4 MÁQUINAS DE REGISTRADORES NORMA

4.1 Codificação de Conjuntos Estruturados

4.2 Definição da Máquina Norma

4.3 Máquina Norma como Máquina Universal

4.3.1 Operações e Testes

4.3.2 Valores Numéricos

4.3.3 Dados Estruturados

4.3.4 Endereçamento Indireto e Recursão

4.3.5 Cadeias de Caracteres

4.4 Conclusões

4.5 Exercícios

4 MÁQUINAS DE REGISTRADORES - NORMA

Algoritmo

- termo intuitivamente usado como solução de um problema,
- uma forma de descrever se determinada propriedade pode ser verificada ou não para uma dada classe de entrada.

Investigação da solucionabilidade de um problema

- é a investigação da existência de um algoritmo capaz de resolvê-lo.

Noção intuitiva de algoritmo

- sua descrição deve ser finita e não-ambígua;
- deve consistir de passos discretos, executáveis mecanicamente e em um tempo finito.
- limitações de **tempo** ou de **espaço não são restrições teóricas**, mas podem determinar se um algoritmo pode ou não ser usado na prática.
- estudo será restrito aos **algoritmos naturais**, ou seja, definidos sobre o conjunto dos números naturais.
- qualquer conjunto contável pode ser equivalente ao conjunto dos naturais, através de uma codificação.
- conceito de programa, como introduzido anteriormente, satisfaz à noção intuitiva de algoritmo.

Máquina

- ***simples*,** para permitir estudos de propriedades, sem a necessidade de considerar características não-relevantes, bem como permitir estabelecer conclusões gerais sobre a classe de funções computáveis;

- ***poderosa*,** capaz de simular qualquer característica de máquinas reais ou teóricas, de tal forma que os resultados provados sejam válidos para modelos aparentemente com mais recursos.

Máquina Universal

Se for possível representar qualquer algoritmo como um programa em tal máquina, então esta é denominada de máquina universal.

As evidências de que uma máquina é universal:

- ***Evidência Interna.*** Consiste na demonstração de que qualquer extensão das capacidades da máquina universal proposta computa, no máximo, a mesma classe de funções, ou seja, não aumenta o seu poder computacional;
- ***Evidência Externa.*** Consiste no exame de outros modelos que definem a noção de algoritmo, juntamente com a prova de que são, no máximo, computacionalmente equivalentes.

Máquina Norma, *uma máquina universal*

- um conjunto de registradores naturais
- três instruções sobre os registradores:
  - *operação de incrementar um - sucessor*
  - *operação de decrementar um - antecessor*
  - *teste se o valor armazenado é zero.*

4.1 Codificação de conjuntos estruturados

- ***codificação de conjuntos estruturados***, onde elementos de tipos de dados estruturados são representados como números naturais.

  Para um dado conjunto estruturado X, define-se uma função injetora c: $X \to N$, onde, para todo x, y $\in$ X, tem-se que:

  se c(x) = c(y), então x = y

  o número natural c(x) é a codificação do elemento estruturado x.

Exemplo 4.1 Número de Gödel - codificação em n-uplas

Codificar de forma unívoca, elementos de N^n como números naturais, ou seja, deseja-se uma função injetora

$c: N^n \to N$. Uma codificação simples é a seguinte:

a) lembre-se de que, pelo *Teorema Fundamental da Aritmética*, cada número natural é univocamente decomposto em seus fatores primos;

b) suponha os n primeiros números primos denotados por:

$p_1 = 2$, $p_2 = 3$, $p_3 = 5$ e assim sucessivamente.

A codificação unívoca $c: N^n \to N$ é definida como abaixo, supondo $(x_1, x_2, \ldots, x_n)$ em N^n e o símbolo $\bullet$ denotando a operação de multiplicação nos naturais):

$$c(x_1, x_2, \ldots, x_n) = p_1^{x_1} \bullet p_2^{x_2} \bullet \ldots \bullet p_n^{x_n}$$

Deve-se reparar que essa codificação (Número de Gödel) não constitui uma função bijetora, ou seja, nem todo número natural corresponde a uma n-upla.

Entretanto, todo número naturais é decomponível em n primeiros números primos correspondendo a uma n-upla.

Exemplo 4.2
Codificação de programas monolíticos.

- Um programa monolítico pode ser codificado como um número natural.
- Suponha um programa monolítico $P = (I, r)$ com m instruções rotuladas onde $\{F1, F2, \ldots, Ff\}$ e $\{T1, T2, \ldots, Tt\}$ são os correspondentes conjuntos de identificadores de operações e testes, respectivamente.
- Seja $P' = (I, 1)$ como P, exceto pelos rótulos, os quais são renomeados como números naturais, onde 1 é o rótulo inicial, e o 0 único rótulo final (se existir).
- Assim, uma instrução rotulada pode ser de uma das duas seguintes formas:

  a) *Operação*. r_1: **faça** F_k vá_para $\mathbf{r_2}$

  b) *Teste*.

  r_1:**se** T_k então vá_para $\mathbf{r_2}$ senão vá_para $\mathbf{r_3}$

- Cada instrução rotulada pode ser denotada por uma quádrupla ordenada, onde a primeira componente identifica o tipo da instrução:

  a)*Operação*. $(\mathbf{0}, k, r_2, r_2)$

  b) *Teste*. $(\mathbf{1}, k, r_2, r_3)$

Usando a codificação do exemplo anterior, o programa monolítico P', visto como quádruplas ordenadas podem ser codificadas como segue:

- cada quádrupla (instrução rotulada) é codificada como um número natural, usando a codificação. Assim, o programa monolítico P' com **m** instruções rotuladas pode ser visto como uma **m-upla**;
- por sua vez, a **m-upla** correspondente ao programa monolítico P' é codificada como um número natural, usando a codificação.
- codificação de programas monolíticos apresentada não é uma função bijetora.

