003 0417	168	=	
003 0100	169		
	170		
	171		
	172		
003 0413	173	<===	ADDR
003 0427	174	=	
003 0100	175		
	176		
	177		
000 0413	178	<===	SUPR

003 0457	179	=
000 0000	180	
	181	
000 0413	182	<=== SPRM
003 0467	183	=
000 0000	184	
	185	
	186	
000 0413	187	<=== AMAP
003 0457	188	=
000 0000	189	
	190	
	191	
000 0413	192	<=== XRAM
003 0467	193	=
000 0000	194	
	195	
	196	
000 0413	197	<=== DRAM
003 0477	198	=
000 0000	199	
	200	
	201	
000 0413	202	<=== CUPM
003 0467	203	=
000 0000	204	
	205	
	206	
000 0501	207	<=== RET COND
007 0510	208	=
007 0510	209	
007 0510	210	
000 0000	211	
	212	
000 0529	213	<=== RFT
007 0510	214	=
007 0510	215	
007 0510	216	
000 0000	217	
	218	
000 0529	219	<=== PCP RP
007 0529	220	=
007 0529	221	
007 0533	222	
000 0000	223	
	224	
000 0529	225	<=== PCP PSA
007 0529	226	=
007 0529	227	
007 0529	228	
000 0000	229	
	230	
000 0529	231	<=== PCHL
000 0000	232	=
	233	
	234	
000 0529	235	<=== SFHL
000 0000	236	=
	237	
	238	
	239	

000	0507	240	<===	JMP	COND
001	0506	241	=		
001	0500	242			
000	0000	243			
		244			
		245			
000	0508	246	<===	JMP	
001	0506	247	=		
001	0500	248			
000	0000	249			
		250			
000	0508	251	<===	OUT	
001	0505	252	=		
010	0600	253			
000	0000	254			
		255			
		256			
000	0508	257	<===	IN	
001	0505	258	=		
009	0607	259			
000	0000	260			
		261			
		262			
		263			
000	0604	264	<===	DI	
000	0000	265	=		
		266			
		267			
000	0601	268	<===	EI	
000	0000	269	=		
		270			
		271			
000	0508	272	<===	XTHL	
007	0608	273	=		
007	0600	274			
008	0606	275			
008	0600	276			
000	0000	277			
		278			
		279			
000	0608	280	<===	XCHG	
000	0000	281	=		
		282			
		283			
000	0657	284	<===	CALL	COND
001	0603	285	=		
001	0667	286			
008	0602	287			
008	0605	288			
000	0000	289			
		290			
		291			
000	0670	292	<===	CALL	
001	0603	293	=		
001	0606	294			
009	0672	295			
008	0675	296			
000	0000	297			
		298			
		299			
000	0670	300	<===	PUSH	RP

004	0669	301	=
004	0661	302	
000	0000	303	
		304	
		305	
000	0670	306	<=== PUSH PSW
004	0667	307	=
004	0700	308	
000	0000	309	
		310	
		311	<=== ACI
000	0707	312	=
001	0713	313	
000	0000	314	
		315	
		316	
000	0707	317	<=== ACI
001	0723	318	=
000	0000	319	
		320	
		321	
000	0707	322	<=== SUI
001	0713	323	=
000	0000	324	
		325	
		326	
000	0707	327	<=== SUI
001	0713	328	=
000	0000	329	
		330	
		331	
000	0707	332	<=== ANI
001	0753	333	=
000	0000	334	
		335	
		336	
000	0707	337	<=== XRI
001	0753	338	=
000	0000	339	
		340	
		341	
000	0707	342	<=== ORI
001	0773	343	=
000	0000	344	
		345	
		346	
000	0707	347	<=== CPI
001	0753	348	=
000	0000	349	
		350	
		351	
000	0707	352	<=== RST
004	0700	353	=
004	0602	354	
000	0000	355	

APENDICE B

MICROPROGRAMA





PROGRAMA FONTE

```

=====
070 0003 0001  ===== INSTRUCAO NOP          CODIGO DO RPO 000
071 0004 0002  =====
072 0005 0003  <==== ESTADOS T3 E T4 DO CICLO(=0) DE FETCH
073 0006 0004  =====
074 0007 0005  <==== INSTRUCAO LXI RP          CODIGO DO RPO 001
075 0008 0006  =====
076 0009 0007  <==== ESTADO T3 DO CICLO(=0) DE FETCH
077 0010 0008  <==== ESTADO T4
078 0011 0009  =====
079 0012 0010  <==== ESTADO T3 DO CICLO(=1) DE MEMBRAS
080 0013 0011  =====
081 0014 0012  <==== ESTADO T3 DO CICLO(=1) DE MEMBRAS
082 0015 0013  =====
083 0016 0014  <==== ESTADO T3 DO CICLO(=1) DE MEMBRAS
084 0017 0015  =====
085 0018 0016  <==== ESTADO T3 DO CICLO(=1) DE MEMBRAS
086 0019 0017  =====
087 0020 0018  <==== INSTRUCAO DAD RP          CODIGO DO RPI 001
088 0021 0019  =====
089 0022 0020  <==== ESTADOS T3 E T4 DO CICLO(=0) DE FETCH
090 0023 0021  =====
091 0024 0022  <==== ESTADO T3 DO CICLO(=14) DE REGIÃO
092 0025 0023  =====
093 0026 0024  <==== ESTADO T3 DO CICLO(=14) DE REGIÃO
094 0027 0025  =====
095 0028 0026  <==== ESTADO T3 DO CICLO(=14) DE REGIÃO
096 0029 0027  =====
097 0030 0028  <==== ESTADO T3 DO CICLO(=14) DE REGIÃO
098 0031 0029  =====
099 0032 0030  <==== ESTADO T3 DO CICLO(=14) DE REGIÃO
100 0033 0031  =====
101 0034 0032  <==== INSTRUCAO STAX RP          CODIGO DO RPO 010
102 0035 0033  =====
103 0036 0034  <==== ESTADOS T3 E T4 DO CICLO(=0) DE FETCH
104 0037 0035  =====
105 0038 0036  <==== ESTADO T3 DO CICLO(=6) DE MEMBRAS
106 0039 0037  =====
107 0040 0038  <====
108 0041 0039  =====
109 0042 0040  <==== INSTRUCAO LDAX RP          CODIGO DO RPI 010
110 0043 0041  =====
111 0044 0042  <==== ESTADOS T3 E T4 DO CICLO(=0) DE FETCH
112 0045 0043  =====
113 0046 0044  <==== ESTADO T3 DO CICLO(=2) DE MEMBRAS
114 0047 0045  =====
115 0048 0046  <====
116 0049 0047  =====
117 0050 0048  <====
118 0051 0049  =====
119 0052 0050  <====
120 0053 0051  =====
121 0054 0052  <====
122 0055 0053  =====
123 0056 0054  <==== INSTRUCAO SALD          CODIGO IC 100 C10
124 0057 0055  =====
=====

