SIMULADOR/MONTADOR MIX EM MINI-COMPUTADOR COM SISTEMA DE TEMPO COMPARTILHADO

Lilian Markenzon

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS (M.Sc.)

Aprovada por:

Prof. Estevam De Simone
(Presidente)

Prof. José Lucas M. Rangel Netto

Profa. Sueli Mendes dos Santos

RIO DE JANEIRO, RJ - BRASIL

SETEMBRO DE 1978

MARKENZON, LILIAN

SIMULADOR/MONTADOR MIX EM MINI-COMPUTADOR COM SISTEMA DE TEMPO COMPARTILHADO |Rio de Janeiro| 1978.

X, 89p. 29,7cm (COPPE-UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1978)

Tese - Univ. Fed. Rio de Janeiro. Fac. Engenharia

1. Simulador e Montador de Computador I. COPPE/UFRJ

II.Título: Simulador/Montador MIX em Mini-Computador Com Sistema de Tempo Compartilhado

A MEUS PAIS

AGRADECIMENTOS

A meu orientador, prof. Estevam Gilberto de Simone, pela dedicação e competência.

Aos professores Gerhard Schwarz e José Lucas Mourão Rangel Netto, pelo interêsse e pela colaboração prestada.

Ao Laboratório de Automação e Simulação de Sistemas na pessoa de seu chefe Engº Jean-Michel Nayrac e dos técnicos Adilson Magalhães e Eduardo Pereira, pela atenção e gentileza.

A meus amigos, pelo incentivo.

Este trabalho contou com o auxílio financeiro do CNPq e da FINEP.

RESUMO

São apresentados simulador do computador hipotético MIX e montador da linguagem MIXAL conforme definidos por D. E. Knuth. A implementação foi feita no minicomputador MITRA 15 do Laboratório de Automação e Simulação de Sistemas do Programa de Engenharia de Sistemas em forma reentrante para ser utilizada em sistema de tempo compartilhado por até 16 usuários simultâneos.

O simulador, dotado de facilidade de depuração de programas, apresenta eficientes níveis de aproveitamento de memória e tempo de execução.

O trabalho, além de descrever a especificação completa dos programas, contém um capítulo destinado a servir como apostila introdutória sobre a máquina hipotética. Espera-se que sua utilização seja bastante grande como instrumento didático auxiliar de diversos cursos de graduação e pós-graduação, especialmente os que utilizam a série "The Art of Computer Programming" de D.E. Knuth como texto base.

ABSTRACT

This work presents a simulator of the hypothetical MIX computer, and a MIXAL language assembler, in agreement with the definition given by D.E. Knuth. Both were implemented on a MITRA-15 minicomputer, at the Automation and Systems Simulation Laboratory of the Systems Engineering and Computation Department of COPPE-UFRJ. Both programs were written as reentrant, and are meant to be used in conjunction with a time-sharing system by up to 16 simultaneous users.

The simulator has built-in facilities for debugging and was designed with time and space efficiency in mind.

The thesis also includes a chapter meant to be used as a primer for the teaching of the use of the hypothetical machine. The system described here is expected to be intensely used as an auxiliary tool for graduate and undergraduate courses, in particular those for which books of D. E. Knuth's "The Art of Computer Programming" series are used as text.

ÍNDICE

Páginas

CAPÍTULO I - Introdução 1

CAPÍTULO II - O computador MIX 3

II.1. - Descrição da máquina 3

II.1.1. - Memória 3

II.1.2. - Registros 4

II.1.3. - Dispositivos de entrada e saída 5

II.2. - Linguagem de máquina 6

II.2.1. - Formato de Instrução 6

II.2.2. - Instruções 8

II.2.2.1. - Instruções de carga. 9

II.2.2.2. - Instruções de armazenamento 10

II.2.2.3. - Instruções aritméticas 11

II.2.2.4. - Instruções imediatas 12

II.2.2.5. - Instruções de comparação 13

II.2.2.6. - Instruções de desvio 13

II.2.2.7. - Instruções de entrada e saida 15

II.2.2.8. - Instruções de conversão 17

II.2.2.9. - Outras instruções . 18

Páginas

II.3. - Linguagem simbólica MIXAL 21

II.3.1. - Formato da instrução 21

II.3.2. - Descrição do MIXAL 22

II.3.2.1. - Conjunto de caracteres 22

II.3.2.2. - Símbolo 22

II.3.2.3. - Número 24

II.3.2.4. - Expressão 24

II.3.2.5. - Instrução 25

II.3.2.6. - Pseudo-instruções . 26

II.3.2.7. - Comentários 29

II.3.2.8. - Estrutura de Programa 30

II.3.2.9. - Árvore sintática (BNF) 30

CAPÍTULO III - O computador MITRA 32

III.1. - Dados técnicos sobre o MITRA 33

III.1.1. - Tipo 33

III.1.2. - CPU 33

III.1.3. - ROM 33

III.1.4. - Memória Principal 34

III.1.5. - Registros 34

III.1.6. - Software 35

III.1.7. - Linguagem de implementação . . 35

CAPÍTULO IV - Simulador 37

IV.1. - Linha geral de simulação 37

Páginas

IV.2. - Especificações do simulador 38

IV.2.1. - Byte MIX 38

IV.2.2. - Palavra MIX 39

IV.2.3. - Memória MIX 39

IV.2.4. - Registros MIX 41

IV.3. - Execução de um programa em linguagem de máquina 42

IV.4. - Simulação da execução de uma instrução. 43

IV.4.1. - Instruções de carga 43

IV.4.2. - Instruções de armazenamento . 44

IV.4.2.1. - Armazenamento de registros 44

IV.4.2.2. - Armazenamento do valor zero . . . 46

IV.4.3. - Instruções aritméticas . . . 46

IV.4.3.1. - ADD e SUB 46

IV.4.3.2. - MUL 48

IV.4.3.3. - DIV 49

IV.4.4. - Instruções imediatas 50

IV.4.5. - Instruções de comparação . .. 51

IV.4.6. - Instruções de desvio 52

IV.4.7. - Instruções de entrada e saída 54

IV.4.7.1. - Instruções IN e OUT 54

IV.4.7.2. - Instrução IOC . . 57

IV.4.7.3. - JRED 58

Páginas

IV.4.7.4. - JBUS 58

IV.4.8. - Instruções de conversão 59

IV.4.8.1. - NUM 59

IV.4.8.2. - CHAR 60

IV.4.9. - Outras instruções 61

IV.4.9.1. - MOVE 61

IV.4.9.2. - NOP 61

IV.4.9.3. - HLT 61

IV.4.9.4. - SHIFT 61

IV.4.9.5. - CLOC 64

IV.4.9.6. - LOOP 64

IV.4.9.7. - OUTL 64

IV.4.9.8. - TRCE 65

IV.4.9.9. - NTRC 65

CAPÍTULO V - Montador 66

V.1. - Introdução 66

V.2. - Tabelas: organização e acesso 67

V.2.1. - Tabela de Mnemônicos 67

V.2.2. - Tabela de Símbolos 69

V.2.3. - Tabela H e Tabela F 72

V.3. - Rotinas básicas 72

V.3.1. - Validade de símbolos 72

V.3.2. - Cálculo de expressões 74

V.3.3. - Valor W 75

V.4. - Instruções 75

V.5. - Pseudo-instruções 80

Páginas

V.5.1. - Pseudo EQU 80

V.5.2. - Pseudo ORIG 81

V.5.3. - Pseudo CON 81

V.5.4. - Pseudo ALF 82

V.5.5. - Pseudo END 83

V.5.6. - Pseudo CDON 83

V.5.7. - Pseudo CDOF 84

V.5.8. - Pseudo LTON 84

V.5.9. - Pseudo LTOF 84

V.5.10.- Pseudo SPAC 84

CAPÍTULO VI - CONCLUSÕES 86

CAPÍTULO I

INTRODUÇÃO

Este trabalho se propõe a elaborar a simulação do computador MIX, computador hipotético criado por Donald Knuth |[1]|, fazendo-a residente num computador MITRA, existente no Laboratório de Sistemas da COPPE-UFRJ.

O computador MIX é um computador hipotético criado com fins didáticos; sendo assim, sua estrutura e sua linguagem foram construidas para serem suficientemente poderosas a fim de permitir que pequenos programas sejam escritos para a maioria dos algoritmos utilizados nos cursos de graduação e pós-graduação em Informática. A simulação deste computador se propõe a ser um auxílio didático eficiente ao ensino de disciplinas que envolvam arquitetura e programação em computadores, como cursos de Programação Assembler, Estrutura de Dados, que utilizam como livro básico "Fundamental Algorithms" |[1]|, Busca em Arquivos e Técnicas de Ordenação que utilizam "Sorting and Searching" |[2]|.

Como a simulação é feita a partir da linguagem de máquina MIX, o trabalho inclui um tradutor de linguagem simbólica MIXAL para linguagem de máquina; foram desenvolvidas também algumas instruções visando a facilidade de depuração de programas e melhor utilização didática.

O Capítulo II é a especificação do computador

MIX e foi escrito de modo a servir como apostila de referência para os usuários do Simulador. O capítulo III é a especificação do computador MITRA disponível no Laboratório de Automação e Simulação de Sistemas, segundo informações colhidas em |[3]|. O capítulo IV é a descrição da simulação da execução de cada instrução da linguagem, a partir do código de máquina gerado pelo Montador. O Capítulo V apresenta a montagem de instruções e a descrição da execução das ações provocadas por pseudo-instruções.

CAPÍTULO II

O COMPUTADOR MIX

O MIX é um computador hipotético criado com fins didáticos por Donald E. Knuth |[1]|.

De acordo com o fim a que se propõe, sua estrutura e sua linguagem são bastante simples, visando a fácil apreensão e no entanto poderosas o suficiente para que pequenos programas possam ser escritos para a maioria de tarefas necessárias.

O MIX é apresentado no livro "Fundamental Algorithms" |[1]| no Capítulo I, seções 1.3.1 e 1.3.2, para onde deve se dirigir o leitor em busca de informações suplementares.

II.1. - Descrição da máquina

II.1.1. - Memória

A unidade básica de informação é o byte. Cada byte contem uma quantidade de informação não especificada (o MIX tem a particularidade de poder ser binário ou decimal) porém é capaz de armazenar no mínimo 64 valores distintos e no máximo 100 valores distintos. Nesta implementação foi escolhido o limite de 64 valores, entretanto, programas escritos em linguagem de máquina MIX devem poder ser executados em qualquer implementação.

Uma palavra do MIX é composta de 5 bytes mais

um sinal (+ ou -).

A memória do MIX é composta de 4000 palavras.

II.1.2. - Registros

O MIX possui nove registros:

- Registro A (Acumulador)
  Composto de cinco bytes mais sinal. Usado especialmente para operações em dados.
- Registro X (Extensão)
  Composto de cinco bytes mais sinal. Extensão à direita do registro A,é usado para inteirar 10 bytes de um produto ou dividendo,ou para armazenar informações produzidas por "shifts" para a direita do registro A.
- Registros I(Registros de índice)I1,I2,I3,I4,I5 e I6.
  Compostos de dois bytes mais sinal. Usados geralmente como contadores e para referenciar endereços variáveis de memória.
- Registro J (endereço de desvio)
  Composto de dois bytes e sinal sempre positivo. Usado para armazenar o endereço da instrução seguinte de uma instrução de desvio, basicamente para retorno de subrotinas.

E mais:

- Bit de overflow
- Indicador de comparação: assume tres valores menor ,

igual ou maior.

A notação rA será utilizada para representar o registro A,rIi para o registro Ii, $1 \geq i \geq 6$, rR para qualquer registro; CI representa o indicador de comparação e OVF o bit de overflow.

II.1.3. - Dispositivos de entrada e saída

- Disco MIX.

O disco MIX possui as seguintes especificações, fornecidas na página 362 do livro "Sorting and Searching" |[2]|:

1 unidade de disco = 200 cilindros

1 cilindro = 20 trilhas

1 trilha = 5000 caracteres.

Assim a capacidade de uma unidade de disco é de 20 milhões de caracteres.

- Fita MIX

A unidade de fita MIX, apresenta as seguintes especificações fornecidas na página 320 do livro "Sorting and Searching" |[2]|; a unidade lê e escreve 800 caracteres por polegada de fita, com velocidade de 75 polegadas por segundo. Isto significa que um caracter é lido ou escrito cada 1/60 ms. Cada carretel contém 2400 pés de fita. Informações são armazenadas por blocos na fita, sendo que cada instrução de leitura ou impressão acareta a transmissão de um único bloco. Cada bloco contém 100 palavras e o intervalo entre os blocos possui 480

caracteres.

- Teletipo

Definida conforme as características do modelo ASR33 |[4]| que apresenta velocidade de impressão de 10 caracteres por segundo. Note-se que cada registro possui 70 caracteres, equivalente a 14 palavras MIX.

- Leitora de cartões

Definida conforme as especificações do modelo LC300 |[4]| que lê cartões standard de 80 colunas (16 palavras MIX) à velocidade de 300 cartões/minuto.

II.2. - Linguagem de máquina

II.2.1. - Formato da instrução

Os cinco bytes e o sinal de uma palavra MIX são numerados da seguinte forma:

0	1	2	3	4	5
±	Byte	Byte	Byte	Byte	Byte

Numa palavra utilizada para instrução a distribuição será:

0	1	2	3	4	5
±	AA		I	F	C

onde:

- $\pm$ AA : endereço

A instrução MIX utiliza endereçamento direto. O sinal da palavra (byte 0) pertence ao endereço.

- I : especificação de índice

É usado para alterar $\pm$AA. Se I = 0 o endereço $\pm$AA é usado sem modificação; caso contrário I deve conter um valor i, sendo $1 \leq i \leq 6$ e o conteúdo de rIi é adicionado algebricamente a $\pm$AA, sendo o resultado utilizado como endereço. Este processo de indexação ocorre para qualquer instrução; M será a notação utilizada para indicar o endereço resultante deste processo e c(M) a notação do conteúdo desta posição.

- F : modificação do código de instruções

Geralmente esta modificação será uma especificação de campo. Muitas instruções permitem ao programador utizar somente parte da palavra. Neste caso será dada uma especificação de campo do tipo (L:R) onde L é o número do byte à esquerda do campo desejado e R o número do byte a direita.

Exemplos:

(0:0) - somente o sinal é considerado

(0:2) - o sinal e os dois primeiros bytes

(0:5) - a palavra toda

(4:4) - somente o quarto byte

A especificação de campo F é calculada através da fórmula F = 8L + R o que permitirá sua representação num só byte.

Outros usos do campo F serão descritos quando da especificação da instrução correspondente.

- C : Código de operação

Para melhor visualização uma instrução será representada por:

OP ±AA, I(F)

onde OP é o mnemônico do código de instrução (C). Admitem-se opcionalmente as seguintes alterações:

- se I = 0, pode ser omitido
- se F é a especificação padrão do operador OP, (F) pode ser omitido.

Exemplos: LDA 2000, 5(4:4)

ADD -1000,4 é equivalente a ADD -1000,4(0:5)

STA 100 é equivalente a STA 100,0(0:5)

II.2.2. - Instruções

Será usada nesta seção a seguinte notação:

rR : registros quaisquer

rAX: registros rA e rX juntos (10 bytes)

i : número entre 1 e 6

M : endereço resultante após o processo de indexação

c(M) : conteúdo da posição de memória M

campo: especificação de bytes de uma palavra

V : campo específico de c(M) tal como seria carregado em qualquer registro.

A ← B: A recebe o valor de B

PC : endereço da próxima instrução a ser executada

II.2.2.1. - Instruções de carga

LDA (load A) C = 8 ; F = campo

LDX (load X) C = 15; F = campo

LDi (load Ii) C = 8+i; F = campo

Ação : rR ← c(M)

LDAN (load A negativo) C = 16 ; F = campo

LDXN (load X negativo) C = 23 ; F = campo

LDiN (load Ii negativo) C = 1b+i ; F = campo

Ação : rR ← -c(M)

Em todas as operações onde um campo parcial é utilizado, o sinal é usado se ele é parte do campo, caso contrário o sinal + é subentendido. O campo é deslocado para a direita do registro onde será carregado.