Exemplo 4.3
Codificação de programas monolíticos

Considere o programa monolítico Q da Figura 4.1.

```
Programa Monolítico Q
1:    faça F vá_para 2
2:    se T então vá_para 3 senão vá_para 5
3:    faça G vá_para 4
4:    se T então vá_para 1 senão vá_para 0
5:    faça F vá_para 6
6:    se T então vá_para 7 senão vá_para 2
7:    faça G vá_para 8
8:    se T então vá_para 6 senão vá_para 0
```

Figura 4.1 Programa monolítico

A correspondente codificação $c(i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8)$ é o numero natural q dado por:

$$q = 2^{i_1} \bullet 3^{i_2} \bullet 5^{i_3} \bullet 7^{i_4} \bullet 11^{i_5} \bullet 13^{i_6} \bullet 17^{i_7} \bullet 19^{i_8}$$

sendo que (tomando F como 1, G como 2 e o teste T como 1 na determinação de k, ou seja, o expoente do número primo 3):

$i_1 = 2^0 \bullet 3^1 \bullet 5^2 \bullet 7^2$	obtido da quádrupla: (0, 1, 2, 2)
$i_2 = 2^1 \bullet 3^1 \bullet 5^3 \bullet 7^5$	(1, 1, 3, 5)
$i_3 = 2^0 \bullet 3^2 \bullet 5^4 \bullet 7^4$	(0, 2, 4, 4)
$i_4 = 2^1 \bullet 3^1 \bullet 5^1 \bullet 7^0$	(1, 1, 1, 0)
$i_5 = 2^0 \bullet 3^1 \bullet 5^6 \bullet 7^6$	(0, 1, 6, 6)
$i_6 = 2^1 \bullet 3^1 \bullet 5^7 \bullet 7^2$	(1, 1, 7, 2)
$i_7 = 2^0 \bullet 3^2 \bullet 5^8 \bullet 7^8$	(0, 2, 8, 8)
$i_8 = 2^1 \bullet 3^1 \bullet 5^6 \bullet 7^0$	(1, 1, 6, 0)

Exemplo 4.4
Decodificação de programas monolíticos

A decodificação de um número natural p que denota um programa monolítico em seu correspondente programa é ilustrada.
Suponha o número natural **p** decomposto em fatores primos.

$$p = (2^{150}) \bullet (3^{105})$$

Pode-se recuperar o programa original. Como a decomposição apresentou dois numeros primos, então o programa possui duas instruções rotuladas correspondentes aos números **150** e **105**.

Relativamente às decomposições em seus fatores primos, tem-se que:

$$150 = 2^{1} \bullet 3^{1} \bullet 5^{2} \bullet 7^{0} \quad \text{e} \quad 105 = 2^{0} \bullet 3^{1} \bullet 5^{1} \bullet 7^{1}$$

o que corresponde às quádrulas:

$$(1, 1, 2, 0) \quad \text{e} \quad (0, 1, 1, 1)$$

```
1:   se T1 então vá_para 2 senão vá_para 0
2:   faça F1 vá_para 1
```

Exercícios de Codificação

Exercício 4.1

Analise o programa monolítico abaixo e assinale a opção que contém a codificação correta (considere a seguinte ordem de operações F=1, G=2, H=3 e dos testes T_1=1, T_2=2).

Programa Monolítico

```
1:   faça F vá-para 2
2:   se T1 então vá-para 3 senão vá-para 1
3:   faça G vá-para 4
4:   se T2 então vá-para 1 senão vá-para 5
5:   faça H vá-para 6
6:   se T2 então vá-para 0 senão vá-para 5
```

a) $2^{3675} \bullet 3^{5250} \bullet 5^{135025} \bullet 7^{151260} \bullet 11^{496371875} \bullet 13^{30256}$

b) $2^{375} \bullet 3^{525} \bullet 5^{1350625} \bullet 7^{151630} \bullet 11^{4963311875} \bullet 13^{3025}$

c) $2^{3675} \bullet 3^{5250} \bullet 5^{13505625} \bullet 7^{1512630} \bullet 11^{49633171875} \bullet 13^{302526}$

d) $2^{375} \bullet 3^{5250} \bullet 5^{1355625} \bullet 7^{151260} \bullet 11^{49633171875} \bullet 13^{30256}$

e) $2^{3675} \bullet 3^{5250} \bullet 5^{13505625} \bullet 7^{152630} \bullet 11^{4963171875} \bullet 13^{3526}$

Exercício 4.2

Codifique o seguinte programa monolítico e escolha a alternativa que representa o resultado de forma correta.

Programa Monolítico
1: se T_1 então vá-para 2 senão vá-para 3
2: faça F_1 vá-para 3
3: faça F_2 vá-para 4
4: se T_1 então vá-para 1 senão vá-para 0

a) $2^{51450} \bullet 3^{128625} \bullet 5^{13505625} \bullet 7^{30}$

b) $2^{2} \bullet 3^{12825} \bullet 5^{13505625} \bullet 7^{10}$

c) $2^{450} \bullet 3^{108625} \bullet 4^{13505625} \bullet 7^{20}$

d) $2^{40532} \bullet 3^{128888} \bullet 4^{13505625} \bullet 7^{70}$

e) $2^{2345} \bullet 3^{1555555} \bullet 5^{13505625} \bullet 7^{30}$

Exercício 4.3

Dado um programa monolítico abaixo, sabe-se que sua codificação é da forma $p = 2^x \bullet 3^y \bullet 5^z$.

```
1:    se T1 então vá-para 4 senão vá-para 3
2:    faça F1 vá-para 1
3:    se T1 então vá-para 2 senão vá-para 1
```