```

```
070 0058 0057 #
071 0059 0058 # ESTADOS T3 E T4 DO CICLO(=0) DE FETCH
072 0060 0059 <===
073 0061 0060 <===
074 0062 0061 # ESTADO T3 DO CICLO(=1) DE MEMREAD
075 0063 0062 #
076 0064 0063 <===
077 0065 0064 <===
078 0066 0065 #
079 0067 0066 <===
080 0068 0067 <===
081 0069 0068 # ESTADO T3 DO CICLO(=5) DE MEMWRITE
082 0070 0069 <===
083 0071 0070 <===
084 0072 0071 <===
085 0073 0072 #
086 0074 0073 <===
087 0075 0074 #
088 0076 0075 #
089 0077 0076 <===
090 0078 0077 # INSTRUCAO LHL0 CODIGO DO L3 010
091 0079 0078 # ===
092 0080 0079 # ESTADOS T3 E T4 DO CICLO(=0) DE FETCH
093 0081 0080 <===
094 0082 0081 <===
095 0083 0082 #
096 0084 0083 # ESTADO T3 DO CICLO(=1) DE MEMREAD
097 0085 0084 <===
098 0086 0085 <===
099 0087 0086 #
100 0088 0087 # ESTADO T3 DO CICLO(=1) DE MEMREAD
101 0089 0088 <===
102 0090 0089 <===
103 0091 0090 #
104 0092 0091 # ESTADO T3 DO CICLO(=2) DE MEMREAD
105 0093 0092 <===
106 0094 0093 <===
107 0095 0094 #
108 0096 0095 # ESTADO T3 DO CICLO(=2) DE MEMREAD
109 0097 0096 <===
110 0098 0097 #
111 0099 0098 #
112 0100 0099 #
113 0101 0100 # INSTRUCAO STA CODIGO DO L3 010
114 0102 0101 # ===
115 0103 0102 # ESTADOS T3 E T4 DO CICLO(=0) DE FETCH
116 0104 0103 <===
117 0105 0104 <===
118 0106 0105 #
119 0107 0106 # ESTADO T3 DO CICLO(=1) DE MEMREAD
120 0108 0107 <===
121 0109 0108 <===
122 0110 0109 #
123 0111 0110 # ESTADO T3 DO CICLO(=1) DE MEMREAD
124 0112 0111 <===
125 0113 0112 <===
126 0114 0113 #
127 0115 0114 # ESTADO T3 DO CICLO(=5) DE MEMWRITE
128 0116 0115 <===
129 0117 0116 #
130 0118 0117 <===
```

```

071 0121 0119
070 0120 0120
073 0123 0121
070 0000 0122
071 0125 0123
001 0126 0124
019 0127 0125
070 0003 0126
071 0129 0127
001 0130 0128
017 0131 0129
076 0003 0130
071 0133 0131
001 0134 0132
021 0135 0133
070 0000 0134
000 0000 0135
000 0000 0136
000 0000 0137
071 0140 0138
070 0141 0139
040 0142 0140
070 0143 0141
041 0144 0142
070 0000 0143
000 0000 0144
000 0000 0145
000 0000 0146
071 0150 0147
072 0151 0149
073 0151 0149
076 0000 0150
071 0153 0151
001 0154 0152
031 0155 0153
036 0156 0154
035 0000 0155
070 0158 0156
071 0160 0157
032 0162 0160
002 0161 0159
070 0000 0160
000 0000 0161
000 0000 0162
073 0165 0163
070 0166 0164
039 0167 0165
036 0168 0166
035 0169 0167
030 0170 0168
030 0171 0169
070 0000 0170
000 0000 0171
000 0000 0172
071 0175 0173
070 0176 0174
073 0177 0175
070 0000 0176
071 0179 0177
001 0180 0178
031 0181 0179

```

```

=====
INSTRUCAO LDA          CODIGO DO DIF 010
ESTADOS T3 E T4 DO CICLO(=0) DE MEMREAD
=====
ESTADO T3 DO CICLO(=1) DE MEMREAD
=====
ESTADO T3 DO CICLO(=1) DE MEMREAD
=====
ESTADO T3 DO CICLO(=2) DE MEMREAD
=====
INSTRUCAO INX RP          CODIGO DO DIF 011
***** CORX RP          OC 01 011
=====
ESTADOS T3, T4 E T5 DO CICLO(=0) DE FETCH
=====
INSTRUCAO YRM          CODIGO DO DIF 100
=====
ESTADOS T3 E T4 DO CICLO(=0) DE FETCH
=====
ESTADO T3 DO CICLO(=3) DE MEMREAD
=====
ESTADO T3 DO CICLO(=4) DE MEMWRITE
=====
INSTRUCAO INRP          CODIGO DO DIF 100
=====
ESTADOS T3, T4 E T5 DO CICLO(=0) DE FETCH
=====
INSTRUCAO OGRM          CODIGO DO DIF 101
=====
ESTADOS T3 E T4 DO CICLO(=0) DE FETCH
=====
ESTADO T3 DO CICLO(=3) DE MEMREAD
=====

```

```

037 0182 0180
038 0183 0181
039 0184 0182
040 0185 0183
041 0186 0184
042 0187 0185
043 0188 0186
044 0189 0187
045 0190 0188
046 0191 0189
047 0192 0190
048 0193 0191
049 0194 0192
050 0195 0193
051 0196 0194
052 0197 0195
053 0198 0196
054 0199 0197
055 0200 0198
056 0201 0199
057 0202 0200
058 0203 0201
059 0204 0202
060 0205 0203
061 0206 0204
062 0207 0205
063 0208 0206
064 0209 0207
065 0210 0208
066 0211 0209
067 0212 0210
068 0213 0211
069 0214 0212
070 0215 0213
071 0216 0214
072 0217 0215
073 0218 0216
074 0219 0217
075 0220 0218
076 0221 0219
077 0222 0220
078 0223 0221
079 0224 0222
080 0225 0223
081 0226 0224
082 0227 0225
083 0228 0226
084 0229 0227
085 0230 0228
086 0231 0229
087 0232 0230
088 0233 0231
089 0234 0232
090 0235 0233
091 0236 0234
092 0237 0235
093 0238 0236
094 0239 0237
095 0240 0238
096 0241 0239
097 0242 0240
098 0243 0241
099 0244 0242

```

ESTADO T3 DO CICLO(=4) DE MEMORIA

INSTRUCAO DCRR CODIGO DO DDD 101

ESTADOS T3, T4 E T5 DO CICLO(=0) DE FETCH

INSTRUCAO MVIH CODIGO DO DDD 110

ESTADOS T3 E T4 DO CICLO(=0) DE FETCH

ESTADO T3 DO CICLO(=1) DE MEMORIA

ESTADO T3 DO CICLO(=4) DE MEMORIA

INSTRUCAO MVIH CODIGO DO DDD 110

ESTADOS T3 E T4 DO CICLO(=0) DE FETCH

ESTADO T3 DO CICLO(=1) DE MEMORIA

INSTRUCAO RLC CODIGO DO DDD 111

ESTADOS T3 E T4 DO CICLO(=0) DE FETCH

INSTRUCAO RFC CODIGO DO DDD 111

ESTADOS T3 E T4 DO CICLO(=0) DE FETCH

```

071 0243 0241 <=== INSTRUCAO RAL          CODIGO DE 100 111
070 0240 0242 == ESTADOS T3 E T4 DO CICLO(=0) DE FETCH
068 0245 0243 =
033 0284 0244 <===
070 0000 0245
000 0000 0246
000 0000 0247
000 0000 0248
071 0251 0249 <=== INSTRUCAO RAR          CODIGO DE 111 111
070 0250 0250 = ESTADOS T3 E T4 DO CICLO(=0) DE FETCH
069 0253 0251 =
033 0250 0252 <===
070 0000 0253
000 0000 0254
000 0000 0255
000 0000 0256
071 0250 0257 <=== INSTRUCAO DAA          CODIGO DE 100 111
070 0240 0254 = ESTADOS T3 E T4 DO CICLO(=0) DE FETCH
065 0261 0259 =
033 0242 0260 <===
070 0000 0261
000 0000 0262
000 0000 0263
000 0000 0264
071 0267 0265 <=== INSTRUCAO CHA          CODIGO DE 101 111
070 0266 0266 = ESTADOS T3 E T4 DO CICLO(=0) DE FETCH
052 0240 0267 <===
071 0200 0266
000 0000 0269
000 0000 0270
000 0000 0271
071 0274 0272 <=== INSTRUCAO STC          CODIGO DE 110 111
070 0275 0273 = ESTADOS T3 E T4 DO CICLO(=0) DE FETCH
053 0276 0274 <===
070 0000 0275
000 0000 0276
000 0000 0277
000 0000 0278
071 0281 0279 <=== INSTRUCAO CMC          CODIGO DE 111 111
070 0282 0280 = ESTADOS T3 E T4 DO CICLO(=0) DE FETCH
054 0283 0281 <===
070 0000 0282
000 0000 0283
000 0000 0284
000 0000 0285
000 0000 0286
000 0000 0287 <=== INSTRUCAO HALT          CODIGO DE 110 110
071 0290 0288 = ESTADOS T3 E T4 DO CICLO(=0) DE FETCH
070 0291 0289 = PROXIMO CICLO(=12) HALTACK ==> WAIT WAKE
073 0292 0290 <===
070 0000 0291
000 0000 0292
000 0000 0293
000 0000 0294
071 0297 0295 <=== INSTRUCAO MOVBY          CODIGO DE 100 110
070 0298 0296 = ESTADOS T3 E T4 DO CICLO(=0) DE FETCH
074 0299 0297 <===
070 0000 0298
071 0301 0299 <===
001 0302 0300 =
025 0303 0301 = ESTADO T3 DO CICLO(=3) DE MEMORIA