Exemplos: Suponha-se a posição 2000

-	80	5	7	9

Instrução	rA após a execução				
LDA 2000	-	80	5	7	9
LDA 2000(0:1)	-	0	0	0	0	?
LDA 2000(4:4)	+	0	0	0	0	7
LDA 2000(1:1)	+	0	0	0	0	?
LDAN 2000(3:4)	-	0	0	0	5	7

Para os demais registros a operação é análoga.

II.2.2.2. - Instruções de armazenamento

STA (store A) C = 24 ; F = campo

STX (store X) C = 31 ; F = campo

STi (store Ii) C = 24+i ; F = campo

STJ (store J) C = 32 ; F = campo

STZ (store zero) C = 33 ; F = campo

Ação : M ← rR

Nas instruções de armazenamento o número de bytes do campo desejado é tomado do lado direito do registro e inserido na posição especificada pelo campo.

Exemplos: Suponha-se

Posição 2000:

-	1	3	5	7	9

rA

+	0	2	4	6	8

Instrução	c(2000) após a execução:
STA 2000(1:5)	- 0 2 4 6 8
STA 2000(1:2)	- 6 8 5 7 9
STA 2000(0:1)	+ 8 3 5 7 9
STA 2000(3:5)	- 1 3 4 6 8

II.2.2.3. - Instruções aritméticas

ADD(soma) C = 1 ; F = campo

Ação : rA ← rA + V

Se o resultado excede a capacidade de rA, OVF é ligado (OVF ← True) e rA armazena os cinco bytes à direita do resultado e seu sinal.

SUB(subtração) C = 2 ; F = campo

Ação: rA ← rA - V

Pode ocorrer overflow da mesma forma que na adição.

MUL(multiplicação) C = 3 ; F = campo

Ação : rAX ← rA * V

Os sinais de rA e rX recebem o sinal algébrico do produto (isto é, + se rA e V são de mesmo sinal, - em caso contrário)

DIV(divisão) C = 4 ; F = campo

Ação: rA ← rAX/V ; rX ← (rAX) MOD V

O sinal de rA é considerado como sinal de rAX. Se V = 0 ou o resultado é maior que 5 bytes rA e rX ficam indefinidos e o bit de overflow é ligado. Após a operação rX contém o resto com o sinal de rA; rA contém o quociente com seu sinal algébrico.

II.2.2.4. - Instruções imediatas

ENTA (enter A) C = 48 ; F = 2

ENTX (enter X) C = 55 ; F = 2

ENTi (enter Ii) C = 48+i ; F = 2

Ação : rR ← M

Se M = 0 o sinal da instrução é carregado.

ENNA (enter negative A) C = 48 ; F = 3

ENNX (enter negative X) C = 59 ; F = 3

ENNi (enter negative Ii) C = 48+i; F = 3

Ação : rR ← -M

INCA (increase A) C = 48 ; F = 0

INCX (increase X) C = 55 ; F = 0

INCi (increase Ii) C = 48+i ; F = 0

Ação : rR ← rR + M

DECA (decrease A) C = 48 ; F = 1

DECX (decrease X) C = 55 ; F = 1

DECi (decrease Ii) C = 48+i ; F = 1

Ação: rR ← rR - M sendo $-4095 \leq M \leq +4095$ se I = 0

$-8190 \leq M \leq +8190$ se $I \neq 0$

II.2.2.5. - Instruções de comparação

CMPA (compare A) C = 56 ; F = campo

CMPX (compare X) C = 63 ; F = campo

CMPi (compare Ii) C = 56+i ; F = campo

Ação : comparar rR com c(M) resultando:

Se rR > c(M) acarreta CI ← MAIOR

Se rR < c(M) acarreta CI ← MENOR

Se rR = c(M) acarreta CI ← IGUAL

II.2.2.6. - Instruções de desvio

JMP (jump) C = 39 ; F = 0

Ação : rJ ← PC;

PC ← M ;

Habitualmente as instruções são executadas sequencialmente; as instruções de desvio (JMP e as outras que serão vistas a seguir) servem para modificar esta sequência. O registro J serve para armazenar qual seria a sequência normalmente executada uma vez que ele conterá o endereço da instrução seguinte.

JSJ (jump, save J) C = 39 ; F = 1

Ação : mesma que o JMP, exceto que rJ permanece inalterado.

JOV (jump on overflow) C = 39 ; F = 2

Ação: Se OVF = true então rJ ← PC;
PC ← M ;
senão;

JNOV (jump on no overflow) C = 39 ; F = 3

Ação : Se OVF = false então rJ ← PC;
PC ← M ;
senão OVF ← false;

JL (jump on less) C = 39 ; F = 4

JE (jump on equal) C = 39 ; F = 5

JG (jump on greater) C = 39 ; F = 6

JGE(jump on non-less) C = 39 ; F = 7

JNE(jump on non-equal) C = 39 ; F = 8

JLE(jump on non-greater) C = 39 ; F = 9

Ação : o desvio ocorre se o indicador de comparação apresenta a condição indicada.

JAN (jump A negative) C = 40 ; F = 0

JAZ (jump A zero) C = 40 ; F = 1

JAP (jump A positive) C = 40 ; F = 2

JANN (jump A nonnegative) C = 40 ; F = 3

JANZ (jump A nonzero) C = 40 ; F = 4

JANP (jump A nonpositive) C = 40 ; F = 5

JXN (jump X negative) C = 47 ; F = 0

JXZ (jump X zero) C = 47 ; F = 1

JXP (jump X positive) C = 47 ; F = 2

JXNN (jump X nonnegative) C = 47 ; F = 3

JXNZ (jump X nonzero) C = 47 ; F = 4

JXNP (jump X nonpositive) C = 47 ; F = 5

JiN (jump Ii negative) C = 40+i ; F = 0

JiZ (jump Ii zero) C = 40+i ; F = 1

JiP (jump Ii positive) C = 40+i ; F = 2

JiNN (jump Ii nonnegative) C = 40+i ; F = 3

JiNZ (jump Ii nonzero) C = 40+i ; F = 4

JiNP (jump Ii nonpositive) C = 40+i ; F = 5

Ação: o desvio ocorre se o conteúdo de rR satisfaz a condição, caso contrário nada ocorre.

Note-se que positivo significa maior que zero (não zero); não positivo significa o contrário, isto é, zero ou negativo.

II.2.2.7. - Instruções de entrada e saída

IN (entrada) C = 36 ; F = unidade

Ação: inicia a transferência da informação da unidade de entra

da especificada para posições consecutivas começando em M. O número de posições transferidas é o tamanho do bloco da unidade.

OUT (saída) C = 37 ; F = unidade

Ação : inicia a transferência da informação de posições de memória a partir de M para a unidade de saída especificada.

A máquina aguarda neste ponto se uma operação precedente da mesma unidade estiver incompleta. A transferência de informação que começa nesta instrução só será terminada algum tempo após, dependendo do dispositivo de entrada ou saída que está sendo utilizado; assim, não é aconselhável fazer referências, sem precauções, à posições de memória que estão sendo alteradas, até que a operação seja completada.

IOC (controle de entrada e saída) C = 35 ; F = unidade

Ação : uma operação de controle é executada, dependendo do dispositivo:

Fita magnética : Se M = 0 a fita é reenrolada. Se M < 0 a fita volta M registros ou volta para o início da fita, o que ocorrer primeiro. Se M > 0 a fita avança (a operação será ignorada e a unidade suspensa se avançar mais do que o último registro escrito na fita)

Disco : M deve ser zero. Posiciona o dispositivo de acordo com rX (ver observação abaixo) para economizar tempo da próxima instrução IN ou OUT.

Impressora : M deve ser zero. Posiciona a impressora no topo da próxima página.

Fita de papel : M deve ser zero. Reenrola a fita.

Nas instruções IN, OUT e IOC relativas a disco o bloco de 100 palavras referido é especificado pelo conteúdo dos dois bytes menos significantes de rX (bytes 4 e 5).

JRED (jump ready) C = 38 ; F = unidade

Ação : o desvio ocorre se a unidade especificada terminou a operação iniciada por IN, OUT ou IOC

JBUS (jump busy) C = 34 ; F = unidade

Ação : mesma que JRED exceto que o desvio ocorre em condição oposta.

II.2.2.8. - Instruções de conversão

NUM (conversão a numérico) C = 5 ; F = 0

Ação : converte código de caracteres para código numérico.M é ignorado. rAX é suposto contendo um número de 10 bytes escrito em caracteres que são convertidos a um número decimal armazenado em rA. A conversão é feita da seguinte forma: 00,10,20,... são convertidos a 0; 01, 11,21,... convertido a 1 e assim sucessivamente. O valor de rX e o sinal de rA permanecem sem alteração.Overflow é possível; neste caso o resto é armazenado.

CHAR (conversão a caracteres) C = 5 ; F = 1

Ação : converte código numérico a código de caracteres, código este necessário para a saída em cartões ou impressora. O valor contido em rA é convertido a um número decimal

de 10 bytes em rAX. Os sinais de rA e rX não sofrem alterações. M é ignorado. Exemplo:

Operação	Registro A						Registro X					
conteúdo inicial	-	00	40	27	31	85	+	38	94	15	30	20
NUM 0	-					71584500	+	38	94	15	30	20
CHAR 0	-	30	30	37	31	35	+	38	34	35	30	30

II.2.2.9. - Outras instruções

MOVE C = 7 ; F = número

Ação : Transferir o número de palavras especificadas por F começando da posição de memória M para a posição de memória dada pelo conteúdo de rI1. É transferida uma palavra de cada vez e rI1 é incrementado do valor de F ao fim da operação. Se F = 0 nada acontece.

SLA (shift left A) C = 6 ; F = 0

SRA (shift right A) C = 6 ; F = 1

SLAX (shift left AX) C = 6 ; F = 2

SRAX (shift right AX) C = 6 ; F = 3

SLC (shift left AX circularly) C = 6 ; F = 4

SRC (shift right AX circularly) C = 6 ; F = 5

Ação: rR é movimentado o número de bytes especificado por M.

Os sinais de rA e rX não são alterados. SLA e SRA não alteram rX; os outros operadores alteram rAX. As intruções SRA, SLA, SRAX e SLAX movimentam bytes para um lado, completando com zero os bytes necessários; as instruções SRC e SLC movimentam circularmente, i.e., os bytes que desaparecem de um lado, surgem do outro.

Exemplo

	Registro A						Registro X					
	+	1	2	3	4	5	-	6	7	8	9	10
SLA 1	+	2	3	4	5	0	-	6	7	8	9	10
SRC 3	+	8	9	10	2	3	-	4	5	0	6	7
SRAX 2	+	0	0	8	9	10	-	2	3	4	5	0

NOP (no operation) C = 0

Ação : nada ocorre. M e F são ignorados.

HLT (halt) C = 5 ; F = 2

Ação : a máquina para.

As instruções abaixo não constam da definição original mas foram acrescentadas visando facilitar a depuração e a medida de desempenho dos programas. Para estas instruções o relógio do simulador não é incrementado.

CLOC C = 5 ; F = 3

Ação : o relógio MIX é impresso

Unidade de tempo MIX : 1 U = 10μs

Formato de impressão:

**TEMPO:0000000000 U

LOOP C = 5 ; F = 4

Ação: o campo (18 bites) formado pelas partes A e I da instrução é incrementado de 1 e o novo valor é guardado (no mesmo campo) da posição de memória da instrução original.

OUTL C = 5 ; F = 5

Ação : imprime o campo formado pelas partes A e I do endereço mencionado.

Formato de impressão:

**LOOP:000000

TRCE (trace) C = 5 ; F = 6

Ação : Imprime a partir deste ponto, em cada instrução: a instrução corrente, e os conteúdos de rA, rX, rI1, rI2, rI3, rI4, rI5, rI6.

Formato de impressão:

INST=0000 A=±0000000000 I1=±0000 I2=±0000 I3=±0000

X=±0000000000 I4=±0000 I5=±0000 I6=±0000

NTRC(no trace) C = 5 ; F = 7

Ação : desligar o rastreamento, se ligado. Caso contrário nada ocorre.

II.3. - Linguagem simbólica MIXAL

MIXAL é a linguagem simbólica do computador MIX (MIX Assembly Language).

MIXAL é uma extensão da notação usada para as instruções em linguagem de máquina já vistas na seção anterior.Os aspectos principais das modificações introduzidas são:

- Uso de símbolos representando endereços (antes numéricos).Para tal é necessário a criação de um novo campo onde estes símbolos aparecem e são associados à posição de memória componente da instrução.
- Pseudo-instruções. Não são traduzidas para linguagem de máquina mas produzem alterações no código gerado, como por exemplo a pseudo CON que define constantes em posições de memória ou a pseudo ORIG que determina onde será montada a instrução seguinte (todas as pseudos serão vistas na descrição da linguagem).

II.3.1. - Formato da instrução

Um comando MIXAL (incluindo aqui as pseudo-intruções e as instruções já conhecidas pela sua associação direta com as instruções da linguagem de máquina) conterá basicamente tres campos:

Campo LOC : os nomes aí definidos são associados a esta posição de memória

Campo OP : mnemônico da instrução ou pseudo.

Campo ADDRESS : endereço, especificação de índice e modificação do código de operação.

Ao elaborar um programa MIX o programador deverá levar em conta as posições fixas utilizadas por estes campos:

Colunas 1-10 : campo LOC
Colunas 12-15 : campo OP
Colunas 17-80 : campo ADDRESS e comentários opcionais.

II.3.2. - Descrição da MIXAL

II.3.2.1. - Conjunto de caracteres

São os seguintes os caracteres utilizados pela linguagem:

Letras : 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 e Z
Dígitos: 0,1,2,3,4,5,6,7,8,9
Operadores: +, -, *, /, //, :
Delimitadores: (,), , = , ,

II.3.2.2. - Símbolo

Um símbolo é uma sequência de um a dez letras e/ou dígitos contendo no mínimo uma letra.
Exemplos: 21A34 A2 2A TEMPORARIA

Um símbolo é associado a um valor numérico no momento em que aparece no campo LOC de um comando MIXAL(valor nu

mérico este que é o endereço de memória correspondente ao comando). Assim um símbolo pode ser um "símbolo definido" que é aquele que já apareceu no campo LOC de um comando precedente do programa ou uma "referência futura" que é o símbolo que ainda não foi definido desta forma (e o deverá ser).

Os símbolos nF, nB e nH (onde n é um único dígito) são chamados "símbolos locais" e possuem um significado especial, uma vez que podem ser redefinidos. Um símbolo como ABC terá um único significado no decorrer de todo o programa; já por exemplo, 2H ("2 aqui") no campo LOC, 2B("2 atras") e 2F("2 a frente") no campo ADDRESS significarão:

2B → a localização anterior de 2H

2F → a localização posterior de 2H

Como se pode ter diversos 2H no programa estes "anterior" e "posterior" se referem, para ser mais exato, a "anterior" e "posterior" mais próximos no programa fonte. Exemplo:

```
2H    LDA   A
      ADD   B
2H    ENNA  1
      INC1  1
      JAP   2B
```

neste trecho de programa a instrução JAP 2B desvia, se for o caso, para a instrução ENNA 1.

II.3.2.3. - Número

Um número é uma sequência de 1 a 10 dígitos.

Exemplo: 48 0032 1000000000

O maior número que pode ser armazenado numa palavra MIX é 1073741823, e o menor -1073741823.

II.3.2.4. - Expressão

Uma expressão atômica é um número, um símbolo definido ou um * significando a localização da linha. (A partir daqui * com este significado será representado por (*) e chamado contador de posição).

Uma expressão é uma expressão atomica, um operador + ou - seguido de uma expressão atomica ou uma expressão seguida de um operador, seguida de uma expressão atomica.

Exemplos de expressões:

1024	* * * significando (*) * (*)
-38	20-3*4-2
AB	
AB+3	
*-3*8	

As seis operações binárias são +, -, *, /, //, : . Elas podem ser definidas da seguinte forma.

C = A + B	LDA AA ; ADD BB; STA CC
C = A - B	LDA AA , SUB BB; STA CC

```
C = A*B          LDA AA; MUL BB; STX CC
C = A/B          LDX AA; DIV BB; STA CC
C = A//B         LDA AA; ENTX 0; DIV BB; STA CC
C = A:B          LDA AA; MUL =8=; SLAX 5; ADD BB; STA C
```

onde AA, BB, CC significam posições de memória contendo respectivamente os valores A,B e C.

As operações são realizadas da esquerda para a direita sem prioridades.