Marque a alternativa correta:

a) $x = 643125$, $y = 128625$ e $z = 26250$

b) $x = 1286250$, $y = 105$ e $z = 1050$

c) $x = 643125$, $y = 210$ e $z = 525$

d) $x = 36750$, $y = 128625$ e $z = 26250$

e) $x = 1286250$, $y = 128625$ e $z = 1050$

Exercício 4.4

Considere o seguinte programa monolítico:

```
1:    se T então vá-para 2 senão vá-para 3
2:    faça F vá-para 6
3:    se U então vá-para 5 senão vá-para 4
4:    faça G vá-para 0
5:    faça F vá-para 0
6:    se U então vá-para 4 senão vá-para 1
```

Assinale a quádrupla ordenada que *não* corresponde a nenhuma codificação das instruções rotuladas existentes neste programa:

a) (0, 1, 1, 0)

b) (0, 2, 0, 0)

c) (1, 1, 2, 3)

d) (1, 2, 4, 1)

e) (1, 2, 5, 4)

Exercício 4.5

Considere o seguinte programa monolítico:

```
1:    faça F vá-para 2
2:    se T1 então vá-para 1 senão vá-para 3
3:    faça G vá-para 4
4:    se T2 então vá-para 5 senão vá-para 1
```

Marque a correspondente codificação correta:

a) $2^{(2^0 3^1 5^2 7^2)} + 3^{(2^1 3^1 5^1 7^3)} + 5^{(2^0 3^2 5^4 7^4)} + 7^{(2^1 3^2 5^5 7^1)}$

b) $2^{(2^1 3^2 5^5 7^1)} \bullet 3^{(2^1 3^1 5^1 7^3)} \bullet 5^{(2^0 3^2 5^4 7^4)} \bullet 7^{(2^0 3^1 5^2 7^2)}$

c) $2^{(2^0+3^1+5^2+7^2)} \bullet 3^{(2^1+3^1+5^1+7^3)} \bullet 5^{(2^0+3^2+5^4+7^4)} \bullet 7^{(2^1+3^2+5^5+7^1)}$

d) $2^{(2^2 3^1 5^2 7^2)} \bullet 3^{(2^3 3^1 5^1 7^3)} \bullet 5^{(2^0 3^2 5^4 7^4)} \bullet 7^{(2^1 3^2 5^5 7^1)}$

e) $2^{(2^0 3^1 5^2 7^2)} \bullet 3^{(2^1 3^1 5^1 7^3)} \bullet 5^{(2^0 3^2 5^4 7^4)} \bullet 7^{(2^1 3^2 5^5 7^1)}$

Exercício 4.6

Considere o seguinte programa monolítico:

```
1:    faça F vá-para 2
2:    se T1 então vá-para 1 senão vá-para 3
3:    faça F2 vá-para 0
```

Marque a alternativa que representa o número natural associado pela função de codificação codigo(P):

a) 340341750

b) $2^{7350} \bullet 3^{5145} \bullet 5^{18}$

c) $2^{3675} \bullet 3^{10290} \bullet 5^{9}$

d) 583200

e) 77145

Exercício 4.7

Qual dos números abaixo representa a codificação de um programa monolítico?

a) $2^{3675} \bullet 3^{18375} \bullet 5^{4410}$

b) $2^{11025} \bullet 3^{1470} \bullet 5^{5292}$

c) $2^{9261} \bullet 3^{6174} \bullet 5^{9450}$

d) $2^{8505} \bullet 3^{13230} \bullet 5^{2835}$

e) 9447840

Exercício 4.8

Qual dos números abaixo *não* representa a codificação de uma instrução de programa monolítico?

a) 210

b) 450

c) 315

d) 750

e) 525

Exercício 4.9

Considere o seguinte programa monolítico codificado:

$$p = 2^{51450} \bullet 3^{105} \bullet 5^{9}$$

Marque a alternativa que representa a reescrita como um programa iterativo:

a) enquanto T faça (F;G)

b) até T faça (F);G

c) até T faça (F;G)

d) enquanto T faça (F);G

e) enquanto T faça (F)

Exercício 4.10

Codifique o seguinte programa iterativo e marque a opção correta:

```
enquanto T1 faça (F1);
F2;
(se T2
então faça até T1 faça (F3;F1); √
senão faça √)
```

a) $2^{(2^1 3^1 5^2 7^3)} \bullet 3^{(2^0 3^1 5^1 7^1)} \bullet 5^{(2^1 3^2 5^4 7^4)} \bullet 7^{(2^0 3^2 5^5 7^0)} \bullet 11^{(2^1 3^3 5^6 7^6)} \bullet 13^{(2^1 3^1 5^7 11^7)} \bullet 17^{(2^0 3^1 5^0 7^5)}$

b) $2^{(2^1 3^1 5^2 7^3)} \bullet 3(2^0 3^1 5^1 7^1) \bullet 5^{(2^0 3^2 5^4 7^4)} \bullet 7^{(2^1 3^2 5^5 7^0)} \bullet 11^{(2^1 3^1 5^0 7^6)} \bullet 13^{(2^0 3^3 5^7 7^7)} \bullet 17^{(2^0 3^1 5^5 7^5)}$

c) $2^{(2^0 3^1 5^2 7^3)} \bullet 3^{(2^1 3^6 5^1 7^1)} \bullet 5^{(2^1 3^2 5^4 7^4)} \bullet 7^{(2^0 3^2 5^5 7^0)} \bullet 11^{(2^1 3^3 5^6 7^6)} \bullet 13^{(2^1 3^1 5^7 7^7)} \bullet 17^{(2^0 3^1 5^0 7^7)}$