```

```
070 0000 0302 <===  
000 0000 0303  
000 0000 0304  
000 0000 0305  
071 0308 0306 = INSTRUCAO MOVBR          CODIGO 01 10 SSS  
= =====  
= ESTADOS T3 E T4 DO CICLO(=0) DE FETCH  
070 0309 0307 <===  
070 0000 0309  
024 0312 0310 <===  
002 0313 0311 = ESTADO T3 DO CICLO(=4) DE MEMWRITE  
070 0000 0312 <===  
000 0000 0313  
000 0000 0314  
000 0000 0315  
071 0318 0316 = INSTRUCAO MOVBR2        CODIGO 01 100 SSS  
= =====  
= ESTADOS T3, T4 E T5 DO CICLO(=0) DE FETCH  
070 0319 0317 <===  
051 0320 0318  
070 0321 0319 =  
050 0322 0320 <===  
070 0323 0321  
000 0000 0322  
000 0000 0323  
000 0000 0324  
071 0327 0325 = INSTRUCAO ADDR          CODIGO 10 000 SSS  
= =====  
= ESTADOS T3 E T4 DO CICLO(=0) DE FETCH  
070 0328 0326 <===  
051 0329 0327  
049 0330 0328 =  
070 0331 0329  
032 0332 0330  
035 0333 0331 =  
033 0000 0332 <===  
000 0000 0333  
000 0000 0334  
000 0000 0335  
071 0338 0336 = INSTRUCAO ADDR          CODIGO 10 000 SSS  
= =====  
= ESTADOS T3 E T4 DO CICLO(=0) DE FETCH  
070 0339 0337 <===  
051 0340 0338  
049 0341 0339 =  
070 0342 0340  
043 0343 0341 =  
035 0344 0342 <===  
033 0000 0343  
000 0000 0344  
000 0000 0345  
000 0000 0346  
071 0349 0347 = INSTRUCAO SUBR          CODIGO 10 000 SSS  
= =====  
= ESTADOS T3 E T4 DO CICLO(=0) DE FETCH  
070 0350 0348 <===  
051 0351 0349  
049 0352 0350 =  
070 0353 0351  
044 0354 0352 =  
035 0355 0353 <===  
033 0000 0354  
000 0000 0355  
000 0000 0356  
000 0000 0357  
071 0360 0358 = INSTRUCAO SUBR          CODIGO 10 000 SSS  
= =====  
= ESTADOS T3 E T4 DO CICLO(=0) DE FETCH  
070 0361 0359 <===  
051 0362 0360  
049 0363 0361 =  
070 0364 0362 =
```



0424					
0425					
0426					
0427					
0428					
0429					
0430					
0431					
0432					
0433					
0434					
0435					
0436					
0437					
0438					
0439					
0440					
0441					
0442					
0443					
0444					
0445					
0446					
0447					
0448					
0449					
0450					
0451					
0452					
0453					
0454					
0455					
0456					
0457					
0458					
0459					
0460					
0461					
0462					
0463					
0464					
0465					
0466					
0467					
0468					
0469					
0470					
0471					
0472					
0473					
0474					
0475					
0476					
0477					
0478					
0479					
0480					
0481					
0482					
0483					
0484					
0485					
0486					
0487					
0488					
0489					
0490					
0491					
0492					
0493					
0494					
0495					
0496					
0497					
0498					
0499					
0500					







```

010 0473 0671 <=== ESTADO T3 DO CICLO(=8) DE STACKFRAME 4
002 0574 0672 <===
070 0600 0673 <===
004 0576 0674 <===
002 0677 0675 <===
070 0678 0676 <===
056 0600 0677 <=== ESTADO T3 DO CICLO(=8) DE STACKFRAME 5
071 0680 0678 <=== INSTRUCAO CALL CODIGO 1 001 101
070 0681 0679 <=== ***** PUSH RP, PSW CODIGO 1 000 101
070 0682 0680 <=== ESTADOS T3, T4 E T5 DO CICLO(=0) DE FETCH
093 0683 0681 <===
070 0684 0682 <===
071 0685 0683 <=== ESTADO T3 DO CICLO(=1) DE MEMBRAS 3
001 0686 0684 <=== DESCRICAO DOS OUTROS ESTADOS VIO: CALL CONO
017 0687 0685 <===
070 0688 0686 <=== PUSH RP
027 0689 0687 <=== ESTADO T3 DO CICLO(=8) DE STACKFRAME 2
070 0690 0688 <===
070 0691 0689 <===
026 0692 0690 <=== PUSH RP
002 0693 0691 <=== ESTADO T3 DO CICLO(=8) DE STACKFRAME 3
073 0694 0692 <===
0694 <===
0695 <===
020 0692 0694 <=== PUSH PSW
002 0693 0697 <=== ESTADO T3 DO CICLO(=8) DE STACKFRAME 2
070 0698 0698 <===
022 0701 0699 <=== PUSH PSW
002 0702 0700 <=== ESTADO T3 DO CICLO(=8) DE STACKFRAME 3
070 0703 0701 <===
0704 <===
0705 <===
071 0708 0706 <=== OPERACOES LOGICAS F ARITMETICAS ENTRE A E <R?>
070 0709 0707 <=== ESTADOS T3 F T4 DO CICLO(=0) DE FETCH
049 0710 0709 <===
070 0700 0709 <===
0710 <===
071 0710 0710 <=== INSTRUCAO ADI CODIGO 1 000 110
001 0711 0713 <===
031 0716 0714 <===
070 0717 0715 <=== ESTADO T3 DO CICLO(=1) DE MEMBRAS
042 0718 0716 <===
035 0719 0717 <===
033 0720 0718 <===
0720 <===
0721 <=== INSTRUCAO ACI CODIGO 11 001 110
0722 <===
001 0725 0723 <===
031 0724 0724 <===
070 0727 0725 <=== ESTADO T3 DO CICLO(=1) DE MEMBRAS
043 0728 0726 <===
033 0729 0727 <===
033 0730 0728 <===
0729 <===
0730 <===
0731 <===