II.3.2.5. - Instrução

Na tradução da linguagem simbólica para linguagem de máquina será utilizado o contador de posição inicializado em zero; a medida que as instruções forem traduzidas serão colocadas na posição apontada pelo contador. Assim, cada instrução acarreta um incremento de uma unidade no contador de posição que aponta sempre para a primeira posição livre de memória.

Na instrução MIXAL os campos LOC, OP e ADDRESS terão o seguinte significado:

O campo LOC pode ser um símbolo, que a partir deste momento será um "símbolo definido", uma vez que é feita a correspondência entre o símbolo e o contador de posição, um símbolo local (neste campo o símbolo local só pode ser do tipo nH, $0 \leq n \leq 9$) ou vazio.

O campo OP deve conter um mnemônico de instrução (NOP, ADD, SUB, MUL,...). A lista completa é apresentada na árvore sintática (II.3.2.9).

O campo ADDRESS deve ser composto de uma parte A (relativa ao endereço da instrução) seguida opcionalmente por uma parte I (relativa a especificação do índice) e de uma parte F (relativa a modificação do código de instrução). Por sua vez a parte A pode ser uma expressão, um símbolo local (neste campo só pode ser do tipo nB ou nF, $0 \leq n \leq 9$), uma referência futura, uma constante literal (descrita em II.3.2.6) ou vazio (significando zero); a parte I será um delimitador "," seguido de uma expressão ou vazio (significando zero); a parte F será um delimitador "("seguido de uma expressão, seguido de uma expressão, seguido de um delimitador")" ou vazio (significando o F padrão baseado no contexto).

II.3.2.6. - Pseudo-instruções

O campo OP das pseudos deve conter EQU, ORIG, CON, ALF, END, CDON, CDOF, LTON,LTOF ou SPAC. De acordo com este campo os campos LOC e ADDRESS tem diferentes significados. Na maioria das pseudos entretanto (EQU,ORIG,CON,END e SPAC), o campo ADDRESS deve ser um valor W, que é usado para descrever uma palavra MIX ocupando o campo (0:5). Um valor W será da forma $E_1(F_1),E_2(F_2),\ldots,E_n(F_n)$ onde $n \geq 1$, $E_i (1 \leq i \leq n)$ é uma expressão e $F_i (1 \leq i \leq n)$ é um campo. O resultado é o valor final que apareceria na posição CON se o seguinte programa hipotético fosse executado:

```
STZ    CON
LDA    C1
STA    CON(F1)
  :
  :
  :
LDA    Cn
STA    CON(Fn)
```

onde $C_1, C_2, \ldots, C_n$ significam posições de memória que contém os valores de $E_1, E_2, \ldots, E_n$.

Um valor W de no máximo 9 caracteres encerrada entre delimitadores "=" (um à esquerda, outro a direita) é chamado uma "constante literal".

EX: LDA =8=

Significa a criação de uma palavra na primeira posição de memória disponível após a montagem completa do programa que conterá a constante 8 e cujo endereço será referenciado pelo LDA.

Cuidados especiais deverão ser tomados pelo programador para evitar alocação de outros dados nesta área.

- Pseudo EQU

Se o campo LOC não for vazio êste símbolo será associado ao valor W existente no campo ADDRESS; se for vazio é erro. O contador de posição permanece inalterado.

- Pseudo ORIG

O campo LOC tem o mesmo tratamento que na instrução.

O campo ADDRESS deve ser um valor W; o contador de posição será igualado a este valor.
Note-se que o símbolo existente no campo LOC é equivalente ao valor do contador de posição antes que este sofra a alteração.

- Pseudo CON

O campo LOC tem o mesmo tratamento que na instrução.
O campo ADDRESS deve ser um valor W; é montada uma palavra com este valor na posição assinalada pelo contador de posição. O contador de posição é incrementado de 1.

- Pseudo ALF

Semelhante à CON com a única diferença que o campo ADDRESS é formado por cinco caracteres.

- Pseudo END

A pseudo END assinala o fim de um programa MIXAL. O valor W que deve estar no campo ADDRESS assinala qual a posição de memória em que a execução do programa deve começar.

- Pseudo CDON

Os campos LOC e ADDRESS são ignorados. A partir do aparecimento desta pseudo, o código gerado pelas instruções seguintes é listado. É importante notar que, no caso de instruções que possuem a parte A ainda incompleta e serão terminadas mais tarde (por exemplo, referências futuras, símbolos locais nF, constantes literais) serão impressas com as informações disponíveis no momento.

- Pseudo CDOF

Os campos LOC e ADDRESS são ignorados. A partir do aparecimen

to desta pseudo o código gerado deixa de ser listado, o estiver sendo. Caso contrário, nada ocorre.

- Pseudo LTON

Os campos LOC e ADDRESS são ignorados. A partir do aparecimento desta pseudo o programa fonte é listado. O montador gera sempre uma ocorrência desta instrução no início do programa.

- Pseudo LTOF

Os campos LOC e ADDRESS são ignorados. A partir do aparecimento deste pseudo o programa fonte não é listado.

- Pseudo SPAC

O campo LOC é ignorado. O campo ADDRESS deve ser um valor W, com valor máximo 63; este valor indica o número de linhas a serem puladas a seguir. Caso o campo ADDRESS seja vazio é assumido o valor 1.

II.3.2.7. - Comentários

Comentários podem ser inseridos em qualquer ponto do programa de duas maneiras distintas:

- em qualquer comando MIXAL após o campo ADDRESS deixando um espaço em branco
- um cartão tendo na primeira coluna o símbolo "*".

II.3.2.8. - Estrutura de programa

Um programa MIXAL é um conjunto de comandos. Terminando obrigatoriamente com um comando que possue a pseudo END.

II.3.2.9. - Árvore sintática (BNF)

<ALFABETO>::=<LET>|<DIG>|<OPER>|<DEL>

<LET>::=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

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

<OPER>::= +|-|*|/|//|:

<DEL>::= (|)| |=| ,

<OP1>::= NOP|ADD|SUB|MUL|DIV|NUM|CHAR|HLT|SLA|SRA|SLAX|
SRAX|SLC|SRC|MOVE|LDA|LD1|LD2|LD3|LD4|LD5|LD6|LDX|
LDAN|LD1N|LD2N|LD3N|LD4N|LD5N|LD6N|LDXN|STA|
ST1|ST2|ST3|ST4|ST5|ST6|STX|STJ|STZ|JBUS|IOC|
IN|OUT|JRED|JMP|JSJ|JOV|JNOV|JL|JE|JG|JGE|
JNE|JLE|JAN|JAZ|JAP|JANN|JANZ|JANP|J1N|
J1Z|J1P|J1NN|J1NZ|J1NP|J2N|J2Z|J2P|J2NN|
J2NZ|J2NP|J3N|J3Z|J3P|J3NN|J3NZ|J3NP|
J4N|J4Z|J4P|J4NN|J4NZ|J4NP|J5N|J5Z|
J5P|J5NN|J5NZ|J5NP|J6N|J6Z|J6P|J6NN|
J6NZ|J6NP|JXN|JXZ|JXP|JXNN|JXNZ|JXNP|
INCA|DECA|ENTA|ENNA|INC1|DEC1|ENT1|ENN1|
INC2|DEC2|ENT2|ENN2|INC3|DEC3|ENT3|ENN3|
INC4|DEC4|ENT4|ENN4|INC5|DEC5|ENT5|ENN5|
INC6|DEC6|ENT6|ENN6|INCX|DECX|ENTX|ENNX|

CMPA|CMP1|CMP2|CMP3|CMP4|CMP5|CMP6|CMPX|

CLOC|LOOP|OUTL|TRCE|NTRC

<OP2>::= EQU|ORIG|CON|ALF|END|CDON|CDOF|LTON|LTOF|SPAC

<LS1>::= <DIG> H

<LS2>::= <DIG> B | <DIG> F

<SIMB>::= $\{<LET>|<DIG>\}_k$ <LET> $\{<LET>|<DIG>\}_m$

$k+m \leq 9$

<NUM>::= <DIG>{<DIG>}

<ATOM>::= <NUM>|<SIMB>| *

<EXP>::= [+|-] <ATOM> | <EXP><OPER><ATOM>

<LOC>::= [<SIMB>|<LS1>]

<INST>::= <LOC><OP1><ADDR>

<ADDR>::= <AA>[<I>] [<F>]

<AA>::= [<EXP>|<CL>|<LS2>]

<I>::=,<EXP>

<F>::= (<EXP>)

<CL>::= =<W>=

<W> ::= <EXP>[<F>] |<W>,<EXP>[<F>]

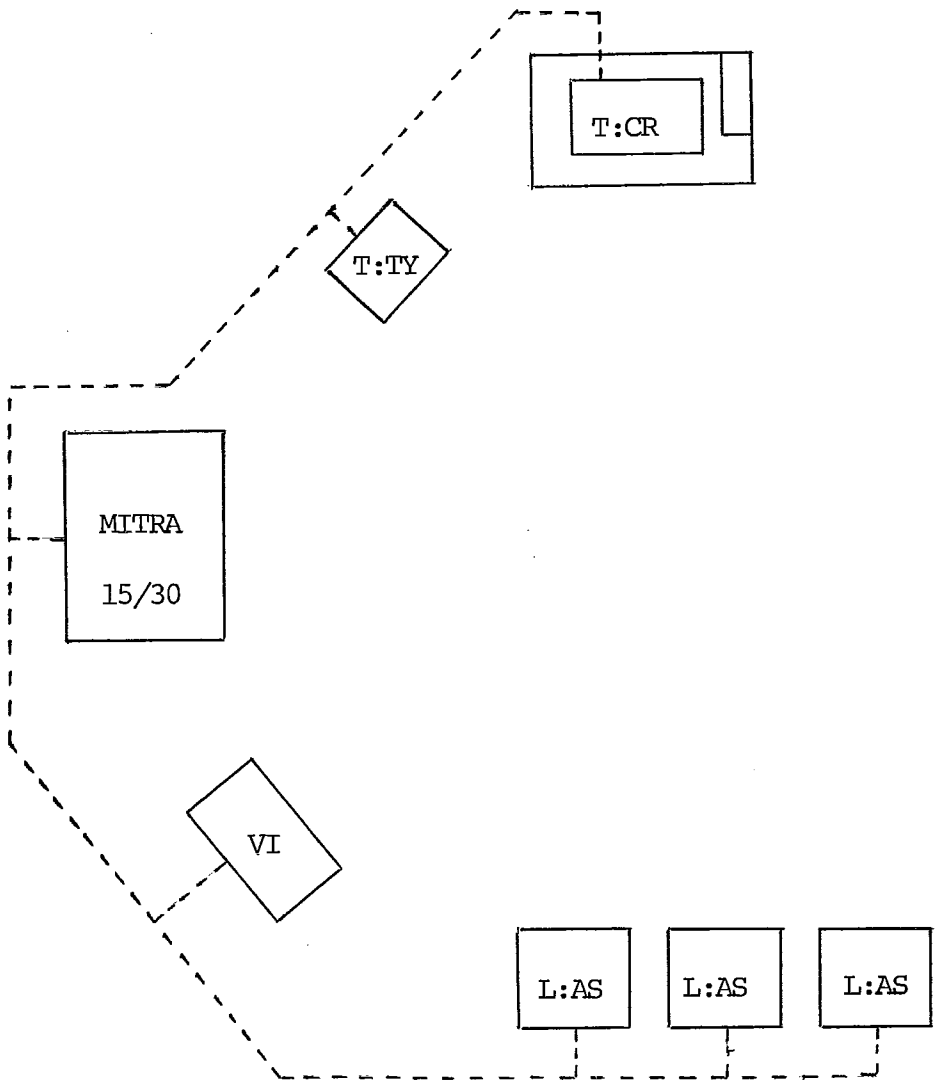
<PSEUDO>::= <LOC><OP2><W>

<COM>::= <INST>|<PSEUDO>

CAPÍTULO III

O COMPUTADOR MITRA

A configuração do computador MITRA 15/30 existente no Laboratório de Automação e Simulação de Sistemas da COPPE, utilizada para a implantação do trabalho de simulação do MIX é a seguinte:



MITRA 15/30 : Computador Digital, 16K palavras de 16 bits

ciclo : 800 nanosegundos

tempo médio/instrução: 3 microsegundos

T : CR : Leitora de cartões

T : TY : teletipo de serviço

VI : video (compatível com o teletipo L:AS)

L : AS : teletipo ligado via linhas assíncronas

III.1. - Dados técnicos sobre o MITRA

III.1.1. - Tipo : MITRA 15/30 da Compagnie Internationale pour l'Informatique (CII)

III.1.2. - CPU :
- Tipo multiprogramável
- Tamanho da palavra: 16 bits + 1 bit de paridade + 1 bit de proteção de memória
- Modos de endereçamento: direto, indireto, indexado, relativo, imediato, local e geral.
- Opção: 32 níveis de interrupção, possibilidade de multiplexação até 112 níveis e subníveis.
- Operação de ponto flutuante

III.1.3. - ROM :
- Tipo : circuitos integrados bi-polares
- Tamanho: 512-1024 palavras/processador
- Tempo de acesso: 60 nanosegundos
- MCl: executa o código das instruções básicas e as funções para acoplar o terminal lento (existente no LASS): Console (T:TY),

KSR 33 modificado, QWERTY em ASCII, 52 chaves, 10 cps, 72 caracteres/linha

- MC2 : executa o código complementar de instruções opcionais (operações sobre cadeias, ponto flutuante, proteção a memória, "test and set") e as funções para acoplar terminais rápidos:
  -Leitora de cartões (T:CR) com 300 cpm
  -Disco com braço movel: 1 disco fixo,1 disco removível, cada um com 5Mbytes (256 bytes/setor, 24 setores/trilha, 400 trilhas/face, 2faces/disco):
  -Velocidade de rotação: 2400 rev/min
  -Velocidade de transmissão: 312Kbytes/s
  - Tempo de espera: 12,5 ms
  - Tempo médio de posicionamento das cabeças: 38ms

III.1.4. - Memória Principal: - Tipo: Lithium ferrite
- Tamanho: 16 Kpalavras
- Tempo de um ciclo: 800ns (ler + escrever)
- Endereçamento: byte de 8 bits

III.1.5. - Registros: - Capacidade: 64 registros
- Registros principais: acumulador, extensão, apontador de programa, indexadro, base local e base geral
- tempo de acesso: 60 ns

III.1.6. - Software:

- Assembler : MITRAS 1,MITRAS 2(extensão)
- Compiladores: LP15, LP15E
- Interpretadores: FORTRAN IV, BASIC 1, BASIC 4
- Utilitários: Biblioteca Matemática, Gestão de Arquivos
- Sistemas Operacionais MOB, MTR, MTRD

  Funções: processamento de erros, controle interno de I/O, controle interno de interrupções, controle de execução de programas, multiprogramação, comunicação com o operador, proteção de memória, controle de "overlay", "management" do sistema, da biblioteca e dos recursos

- Sistema Operacional MCC00

  O MCC00 é uma versão reduzida do MTRD ocupando o mínimo de memória necessária. Possui o núcleo do Sistema Operacional, o MC2 e responde a interrupções a partir do teletipo. Note-se que alguns subprogramas do núcleo do Sistema Operacional não se encontram no MCC00, como, por exemplo, o tratamento de arquivos. A utilização deste monitor é feita com o compilador LP15E, uma vez que a utilização deste compilador em conjunto com o MTRD não é possível por ultrapassar o tamanho da memória.

- Link-editor.

III.1.7. - Linguagem de implementação

A linguagem utilizada na implantação do trabalho apresentado é o LP15E (LP15 extendido). A idéia básica desta linguagem é incorporar ao assembler (que permite um con-

trole preciso do código binário gerado e das zonas de dados, mas impõe restrições de formato, número elevado de mnemônicos, documentação trabalhosa, etc) algumas características que fazem a comodidade das linguagens de mais alto nível, mas somente aquelas que não impedem o controle dos registros e memória em tempo de execução, tais como:

- Formato livre
- Declarações de tipo correspondendo aos diversos tipos de quantidade endereçavel (byte, palavra, longa, tabela)
- Expressões aritméticas e lógicas
- Instruções condicionais : IF...THEN...ELSE...;
- Instruções iterativas: FOR...STEP...UNTIL...DO;
  REPEAT...;
  CYCLE;
  WHILE...DO;
- Instruções de desvio por tabela (CASE...OF...)