d) $2^{(2^1 3^1 5^2 7^3)} \bullet 3^{(2^0 3^1 5^1 7^1)} \bullet 5^{(2^0 3^2 5^4 7^4)} \bullet 7^{(2^1 3^2 5^5 7^0)} \bullet 11^{(2^0 3^3 5^6 7^6)} \bullet 13^{(2^0 3^1 5^7 7^7)} \bullet 19^{(2^1 3^1 5^0 7^5)}$

e) $2^{(2^0 3^1 5^2 7^3)} \bullet 3^{(2^1 3^1 5^1 7^1)} \bullet 5^{(2^1 3^2 5^4 7^4)} \bullet 7^{(2^0 3^2 5^5 7^0)} \bullet 11^{(2^1 3^3 5^6 7^6)} \bullet 13^{(2^5 3^1 5^7 7^7)} \bullet 19^{(2^0 3^1 5^0 7^5)}$

4.2 Definição da Máquina Norma

A *Máquina Universal Norma* (***N**umber TheO**r**etic **R**egister* ***MA**chine*) possui como memória um conjunto infinito de registradores naturais e três instruções sobre cada registrador: **adição** e **subtração do valor um** e **teste se o valor armazenado é zero**.

- N^{∞} denota o conjunto de todas as uplas com infinitos (mas contáveis) componentes sobre o conjunto dos números naturais .
- As componentes das uplas são denotadas por letras maiúsculas como X, Y, A, B,..., registradores na Máquina Norma.

Definição 4.1. Máquina Norma.

A *Máquina Norma* é uma sete-upla (suponha que K seja um registrador, $K \in \{ X, Y, A, B, ... \}$):

$$\text{Norma} = (N^{\infty}, N, N, ent, sai, \{ ad_K, sub_K \}, \{ zero_K \})$$

a) Cada elemento do conjunto de valores de memória N^{∞} denota uma configuração de seus infinitos *registradores*, os quais são denotados por: X, Y, A, B,...

b) A *função de entrada*: $ent: N \to N^{\infty}$ é tal que carrega no registrador denotado por X o valor de entrada, inicializando todos os demais registradores com zero;

c) A *função de saída*: $sai: N^{\infty} \to N$ é tal que retorna o valor corrente do registrador denotado por Y;

d) O conjunto de *interpretações de operações* é uma família de operações indexada pelos registradores, na qual, para cada registrador $K \in \{ A, B, X, Y,... \}$, tem-se que:

$ad_K: N^{\infty} \to N^{\infty}$ adiciona um à componente correspondente ao registrador K, deixando as demais com seus valores inalterados. K:=K+1

$sub_K: N^{\infty} \to N^{\infty}$ subtrai um da componente correspondente ao registrador K, se o seu valor for maior que zero (caso contrário, mantém o valor zero), deixando as demais com seus valores inalterados. K:=K-1

e) O conjunto de *interpretações de testes* é uma família de testes indexada pelos registradores na qual, para cada registrador K, tem-se que:

$zero_K: N^{\infty} \to \{ \text{verdadeiro, falso} \}$ resulta em verdadeiro, se a componente correspondente ao registrador K for zero e em falso, caso contrário. K=0?

4.3 Máquina Norma como Máquina Universal

É uma máquina extremamente simples, de tal forma que parece difícil acreditar que o seu poder computacional é, no mínimo, o de qualquer computador moderno.

Características de máquinas reais são simuladas usando a Máquina Norma, reforçando as evidências de que se trata de uma máquina universal.

As características são as seguintes:

a) *Operações e Testes*. Definição de operações e testes mais complexos como adição, subtração, multiplicação e divisão de dois valores e tratamento de valores diversos como os números primos;

b) *Valores Numéricos*. Armazenamento de tratamento de valores numéricos de diversos tipos como inteiros (negativos e não-negativos) e racionais;

c) *Dados Estruturados*. Armazenamento de tratamento de dados estruturados como em arranjos (vetores uni e multidimensionais), pilhas, etc;

d) *Endereçamento Indireto e recursão*. Desvio para uma instrução determinada pelo conteúdo de um registrador;

e) *Cadeia de Caracteres*. Definição e manipulação de cadeias de caracteres.

4.3.1 Operações e Testes

As seguintes operações e teste não-definidos na Máquina Norma são exemplificadas a seguir:

- atribuição de valor a um registrador;
- adição de dois registradores;
- atribuição do valor de um registrador a outro registrador;
- multiplicação de dois registradores;
- atribuição de um número primo a um registrador;
- teste se o valor de um registrador é menor que o valor de outro registrador;
- teste se o valor de um registrador é múltiplo do valor de outro registrador (ou teste de divisão inteira com resto zero);
- teste se o valor de um registrador é um número primo.

Exemplo 4.5
Atribuição do valor zero a um registrador A.

- Programa Iterativo **A:= 0**:

```
até     A = 0
faça    (A:= A – 1)
```

- Programa na forma de Instruções Rotuladas

```
1: se A = 0 então vá_para 3 senão vá_para 2
2 faça A := A – 1 vá_para 1
```

- pode-se tratar a operação **A:= 0** como uma ***macro***, ou seja, um trecho de programa que é substituído pela sua definição sempre que referenciado.
- Usando a macro A:= 0, é fácil construir macros para definir operações de atribuição de um valor qualquer.

Exemplo 4.6
Atribuição de um valor natural a um registrador.

- macro da atribuição de um valor natural n a um registrador A, **A:= n**, é a generalização do programa abaixo.
- Programa Iterativo n := 3:

```
A:= 0;
A:= A+1;
A:= A+1;
A:= A+1
```

Exemplo 4.7
Adição de dois registradores.