```

071	0730	0732	<===	INSTRUCAO SUI	===	CODIGO 11 010 110
001	0735	0733	=			
031	0736	0730	=	ESTADO T3 DO CICLO(=1) DE MEMORIA		
070	0737	0735	=			
034	0738	0736	=			
035	0739	0737	=			
033	0000	0738	<===			
		0739				
		0740				
		0741				
074	0740	0742	<===	INSTRUCAO SRI	===	CODIGO 11 011 110
001	0745	0743	=			
031	0746	0744	=	ESTADO T3 DO CICLO(=1) DE MEMORIA		
070	0747	0745	=			
045	0748	0746	=			
035	0749	0747	=			
033	0750	0748	<===			
		0749				
		0750				
		0751				
071	0750	0752	<===	INSTRUCAO ANI	===	CODIGO 11 100 110
001	0755	0753	=			
031	0756	0754	=	ESTADO T3 DO CICLO(=1) DE MEMORIA		
070	0757	0755	=			
046	0758	0756	=			
035	0759	0757	=			
033	0760	0759	<===			
		0760				
		0761				
071	0760	0762	<===	INSTRUCAO XRI	===	CODIGO 11 101 110
001	0765	0763	=			
031	0766	0764	=	ESTADO T3 DO CICLO(=1) DE MEMORIA		
070	0767	0765	=			
047	0768	0766	=			
035	0769	0767	=			
033	0000	0768	<===			
		0769				
		0770				
		0771				
071	0770	0772	<===	INSTRUCAO GPI	===	CODIGO 11 110 110
001	0775	0773	=			
031	0776	0774	=	ESTADO T3 DO CICLO(=1) DE MEMORIA		
070	0777	0775	=			
048	0778	0776	=			
035	0779	0777	=			
033	0000	0778	<===			
		0779				
		0780				
		0781				
071	0780	0782	<===	INSTRUCAO CPI	===	CODIGO 11 111 110
001	0785	0783	=			
031	0786	0784	=	ESTADO T3 DO CICLO(=1) DE MEMORIA		
070	0787	0785	=			
044	0788	0786	=			
035	0000	0787	<===			
		0788				
		0789				
		0790				
		0791				
071	0790	0792	<===	INSTRUCAO RST	===	CODIGO 11 111 111



APÊNDICE C

EXEMPLO DE SIMULAÇÃO

SIMBOLO -- SIMULADOR DO MICROPROCESSADOR INTEL 8080 NO B-6700 -- VERSAO 1  
SIMULACAO EXECUTADA EM 21/12/76 AS 14 HORAS E 15 MINUTOS

```

1 2
2 3
3 4
4 5
5 6
6 7
7 8
8 9
9 10
10 11
11 12
12 13
13 14
14 15
15 16
16 17
17 18
18 19
19 20
20 21
21 22
22 23
23 24
24 25
25 26
26 27
27 28
28 29
29 30
30 31
31 32
32 33
33 34
34 35
35 36
36 37
37 38
38 39
39 40
40 41
41 42
42 43
43 44
44 45
45 46
46 47
47 48
48 49
49 50

```

```

* SIMULACAO PARA TESTAR O SIMBOLO
*
* O PROGRAMA TESTE FAZ A ADICAO DECIMAL DOS CONTEUDOS DAS POSICOES DE
* MEMORIA M(30) A M(37) COM AS OF M(38) A M(45)
* O RESULTADO EH COLOCADO NAS POSICOES DE MEMORIA M(30) A M(37)
* MEMORY(01,100);
* INTERRUPTOES A SEREM SIMULADAS
* INTER(6) INST(5);
* HOLD(10) INST(15);
*
* WRITE INST(1,1) M(30),M(31),M(32),M(33),M(34),M(35),M(36),M(37),M(38),
* M(39);
*
* LOAD M(1)= 021, * LXI D,ALPHA 1
* 036, * 2
* 000, * 3
* 041, * LXI H,BETA 4
* 046, * 5
* 000, * MVI C,8 6
* 016, * 7
* 010, * XRA 8
* 257, * LOOP/DX 9
* 032, * 10
* 216, * 11
* 047, * 12
* 022, * 13
* 043, * 14
* 023, * 15
* 015, * 16
* 302, * 17
* 012, *
* 000, *
*
* M(10)=001,002,003,004,005,006,007,000, * ALPHA
* 001,002,003,004,005,006,007,000, * BETA
LOAD PC=000001, SP=000140,
LOAD INIE=1,
LOAD M(AR)= 373, * EI
311, * RE
LOAD M(56)= 373, * EI
311, * RT
*
* RESET INST(60);

```

\*\*\*\*\* ESTADO INICIAL DO SISTEMA \*\*\*\*\*



M(00030)	M(00031)	M(00032)	M(00033)	M(00034)	M(00035)	M(00036)	M(00037)	M(00038)	M(00039)								
0000001	0000010	0000011	0000100	0000101	0000110	0000111	0000000	0000001	0000010								
FLCK	CYCLE	INST	IR	PC	SP	M(SP)	A	B	C	D	E	H	L	FLAGS	ARUS	DBUS	PINDS
0000060	000017	00008	373	000061	000136	014	002	000	010	000	036	000	044	00000	000060	373	1010000000
M(00030)	M(00031)	M(00032)	M(00033)	M(00034)	M(00035)	M(00036)	M(00037)	M(00038)	M(00039)								
0000001	0000010	0000011	0000100	0000101	0000110	0000111	0000000	0000001	0000010								
FLCK	CYCLE	INST	IR	PC	SP	M(SP)	A	B	C	D	E	H	L	FLAGS	ARUS	DBUS	PINDS
0000070	000020	00009	311	000014	000140	000	002	000	010	000	036	000	046	00000	000137	000	1010000000
M(00030)	M(00031)	M(00032)	M(00033)	M(00034)	M(00035)	M(00036)	M(00037)	M(00038)	M(00039)								
0000001	0000010	0000011	0000100	0000101	0000110	0000111	0000000	0000001	0000010								
FLCK	CYCLE	INST	IR	PC	SP	M(SP)	A	B	C	D	E	H	L	FLAGS	ARUS	DBUS	PINDS
0000074	000021	00010	047	000015	000140	000	002	000	010	000	036	000	046	00000	000037	000	1010000000
M(00030)	M(00031)	M(00032)	M(00033)	M(00034)	M(00035)	M(00036)	M(00037)	M(00038)	M(00039)								
0000001	0000010	0000011	0000100	0000101	0000110	0000111	0000000	0000001	0000010								
FLCK	CYCLE	INST	IR	PC	SP	M(SP)	A	B	C	D	E	H	L	FLAGS	ARUS	DBUS	PINDS
0000081	000023	00011	022	000016	000140	000	002	000	010	000	036	000	047	00000	000036	002	1010000000
M(00030)	M(00031)	M(00032)	M(00033)	M(00034)	M(00035)	M(00036)	M(00037)	M(00038)	M(00039)								
0000010	0000010	0000011	0000100	0000101	0000110	0000111	0000000	0000001	0000010								
FLCK	CYCLE	INST	IR	PC	SP	M(SP)	A	B	C	D	E	H	L	FLAGS	ARUS	DBUS	PINDS
0000086	000024	00012	043	000017	000140	000	002	000	010	000	036	000	047	00000	000037	000	1010000000
M(00030)	M(00031)	M(00032)	M(00033)	M(00034)	M(00035)	M(00036)	M(00037)	M(00038)	M(00039)								
0000010	0000010	0000011	0000100	0000101	0000110	0000111	0000000	0000001	0000010								
FLCK	CYCLE	INST	IR	PC	SP	M(SP)	A	B	C	D	E	H	L	FLAGS	ARUS	DBUS	PINDS
0000091	000025	00013	073	000020	000140	000	002	000	010	000	037	000	047	00000	000037	000	1010000000
M(00030)	M(00031)	M(00032)	M(00033)	M(00034)	M(00035)	M(00036)	M(00037)	M(00038)	M(00039)								
0000010	0000010	0000011	0000100	0000101	0000110	0000111	0000000	0000001	0000010								
FLCK	CYCLE	INST	IR	PC	SP	M(SP)	A	B	C	D	E	H	L	FLAGS	ARUS	DBUS	PINDS
0000096	000025	00014	015	000021	000140	000	002	000	010	000	037	000	047	00000	000037	000	1010000000
M(00030)	M(00031)	M(00032)	M(00033)	M(00034)	M(00035)	M(00036)	M(00037)	M(00038)	M(00039)								
0000010	0000010	0000011	0000100	0000101	0000110	0000111	0000000	0000001	0000010								
FLCK	CYCLE	INST	IR	PC	SP	M(SP)	A	B	C	D	E	H	L	FLAGS	ARUS	DBUS	PINDS
0000107	000029	00015	302	000012	000140	000	002	000	010	000	037	000	047	00000	000037	000	1010000000
M(00030)	M(00031)	M(00032)	M(00033)	M(00034)	M(00035)	M(00036)	M(00037)	M(00038)	M(00039)								
0000010	0000010	0000011	0000100	0000101	0000110	0000111	0000000	0000001	0000010								