Por outro lado deve-se observar que a linguagem permite a utilização de qualquer instrução especial da máquina. A linguagem contém o assembler e todas as instruções de máquina podem ser declaradas como funções da linguagem.

- Extensão AMAP para "debugging"

Para maiores detalhes consultar os itens |[3]|,|[5]|, |[6]|, |[7]| e |[8]| da bibliografia.

CAPÍTULO IV

SIMULADOR

IV.1. - Linha geral de simulação

A simulação do computador MIX no MITRA 15 é feita em nível de instrução. A partir de uma configuração inicial codificada da memória simulada e do endereço inicial de execução são interpretadas as intruções. Como no MIX as instruções de entrada e saida são executadas simultaneamente ao trabalho do processador central, o simulador decompõe todas as instruções em eventos, através de procedimentos próprios. Estes procedimentos necessitam de algumas informações para as quais foram criadas as seguintes tabelas:

- Tabela de Ocupação: indica o uso de determinado equipamento. É o primeiro item a ser observado ao se interpretar uma operação de entrada e saida.

- Tabela de Contadores: indica o número de transferências (eventos) a serem realizados para cada unidade, caso a unidade esteja em uso. É preenchida, ao ocorrer uma operação de entrada e saída, com o número de posições de memória a serem transferidas, que depende do tamanho do bloco da unidade.

- Lista Ordenada de Unidades: lista ordenada, segundo o momento da próxima transferência, das unidades em uso. Esta lista é reorganizada, se necessário, cada vez que um novo

momento de cada unidade é inserido.

- Tabela de Eventos: contém para cada unidade o momento em que deverá acontecer a próxima transferência de informação. A inserção nesta tabela é feita da seguinte forma:

a) É calculado o momento em que deve ocorrer o próximo evento da operação considerada, de acordo com a unidade utilizada.

b) A Lista Ordenada de Unidades é consultada a fim de ser calculada a nova posição da unidade, de acordo com o momento do evento previsto, sendo então reorganizada.

c) É feita a inserção propriamente dita na lista de eventos.

- Tabela de Operações e Endereços: indica para cada unidade em uso qual a operação que está sendo executada e qual o endereço referenciado pela intrução. Torna-se necessária pelo fato de uma operação de entrada e saída não ser inteiramente simulada ao surgir, vindo a ser terminada algum tempo mais tarde. É alterada ao ser realizada uma operação de entrada e saída e permanece com o valor atribuido neste ponto do programa até que a operação seja dada por completa, algumas instruções mais tarde.

IV.2. - Especificações do simulador

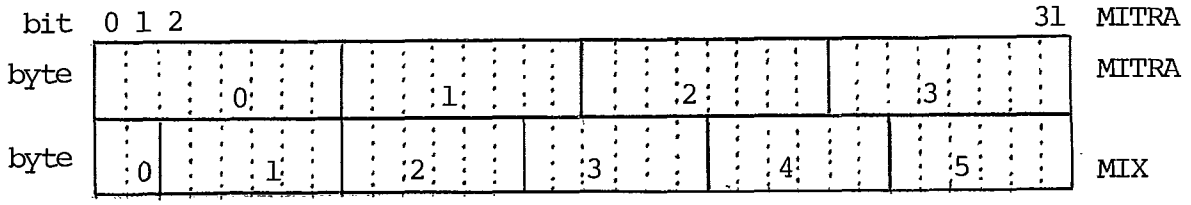
IV.2.1. - Byte MIX

Um byte MIX deve ser capaz de representar 64 va-

lores distintos; sendo o MITRA um computador binário 1 byte MIX será representado por 6 bits.

IV.2.2. - Palavra MIX

A palavra MIX contém cinco bytes MIX e sinal. Sua simulação no MITRA utiliza uma palavra longa com a seguinte distribuição:



IV.2.3. - Memória MIX

É impossível mater toda a memória simulada MIX na memória real MITRA, uma vez que a memória MIX ocupa 16000 bytes e o sistema está previsto para vários usuários simultâneos. É usado um arquivo para cada usuário que contém a sua memória MIX(63 setores) e na memória MITRA são reservadas 512 longas (2 Kbytes) divididas em 8 partes de 64 longas que serão chamadas páginas, cada página correspondendo, em tamanho, a u setor do disco MITRA e a 64 palavras MIX. O sistema de memória virtual apresentado é solucionado pelo método "First In First Out" como será visto a seguir.

São fornecidas ao usuário duas opções:

Opção rápida:

Foi verificado que a maior parte dos programas não utiliza mais de 512 palavras de memória. Neste caso o usuário deve utilizar a memória MIX no intervalo de posições

[0,511]. As oito páginas representam então esta parte da memória MIX e o simulador verifica o respeito a esse intervalo.

Opção lenta:

O usuário tem total liberdade de uso da memória MIX, porém apenas 8 páginas estarão simultaneamente disponíveis em memória.

Para o tratamento são usadas, por usuário, duas listas:

- Fila de Setores (8 bytes): lista que contém os 8 últimos setores que foram trazidos do arquivo.

- Lista de Posição de Setor (63 elementos - 32 bytes): cada setor tem nesta lista sua posição:
  a) se não está em uso: 0
  b) se está em uso: página que está ocupando.
  Como se tem oito páginas disponíveis, cada elemento desta lista ocupa 4 bits.

Nesta opção o procedimento é o seguinte:

É calculado o setor onde está a posição de memória determinada; é consultada a Lista de Posição de Setor. Se o resultado desta consulta é diferente de zero, indicando que o setor já se encontra na memória na página determinada por este número, é calculada simplesmente a posição real desta palavra MIX na memória MITRA. (Este cálculo é feito observando-se a distância da posição ao início do setor e situando esta distân

cia na página indicada). Se o resultado da consulta é igual a zero isto indica que o setor não está ainda na memória oferecendo duas opções:

- Há página disponível para inserir um novo setor.

  Neste caso a Fila de Setores recebe o acréscimo deste setor, a Lista de Posição do Setor é atualizada colocando-se a página correspondente ao setor e a posição real é calculada (processo já visto no caso do setor já se achar na memória).

- Não há página disponível.

  Neste caso o primeiro setor da Fila de Setores (o mais antigo na memória) volta para o disco deixando uma página livre.

  A Fila de Setores é atualizada e recai-se no caso anterior.

IV.2.4. - Registros MIX

Como todas as palavras MIX, os registros MIX são simulados também em uma palavra longa. Aproveitando porém o fato da representação numérica MIX ser do tipo sinal e magnitude, os sinais de rA, rX,rI1, rI2, rI3, rI4, rI5 e rI6 foram armazenados separadamente contendo a palavra longa apenas o valor absoluto do registro. Esta representação visa maior facilidade nas operações aritméticas. (ver IV.4.3)

IV.3. - Execução de um programa em linguagem de máquina MIX

É usado no decorrer da execução um apontador de memória chamado "contador de posição" que indica a próxima instrução a ser executada. Este apontador é inicializado pela rotina de carga com o valor fornecido pelo Montador juntamente com a opção do tratamento de memória (rápido ou lento) que é necessária na execução do programa MIX; todas as referências à posições de memória são testadas de acordo com a opção. A simulação é então iniciada sob o controle da chave de execução, desligada ao ser interpretado um comando HLT ou descoberto um erro, com a execução de ciclos, cada ciclo correspondendo à simulação de uma instrução sendo executada, com os seguintes passos:

- incrementar o contador de posição
- testar a variável de controle da impressão do trace.
- incrementar o relógio. (Neste caso de 1 unidade de tempo, definida no MIX como 10 microsegundos, aproximadamente). Em virtude dos incrementos de tempo das operações de entrada e saída serem muito grandes o relógio foi simulado utilizando 30 bits. Assim, a cada incremento feito na palavra que armazena a parte menos significativa do relógio deve ser verificado o transbordo, caso em que é utilizada a segunda palavra.
- retirar o código de operação (parte C)
- retirar o endereço M (parte A)
- retirar a parte I (se $1 \leq i \leq 6$ o endereço M é incrementado do conteúdo de rIi)

- retirar a parte F, estabelecendo L e R.
- execução da instrução, segundo o código de operação (visto detalhadamente mais adiante)
- testar se é momento de ocorrência de evento em operação de entrada e saída. Os tempos em que as operações de entrada e saída devem ser realizadas estão armazenadas na Tabela de Eventos.

  O momento da primeira operação da Lista Ordenada de Unidades é comparado com o valor do relógio; se for o caso a rotina conveniente é chamada e mais um passo da operação de entrada e saída é realizado. (estes passos serão vistos nas operações de entrada e saída propriamente ditas).
- Fim do ciclo.

IV.4. - Simulação da execução de uma instrução

IV.4.1. - Instruções de carga

Para a execução das intruções de carga (LDA,LDX, LDi, $1 \leq i \leq 6$, LDAN, LDXN, LDiN, $1 \leq i \leq 6$) primeiramente a posição de memória indicada por M é buscada e seu sinal é separado, isto é, se passa a ter o valor absoluto da posição de memória e seu sinal. Esta medida é conveniente uma vez que o sinal ocupa 2 bits da palavra não podendo, por isto, ser tratado da mesma forma que os outros bytes MIX (6 bits). Por este motivo os registros (ver IV.1.4) possuem valor absoluto e sinais separados. O campo máximo considerado será então (1:5) que corresponde ao valor absoluto da posição de memória; no caso

do campo especificado possuir L = 0 então o sinal será anexado ao final do tratamento. Após esta operação o campo especificado por L e R é separado na posição de memória da seguinte forma: (Esta rotina será usada em outras instruções: intruções de soma e subtração, multiplicação, divisão e instruções de comparação) o byte R é encostado à direita da variável de trabalho eliminando-se os bytes à direita de R que não são necessários e ao mesmo tempo já preparando a posição correta para carregar o registro (o campo especificado deve ser carregado na extrema direita do registro); a variável de trabalho, em seguida, é mascarada (existem cinco máscaras, relativas, cada uma, ao número de bytes que devem ser deixados à mostra).

O campo está pronto na variável de trabalho e é então carregado no registro conveniente. No caso das instruções LDi e LDiN é verificado se a operação de carga envolve mais de dois bytes, caso em que o excesso é zerado. Se o campo especificado contém o sinal, primeiramente é verificada a instrução: nos casos de carga negativa o sinal deve ser invertido, depois então o sinal é atribuído ao sinal do registro;em caso contrário é atribuído sinal positivo.

IV.4.2. - Instruções de armazenamento

IV.4.2.1. - Armazenamento de registros

Pode-se dividir a execução das instruções de armazenamento de registros (STA,ST1,ST2,ST3,ST4,ST5,ST6,STX, STJ) basicamente em tres partes: a preparação do registro a

ser armazenado, a preparação da posição de memória que receberá o registro e a reunião dos dois.

- preparação do registro a ser armazenado: primeiramente é verificado se o armazenamento inclui o byte do sinal; neste caso o tratamento é idêntico ao utilizado nas instruções de carga: o sinal é separado e se passa a trabalhar com L = 1. Apesar do registro não utilizar o byte do sinal a medida é necessária para o cálculo do número de bytes a ser armazenado, o que é feito a seguir. Este número de bytes deve ser retirado da extrema direita do registro indicado, o que é feito mascarando-se os bytes que não serão utilizados (aqueles que não serão armazenados).

- preparação da posição de memória: a idéia fundamental é preparar um "vazio" na posição de memória (este "vazio" possui o nº de bytes calculado acima) a ser preenchido pelo registro já preparado. Isto é feito encostando-se o byte R na extrema direita, mascarando-se então os bytes que serão substituídos (a máscara utilizada é a negação lógica da máscara utilizada no registro). Esta movimentação da posição de memória é necessária para que as máscaras utilizadas e a reunião do registro e da posição de memória variem apenas em função do número de bytes que devem ser armazenados. Por exemplo: o armazenamento no campo (4:5) e no campo (2:3) utiliza a mesma máscara, variando somente a movimentação dos bytes.

- reunião do registro a ser armazenado e da posição de memória: após a preparação os dois são reunidos através da operação

lógica OU, retornando à posição correta dos bytes. O sinal em seguida é verificado uma vez que deverá, se for o caso, ser introduzido na posição de memória.

IV.4.2.2. - Armazenamento do valor zero

A instrução STZ é executada utilizando a rotina de armazenamento em registros, simplesmente considerando-se o registro a ser armazenado como um registro nulo, de sinal positivo.

IV.4.3. - Instruções aritméticas

IV.4.3.1. - ADD e SUB

As instruções de soma e subtração passam pelo mesmo processo de simulação, uma rotina que executa a soma de dois valores em notação MIX de mesmo sinal ou sinais diferentes, armazenados em palavras longas MITRA representadas por PAL1.PAL2 e PAL3.PAL4. Esta rotina exige uma preparação prévia dos valores: os sinais são retirados e armazenados separadamente, restando em PAL1.PAL2 e PAL3.PAL4 os valores absolutos ocupando 30 bits da palavra longa. Como no MITRA a adição é realizada entre o acumulador e uma palavra de memória (16 bits) PAL1.PAL2 e PAL3.PAL4 deverão ser adicionadas palavra a palavra o que torna necessário uma redistribuição dos 30 bits, de forma a PAL1 possuir 15 bits e PAL2 os outros 15 bits, garantindo palavras positivas. (PAL3 e PAL4 passam pelo mesmo processo).

A operação adição é então executada:

Sinais iguais:

- A variável VAIUM é zerada.
- PAL2 e PAL4 são adicionados. Se o resultado é negativo (isto é, ocupar 16 bits) o bit mais significativo é zerado e VAIUM é incrementado de 1.
- PAL1, PAL3 e VAIUM são adicionados. Se o resultado é negativo o bit mais significativo é zerado (o que significa ser abandonado) e OVF é ligado.
- PAL3.PAL4 que armazena o resultado volta a ser condificada da forma já descrita para a representação das palavras MIX e, como estamos tratando da instrução de soma e subtração, este resultado é transferido para o registro A. O sinal do resultado é o sinal comum.

Sinais diferentes:

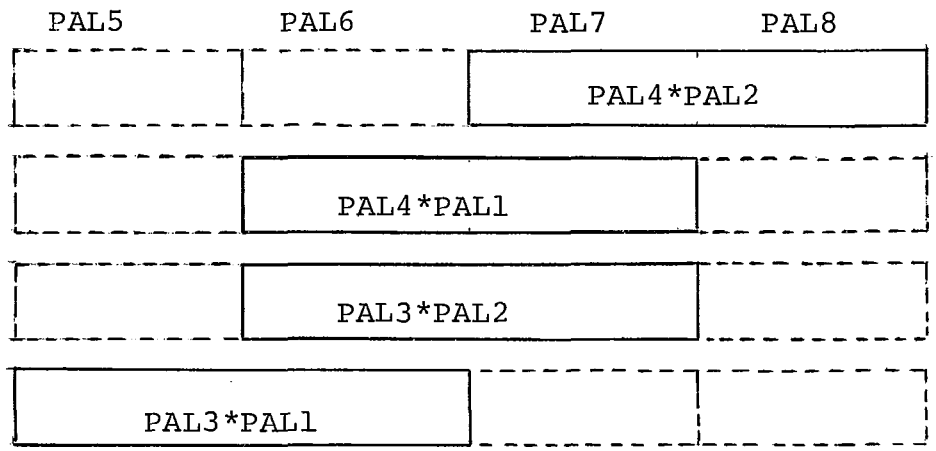
Neste caso deve ser calculada a diferença entre PAL1.PAL2 e PAL3.PAL4. O maior valor absoluto é primeiramente determinado e colocado em PAL1.PAL2.

- PAL2 deve ser maior que PAL4. Caso não seja, PAL2 será adicionado a VEMUM (o conteúdo de VEMUM é 2^{15} o que acarreta a retirada de uma unidade de PAL1)
- as diferenças são calculadas.
- o sinal do resultado é o sinal do maior valor absoluto (sinal correspondente a PAL1.PAL2). O processo de armazenamen-

to já foi visto no caso de sinais iguais.