- macro da correspondente à operação de adição do valor do registrador B ao do registrador A, denotada por: **A: = A + B**
- Programa Iterativo A: = A + B:

```
até    B = 0
faça   (A:= A + 1; B:= B - 1)
```

- Observe que, ao somar o valor de B em A, o registrador B é zerado!

Exemplo 4.8
Adição de dois registradores, preservando B

- Macro da correspondente à operação de adição do valor do registrador B ao do registrador A, preservando o valor em B, necessita usar um registrador auxiliar C
- Programa Iterativo **A:= A + B usando C**

```
C:= 0;
até    B = 0
faça   (A:= A + 1; C:= C + 1; B:= B - 1);
até    C = 0
faça   (B:= B + 1; C:= C - 1)
```

- como este programa não preserva o conteúdo original do registrador de trabalho C, faz-se necessário explicitar o uso deste registrador.
- é necessário escolher um registrador de trabalho C que não seja usado!

Exemplo 4. 9
Atribuição do conteúdo de um registrador.

- Programa Iterativo **A:= B usando C:**

```
A:= 0;
A:= A + B usando C
```

Exemplo 4.10
Multiplicação de dois registradores.

- A definição da macro de multiplicação requer dois registradores de trabalho.
- Programa Iterativo **A:= A • B usando C, D**:

```
C:= 0;
até     A = 0
  faça  (C:= C + 1; A := A - 1);
até     C = 0
  faça  (A:= A + B usando D; C:= C - 1)
```

Ou seja,

```
C := 0;
até     A = 0
  faça  (C:= C + 1; A := A - 1);
até     C = 0
  faça  (D:= 0;
         até    B = 0
           faça  (A:= A + 1; D:= D + 1; B:= B - 1);
          até   D = 0
           faça  (B:= B + 1; D:= D - 1);
          C:= C - 1)
```

- correspondente a uma macro cuja operação é a multiplicação do valor do registrador B pelo valor do registrador A, usando dois registradores de trabalho C e D

Exemplo 4.11
Teste se o valor de um registrador é menor que outro registrador

macro do teste que verifica se o valor de um registrador A é menor que o valor de um registrador B, denotada por:

A < B usando C, D, E

```
C:= A usando E;
D:= B usando E;
até C = 0
faça (se D = 0
       então falso
       senão C:= C-1; D:= D-1);
       (se D = 0
       então falso
       senão verdadeiro)
```

Exemplo 4.12
Teste se o valor da divisão inteira é zero

macro do teste que verifica se o se o resto da divisão inteira do valor de um registrador A pelo valor de um registrador B é zero, denotada por:

Teste_mod(A, B) usando C, D, E, C’, D’, E’

```
C:= A usando E;
(se B = 0
então falso
senão (se A = 0
       então verdadeiro
       senão até C < B usando C’, D’, E’
              faça  (C:= C-D usando E);
       (se C = 0
       então verdadeiro
       senão falso)))
```

Por simplicidade, no texto que segue, na referência a uma macro definida anteriormente, são omitidos os registradores usados. Por exemplo:

teste_mod(A, B) usando C, D, E, C’, D’, E’ é abreviada por teste_mod(A, B)

Exemplo 4.13
Teste se o valor do registrador é um primo.

- Programa Iterativo **teste_primo(A)**

```
(se     A = 0
então   falso
senão   C:= A;
        C:= C - 1;
        (se     C = 0
        então   verdadeiro
        senão   até     teste_mod(A, C)
                faça    (C:= C – 1)
                C:= C - 1;
                (se     C = 0
                então   verdadeiro
                senão   falso) ) )
```

- é uma macro de teste que verifica se o valor de um registrador A é um número primo, retornando o valor verdadeiro, se o valor de A é primo, e o valor falso, caso contrário:
- **teste_mod(A, C)** é um teste que retorna o valor verdadeiro, se o resto da divisão inteira do conteúdo de A por C é zero, e o valor falso, caso contrário.

Exemplo 4.14
Atribuição do n-ésimo número primo ao registrador.

- Programa Iterativo **A:= primo(B)**

```
A:= 1;
D:= B;
até     D = 0
faça    (D:= D - 1;
        A:= A + 1;
        até     teste_primo(A)
        faça    (A:= A + 1) )
```

- é a atribuição do n-ésimo número primo a um registrador A, usando um registrador de trabalho D,
- usa a macro teste_primo construída acima (suponha que 1 é o 0-ésimo número primo)

4.3.2 Valores Numéricos

Norma pode definir inteiros e racionais?

Exemplo 4.15
Inteiros.

Um **valor inteiro m** pode ser representado como um par ordenado: **(s, |m|),** onde:

- **|m|** denota magnitude dada pelo valor absoluto de m;
- **s** denota o sinal de **m: se m < 0,**
  então s = 1 (negativo)
  senão s = 0 (positivo)

Como representar pares ordenados em Norma?

- usando codificação de n-uplas naturais como introduzido em **4.1**;
- usando dois registradores, o primeiro referente ao sinal, e o segundo, à magnitude.

Representação de Inteiros utilizando dois registradores

- Supor que o registrador inteiro A é representado pelo par (A_1, A_2) na representação conhecida como *sinal-magnitude*, ou seja, A_1 (sinal) e A_2 (magnitude).