\*\*\*\*\* HOLD - INSTRUCTION=15



M(00030)	M(00031)	M(00032)	M(00033)	M(00034)	M(00035)	M(00036)	M(00037)	M(00038)	M(00039)								
0000010	0000100	0000011	0000100	0000101	0000110	0000111	0000000	0000001	0000010								
CLUCK	CYCLE	INST	IR	PC	SP	M(SP)	A	B	C	D	E	H	L	FLAGS	ARUS	UBUS	PINDS
0000176	0000044	00024	032	000013	000140	000	003	000	000	000	040	000	050	00010	000040	003	1010000000
M(00030)	M(00031)	M(00032)	M(00033)	M(00034)	M(00035)	M(00036)	M(00037)	M(00038)	M(00039)								
0000010	0000100	0000011	0000100	0000101	0000110	0000111	0000000	0000001	0000010								
CLUCK	CYCLE	INST	IR	PC	SP	M(SP)	A	B	C	D	E	H	L	FLAGS	ARUS	UBUS	PINDS
0000133	0000046	00025	216	000014	000140	000	006	000	000	000	040	000	050	00010	000050	003	1010000000
M(00030)	M(00031)	M(00032)	M(00033)	M(00034)	M(00035)	M(00036)	M(00037)	M(00038)	M(00039)								
0000010	0000100	0000011	0000100	0000101	0000110	0000111	0000000	0000001	0000010								
CLUCK	CYCLE	INST	IR	PC	SP	M(SP)	A	B	C	D	E	H	L	FLAGS	ARUS	UBUS	PINDS
0000187	0000047	00026	047	000015	000140	000	006	000	000	000	040	000	050	00010	000014	047	1010000000
M(00030)	M(00031)	M(00032)	M(00033)	M(00034)	M(00035)	M(00036)	M(00037)	M(00038)	M(00039)								
0000010	0000100	0000011	0000100	0000101	0000110	0000111	0000000	0000001	0000010								
CLUCK	CYCLE	INST	IR	PC	SP	M(SP)	A	B	C	D	E	H	L	FLAGS	ARUS	UBUS	PINDS
0000194	0000049	00027	022	000016	000140	000	006	000	000	000	040	000	050	00010	000040	006	1010000000
M(00030)	M(00031)	M(00032)	M(00033)	M(00034)	M(00035)	M(00036)	M(00037)	M(00038)	M(00039)								
0000010	0000100	0000011	0000100	0000101	0000110	0000111	0000000	0000001	0000010								
CLUCK	CYCLE	INST	IR	PC	SP	M(SP)	A	B	C	D	E	H	L	FLAGS	ARUS	UBUS	PINDS
0000199	0000050	00028	043	000017	000140	000	006	000	000	000	040	000	050	00010	000016	043	1010000000
M(00030)	M(00031)	M(00032)	M(00033)	M(00034)	M(00035)	M(00036)	M(00037)	M(00038)	M(00039)								
0000010	0000100	0000011	0000100	0000101	0000110	0000111	0000000	0000001	0000010								
CLUCK	CYCLE	INST	IR	PC	SP	M(SP)	A	B	C	D	E	H	L	FLAGS	ARUS	UBUS	PINDS
0000204	0000051	00029	023	000020	000140	000	006	000	000	000	040	000	050	00010	000017	023	1010000000
M(00030)	M(00031)	M(00032)	M(00033)	M(00034)	M(00035)	M(00036)	M(00037)	M(00038)	M(00039)								
0000010	0000100	0000011	0000100	0000101	0000110	0000111	0000000	0000001	0000010								
CLUCK	CYCLE	INST	IR	PC	SP	M(SP)	A	B	C	D	E	H	L	FLAGS	ARUS	UBUS	PINDS
0000209	0000052	00030	015	000021	000140	000	006	000	000	000	040	000	050	00010	000020	015	1010000000
M(00030)	M(00031)	M(00032)	M(00033)	M(00034)	M(00035)	M(00036)	M(00037)	M(00038)	M(00039)								
0000010	0000100	0000011	0000100	0000101	0000110	0000111	0000000	0000001	0000010								
CLUCK	CYCLE	INST	IR	PC	SP	M(SP)	A	B	C	D	E	H	L	FLAGS	ARUS	UBUS	PINDS
0000220	0000055	00031	302	000022	000140	000	006	000	000	000	040	000	050	00010	000023	000	1010000000
M(00030)	M(00031)	M(00032)	M(00033)	M(00034)	M(00035)	M(00036)	M(00037)	M(00038)	M(00039)								
0000010	0000100	0000011	0000100	0000101	0000110	0000111	0000000	0000001	0000010								
CLUCK	CYCLE	INST	IR	PC	SP	M(SP)	A	B	C	D	E	H	L	FLAGS	ARUS	UBUS	PINDS
0000229	0000056	00032	303	000023	000140	000	006	000	000	000	040	000	050	00010	000024	000	1010000000

```

0000227 000057 000032 032 000013 000140 000 004 003 305 000 041 000 051 00010 000041 004 1010000000
M(000030) M(000031) M(000032) M(000033) M(000034) M(000035) M(000036) M(000037) M(000038) M(000039)
00000010 00000100 00000110 00000100 00000101 00000111 00000111 00000000 00000001 00000010

FLICK CYCLE INST IR PC SP M(SP) A B C D E H L FLAGS ARUS UBUS PINDS
0000234 000059 000033 216 000014 000140 000 010 000 005 000 041 000 051 00000 000051 004 1010000000
M(000030) M(000031) M(000032) M(000033) M(000034) M(000035) M(000036) M(000037) M(000038) M(000039)
00000010 00000100 00000110 00000100 00000101 00000110 00000111 00000000 00000001 00000010

FLICK CYCLE INST IR PC SP M(SP) A B C D E H L FLAGS ARUS UBUS PINDS
0000234 000060 000034 047 000015 000140 000 010 000 005 000 041 000 051 00000 000014 047 1010000000
M(000030) M(000031) M(000032) M(000033) M(000034) M(000035) M(000036) M(000037) M(000038) M(000039)
00000010 00000100 00000110 00000100 00000101 00000110 00000111 00000000 00000001 00000010

FLICK CYCLE INST IR PC SP M(SP) A B C D E H L FLAGS ARUS UBUS PINDS
0000243 000062 000035 022 000016 000140 000 010 000 005 000 041 000 051 00000 000041 010 1010000000
M(000030) M(000031) M(000032) M(000033) M(000034) M(000035) M(000036) M(000037) M(000038) M(000039)
00000010 00000100 00000110 00000100 00000101 00000110 00000111 00000000 00000001 00000010

FLICK CYCLE INST IR PC SP M(SP) A B C D E H L FLAGS ARUS UBUS PINDS
0000250 000063 000036 043 000017 000140 000 010 000 005 000 041 000 052 00000 000016 043 1010000000
M(000030) M(000031) M(000032) M(000033) M(000034) M(000035) M(000036) M(000037) M(000038) M(000039)
00000010 00000100 00000110 00000100 00000101 00000110 00000111 00000000 00000001 00000010

FLICK CYCLE INST IR PC SP M(SP) A B C D E H L FLAGS ARUS UBUS PINDS
0000255 000064 000037 023 000020 000140 000 010 000 005 000 042 000 052 00000 000017 023 1010000000
M(000030) M(000031) M(000032) M(000033) M(000034) M(000035) M(000036) M(000037) M(000038) M(000039)
00000010 00000100 00000110 00000100 00000101 00000110 00000111 00000000 00000001 00000010

FLICK CYCLE INST IR PC SP M(SP) A B C D E H L FLAGS ARUS UBUS PINDS
0000260 000065 000038 015 000021 000140 000 010 000 004 000 042 000 052 00000 000020 015 1010000000
M(000030) M(000031) M(000032) M(000033) M(000034) M(000035) M(000036) M(000037) M(000038) M(000039)
00000010 00000100 00000110 00000100 00000101 00000110 00000111 00000000 00000001 00000010

FLICK CYCLE INST IR PC SP M(SP) A B C D E H L FLAGS ARUS UBUS PINDS
0000271 000068 000039 302 000012 000140 000 010 000 004 000 042 000 052 00000 000023 000 1010000000
M(000030) M(000031) M(000032) M(000033) M(000034) M(000035) M(000036) M(000037) M(000038) M(000039)
00000010 00000100 00000110 00000100 00000101 00000110 00000111 00000000 00000001 00000010

FLICK CYCLE INST IR PC SP M(SP) A B C D E H L FLAGS ARUS UBUS PINDS
0000278 000070 000040 032 000013 000140 000 005 000 004 000 042 000 052 00000 000042 005 1010000000
M(000030) M(000031) M(000032) M(000033) M(000034) M(000035) M(000036) M(000037) M(000038) M(000039)
00000010 00000100 00000110 00000100 00000101 00000111 00000111 00000000 00000001 00000010