IV.4.3.2. - MUL

A instrução de multiplicação é simulada por uma rotina que executa o produto de dois valores em notação MIX de mesmo sinal ou sinais deferentes armazenados em PAL1.PAL2 e PAL3.PAL4. Como no MITRA a multiplicação é realizada entre o acumulador e uma palavra de memória e o resulta do é armazenado no registro E e no acumulador, PAL1.PAL2 e PAL3. PAL4 passam pela mesma preparação já vista na rotina de adição, uma vez que também o produto será realizado palavra a palavra. O resultado final estará armazenado em quatro variáveis de trabalho PAL5, PAL6, PAL7 e PAL8 onde serão acumulados sucessivamente os resultados dos produtos parciais, obedecendo à seguinte distribuição:



Para cada produto parcial executado (PAL4*PAL2, PAL4*PAL1, etc) é necessário refazer a distribuição do resulta

do em 15 bits para cada palavra, uma vez que, a adição dos resultados parciais será feita também palavra a palavra (no caso o resultado final ocupa quatro palavras MITRA). PAL5 e PAL6 são atribuídos a rA, PAL7 e PAL8 é atribuído a rX e a notação MIX é empregada novamente.

IV.4.3.3. - DIV

A rotina de divisão consta dos seguintes passos, que serão mostrados com o auxílio de variáveis de trabalho PAL1, PAL2, PAL3, PAL4 (onde estão rA e rX), SOR1 e SOR2 (posição de memória que é divisor), QUO1, QUO2 (armazamento do quociente) e RESTO.

- Preparação de PAL1, PAL2, PAL3, PAL4, SOR1 e SOR2 de forma a trabalhar com 15 bits (processo já visto anteriormente).

1º caso : SOR1 $\neq$ 0

- PAL1.PAL2 é dividido por SOR1 com resultado QUO1 e RESTO (transbordo é observado)
- PAL2.PAL3 é substituido por RESTO.PAL3 - QUO1*SOR2. Note-se que esta subtração é feita palavra a palavra e só será satisfatória quando os dois resultados forem positivos isto é obtido para a palavra de peso mais fraco subtraindo-se unidades de RESTO e para a palavra de peso forte diminuindo-se QUO1 enquanto for necessário.
- O processo se repete para PAL2.PAL3 sendo dividido por SOR1

com resultado QUO2 e RESTO.

(PAL3.PAL4 é substituído por RESTO.PAL4 - QUO2*SOR2)

2º caso : SOR1 = 0

- Se PAL1 = 0 (caso contrário pode-se afirmar de imediato que aconteceu transbordo) tem-se PAL2.PAL3 dividido por SOR2 com resultado em QUO1 (transbordo também observado)
- Resto armazenado de imediato em PAL3.
- O processo se repete para PAL3.PAL4 com resultado em QUO2.

Os passos seguintes são comuns:

- rA recebe QUO1.QUO2 (retomando a notação MIX)
- rX recebe PAL3.PAL4 (retomando a notação MIX)

IV.4.4 - Instruções imediatas

Uma vez que nas instruções imediatas o endereço M é utilizado como um número e não como um endereço, primeiramente este valor é preparado, separando-se seu sinal. A parte F(especificação de campo, que não existe no caso) determina a qual dos quatro grupos a instrução pertence:

- instrução de carga de registro imediata (ENTA,ENT1,ENT2,ENT3, ENT4, ENT5, ENT6, ENTX)
- instrução de carga negativa de registro imediata (ENNA,ENN1, ENN2, ENN3, ENN4, ENN5, ENN6, ENNX)

- instrução de incremento de registro (INCA,INC1,INC2,INC3, INC4,INC5,INC6,INCX)
- instrução de decremento de registro (DECA,DEC1,DEC2,DEC3, DEC4, DEC5,DEC6, DECX)

Nos primeiros casos o sinal de M é tratado primeiramente (no caso de carga negativa é invertido); o valor de M é atribuido então ao registro conveniente. Se M é igual a zero, o sinal da instrução é carregado.

Nos casos que envolvem incremento e decremento dos registros se torna necessária a utilização da rotina de soma (já utilizada em IV.2.5.1 - instruções de soma e subtração) para realizar a operação entre o registro referenciado e o valor de M. O resultado da operação é armazenado no registro conveniente. Mas instruções relativas a rIi($i \leq i \leq 6$) é necessário verificar se o resultado não ultrapassa os dois bytes do registro, caso em que o transbordo será anulado.

IV.4.5. - Instruções de comparação

As instruções de comparação (CMPA,CMP1,CMP2, CMP3,CMP4,CMP5,CMP6 ,CMPX) comparam o campo especificado do registro com o mesmo campo da posição de memória. Sendo assim a primeira providência na simulação da instrução é separar o campo especificado do registro referenciado (que é colocado em uma variável de trabalho longa PAL1.PAL2) e fazer o mesmo para a posição de memória (que é armazenada em PAL3.PAL4). PAL1.PAL2 e PAL3.PAL4 contem valores absolutos; os sinais respectivos se

encontram armazenados em S1 e S2.

A comparação é feita para os valores absolutos: PAL1 é comparado a PAL3 : se maior ou menor CI é ligado em maior ou menor; se igual, PAL2 e PAL4 passam a ser comparados da mesma forma. Os sinais então são estudados, caso o byte de sinal esteja incluído no campo especificado. Se os sinais são iguais positivos, nada acontece; iguais negativos CI é invertido. Se os sinais são diferentes e o sinal do registro é positivo CI passa a indicar maior; se o sinal do registro é negativo passa a indicar menor.

IV.4.6 - Instruções de desvio

A simulação destas instruções é reduzida a dez alternativas de ações que decidem se será ou não realizada a rotina de desvio.

Considera-se a rotina de desvio o armazenamento da contador de instrução em rJ (corresponde à instrução que seria a seguinte da instrução que está sendo executada) e a substituição do contador de instrução pelo valor de M (que indica a próxima instrução a ser executada).

São vistos a seguir os dez grupos e as respectivas instruções à que correspondem, sendo que para os casos e, f, g, h, i, j foi criada uma variável ESTADO que recebe CI (que é codificado para maior valendo 1, igual valendo 0 e menor valendo -1) ou o sinal do registro indicado (positivo 1,negativo -1) segundo a operação a ser executada. No caso do re-

gistro ser nulo, ESTADO recebe zero.

a) JMP - rotina de desvio sempre executada

b) JSJ - a rotina de desvio não é completamente executada: rJ não sofre alteração.

c) JOV - Se OVF está ligado ele é desligado e a rotina de desvio é executada; caso contrário nada acontece.

d) JNOV - Se OVF não está ligado a rotina de desvio é executada; caso contrário OVF é desligado.

e) JL,JAN,J1N,J2N,J3N,J4N,J5N,J5N,JXN
Se a variável ESTADO é negativa a rotina de desvio é executada.

f) JE,JAZ,J1Z,J2Z,J3Z,J4Z,J5Z,J6Z,JXZ
Se a variável ESTADO é nula a rotina de desvio é executada.

g) JG, JAP,J1P,J2P,J3P,J4P,J5P,J6P,JXP
Se a variável ESTADO é positiva a rotina de desvio é executada.

h) JGE,JANN,J1NN,J2NN,J3NN,J4NN,J5NN,J6NN,JXNN
Se a variável ESTADO é não negativa a rotina de desvio é executada.

i) JNE,JANZ,J1NZ,J2NZ,J3NZ,J4NZ,J5NZ,J6NZ,JXNZ
Se a variável ESTADO é não nula a rotina de desvio é executada.

j) JLE,JANP,J1NP,J2NP,J3NP,J4NP,J5NP,J6NP,J6NP,JXNP
Se a variável ESTADO é não positiva a rotina de desvio é executada.

IV.4.7. - Instruções de entrada e saída

IV.4.7.1. - Instruções IN e OUT

Uma vez que existem diversos elementos comuns as instruções IN e OUT são simuladas pela mesma rotina, que consta de:

a) É consultada a Tabela de Ocupação. Se a unidade está ocupada é simulada uma instrução "JBUS *", uma vez que é necessário aguardar o término da operação anterior da unidade. (ver IV.1)

b) Estando livre a unidade, são inicializados a Tabela de Ocupação, a Tabela de Contadores e a Tabela de Operações e Endereços.

c) É calculado o tempo gasto para iniciar uma operação de entrada e saída da unidade; este tempo é acrescido do tempo de transferência de uma palavra (a primeira, no caso), sendo então inserido na Tabela de Eventos (ver IV.1). Para cada unidade procede-se da seguinte forma:

- Fita:

  Tempo de aceleração: 3ms (apenas no início de uma operação, se for o caso)

  É interessante notar que a fita continua rodando por 66/60 ms após uma operação de entrada e saída; se uma nova instrução for dada neste intervalo o tempo de aceleração será nulo e este fato é controlado através da Tabela de Eventos.

  Tempo de transferência (de uma palavra): 5/60 ms

Este é o incremento de tempo que será acrescido ao relógio que marcar o momento do próximo evento, procedimento repetido 100 vezes.

- Leitora de cartões:

Tempo de Transferência (de uma palavra): 10240 μs
Este é o incremento de tempo que será acrescido ao relógio para marcar o momento da próxima transferência, procedimento repetido 16 vezes.

- Teletipo:

Tempo de transferência (de uma palavra) 655360 μs
Mesmo procedimento da leitora de cartões, repetido 14 vezes.

- Disco:

Tempo de "seek":o tempo requerido para que o braço de acesso se movimente do cilindro i ao cilindro j é 25+1/2|i-j|ms.

Tempo de meia rotação: 12.5 ms

Velocidade de Transmissão: 200 Kbytes/s

d) No caso da instrução IN é executada a entrada real da informação, isto é, a transferência do bloco para o buffer da unidade. No sistema simulado foi determinada a utilização para cada usuário de, no máximo, duas fitas, um disco, leitora de cartões e teletipo (com a reserva de buffers correspondentes). No momento em que uma unidade de fita ou disco é utilizada é criado um arquivo em disco para simular a utilização de uma

parte da fita (cada unidade de fita foi limitada a 40 blocos, 4000 palavras MIX, ocupando cada bloco dois setores de disco MITRA) ou uma parte do disco (neste caso o limite estabelecido foi de 40 cilindros, cada cilindro com duas trilhas, cada trilha armazenando 500 caracteres, o que corresponde a um limite de 80 blocos, isto é, 8000 palavras MIX). Cada unidade possui um buffer do tamanho de um bloco; a transferência simulada, palavra a palavra, será feita a partir deste buffer.

A rotina vista acima corresponde ao primeiro passo das instruções IN e OUT, uma vez que, em termos de simulação, aí é apenas iniciada a transferência. O prosseguimento e o término da operação se darão a intervalos constantes (o incremento de tempo relativo à cada unidade); a primeira posição da Lista Ordenada de Unidades é pesquisada após a execução de cada instrução subsequente e quando o relógio e o momento previsto para a transferência coincidem, mais um passo da instrução de entrada e saída é executado:

a) Pesquisa à Tabela de Operações e Endereços. É necessario voltar à situação original da instrução de entrada e saída.

b) É transferida mais uma palavra (do buffer para a memória no caso da instrução IN, e vice-versa no caso da instrução OUT); a Tabela de Contadores é atualizada.

c) Verifica-se o resultado da Tabela de Contadores:

   - o bloco ainda não foi inteiramente transferido: o próximo momento é calculado e inserido na Tabela de Eventos.

- o bloco foi inteiramente transferido: a posição da Tabela de Ocupação é liberada e a unidade retirada da Lista Ordenada de Unidades (o que acarreta a retirada da Tabela de Eventos e a Tabela de Operações e Endereços). No caso de fita magnética deve se estabelecer o momento de parada. Se a instrução é OUT a saída real é executada (do buffer para a unidade.

IV.4.7.2. - Instrução IOC

A simulação da instrução IOC consta de:

a) É consultada a Tabela de Ocupação. Se a unidade está ocupada é aplicado o mesmo processo que nas instruções IN e OUT.

b) É inicializada a Tabela de Ocupação.

c) Dependendo da unidade em questão são os seguintes os procedimentos:

- Fita: É verificado o valor de M e calculada a diferença entre a posição atual do cabeçote de leitura e a posição futura (DIF). O tempo necessário para a operação de controle é então o somatário de:

  Tempo de aceleração: 3 ms
  Tempo de seek: DIF x 980/60 ms
  Tempo de parada: 9 ms
  O tempo calculado é inserido na Tabela de Eventos.

- Disco: É calculada a diferença entre os cilindros. O tempo necessário para a opção de controle é calculado

então de maneira análoga ao da operação de entrada e saída.

IV.4.7.3. - JRED

A Tabela de Ocupação, no item referente à unidade, é testada. Caso a unidade esteja desocupada a rotina de desvio (ver IV.4.6) é executada.

IV.4.7.4. - JBUS

A Tabela de Ocupação, no item referente à unidade é testada. Caso a unidade esteja ocupada é verificado o endereço da instrução para onde deve ser desviada a sequência do programa. Se este endereço for diferente da instrução corrente a rotina de desvio é executada; caso contrário tem-se que o programa permaneceria efetuando desvios para a própria instrução até que a unidade estivesse desocupada. Se existirem diversos dispositivos em uso, isto realmente acontece, porém de maneira simplificada, isto é, apenas o relógio é incrementado e imediatamente é realizado o teste de entrada e saída (são omitidas todas as rotinas de partes A,C,F e I); se somente a unidade referenciada estiver na Lista Ordenada de Unidades a simplificação é maior ainda:a transferência da parte do bloco que resta é executada de uma vez (nas instruções IN e OUT), o relógio é incrementado do tempo total gasto na operação e a unidade retirada da Tabela de Eventos.

IV.4.8. - Instruções de conversão

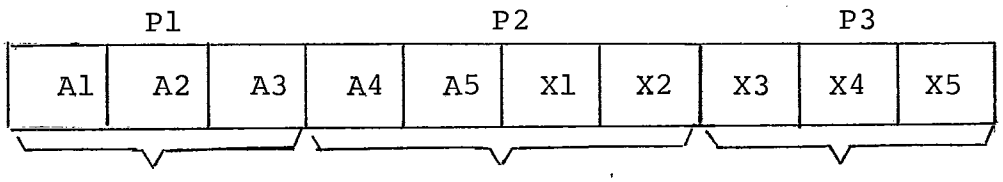
IV.4.8.1. - NUM

Os registros A e X estão em código alfanumérico MIX. A instrução MIX converte bytes contendo 00,10,20... ao dígito zero; 01,11,21... ao dígito um, e assim sucessivamente, formando um número decimal. Na simulação da instrução é usado o módulo MITRA DCBN que converte uma cadeia de algarismos decimais em EBCDIC a um número binário ocupando 16 bits.Para que o módulo possa ser usado a contento a rotina deve possuir os seguintes passos:

a) Cada byte MIX é dividido por 10 para encontrar o dígito que comporá o número decimal

b) Cada dígito é escrito em código EBCDIC

c) É obtida uma cadeia de 10 bytes MITRA; esta cadeia é dividida em tres partes P1, P2 e P3 que são convertidos a binário separadamente. A composição de P1, P2 e P3 é a seguinte:



onde A1: 1º dígito de rA escrito em EBCDIC

A2: 2º dígito de rA escrito em EBCDIC

'

e assim sucessivamente.

O número resultante será:

(P1*10000 + P2) * 1000 + P3

d) Este número é testado numa verificação de transbordo, caso em que OVF é ligado.

IV.4.8.2. - CHAR

É usado na simulação da instrução CHAR o módulo MITRA BNDC que converte um número binário de 15 bits a uma cadeia de cinco dígitos decimais em código EBCDIC; necessita-se então de uma redistribuição prévia dos 30 bits de palavra MIX em duas palavras MITRA que são tratadas separadamente, cada uma contendo 15 bits do número, o que corresponde a dividi-lo por 32768 armazenando na primeira palavra (PAL1) o quociente e na segunda (PAL2) o resto. Os passos seguidos pela rotina são os seguintes:

a) PAL2 é convertido a decimal e a resposta armazenada num vetor de trabalho de 10 bytes. As conversões futuras serão sucessivamente acumuladas neste vetor.

b) PAL1 é multiplicado sucessivamente por 8,6,7,2 e 3 sendo as respostas convertidas a decimal e acumuladas no vetor de trabalho. É interessante notar que como o resultado destes produtos pode ser maior que 32767, este deve também ser convertido em duas partes; desta vez entretanto é suficiente dividir o número por 10000 e tratar quociente e resto. Após cada multiplicação deve ser o tratamento do vetor de trabalho: verificar se cada posição deste vetor contém apenas um dígito ou não, caso em que o excesso deve ser retirado e passado ao byte seguinte sucessivamente.

c) Conversão do resultado a código MIX e armazenamento em rA e rX.