- É necessário desenvolver programas em Norma para executar as operações inteiras **A:= A+1, A:= A-1** e o teste **A = 0?**

A	A_1 (sinal)	A_2 (mag.)	A + 1	A_1 (sinal)	A_2 (mag.)	A - 1	A_1 (sinal)	A_2 (mag.)
-2	1	2	-1	1	1	-3	1	3
-1	1	1	0	0	0	-2	1	2
0	0	0	1	0	1	-1	1	1
1	0	1	2	0	2	0	0	0
2	0	2	3	0	3	1	0	1

Figura 4.2 Operações inteiras A:= A + 1 e A:= A - 1

- ***Programa Iterativo A:= A + 1:***

```
(se     A1 = 0
então  A2:= A2 + 1
senão  A2:= A2 - 1;
       (se     A2 = 0
       então  A1:= A1 – 1
       senão  √) )
```

- ***Programa Iterativo A:= A - 1:***

```
(se   A1 = 0
 então    (se   A2 = 0
            então    A1 := A1 + 1;
                     A2 := A2 + 1
            senão    A2 := A2 – 1)
 senão    A2 := A2 + 1)
```

- ***Programa Iterativo A = 0 ?***

```
(se   A2 = 0
então     verdadeiro
senão     falso)
```

Exemplo 4.16
Racionais.

- Um valor racional r pode ser denotado como um par ordenado: (a, b) tal que $b > 0$ e $r = a/b$.
- A representação não é única pois, por exemplo, o valor racional 0.75 pode ser representado pelos pares (3, 4) e (6, 8), entre outros (na realidade, os pares (3, 4) e (6, 8) pertencem à mesma classe de equivalência).
- Neste contexto, as operações de adição, subtração, multiplicação e divisão, bem como o teste de igualdade, podem ser definidos como segue:

$$(a, b) + (c, d) = (a \bullet d + b \bullet c,\ b \bullet d)$$

$$(a, b) - (c, d) = (a \bullet d - b \bullet c,\ b \bullet d)$$

$$(a, b) \bullet (c, d) = (a \bullet c,\ b \bullet d)$$

$$(a, b) / (c, d) = (a \bullet d,\ b \bullet c) \ (\text{com } c \neq 0)$$

$$(a, b) = (c, d) \quad \text{se, e somente se,} \quad a \bullet d = b \bullet c$$

4.3.3 Dados Estruturados

Arranjos podem ser definidos em Norma?

E arranjos multidimensionais?

Exemplo 4.17
Arranjo Unidimensional.

- Uma estrutura do tipo arranjo unidimensional da forma A(1), A(2), …, pode ser definida por um único registrador A usando a codificação de n-uplas naturais. Por exemplo: Se
  A(1) = 5 A(2) = 2 A(3) = 0 A(4) = 3 A(5) = 1
  então o valor a armazenada em A é o resultado de:
  $$a = 2^5 \bullet 3^2 \bullet 5^0 \bullet 7^3 \bullet 11^1 \bullet 13^0 \bullet 17^0 \ldots$$
- Note-se que o arranjo não necessita ter tamanho máximo predefinido (número de posições indexáveis). Lembre-se de que a função de entrada é tal que carrega o valor da entrada no registrador X, zerando todos os demais, incluindo o arranjo.
- Para manter a coerência com a definição da Máquina Norma, é necessário definir as operações de adição e de subtração do valor 1, bem como o teste se o valor é zero.
  - o arranjo é implementado usando o registrador A;
  - p_n denota o n-ésimo número primo;
  - teste_mod(A, C) é um teste que retorna o valor verdadeiro, se a divisão inteira do conteúdo de A pelo conteúdo de C é zero, e o valor falso, caso contrário;
  - A:= div(A, C) denotada uma macro de divisão de dois registradores;
- Em uma estrutura do tipo arranjo, é desejável indexar as suas posições de forma direta (número natural) ou indireta (conteúdo de um registrador).

a) Indexação Direta.

Programa Iterativo ad$_{A(n)}$

```
C := pn;
A := A • C
```

Programa Iterativo sub$_{A(n)}$

```
C := pn;
(se     teste_mod(A, C)
então  A := div(A, C)
senão  √)
```

Programa Iterativo zero$_{A(n)}$

```
C := pn;
(se     teste_mod(A, C)
então  falso
senão  verdadeiro)
```

b) Indexação Indireta.

Programa Iterativo ad$_{A(B)}$

```
C := primo(B)
A := A • C
```

Programa Iterativo sub$_{A(B)}$

```
C := primo(B)
(se     teste_mod(A, C)
então  A := div(A, C)
senão  √)
```

Programa Iterativo zero$_{A(B)}$

```
C := primo(B)
(se     teste_mod(A, C)
então  falso
senão  verdadeiro)
```

Observação 4.2
Arranjo Unidimensional × Norma com 2 Registradores.

- Usando a estrutura de arranjo unidimensional com indexação direta, pode-se mostrar que os registradores X e Y são suficientes para realizar qualquer processamento;
- De fato, é suficiente usar X para armazenar um arranjo unidimensional onde cada posição corresponde a um registrador (por exemplo: X, Y, A, B, …, correspondem às posições do arranjo indexadas por 0, 1, 2, 3, …).
- Assim, para um determinado registrador K, as operações e testes de Norma: ad_K, sub_K e $zero_K$ podem ser simulados pelas operações e testes indexados introduzidos no ***Exemplo 4.17***, ou seja, $ad_{X(k)}$, $sub_{X(k)}$ e $zero_{X(k)}$ onde X(k) denota a k-ésima posição do arranjo em X.

Exemplo 4.18
Pilha.

- Estruturalmente, a principal característica de uma *pilha* é que o último valor gravado é o primeiro a ser lido.