```

CLUCK CYCLE INST IR PC SP M(SP) A B C D E H L FLAGS ARUS UBUS PINDS  
 0000285 000072 00041 216 000014 000140 000 012 000 004 000 042 000 052 00010 000052 005 1010000000

M(00030) M(00031) M(00032) M(00033) M(00034) M(00035) M(00036) M(00037) M(00038) M(00039)  
 0000010 0000100 0000110 0000100 0000101 0000110 0000111 0000000 0000001 0000010

CLUCK CYCLE INST IR PC SP M(SP) A B C D E H L FLAGS ARUS UBUS PINDS  
 0000289 000073 00042 047 000015 000140 000 020 000 004 000 042 000 052 00010 000014 047 1010000000

M(00030) M(00031) M(00032) M(00033) M(00034) M(00035) M(00036) M(00037) M(00038) M(00039)  
 0000010 0000100 0000110 0000100 0000101 0000110 0000111 0000000 0000001 0000010

CLUCK CYCLE INST IR PC SP M(SP) A B C D E H L FLAGS ARUS UBUS PINDS  
 0000296 000075 00043 022 000016 000140 000 020 000 004 000 042 000 052 00010 000042 022 1010000000

M(00030) M(00031) M(00032) M(00033) M(00034) M(00035) M(00036) M(00037) M(00038) M(00039)  
 0000010 0000100 0000110 0000100 0001000 0000110 0000111 0000000 0000001 0000010

CLUCK CYCLE INST IR PC SP M(SP) A B C D E H L FLAGS ARUS UBUS PINDS  
 0000301 000076 00044 043 000017 000140 000 020 000 004 000 042 000 053 00010 000016 043 1010000000

M(00030) M(00031) M(00032) M(00033) M(00034) M(00035) M(00036) M(00037) M(00038) M(00039)  
 0000010 0000100 0000110 0000100 0001000 0000110 0000111 0000000 0000001 0000010

CLUCK CYCLE INST IR PC SP M(SP) A B C D E H L FLAGS ARUS UBUS PINDS  
 0000306 000077 00045 023 000020 000140 000 020 000 004 000 043 000 053 00010 000017 023 1010000000

M(00030) M(00031) M(00032) M(00033) M(00034) M(00035) M(00036) M(00037) M(00038) M(00039)  
 0000010 0000100 0000110 0000100 0001000 0000110 0000111 0000000 0000001 0000010

CLUCK CYCLE INST IR PC SP M(SP) A B C D E H L FLAGS ARUS UBUS PINDS  
 0000311 000078 00046 015 000021 000140 000 020 000 003 000 043 000 053 00010 000020 015 1010000000

M(00030) M(00031) M(00032) M(00033) M(00034) M(00035) M(00036) M(00037) M(00038) M(00039)  
 0000010 0000100 0000110 0000100 0001000 0000110 0000111 0000000 0000001 0000010

CLUCK CYCLE INST IR PC SP M(SP) A B C D E H L FLAGS ARUS UBUS PINDS  
 0000322 000081 00047 302 000012 000140 000 020 000 003 000 043 000 053 00010 000023 000 1010000000

M(00030) M(00031) M(00032) M(00033) M(00034) M(00035) M(00036) M(00037) M(00038) M(00039)  
 0000010 0000100 0000110 0000100 0001000 0000110 0000111 0000000 0000001 0000010

CLUCK CYCLE INST IR PC SP M(SP) A B C D E H L FLAGS ARUS UBUS PINDS  
 0000329 000083 00048 032 000013 000140 000 006 000 003 000 043 000 053 00010 000043 005 1010000000

M(00030) M(00031) M(00032) M(00033) M(00034) M(00035) M(00036) M(00037) M(00038) M(00039)  
 0000010 0000100 0000110 0000100 0001000 0000110 0000111 0000000 0000001 0000010

CLUCK CYCLE INST IR PC SP M(SP) A B C D E H L FLAGS ARUS UBUS PINDS  
 0000336 000085 00049 216 000014 000140 000 014 000 003 000 043 000 053 00010 000053 006 1010000000