IV.4.9. - Outras instruções

IV.4.9.1. - MOVE

O número de palavras especificadas pela parte F é movido, a partir da posição especificada por M para o endereço contido em rI1. Ao ser executada a transferência de cada palavra a validade dos endereços de transferência e recepção é testada; caso haja erro a operação é interrompida. Caso a simulação da instrução termine normalmente rI1 é incrementado do valor de F.

IV.4.9.2. - NOP

Nada acontece.

IV.4.9.3. - HLT

A chave de execução de programa é desligada.

IV.4.9.4. - SHIFT

É interessante observar que as instruções SHIFT do MIX que permitem o deslocamento de qualquer número de bytes, são simuladas utilizando-se as instruções SHIFT do MITRA que permitem apenas um deslocamento de no máximo 31 bits, o que

torna necessário a existencia de testes e tratamento diferentes para instruções aparentemente de mesmo efeito.

SLA e SRA

Se o número de deslocamentos é menor que cinco, este é realizado; caso contrário rA é zerado, não alterando seu sinal.

SLAX e SRAX

Se o número de deslocamentos é menor que nove, ele é realizado (ver abaixo); caso contrário rA e rX são zerados.

O deslocamento de rA e rX é executado de formas diferentes para dois casos:

1º caso: o número de deslocamentos é menor ou igual a cinco (isto é, alguns bytes de rA estarão em rX após a execução da instrução SRAX ou vice-versa no caso de SLAX). Os passos da rotina são:

a) rA e rX sofrem separadamente o deslocamento indicado por M para a esquerda no caso de SLAX, ou para a direita no caso de SRAX.
b) É criada uma variável de trabalho que recebe rX deslocado de (5-M) bytes para a direita para a instrução SLAX (ou o simétrico para rA no caso de SRAX).

c) É feita a reunião entre rA e a variável de trabalho (caso SLAX) ou rX e a variável de trabalho (caso SRAX)

2º caso: o número de deslocamentos é maior que cinco.

a) Para SLAX: rX é deslocado (M-5) bytes; rA recebe o resultado e rX é zerado.

b) Para SRAX: rA é deslocado (M-5) bytes; rX recebe o resultado e rA é zerado.

SLC e SRC

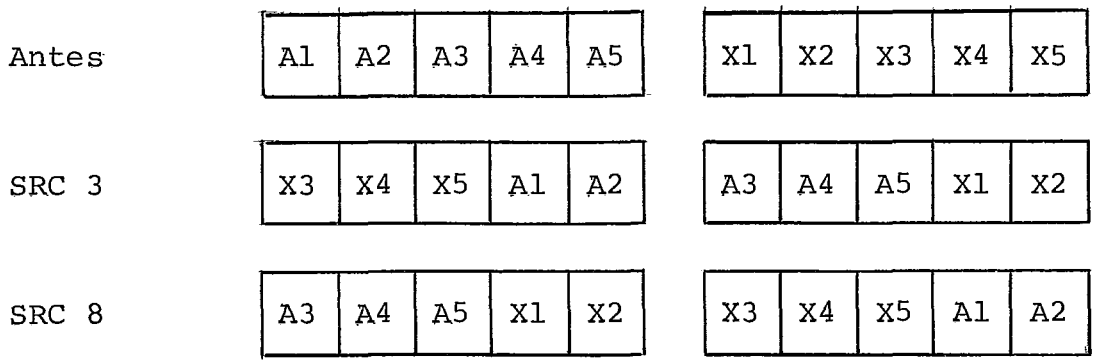
Primeiramente é pesquisado o resto da divisão de M por 10 que corresponderá ao número real de deslocamentos que deverá ser feito; se este número é maior que cinco é ligada uma chave e este excesso é retirado, recaindo-se num caso permitido pelo MITRA. Assim, por exemplo, SRC 3 e SRC 8 serão tratados da mesma forma modificando-se apenas a atribuição final. Os passos seguintes da rotina são então:

a) Para SLC: rA e rX são deslocados M bytes para a esquerda, devidamente armazenados em variáveis de trabalho; rA e rX são deslocados (5-M) bytes para a esquerda e armazenados em variáveis de trabalho. No caso de SRC o procedimento é simétrico do visto acima.

b) As variáveis de trabalho são reunidas.

c) A atribuição dos resultados é feita à rA e rX. Note-se que para o caso de M inicialmente maior que cinco esta atribuição é invertida.

Exemplo:



IV.4.9.5. - CLOC

O valor do relógio no momento é convertido a decimal e é impresso.

IV.4.9.6. - LOOP

A posição de memória indicada pelo contador de posição é incrementada de 1 num campo formado pela reunião das partes A e I. Este valor é testado para não ultrapassar o maior valor possível, quando então é zerado e recomeça a contagem.

IV.4.9.7. - OUTL

As partes A e I da posição de memória indicada por M são reunidas, formando um campo que é convertido a decimal e impresso.

IV.4.9.8. - TRCE

É ligada a variável de controle de impressão da instrução corrente a dos registros rA,rX,rI1,rI2,rI3,rI4,rI5 e rI6.

IV.4.9.10 - NTRC

É desligada a variável de controle de impressão do trace.

CAPÍTULO V

MONTADOR

V.1. - Introdução

A tradução da linguagem simbólica MIXAL para linguagem de máquina é feita utilizando-se montador de um passo. A estrutura da linguagem permite esta solução de tal forma que ao findar este passo (aparecimento do cartão END) estarão incompletas apenas as instruções que apresentaram constante literais, que serão montadas após o final físico do programa e as instruções que as referenciam completadas com estes endereços.

O montador passa por um ciclo completo para cada cartão contendo um comando MIXAL:

- leitura do comando
- impressão ou não do comando, segundo variável de controle
- inicialização de variáveis necessárias à montagem
- retirada do campo OP (mnemônico da instrução ou pseudo)
- montagem da instrução ou pseudo (diferença fornecida pelo campo OP.

A descrição mais pormenorizada do último item será feita adiante.

V.2. - Tabelas: organização e acesso

V.2.1. - Tabela de Mnemônicos

Tabela "hashing" que contém apenas os mnemônicos das das instruções e pseudo-instruções que aparecem no campo OP de um comando MIXAL e fornece o código de instrução (C) e a modificação do código da instrução (F) padrão.

O número total de mnemônicos é 159, o que leva a optar por uma tabela com 256 posições, que é a menor potência de 2 que contém 159. Escolheu-se esta forma de modo a facilitar a operação módulo, que torna a tabela circular.

Cada mnemônico tem no máximo 4 letras ocupando, portanto, 4 bytes. Utilizou-se para transformação da chave em endereço a seguinte função:

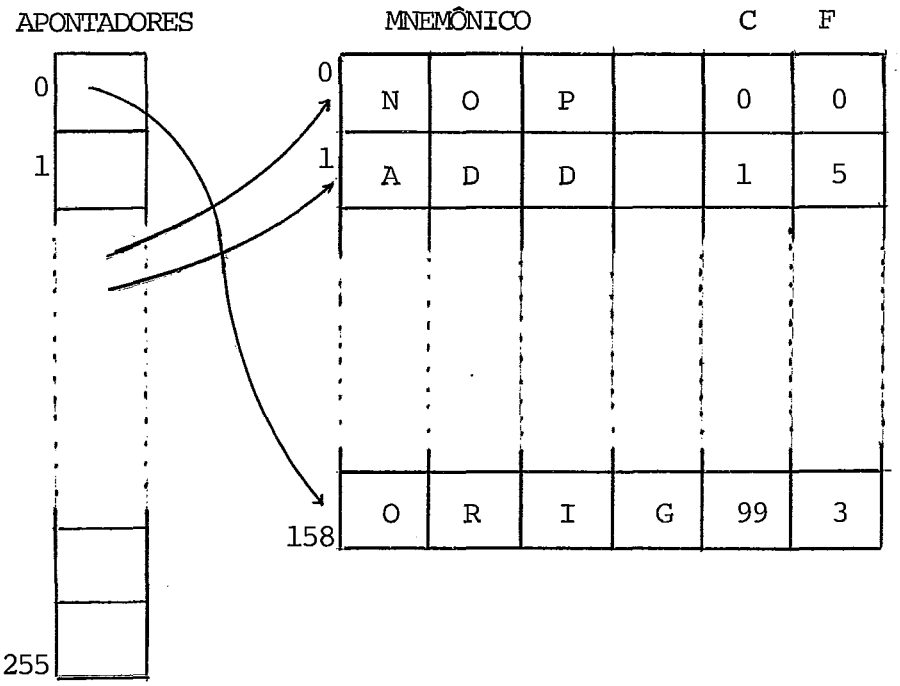
dado K_j = BT1.BT2.BT3.BT4 onde BT1 representa o primeiro caracter do mnemônico, BT2 o segundo e assim sucessivamente tem-se:

$$h_i(K_j) = [(BT1+BT3) \oplus (BT2+BT4) + i^2] \text{ MOD } 256$$

Esta função efetua o dobramento da chave e fornece o endereço em módulo 256. Como a Tabela de Mnemônicos é estática, chegou-se a esta função através de pesquisa entre funções de igual simplicidade e rapidez de cálculo, feita por programa auxiliar. Escolheu-se a função que apresentou valores de endereços "mais uniformemente distribuidos", ou seja, com o menor número médio de colisões primárias. (Método de

Busca Quadrática)|[11]|.

A tabela possui a seguinte organização:



O uso de apontadores permite uma economia de espaço frente a tabela simples de 20%(≃ 0,4 Kbytes). Além disso permite acelerar um pouco o tempo de busca sem sucesso pois as posições não utilizadas da lista de apontadores são verificadas com facilidade.

Como a tabela é estática foi adotado o algoritmo de BRENT |[9]|, procurando-se minimizar o custo das buscas com sucesso.

A média de comparações para buscas com sucesso obtida é de 1.7, para aproximadamente 58% de taxa de ocupação. Esta média, um pouco alta frente aos valores teóricos, pode ser explicada devido ao grande número de sinônimos (chaves com

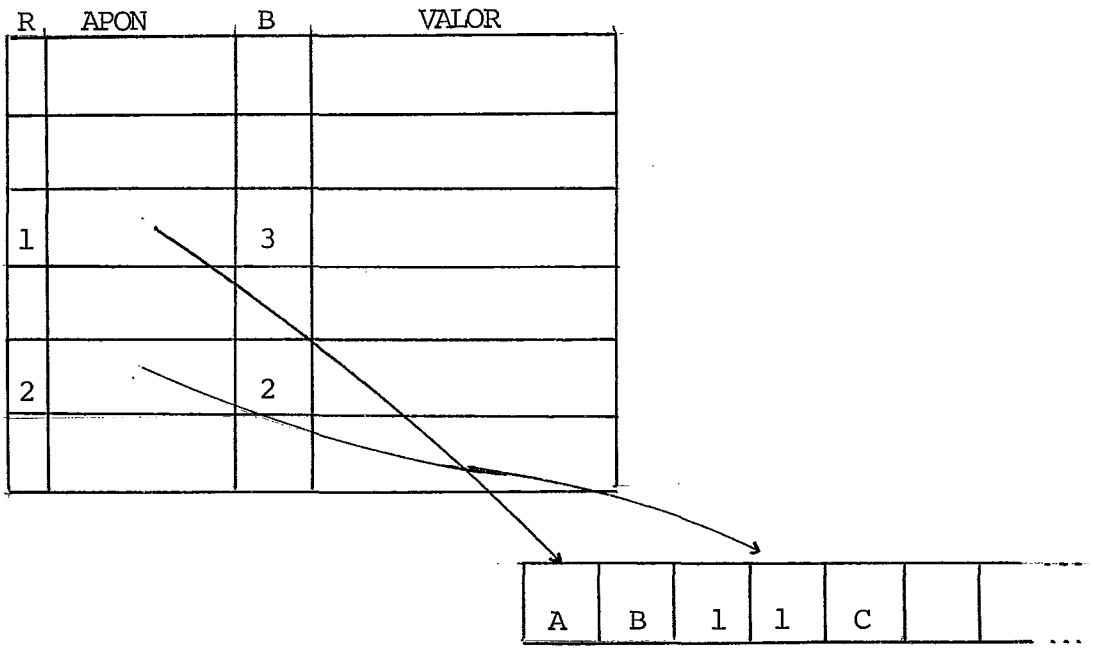
colisão primária) existentes na lista de mnemônicos. Na verdade, tem-se poucos mnemônicos básicos (LD,ST,JMP,etc), seguidos do identificador numérico do registro. Isto causa o aparecimento de "famílias de mnemônicos sinônimos", com a mesma lei de formação. Se se acrescentar a isso a dificuldade de tratar chaves em EBCDIC |[12]|, também devido a lei de formação do código, pode-se verificar que a dificuldade principal foi encontrar uma função de transformação de chaves em endereços razoável e, apesar da otimização feita pelo algoritmo de BRENT, ainda chegou-se a uma média de comparações alta, embora plenamente razoável para o fim visado.

V.2.2. - Tabela de Símbolos

A Tabela de Símbolos é utilizada para armazenar identificadores, referências futuras e constantes literais.

Trata-se de uma tabela hashing onde cada posição ocupa seis bytes e, no caso de identificadores (símbolos definidos) e referências futuras, mais tantos bytes quantos forem seus caracteres, situados em lista à parte.

Organização da Tabela:



No exemplo, AB1 é um símbolo definido e 1C é uma referencia futura.

Para símbolos definidos e referências futuras os campos R,APON,B e VALOR da Tabela apresentam o seguinte conteúdo:

R(2 bits): código indicativo do conteúdo da posição (tabela inicializada com R = 0 equivalente a posição vazia). O código pode ser:

1 - símbolo definido

2 - referência futura

APON(10 bits): apontador para o byte correspondente ao primeiro caracter do símbolo na lista de símbolos

B(4 bits): número de caracteres do símbolo (máximo 10)

VALOR(32 bits):

- se símbolo definido: valor do contador de posição no momento do aparecimento do símbolo.

- se referência futura: endereço da última instrução onde apareceu o símbolo, que por sua vez contém o endereço da penúltima instrução e assim sucessivamente.

A transformação da chave (10 caracteres) em endereço é feita através do método do Quociente Linear [10]:

dado K_j = PAL1.PAL2.PAL3.PAL4.PAL5 onde PAL1 representa os dois primeiros caracteres do símbolo, PAL2 os dois próximos, e assim sucessivamente.

$$h_i(K_j) = [((PAL1 \oplus PAL2 \oplus PAL3 \oplus PAL4 \oplus PAL5) \text{ AND } FF00) + ix((PAL1 \oplus PAL2 \oplus PAL3 \oplus PAL4 \oplus PAL5) \text{ AND } 000F)] \text{ MOD } 256$$

As constantes literais são armazenadas na Tabela de Símbolos com um tratamento semelhante ao das referências futuras uma vez que, se expandidas, corresponderiam à criação de uma. Assim:

LDA = 8 = equivale a

```
        LDA   CONST
CONST   CON   8
```

O valor da constante é calculado e é armazenado juntamente com o endereço da instrução que referencia a constante. O acesso à tabela é feito sequencialmente em posições ainda não ocupadas. Os campos R,APON,B e VALOR neste caso possuem os seguintes conteúdos:

R(2 bits) : 3 (códigos de constante literal)

APON+B(14bits) : posição de memória da instrução que utiliza a constante literal

VALOR(32 bits) : palavra MIX correspondente à constante literal

V.2.3. - Tabela H e Tabela F

Tabelas de igual organização, constituidas de 10 posições que armazenam o último endereço aonde ocorreu o símbolo local nH ou nF ($0 \leq n \leq 9$), respectivamente.