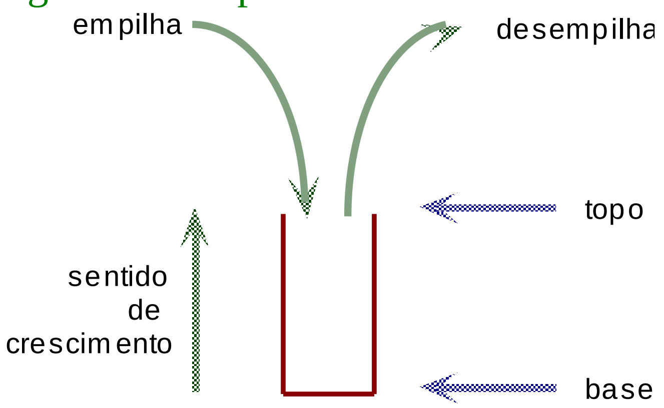

figura 4.3 Estrutura do tipo pilha

- A *base* de uma pilha é fixa e define o seu início.
- O *topo* é variável e define a posição do último símbolo gravado.
- definem-se as operações *empilha* (adiciona o conteúdo de um registrador no topo da pilha) e *desempilha* (retira o valor do topo e armazena em um registrador).
- Uma pilha pode facilmente ser simulada usando um arranjo e um registrador de índice (indexação indireta do arranjo) que aponta para o topo da pilha.

4.3.4 Endereçamento Indireto e Recursão

Como definir desvios, usando endereçamento indireto (determinado pelo conteúdo de um registrador) em programas do tipo monolítico?

Exemplo 4.19
endereçamento indireto em um programa monolítico

- Uma operação com endereçamento indireto da seguinte forma, onde A é um registrador:

r: faça F vá_para A

- pode ser definida pelo seguinte programa monolítico:

r: faça F vá_para End_A

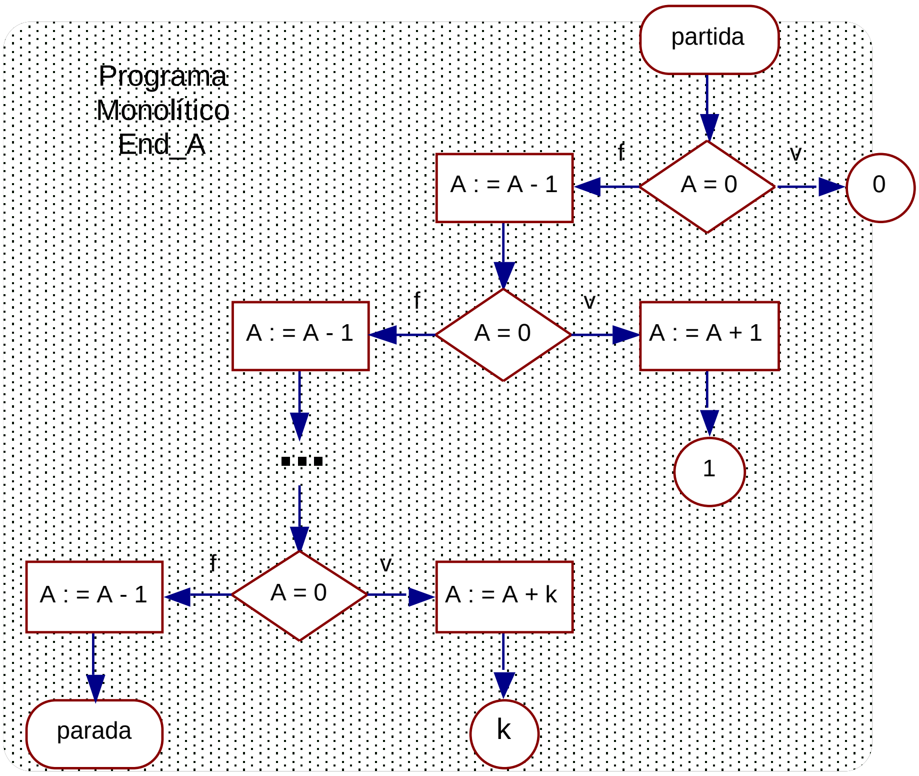

figura 4.4 Fluxograma para tratar endereçamento indireto

- A macro End_A trata o endereçamento indireto de A (cada circunferência rotulada representa um desvio incondicional para a correspondente instrução).

- Teste com endereçamento indireto, onde A e B são registradores:

r: se T então vá_para A senão vá_para B

4.3.5 Cadeias de Caracteres

- Cadeia de caracteres é outro tipo de dado não pré-definido na Máquina Norma.
- O tratamento da definição e da manipulação de cadeias de caracteres será realizado através de outra Máquina Universal, denominada Máquina de Turing, a qual se prova ser equivalente à Norma.

4.4 Conclusões

- Neste capítulo foram introduzidas as máquinas de registradores.
- Em particular, foi estudada a máquina NORMA que possui três tipos de instruções/teste (ad, sub e zero) sobre registradores naturais.
- Os recursos necessários para computar são bem poucos. Todos os demais recursos existentes em máquinas reais são apenas artifícios para facilitar a programação da computação.
- Evidências internas de que NORMA é uma máquina universal foram apresentadas.
- Evidencias externas serão vistas nos próximos capítulos.

4.5 Exercícios

Exercício 4.11

Codifique o fluxograma ilustrado na ***Figura 4.5*** como um número natural, usando a codificação de programas introduzida no ***Exemplo 4.2*** **Codificação de programas monolíticos**.