M(00030) M(00031) M(00032) M(00033) M(00034) M(00035) M(00036) M(00037) M(00038) M(00039)  
 0000010 0000100 0000110 0000100 0001000 0000110 0000111 0000000 0000001 0000010

```

00000010 00000100 00000110 00010000 00000110 00000111 00000000 00000001 00000010
-----
FLUCK  CYCLE  INST  IR  PC  SP  M(SP)  A  B  C  D  E  H  L  FLAGS  ARBUS  UBUS  PINDS
0000340 0000056 000050 047 000015 000140 000 022 000 001 000 043 000 053 00010 000014 047 1010000000
M(00030) M(00031) M(00032) M(00033) M(00034) M(00035) M(00036) M(00037) M(00038) M(00039)
00000010 00000100 00000110 00000110 00010000 00000110 00000111 00000000 00000001 00000010
-----
FLUCK  CYCLE  INST  IR  PC  SP  M(SP)  A  B  C  D  E  H  L  FLAGS  ARBUS  UBUS  PINDS
0000347 0000088 000051 022 000016 000140 000 022 000 003 000 043 000 053 00010 000043 022 1010000000
M(00030) M(00031) M(00032) M(00033) M(00034) M(00035) M(00036) M(00037) M(00038) M(00039)
00000010 00000100 00000110 00000110 00010000 00010010 00000111 00000000 00000001 00000010
-----
FLUCK  CYCLE  INST  IR  PC  SP  M(SP)  A  B  C  D  E  H  L  FLAGS  ARBUS  UBUS  PINDS
0000352 0000089 000052 043 000017 000140 000 022 000 003 000 043 000 054 00010 000016 043 1010000000
M(00030) M(00031) M(00032) M(00033) M(00034) M(00035) M(00036) M(00037) M(00038) M(00039)
00000010 00000100 00000110 00000110 00010000 00010010 00000111 00000000 00000001 00000010
-----
FLUCK  CYCLE  INST  IR  PC  SP  M(SP)  A  B  C  D  E  H  L  FLAGS  ARBUS  UBUS  PINDS
0000357 0000090 000053 023 000020 000140 000 022 000 003 000 044 000 054 00010 000017 023 1010000000
M(00030) M(00031) M(00032) M(00033) M(00034) M(00035) M(00036) M(00037) M(00038) M(00039)
00000010 00000100 00000110 00000110 00010000 00010010 00000111 00000000 00000001 00000010
-----
FLUCK  CYCLE  INST  IR  PC  SP  M(SP)  A  B  C  D  E  H  L  FLAGS  ARBUS  UBUS  PINDS
0000362 0000091 000054 015 000021 000140 000 022 000 002 000 044 000 054 00000 000020 015 1010000000
M(00030) M(00031) M(00032) M(00033) M(00034) M(00035) M(00036) M(00037) M(00038) M(00039)
00000010 00000100 00000110 00000110 00010000 00010010 00000111 00000000 00000001 00000010
-----
FLUCK  CYCLE  INST  IR  PC  SP  M(SP)  A  B  C  D  E  H  L  FLAGS  ARBUS  UBUS  PINDS
0000373 0000094 000055 302 000012 000140 000 022 000 002 000 043 000 054 00000 000023 000 1010000000
M(00030) M(00031) M(00032) M(00033) M(00034) M(00035) M(00036) M(00037) M(00038) M(00039)
00000010 00000100 00000110 00000110 00010000 00010010 00000111 00000000 00000001 00000010
-----
FLUCK  CYCLE  INST  IR  PC  SP  M(SP)  A  B  C  D  E  H  L  FLAGS  ARBUS  UBUS  PINDS
0000380 0000096 000056 032 000013 000140 000 007 000 002 000 044 000 054 00000 000044 007 1010000000
M(00030) M(00031) M(00032) M(00033) M(00034) M(00035) M(00036) M(00037) M(00038) M(00039)
00000010 00000100 00000110 00000110 00010000 00010010 00000111 00000000 00000001 00000010
-----
FLUCK  CYCLE  INST  IR  PC  SP  M(SP)  A  B  C  D  E  H  L  FLAGS  ARBUS  UBUS  PINDS
0000387 0000098 000057 216 000014 000140 000 016 000 002 000 044 000 054 00000 000054 007 1010000000
M(00030) M(00031) M(00032) M(00033) M(00034) M(00035) M(00036) M(00037) M(00038) M(00039)
00000010 00000100 00000110 00000110 00010000 00010010 00000111 00000000 00000001 00000010
-----
FLUCK  CYCLE  INST  IR  PC  SP  M(SP)  A  B  C  D  E  H  L  FLAGS  ARBUS  UBUS  PINDS
0000391 0000099 000058 047 000015 000140 000 024 000 002 000 044 000 054 00000 000014 047 1010000000

```

M(00030)	M(00031)	M(00032)	M(00033)	M(00034)	M(00035)	M(00036)	M(00037)	M(00038)	M(00039)										
0000010	0000100	0000110	0001000	0001000	0001010	0000011	0000000	0000001	0000010										
CLCK	CYCLE	INST	IR	PC	SP	M(SP)	A	B	C	D	E	F	G	H	L	FLAGS	ABUS	UBUS	PINDS
0000396	000101	000059	022	000016	000140	000	024	000	002	000	044	000	054	000	0000044	024	1010000000		
M(00030)	M(00031)	M(00032)	M(00033)	M(00034)	M(00035)	M(00036)	M(00037)	M(00038)	M(00039)										
0000010	0000100	0000110	0001000	0001000	0001010	0000011	0000000	0000001	0000010										
CLCK	CYCLE	INST	IR	PC	SP	M(SP)	A	B	C	D	E	F	G	H	L	FLAGS	ABUS	UBUS	PINDS
0000400	000102	000060	043	000017	000140	000	024	000	002	000	044	000	055	00000	0000016	043	1010000000		
M(00030)	M(00031)	M(00032)	M(00033)	M(00034)	M(00035)	M(00036)	M(00037)	M(00038)	M(00039)										
0000010	0000100	0000110	0001000	0001000	0001010	0000011	0000000	0000001	0000010										
CLCK	CYCLE	INST	IR	PC	SP	M(SP)	A	B	C	D	E	F	G	H	L	FLAGS	ABUS	UBUS	PINDS
0000408	000103	000061	023	000020	000140	000	024	000	002	000	045	000	055	00000	0000017	023	1010000000		
M(00030)	M(00031)	M(00032)	M(00033)	M(00034)	M(00035)	M(00036)	M(00037)	M(00038)	M(00039)										
0000010	0000100	0000110	0001000	0001000	0001010	0000011	0000000	0000001	0000010										
CLCK	CYCLE	INST	IR	PC	SP	M(SP)	A	B	C	D	E	F	G	H	L	FLAGS	ABUS	UBUS	PINDS
0000413	000104	000062	015	000021	000140	000	024	000	001	000	045	000	055	00000	0000020	015	1010000000		
M(00030)	M(00031)	M(00032)	M(00033)	M(00034)	M(00035)	M(00036)	M(00037)	M(00038)	M(00039)										
0000010	0000100	0000110	0001000	0001000	0001010	0000011	0000000	0000001	0000010										
CLCK	CYCLE	INST	IR	PC	SP	M(SP)	A	B	C	D	E	F	G	H	L	FLAGS	ABUS	UBUS	PINDS
0000424	000107	000063	302	000012	000140	000	024	000	001	000	045	000	055	00000	0000023	000	1010000000		
M(00030)	M(00031)	M(00032)	M(00033)	M(00034)	M(00035)	M(00036)	M(00037)	M(00038)	M(00039)										
0000010	0000100	0000110	0001000	0001000	0001010	0000011	0000000	0000001	0000010										
CLCK	CYCLE	INST	IR	PC	SP	M(SP)	A	B	C	D	E	F	G	H	L	FLAGS	ABUS	UBUS	PINDS
0000431	000109	000064	032	000013	000140	000	024	000	001	000	045	000	055	00000	0000045	000	1010000000		
M(00030)	M(00031)	M(00032)	M(00033)	M(00034)	M(00035)	M(00036)	M(00037)	M(00038)	M(00039)										
0000010	0000100	0000110	0001000	0001000	0001010	0000011	0000000	0000001	0000010										
CLCK	CYCLE	INST	IR	PC	SP	M(SP)	A	B	C	D	E	F	G	H	L	FLAGS	ABUS	UBUS	PINDS
0000439	000111	000065	216	000014	000140	000	024	000	001	000	045	000	055	01010	0000055	000	1010000000		
M(00030)	M(00031)	M(00032)	M(00033)	M(00034)	M(00035)	M(00036)	M(00037)	M(00038)	M(00039)										
0000010	0000100	0000110	0001000	0001000	0001010	0000011	0000000	0000001	0000010										
CLCK	CYCLE	INST	IR	PC	SP	M(SP)	A	B	C	D	E	F	G	H	L	FLAGS	ABUS	UBUS	PINDS
0000442	000112	000066	047	000015	000140	000	024	000	001	000	045	000	055	01010	0000014	047	1010000000		
M(00030)	M(00031)	M(00032)	M(00033)	M(00034)	M(00035)	M(00036)	M(00037)	M(00038)	M(00039)										
0000010	0000100	0000110	0001000	0001000	0001010	0000011	0000000	0000001	0000010										
CLCK	CYCLE	INST	IR	PC	SP	M(SP)	A	B	C	D	E	F	G	H	L	FLAGS	ABUS	UBUS	PINDS
0000442	000112	000066	047	000015	000140	000	024	000	001	000	045	000	055	01010	0000014	047	1010000000		

```

FLCK  CYCLE  INST  IR  PC  SP  M(SP)  A  B  C  D  E  H  L  FLAGS  ARUS  DBUS  PINDS
0000449 000114 00067 022 000016 000140 000 000 001 000 0+5 000 055 01010 000045 000 1010000000
M(00030) M(00031) M(00032) M(00033) M(00034) M(00035) M(00036) M(00037) M(00038) M(00039)
0000010 0000100 0000110 0000100 0001000 0001010 0001010 0000000 0000000 0000010