V.3. - Rotinas básicas

V.3.1. - Validade de símbolos

A verificação e distinção de tipos de símbolos é necessária em diversos pontos do montador. Esta rotina utiliza um automata finito. Para tal os caracteres foram separados em classes:

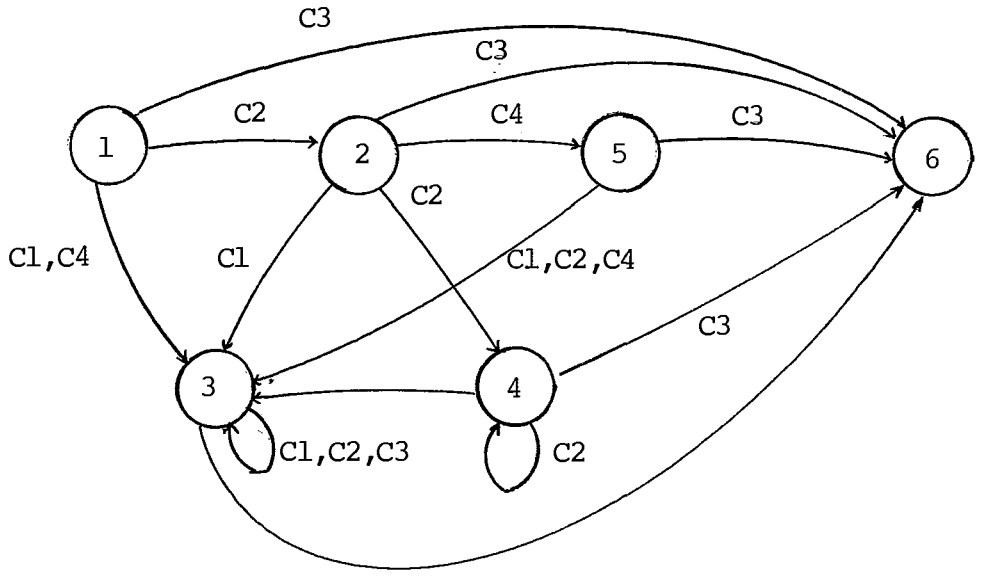
Classe 1 : A,C,D,E, G,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z

Classe 2 : 0,1,2,3,4,5,6,7,8,9

Classe 3 : +,-,*,/,:, ,(,),=, ,

Classe 4 : B,F,H.

Com estas classes foi construido o seguinte autômato:



Expresso pela Tabela:

Estados \ Classes	C1	C2	C3	C4	Observação
E1	E3	E2	E6	E3	Inicial
E2	E3	E4	E6	E5	Recebido 1º dígito
E3	E3	E3	E6	E3	Identificador
E4	E3	E4	E6	E3	Constante numérica
E5	E3	E3	E6	E3	nB,nF,nH
E6	-	-	-	-	Erro

A rotina fornece a seguinte distinção: símbolo válido, símbolo local (nB,nH ou nF separados por sub-classes), símbolo não válido e número (com seu respectivo valor decimal).

V.3.2. - Cálculo de expressões

O cálculo de uma expressão em linguagem simbólica MIX deve ser feito da esquerda para a direita, sem que seja respeitada nenhuma espécie de prioridade. Sendo assim a rotina retira o primeiro operando e executa ciclos sucessivos. Um ciclo pode ser descrito da seguinte forma:

- retirar o operador que vem a seguir, tomando o cuidado de verificar se é um operador representado em um só caracter como "+","-","*","/" e ":" ou um operador representado por dois caracteres "//" caso em que será executado um ciclo vazio aguardando o próximo caracter;
- analisar o próximo operando, que pode ser um símbolo já definido (é tomado seu valor correspondente na Tabela de Símbolos), um símbolo local nB (é consultada a Tabela H para tomar seu valor), um número ou um "*" (representando o cantodor de posição). Qualquer outra possibilidade acarreta erro.
- executar a operação, segundo os critérios já estabelecidos na simulação. As operações que não existem na linguagem de máquina (: e //) são executadas utilizando as operações que existem, uma vez que resultam da composição de instruções de máquina; sendo assim nada foi acrescentado.
- tomar o resultado desta operação como primeiro operando e recomeçar o ciclo.

V.3.3. - Valor W

Como já foi visto em II.3.2.6 um valor W é da forma: $E_1(F_1)$, $E_2(F_2)$,...,$E_n(F_n)$. A rotina que fornece a palavra MIX equivalente a um valor W percorre ciclos sucessivos, sendo o i-ésimo ciclo descrito da seguinte forma:

- cálculo da expressão $E_i (1 \leq i \leq n)$ (Ver V.3.2 - Cálculo de expressões)
- estabelecimento da especificação de campo (L:R). Se F_i não existe é assumido o campo (0:5)
- montagem na palavra MIX do campo especificado da expressão que foi calculada, mesmo que nesta montagem exista superposição em campos já definidos.
- recomeçar o ciclo.

V.4. - Instruções

Lembrando que uma instrução (assim como uma pseudo instrução) MIXAL é constituida de tres campos LOC,OP e ADDRESS e que foi o campo OP que permitiu a classificação de instrução ou pseudo, a montagem da instrução fica reduzida a elaboração de duas etapas, uma referente ao campo LOC e outra ao campo ADDRESS, este por sua vez subdividido em parte A, parte I e parte F.

Esquema de uma instrução:

LOC OP ADDRESS

ou

LOC OP A,I(F)

dando origem à instrução de máquina

±	AA	I	F	C

onde AA corresponde a parte A
I corresponde a parte I
F corresponde a parte F
C corresponde ao código do mnemônico do campo OP

a) Campo LOC: é verificada a existência deste campo, com as seguintes alternativas e ações decorrentes (caso não exista o campo, nada é feito)

a1) símbolo definido (já surgiu em uma instrução anterior no campo LOC) - erro.

a2) símbolo - é inserido na Tabela de Símbolos com código 1 equivalendo ao valor do contador de posição; passa a ser considerado símbolo definido.

a3) referencia futura (já surgiu em uma instrução anterior no campo ADDRESS) - o código desta posição na Tabela de Símbolos é modificado de 2 para 1, (símbolo definido) e todas as instruções anteriores que contém esta referência são revistas e sua parte A definitivamente formada.

a4) símbolo local nH - inserir na Tabela H, na posição relativa a n, o valor do contador de posição; é necessário

também pesquisar na Tabela F a fim de verificar se houve alguma aparição de símbolo local nF. Em caso afirmativo as instruções que apresentarem este símbolo local são revistas (estas instruções são encadeadas) e definitivamente montadas:

a5) símbolos locais nF ou nB - erro

ab) qualquer outra alternativa - erro

b) Campo ADDRESS : subdividido em:

b1) Parte A - se não existe, a parte A assume o valor zero; caso contrário o primeiro elemento da parte é analisado dando origem as seguintes ações:

- símbolo definido:neste caso seu valor associado é tomado da Tabela de Símbolos e este elemento é considerado como primeiro operando de uma expressão então calculada (ver V.3.2 - Cálculo de Expressões).

- referência futura:o endereço desta última aparição (contador de posição) é inserido na Tabela de Símbolos e o endereço da aparição anterior é colocado na parte A da instrução corrente, estabelecendo assim uma cadeia. Esta cadeia será percorrida quando o símbolo surgir num campo LOC; neste momento todas as instruções constantes da cadeia são revistas e a parte A retificada com o endereço correto. Nesta caso a parte A só pode conter a referência futura e um teste é utilizado para assegurar que esta condição seja respeitada.

- símbolo:é considerado a partir daí uma referência futura e introduzido na Tabela de Símbolos com código 2

- símbolo local nB:n é determinado e a Tabela H na posição referente a n é pesquisada, surgindo duas opções: se nH já surgiu é utilizado na instrução corrente como primeiro operando de uma expressão, calculada então; caso contrário, erro.

- símbolo local nF:n é determinado e a Tabela F é atualizada através do mesmo processo utilizado para referências futuras (na posição da tabela relativa a n é colocado o contador de posição, na instrução corrente é colocado o endereço da aparição anterior e assim sucessivamente)

- símbolo local nH:erro

- número:é considerado como primeiro operando de uma expressão, que é então calculada.

- operador + ou -:o primeiro operando é considerado zero e a rotina que calcula expressões é chamada (a primeira operação será adição ou subtração, respectivamente)

- caracter "*":o primeiro operando de uma expressão recebe o valor do contador de posição e a expressão é calculada.

- caracter "="(constante literal):os caracteres "=" que delimitam uma constante literal são ignorados e lhe é dado o tratamento de uma expressão;esta expressão

entretanto, após ser calculada, é inserida na Tabela de Símbolos (com código 3) juntamente com o endereço da instrução. As constantes literais são tratadas no final da montagem.

b2) Parte I:se não existe, a parte I assume o valor zero; caso contrário o primeiro elemento da parte é analisado dando origem às seguintes opções:

- símbolos definidos, símbolos locais nB, números, operadores + ou - e caracteres "*" são tratados da mesma forma que quando ocorrem na parte A.
- referência futura ou símbolo:erro
- símbolos locais nF ou nH:erro

b3) Parte F:se não existe, a parte F assume o valor padrão que foi obtido na Tabela de Mnemônicos juntamente com o valor da parte C (código de operação); caso contrário o procedimento é o mesmo que para a parte I.

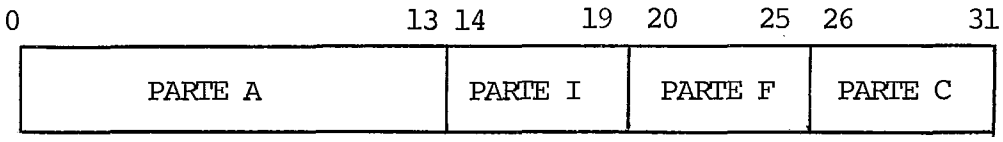
Com a parte A, parte I, parte F e parte C é montada a instrução MIX na posição de memória indicada pelo contador de posição que é incrementado, apontado para a próxima posição vazia. Note-se que esta referência à posição de memória, bem como outras feitas no decorrer deste item como, por exemplo, o caso de referências futuras no campo LOC, levam sempre em consideração a opção relativa à utilização de memória feita pelo usuário e seu tratamento correspondente (ver IV.2.3 - Memória).

Como já foi visto no capítulo referente à simu-

lação, cada posição de memória MIX ocupa uma palavra longa MITRA. A montagem da instrução leva a seguinte distribuição física:

Parte A : 14 bits (sendo 2 bits de sinal)

Parte I,F,C : 6 bits cada na seguinte ordem:



V.5. - Pseudo-instruções

As pseudo-instruções não geram instrução porém dão origem à alterações no processo de montagem. O tratamento destas alterações é o seguinte:

V.5.1. - Pseudo EQU

O campo LOC é retirado e armazenado provisoriamente, uma vez que ele não será associado ao contador de posição como é o procedimento para instruções e sim ao próprio campo ADDRESS. Isto quer dizer que todas as medidas tomadas em relação a este símbolo na instrução devem ser repetidas, porém primeiramente o valor do campo ADDRESS tem que ser estabelecido; esta é a razão do armazenamento provisório. O campo ADDRESS é um valor W (ver V.3.3). Após o cálculo deste valor o símbolo então é testado; se é um símbolo não definido ele é introduzido na Tabela de Símbolos com código 1 (símbolo defi-

nido) ; se é uma referencia futura (código 2) este código é alterado para símbolo definido e a cadeia característica das referencias futuras (já vista anteriormente) é resolvida.

V.5.2. - Pseudo ORIG

O campo LOC é tratado da mesma forma que nas instruções. O campo ADDRESS é um valor W, que é calculado (ver V.3.3) e como este valor será atribuido ao contador de posição ele deve ser testado de acordo com a opção de memória do usuário.

Se a opção é rápida: $0 \leq$ valor W ≤ 511

Se a opção é lenta: $0 \leq$ valor W ≤ 3999

isto é, o valor W deve respeitar o tamanho da memória disponível.

V.5.3. - Pseudo CON

O campo LOC é tratado da mesma forma que nas instruções. O campo ADDRESS é um valor W que é calculado (ver V.3.3), sendo então atribuido a posição de memória MIX indicada pelo contador de posição que é incrementado.

Esta referência à posição de memória obedece ao tratamento de memória determinado pela opção do usuário (ver IV.2.3).

V.5.4. - Pseudo ALF

O campo LOC é tratado da mesma forma que nas instruções. Deve ser criada, como na pseudo CON, uma palavra de memória MIX com os cinco caracteres existentes no campo ADDRESS da pseudo ALF. Estes cinco caracteres são lidos pelo MITRA em código EBCDIC cada um ocupando 8 bits; assim a primeira providência é converter cada caracter ao código MIX (conforme tabela abaixo) compactando-os depois para 6 bits. Os 30 bits obtidos são atribuidos à posição de memória MIX indicada pelo contador da posição que é incrementado.

CÓDIGO MIX:

	00	M 14	Y 28	(42
A	01	N 15	Z 29	) 43
B	02	Θ 16	O 30	+ 44
C	03	P 17	1 31	- 45
D	04	Q 18	2 32	* 46
E	05	R 19	3 33	/ 47
F	06	ϕ 20	4 34	= 48
G	07	π 21	5 35	$ 49
H	08	S 22	6 36	< 50
I	09	T 23	7 37	> 51
θ	10	U 24	8 38	@ 52
J	11	V 25	9 39	; 57
K	12	W 26	. 40	: 54
L	13	X 27	, 41	' 55

V.5.5. - Pseudo END

O campo LOC é tratado da mesma forma que nas instruções. O campo ADDRESS é um valor W que indica a posição de memória onde a execução do programa deve começar; o valor W é então calculado (ver V.3.3) e testado de acordo com a opção de memória do usuário (mesmo tratamento que na pseudo ORIG- V.5.2) após o que é armazenado provisóriamente. Esta pseudo assinala o fim da compilação. Neste momento a Tabela de Símbolos é percorrida e testados os códigos de todas as suas posições com as seguintes possibilidade de ocorrências:

- código 0,1 : nada acontece
- código 2 : erro : referência futura não satisfeita.
- código 3 : contante literal - cada constante é atribuida a uma posição de memória e a instrução que deu origem a esta constante é acessada e sua parte A revista (é colocado o endereço, agora estabelecido, da contante literal). É desligada a chave de compilação, o valor W que foi armazenado provisoriamente é atribuido ao contador de posição estando o programa pronto para a execução, ou não, de acordo com a chave de erro.

V.5.6. - Pseudo CDON

É verificada a inexistência de campo LOC e ADDRESS e é ligada uma variável de controle de impressão de código (por DEFAULT desligada) que é testada ao fim da monta-

gem de cada instrução, gerando a impressão do código caso a variável esteja ligada.

V.5.7. - Pseudo CDOF

Procedimento análogo à CDON, desligando a variável de controle.

V.5.8. - Pseudo LTON

É verificada a inexistência de campos LOC e ADDRESS e é ligada uma variável de controle de impressão de linha fonte (por DEFAULT ligada) que é testada em cada ciclo da montagem.

V.5.9. - Pseudo LTOF

Procedimento análogo à LTON, desligando a variável de controle.

V.5.10 - Pseudo SPAC

É verificada a inexistência do campo LOC. O valor W é calculado e dividido por 64, o resto da divisão é o nº de linhas em branco geradas a seguir.

Note-se que instruções referentes à depuração de programas que normalmente são tratadas como pseudo-instruções foram consideradas instruções e descritas em II.2.2.9;sua

montagem obedece então aos critérios já enunciados em V.4.

CAPÍTULO VI

CONCLUSÕES

A implementação da simulação do computador MIX no computador MITRA acreditamos que satisfaça a finalidade didática a que se propõe, uma vez que a linguagem MIXAL complementará o desenrolar de cursos importantes na área de Computação. Note-se que o fato de acrescentar algumas instruções e pseudo-instruções, é um importante auxílio na aferição dos algoritmos pois permitem o acompanhamento pormenorizado dos mesmos, além de criar um sistema adequado de verificação do tempo de execução.

A impossibilidade de manter a memória MIX completa na memória MITRA impos o estudo de uma solução que fosse ao mesmo tempo simples e eficiente, uma vez que o processo é executado pelo menos uma vez por instrução. A solução utilizada para a paginação acreditamos satisfazer tais especificações.

As diferenças apresentadas pelas duas máquinas (MITRA e MIX) obrigou a um estudo cuidadoso para o melhor uso possível dos recursos do MITRA. A nova partição da palavra longa MITRA permite seu quase total aproveitamento, embora tivessemos enfrentado grandes dificuldades, com a criação de rotinas de separação de campos, rotinas aritméticas em 30 bits, deslocamento para palavras longas e rotinas de conversão da

representação MIX.

O montador e o simulador foram programados de forma reentrante para permitir sua utilização para vários usuários simultâneos. A especificação do programa desta forma exigiu um cuidadoso trabalho de análise e estruturação.