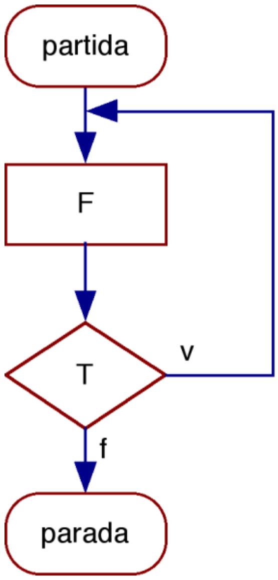

Figura 4.5 Fluxograma

Exercício 4.12

Codifique o programa da ***Figura 4.6*** usando a codificação de programas monolíticos.

```
Programa Monolítico M
1:   faça F vá_para 2
2:   se T então vá_para 3 senão vá_para 1
3:   faça G vá_para 4
4:   se T então vá_para 1 senão vá_para 0
```

Figura 4.6 Programa monolítico

Exercício 4.13

Desenvolva um *software* que seja capaz de processar qualquer programa de Máquina Norma. A entrada para o *software* contem:

- o programa no formato de instruções rotuladas simples;
- o valor de entrada a ser computado pelo programa;

A saída é a computação do programa para a dada entrada. Teste o software para o cálculo do fatorial de 3.

Exercício 4.14

Mostre como a instrução abaixo podem ser construídas como uma macro em Norma:

r_1: se A<2 então vá_para r_2 senão vá_para r_3

Exercício 4.15

Desenvolva os programas, em Norma, que realizam as operações e testes abaixo:

a) A:= B - C
b) div(B, C): divisão inteira de B por C
c) fat(A): fatorial;
d) pot(x, y): potência;
e) teste(A > B)
f) teste(A ≥ B)
g) teste(A ≤ B)
h) mdc(A, B): máximo divisor comum;
i) mod(A, B): resto da divisão inteira;
j) primos(A, B): todos os primos entre os valores de A e de B;
k) teste_nperf(A): verifica se é um número perfeito.

Exercício 4.16

Desenvolva os programas, em Norma, que realizam as operações abaixo nos inteiros (utilize a representação sinal-magnitude introduzida no ***exemplo 4.15 Inteiros***):

a) A:= B + C
b) A:= B • C
c) A:= B - C
d) A:= div(B, C)

Exercício 4.17

Desenvolva os programas, em Norma, que realizam as operações abaixo nos números racionais (utilize a representação de números racionais introduzida no ***exemplo 4.16 Racional***):

a) A:= B + C
b) A:= B • C
c) A:= B - C
d) A:= div(B, C)

Exercício 4.18

Seja a *Máquina NormaNeg* a qual é, em todos os aspectos, idêntica a Norma, exceto pelo fato de poder armazenar números negativos (inteiros) em seus registradores. Prove que, para cada fluxograma *NormaNeg*, pode-se definir um fluxograma Norma equivalente sem o uso de registradores extras.

Sugestão: utilize a representação de inteiros baseada na função codificação e desenvolva os programas para as instruções de Norma que simulam as de *NormaNeg*.

Exercício 4.19

Seja P = {I, 1} definido na Máquina Norma e suponha o conteúdo dos registradores A e B igual a 0 e 5 respectivamente. Marque o programa que realiza A:= B na máquina Norma:

a)

Programa Monolítico	
1:	se $zero_B$ então vá-para 5 senão vá-para 2
2:	faça ad_D vá-para 3
3:	faça sub_B vá-para 4
4:	faça sub_A vá-para 1

b)

Programa Monolítico	
1:	se $zero_A$ então vá-para 4 senão vá-para 2
2:	faça ad_A vá-para 3
3:	faça ad_B vá-para 1

c)

Programa Monolítico	
1:	se $zero_A$ então vá-para 5 senão vá-para 2
2:	faça ad_A vá-para 3
3:	faça sub_B vá-para 1

d)

Programa Monolítico	
1:	se $zero_B$ então vá-para 5 senão vá-para 2
2:	faça ad_A vá-para 3
3:	faça sub_B vá-para 1

e) Nenhuma das alternativas anteriores está correta.

Exercício 4.20

A Máquina Norma é composta basicamente por um conjunto de registradores naturais e apenas três instruções sobre cada um deles (adição e subtração de uma unidade e a verificação se o conteúdo do registrador é zero). Portanto, é correto afirmar que:

a) Seus registradores podem assumir valores naturais tão grandes quanto necessários.
b) Os registradores podem assumir qualquer valor;
c) Seu modelo formal encontra-se baseado em uma estrutura chamada fita;
d) Não é capaz de simular mecanismos de recursão;
e) Pode ser definida por uma 8-upla da forma $(\Sigma, Q, \Pi, q_0, F, V, \beta, ©)$.

Exercício 4.21

Sobre a Máquina Norma, analise as seguintes afirmações:

I. É uma Máquina extremamente simples, porém com poder computacional igual ao de qualquer computador atual;
II. Em Norma, sub-rotinas e mecanismos de recursão, como os definidos em programas recursivos, podem ser simulados por programas monolíticos, usando endereçamento indireto;
III. É impossível programar operações matemáticas complexas (radiciação, seno e co-seno, por exemplo) em Norma, visto que apenas as instruções de somar/diminuir 1 unidade a um registrador não dão suporte para que isso seja feito.

Marque a alternativa correta:

a) Apenas I está correta;
b) Apenas II está correta;
c) Apenas III está correta;
d) Apenas I e II estão corretas;
e) Apenas II e III estão corretas.

Exercício 4.22

Sobre a Máquina Norma, analise as seguintes afirmações:

I. Possui um conjunto finito de registradores naturais como memória;
II. Possui poder computacional, no mínimo, igual ao de qualquer computador moderno;
III. Sub-rotinas e mecanismos de recursão podem ser simulados por programas monolíticos, usando endereçamento indireto.

Marque a alternativa correta:

a) Apenas I está correta;
b) Apenas II está correta;
c) Apenas I e III estão corretas;
d) Apenas II e III estão corretas;
e) I, II e III estão corretas.