```

```

FLCK  CYCLE  INST  IR  PC  SP  M(SP)  A  B  C  D  E  H  L  FLAGS  ARUS  DBUS  PINDS
0000454 000115 00068 043 000017 000140 000 000 001 000 0+5 000 056 01010 000016 003 1010000000
M(00030) M(00031) M(00032) M(00033) M(00034) M(00035) M(00036) M(00037) M(00038) M(00039)
0000010 0000100 0000110 0000100 0001000 0001010 0001010 0000000 0000000 0000010

```

```

FLCK  CYCLE  INST  IR  PC  SP  M(SP)  A  B  C  D  E  H  L  FLAGS  ARUS  DBUS  PINDS
0000459 000116 00069 023 000020 000140 000 000 001 000 0+6 000 056 01010 000017 023 1010000000
M(00030) M(00031) M(00032) M(00033) M(00034) M(00035) M(00036) M(00037) M(00038) M(00039)
0000010 0000100 0000110 0000100 0001000 0001010 0001010 0000000 0000000 0000010

```

```

FLCK  CYCLE  INST  IR  PC  SP  M(SP)  A  B  C  D  E  H  L  FLAGS  ARUS  DBUS  PINDS
0000464 000117 00070 015 000021 000140 000 000 000 000 0+6 000 056 01010 000020 015 1010000000
M(00030) M(00031) M(00032) M(00033) M(00034) M(00035) M(00036) M(00037) M(00038) M(00039)
0000010 0000100 0000110 0000100 0001000 0001010 0001010 0000000 0000000 0000010

```

```

FLCK  CYCLE  INST  IR  PC  SP  M(SP)  A  B  C  D  E  H  L  FLAGS  ARUS  DBUS  PINDS
0000469 000118 00071 302 000024 000140 000 000 000 000 0+6 000 056 01010 000021 302 1010000000
M(00030) M(00031) M(00032) M(00033) M(00034) M(00035) M(00036) M(00037) M(00038) M(00039)
0000010 0000100 0000110 0000100 0001000 0001010 0001010 0000000 0000000 0000010

```

```

FLCK  CYCLE  INST  IR  PC  SP  M(SP)  A  B  C  D  E  H  L  FLAGS  ARUS  DBUS  PINDS
0000473 000119 00072 000 000025 000140 000 000 000 000 0+6 000 056 01010 000024 000 1010000000
M(00030) M(00031) M(00032) M(00033) M(00034) M(00035) M(00036) M(00037) M(00038) M(00039)
0000010 0000100 0000110 0000100 0001000 0001010 0001010 0000000 0000000 0000010

```

```

FLCK  CYCLE  INST  IR  PC  SP  M(SP)  A  B  C  D  E  H  L  FLAGS  ARUS  DBUS  PINDS
0000477 000120 00073 000 000026 000140 000 000 000 000 0+6 000 056 01010 000025 000 1010000000
M(00030) M(00031) M(00032) M(00033) M(00034) M(00035) M(00036) M(00037) M(00038) M(00039)
0000010 0000100 0000110 0000100 0001000 0001010 0001010 0000000 0000000 0000010

```

```

FLCK  CYCLE  INST  IR  PC  SP  M(SP)  A  B  C  D  E  H  L  FLAGS  ARUS  DBUS  PINDS
0000481 000121 00074 000 000027 000140 000 000 000 000 0+6 000 056 01010 000026 000 1010000000
M(00030) M(00031) M(00032) M(00033) M(00034) M(00035) M(00036) M(00037) M(00038) M(00039)
0000010 0000100 0000110 0000100 0001000 0001010 0001010 0000000 0000000 0000010

```

```

FLCK  CYCLE  INST  IR  PC  SP  M(SP)  A  B  C  D  E  H  L  FLAGS  ARUS  DBUS  PINDS
0000485 000122 00075 000 000030 000140 000 000 000 000 0+6 000 056 01010 000027 000 1010000000
M(00030) M(00031) M(00032) M(00033) M(00034) M(00035) M(00036) M(00037) M(00038) M(00039)
0000010 0000100 0000110 0000100 0001000 0001010 0001010 0000000 0000000 0000010

```

```

-----
CLCK  CYCLE  INST  IR  PC  SP  M(SP)  A  B  C  D  E  H  L  FLAGS  ARUS  DBUS  PINDS
0000499 000123 00076 000 000031 000140 000 000 000 000 000 016 000 056 01010 000030 000 1010000000
M(000030) M(000031) M(000032) M(000033) M(000034) M(000035) M(000036) M(000037) M(000038) M(000039)
00000010 00000100 00000110 00000100 00010000 00010010 00010100 00000000 00000001 00000010

CLCK  CYCLE  INST  IR  PC  SP  M(SP)  A  B  C  D  E  H  L  FLAGS  ARUS  DBUS  PINDS
0000493 000124 00077 000 000032 000140 000 000 000 000 000 016 000 056 01010 000031 000 1010000000
M(000030) M(000031) M(000032) M(000033) M(000034) M(000035) M(000036) M(000037) M(000038) M(000039)
00000010 00000100 00000110 00000100 00010000 00010010 00010100 00000000 00000001 00000010
-----

```

```

-----
CLCK  CYCLE  INST  IR  PC  SP  M(SP)  A  B  C  D  E  H  L  FLAGS  ARUS  DBUS  PINDS
0000497 000125 00078 000 000033 000140 000 000 000 000 000 016 000 056 01010 000032 000 1010000000
M(000030) M(000031) M(000032) M(000033) M(000034) M(000035) M(000036) M(000037) M(000038) M(000039)
00000010 00000100 00000110 00000100 00010000 00010010 00010100 00000000 00000001 00000010
-----

```

```

-----
CLCK  CYCLE  INST  IR  PC  SP  M(SP)  A  B  C  D  E  H  L  FLAGS  ARUS  DBUS  PINDS
0000501 000126 00079 000 000034 000140 000 000 000 000 000 016 000 056 01010 000033 000 1010000000
M(000030) M(000031) M(000032) M(000033) M(000034) M(000035) M(000036) M(000037) M(000038) M(000039)
00000010 00000100 00000110 00000100 00010000 00010010 00010100 00000000 00000001 00000010
-----

```

```

-----
CLCK  CYCLE  INST  IR  PC  SP  M(SP)  A  B  C  D  E  H  L  FLAGS  ARUS  DBUS  PINDS
0000505 000127 00080 000 000000 000140 000 000 000 000 000 016 000 056 01010 000034 000 1000000000
M(000030) M(000031) M(000032) M(000033) M(000034) M(000035) M(000036) M(000037) M(000038) M(000039)
00000010 00000100 00000110 00000100 00010000 00010010 00010100 00000000 00000001 00000010
-----

```

\*\*\*\*\*RESET = INSTRUCAD=80

```

-----
CLCK  CYCLE  INST  IR  PC  SP  M(SP)  A  B  C  D  E  H  L  FLAGS  ARUS  DBUS  PINDS
0000509 000128 00081 000 000035 000140 000 000 000 000 000 016 000 056 01010 000035 000 1000000000
M(000030) M(000031) M(000032) M(000033) M(000034) M(000035) M(000036) M(000037) M(000038) M(000039)
00000010 00000100 00000110 00000100 00010000 00010010 00010100 00000000 00000001 00000010
-----

```