Os testes efetuados mostram que as exigências de tempo e memória pelo uso de minicomputador foram resolvidas satisfatóriamente sendo a relação memória real/memória simulada de aproximadamente 18:15,5 e a relação tempo real/tempo simulado de aproximadamente 70:1. Estes dados são especialmente interessantes se consideramos que o MIX é muito mais potente e sofisticado que o MITRA hospedeiro. Exemplos dos testes realizados se encontram no Anexo A.

Na fase de testes observou-se ainda a complexidade de programação em MIXAL. Sugere-se então a criação de uma linguagem de médio nível que mantendo todas as características da linguagem de máquina facilite a elaboração de programas.

REFERÊNCIAS

|[1]| - Knuth, D.E. - "The Art of Computer Programming - vol. 1: Fundamental Algorithm" , Addison-Wesley, Reading, Mass., 1968.

|[2]| - Knuth, D.E. - "The Art of Computer Programming - vol.3: Sorting and Searching", Addison-Wesley, Reading, Mass., 1968.

|[3]| - Schwarz, G. - "O Laboratório de Automação e Simulação de Sistemas (LASS) Descrição e sua Atuação desde a Origem até 1978", COPPE/UFRJ, 1978.

|[4]| - MITRA-15 - "Manuel d'Utilisation:Périfhériques", Compagnie Internationale pour L'Informatique, 1973.

|[5]| - MITRA-15 - "Manuel de Présentation:Software", Compagnie Internacionale pour L'Informatique, 1972.

|[6]| - MITRA-15 - "Manuel de Présentation:Hardware", Compagnie Internacionale pour l'Informatique, 1973.

|[7]| - MITRA-15 - "Manuel d'Utilisation:Language LP15, LP15E", Compagnie Internacionale pour L'Informatique", 1975.

|[8]| - MITRA-15 - "Manuel d'Utilisation:Bibliothécaire BIB", Compagnie Internacionale pour L'Informatique, 1975.

|[9]| - Brent, R.P. - "Reducing The Retrieval Time of Scatter Storage Techniques", Comm. ACM 16(2) Fev.1973, pp.105-109.

|10| - Bell, J.R. e Kaman, C.H. - "The Linear Quocient Hash Code", Comm.ACM 13(11) Nov.1970, pp.675-677.

|11| - Maurer, W.D. - "An Improved Hash Code for Scatter Storage", Comm.ACM 11(1) Jan. 1968, pp.35-38.

|12| - Clapson, Philip - "Improving the Access Time for Random Access Files", Comm.ACM 20(3), Mar.77, pp.127-135.

ANEXO A

```
CHARGEMENT/
SYSTEM MTFIE .5 READY

%F0/@&0:
%C/MIXM/
MIXM
  4026 4042 735C
*****************************************************
*                                                   *
*                    EXEMPLO 1                      *
*                                                   *
*   ALGORITMO:                                      *
*   STRAIGHT INSERTION SORT - KNUTH, VOL. 3, PG. 81 *
*                                                   *
*   O ALGORITMO ORDENA UMA TABELA DE 100 ELEMENTOS  *
*   CERTOS DE FORMA PSEUDO-ALEATORIA                *
*                                                   *
*   FORAM INTRODUZIDAS AS INSTRUCOES CLOC, LOOP E   *
*   OUTL PARA A COMPARACAO DO ALGORITMO COM OS      *
*   APRESENTADOS NO EXEMPLO II E EXEMPLO III.       *
*                                                   *
*****J*****************************************************
          LTOF

          CLOC
INIC      ENT1 2MN              1
LOOP1     LOOP                  1
2H        LIA  INPUT+N, 1       N-1
LOOP2     LOOP                  N-1
          ENT2 N-1, 1           N-1
3H        CMPA INPUT, 2         E+N-1-A
LOOP3     LOOP                  E+N-1-A
          JGE  5F               E+N-1-A
4H        LDX  INPUT, 2         E
LOOP4     LOOP                  E
          STX  INPUT+1, 2       E
          DEC2 1                E
          J2P  3B               E
5H        STA  INPUT+1, 2       N-1
LOOP5     LOOP                  N-1
          INC1 1                N-1
          J1NP 2B               N-1
          CLOC
          OUTL LOOP1
          OUTL LOOP2
          OUTL LOOP3
          OUTL LOOP4
          OUTL LOOP5
          LTOF



PROGRAMA EXECUTAVEL
```

```
%C/MIX S/
MIX S
   40P€ 40AP €C30
INPUT
0000006078      0C000I€235      0000250588      0000064097      0000133258
0000136999      0000010312      0000006973      0000198038      0000245795
0000047092      0000086873      0000228066      0000071263      0000101856
0000040629      0000136302      0000079323      0000194060      0000211281
0000245306      0000162455      0000097400      0000237357      0000238662
0000163987      0000126244      0000101449      0000209554      0000140239
0000087056      0000130213      0000136478      0000098379      0000064828
0000207937      0000145386      0000258567      0000160936      0000038685
0000247542      0000005379      0000230996      0000194873      0000077378
0000247103      0000147008      0000020117      0000203214      0000007867
0000059500      0000250673      0000030106      0000097655      0000135384
0000131853      0000159142      0000228723      0000033668      0000039465
0000028146      0000064175      0000216176      0000169093      0000008830
0000004395      0000112540      0000011809      0000096074      0000138471
0000217352      0000189181      0000170070      0000244195      0000255156
0000093977      0000258466      0000050207      0000229024      0000249461
0000012078      0000022427      0000158412      0000212241      0000015610
0000053335      0000079160      0000145133      0000214790      0000248403
0000043492      0000030729      0000178514      0000139663      0000120016
0000195685      0000147422      0000258571      0000131580      0000000001
** TEMF0: 0019272778  U
** TEMF0: 0019296674  U
**L00P:  000001
**L00F:  000099
**L00P:  002644
**L00P:  002545
**L00F:  000099
0UTPT
0000258571      0000258567      0000258466      0000255156      0000250673
0000250588      0000249461      0000248403      0000247542      0000247103
0000245795      0000245306      0000244195      0000238662      0000237357
0000228066      0000228723      0000228066      0000217352      0000212241
0000216176      0000214790      0000212241      0000211281      0000209554
0000207937      0000203214      0000198038      0000195685      0000194873
0000194060      0000189181      0000178514      0000170070      0000169093
0000163987      0000162455      0000160936      0000159142      0000158412
0000147422      0000147008      0000145133      0000140239      0000138471
0000136999      0000136478      0000136302      0000135384      0000133258
0000131853      0000130213      0000112540      0000101856      0000101449
0000098379      0000097655      0000097400      0000096074      0000093977
0000087056      0000086873      0000079323      0000079160      0000077378
0000071263      0000064828      0000064175      0000064097      0000059500
0000053335      0000050207      0000047092      0000043492      0000040629
0000039465      0000038685      0000033668      0000030729      0000030106
0000028146      0000022427      0000020117      0000016235      0000015610
0000012078      0000011809      0000010312      0000008830      0000007867
0000006973      0000006078      0000005379      0000004395      0000000001
** TEMF0: 0038565436  U
```

```
CHARGEMENT/
SYSTEM MTPDE .5 READY

%F0/@&0)
%C/MIXM/
MIXM
  4096 40A2 735C
*****************************************************************
*                                                               *
*                       EXEMPLO II                              *
*                                                               *
*    ALGORITMO:                                                 *
*    BUBBLE SORT - KNUTH, VOL. 3, PG. 107                       *
*                                                               *
*    O ALGORITMO ORDENA UMA TABELA DE 100 ELEMENTOS             *
*    GERADOS DE FORMA PSEUDO-ALEATORIA                          *
*                                                               *
*    FORAM INTRODUZIDAS AS INSTRUCOES CLOC, LOOP E              *
*    OUTL PARA A COMPARACAO DO ALGORITMO COM OS                 *
*    APRESENTADOS NO EXEMPLO I E EXEMPLO III.                   *
*                                                               *
*****************************************************************
            LTOF


            CLOC
INIC        ENTI N              1
LOOP1       LOOP                1
1H          STI  BOUND(1:2)     A
LOOP2       LOOP                A
            ENT2 1              A
            ENT1 0              A
            JMP  BOUND          A
3H          LDA  INPUT,2        C
LOOP3       LOOP                C
            CMPA INPUT+1,2      C
            JLE  2F             C
            LDX  INPUT+1,2      B
LOOP4       LOOP                B
            STX  INPUT,2        B
            STA  INPUT+1,2      B
            ENT1 0,2            B
2H          INC2 1              C
LOOP5       LOOP                C
BOUND       ENTX -*,2           A+C
LOOP6       LOOP                A+C
            JXN  3B             A+C
LOOP7       LOOP                A
4H          J1P  1B             A
            CLOC
            OUTL LOOP1
            OUTL LOOP2
            OUTL LOOP3
            OUTL LOOP4
            OUTL LOOP5
            OUTL LOOP6
            OUTL LOOP7
            LTOF

PROGRAMA EXECUTAVEL
```

```
%C/MIX S/
MIX S
  4026 40AA 6C3C
INPUT
0000006078      0000016235      0000250588      0000064097      0000133258
0000136999      0000010312      0000006973      0000198038      0000245795
0000047092      0000068673      0000228066      0000071263      0000101856
0000040629      0000136302      0000079323      0000194060      0000211281
0000245306      0000162455      0000097400      0000237357      0000238662
0000163987      0000126244      0000101449      0000209554      0000140239
0000087056      0000130213      0000136478      0000098379      0000064828
0000207937      0000145386      0000258567      0000160936      0000038685
0000247542      0000005379      0000230996      0000194873      0000077378
0000247103      0000147008      0000020117      0000203214      0000007867
0000059500      0000250673      0000030106      0000097655      0000135384
0000131853      0000159142      0000228723      0000033668      0000039465
0000028146      0000064175      0000216176      0000169093      0000008830
0000004395      0000112540      0000011809      0000096074      0000138471
0000217352      0000189181      0000170070      0000244195      0000255156
0000093977      0000258466      0000050207      0000229024      0000249461
0000012078      0000022427      0000158412      0000212241      0000015610
0000053335      0000079160      0000145133      0000214790      0000248403
0000043492      0000030729      0000178514      0000139663      0000120016
0000195685      0000147422      0000258571      0000131580      0000000001
** TEMP0: 0019272778  U
** TEMP0: 0019330946  U
**L00P:  000001
**L00P:  000097
**L00P:  004947
**L00P:  002545
**L00P:  004947
**L00P:  005044
**L00P:  000097
OUTPT
0000258571      0000258567      0000258466      0000255156      0000250673
0000250588      0000249461      0000248403      0000247542      0000247103
0000245795      0000245306      0000244195      0000238662      0000237357
0000230996      0000229024      0000228723      0000228066      0000217352
0000216176      0000214790      0000212241      0000211281      0000209554
0000207937      0000203214      0000198038      0000194873      0000194060
0000189181      0000178514      0000170070      0000169093      0000163987
0000162455      0000147008      0000145386      0000145133      0000140239
0000139663      0000138471      0000136999      0000136478      0000136302
0000135384      0000133258      0000131853      0000131580      0000130213
0000126244      0000120016      0000112540      0000101856      0000101449
0000098379      0000097655      0000097400      0000096074      0000093977
0000087056      0000079160      0000079323      0000068673      0000077378
0000071263      0000064828      0000064175      0000064097      0000059500
0000053335      0000050207      0000047092      0000043492      0000040629
0000039465      0000038685      0000033668      0000030729      0000025106
0000028146      0000022427      0000020117      0000016235      0000015610
0000012078      0000011809      0000010312      0000008830      0000007867
0000006973      0000006078      0000005379      0000004395      0000000001
** TEMP0: 0038599710  U
```

```
CHARGEMENT/
SYSTEM MTRDF .5 READY

%F0/@&0J
%C/MIXM/
MIXM
  4026 40A2 735C
**************************************************************
*                                                            *
*                      EXEMPLØ III                           *
*                                                            *
*    ALGØFITMØ:                                              *
*    STRAIGHT SELECTIØN SØRT - KNUTH, VØL. 3, PG. 140        *
*                                                            *
*    Ø ALGØFITMØ ØRDENA UMA TABELA DE 100 ELEMENTØS          *
*    GERADØS DE FØRMA PSEUDØ-ALEATØRIA                       *
*                                                            *
*    FØRAM INTRØDUZIDAS AS INSTRUCØES CLØC, LØØP E           *
*    ØUTL PARA A COMPARACAØ DØ ALGØFITMØ COM ØS              *
*    APRESENTADØS NØ EXEMPLØ I E EXEMPLØ II                  *
*                                                            *
**************************************************************
            LTØF


            CLØC
INIC        ENT1 N-1            1
LØØP1       LØØF                1
2H          ENT2 0,1            N-1
LØØP2       LØØF                N-1
            ENT3 1,1            N-1
            LDA  INPUT,3        N-1
8H          CMPA INPUT,2        A
LØØP3       LØØF                A
            JGE  *+3            A
            ENT3 0,2            B
LØØP4       LØØF                B
            LDA  INPUT,3        B
            DEC2 1              A
LØØP5       LØØF                A
            J2P  8B             A
            LDX  INPUT+1,1      N-1
LØØP6       LØØF                N-1
            STX  INPUT,3        N-1
            STA  INPUT+1,1      N-1
            DEC1 1              N-1
            J1P  2B             N-1
            CLØC
            ØUTL LØØP1
            ØUTL LØØP2
            ØUTL LØØP3
            ØUTL LØØP4
            ØUTL LØØP5
            ØUTL LØØP6
            LTØF

PRØGRAMA EXECUTAVEL
```

```
%C/MIX S/
MIX S
   4026 40AA 6C30
INPUT
0000006078      0000016235      0000250588      0000064097      0000133258
0000136999      0000010312      0000069773      0000198038      0000245795
0000047092      0000086873      0000228066      0000712663      0000101856
0000040629      0000136302      0000079323      0000194060      0000211281
0000245306      0000162455      0000097400      0000237357      0000238662
0000163987      0000126244      0000101449      0000209554      0000140239
0000087056      0000130213      0000136478      0000098379      0000064828
0000207937      0000145386      0000258567      0000160936      0000038685
0000247542      0000005379      0000230996      0000194873      0000077378
0000247103      0000147008      0000020117      0000203214      0000007867
0000059500      0000250673      0000030106      0000097655      0000135384
0000131853      0000159142      0000228723      0000033668      0000039465
0000028146      0000064175      0000216176      0000169093      0000008830
0000043955      0000112540      0000011809      0000096074      0000138471
0000173952      0000189181      0000170070      0000244195      0000255156
0000939777      0000258466      0000050207      0000229024      0000249461
0000012078      0000022427      0000158412      0000212241      0000015610
0000053335      0000079160      0000145133      0000214790      0000248403
0000042492      0000030729      0000178514      0000139663      0000120016
0000195685      0000147422      0000258571      0000131580      0000000001
** TEMP0: 0019272778  U
** TEMP0: 0019308921  U
**L00P:  000001
**L00P:  000099
**L00F:  0049 50
**L00P:  000304
**L00F:  0049 50
**L00P:  000099
OUTPT
0000258571      0000258567      0000258466      0000255156      0000250673
0000250588      0000249461      0000248403      0000247542      0000247103
0000245795      0000245306      0000244195      0000238662      0000237357
0000230996      0000229024      0000228723      0000228066      0000217352
0000216176      0000214790      0000212241      0000211281      0000209554
0000207937      0000203214      0000198038      0000195685      0000194873
0000194060      0000189181      0000178514      0000170070      0000169093
0000163987      0000162455      0000160936      0000159142      0000158412
0000147422      0000147008      0000145386      0000145133      0000140239
0000139663      0000138471      0000136999      0000136478      0000136302
0000135384      0000133258      0000131853      0000131580      0000130213
0000126244      0000120016      0000112540      0000101856      0000101449
0000098379      0000097655      0000097400      0000096074      0000093977
0000087056      0000086873      0000079323      0000079160      0000077378
0000071263      0000064828      0000064175      0000064097      0000059500
0000053335      0000050207      0000047092      0000043492      0000040629
0000039465      0000038685      0000033668      0000030729      0000030106
0000028146      0000022427      0000020117      0000016235      0000015610
0000012078      0000011809      0000010312      0000008830      0000007867
0000006973      0000006078      0000005379      0000004395      0000000001
** TEMP0: 0038577685  U
```