Máquinas de Turing

Ábacos

Prof. Edson Alves

Faculdade UnB Gama

Sumário

1. Ábacos

2. Exemplos

3. Computabilidade por ábacos

Contexto histórico

- As máquinas de Turing tem muitas limitações: uma delas é trabalhar exclusivamente com inteiros positivos, o que exclui o zero
- Além disso, elas foram propostas antes do surgimento dos computadores digitais
- De fato, as máquinas de Turing contribuíram significativamente no desenvolvimento destes computadores
- Uma importante característica presente nos computadores digitais e ausentes nas máquinas de Turing é o acesso aleatório à memória
- Além disso, o sistema numérico subjacente é o sistema binário, e não o monádico
- O acréscimo destas duas características às máquinas de Turing levam aos ábacos

Ábaco

Definição

Uma **máquina de Lambek** ou uma **máquina de ábaco** é uma versão idealizada de computador, com as seguintes características:

(a) acesso ao um número **ilimitado** de registradores $R_0, R_1, R_2, \ldots$

(b) cada registrador pode armazenar um número natural (positivos e o zero) de tamanho **arbitrário**

(c) cada registrador tem seu próprio **endereço**, de modo que é possível se mover do registrador R_i para o registrador R_j diretamente, sem precisar passar, passo a passo, pelos registradores intermediários $R_{i+1}, R_{i+2}, \ldots, R_{j-1}$

Notação

- Os registradores são representados pela letra maiúscula R e pelo subscrito i, indicando o número do registrador
- A notação $[m]$ indica o número que está armazenado no registrador R_m
- Um registrador pode estar vazio, isto é, armazenar o valor zero
- A instrução "*Coloque a soma dos números armazenados em* R_m *e em* R_n *em* R_p pode ser escrita como

$$[m] + [n] \to p$$

- O número à direita da seta indica o registrador que armazenará o resultado da instrução

Programas em ábaco

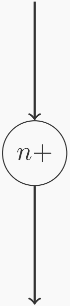
Um **programa** em um ábaco consiste em uma lista de instruções numeradas. Cada uma destas instruções é de uma das duas formas abaixo:

$$(q) \quad \text{acrescente um à caixa } m \text{ e vá para a instrução } r$$

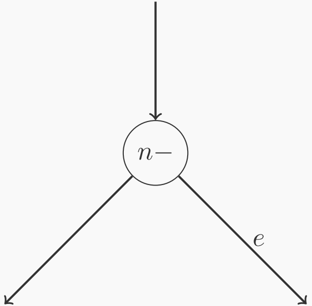
ou

$$(q) \quad \begin{cases} \text{se a caixa } m \text{ não está vazia,} & \text{então subtraia um da caixa } m \text{ e vá para } r \\ \text{se a caixa } m \text{ está vazia,} & \text{então vá para } s \end{cases}$$

Diagramas correspondentes às duas instruções dos ábacos



Instrução: Acrescente um ao número armazenado no registrador R_n



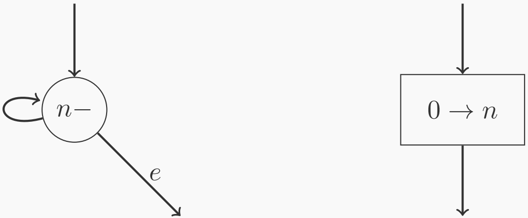
Instrução: Se R_n estiver vazio, saia pela seta e; caso contrário, subtraia um e saia pela outra seta

Exemplo: Esvaziar o registrador R_n

O programa a seguir, que consiste em uma única instrução, esvazia o conteúdo do registrador R_n:

$$(1) \quad \begin{cases} \text{se } [n] \text{ é diferente de zero,} & \text{então subtraia um e permaneça em } 1 \\ \text{se } [n] \text{ é igual a zero,} & \text{então pare} \end{cases}$$

(a) Fluxograma **(b)** Diagrama de Blocos

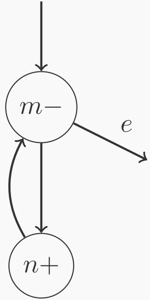


Exemplo: Esvaziar o registrador R_n

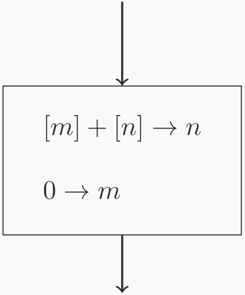
Exemplo: Esvaziar o registrador R_m no registrador R_n

O programa abaixo esvazia o conteúdo do registrador R_m no registrador R_n, assumindo que ambos registradores são distintos.

(a) Fluxograma



(b) Diagrama de Blocos

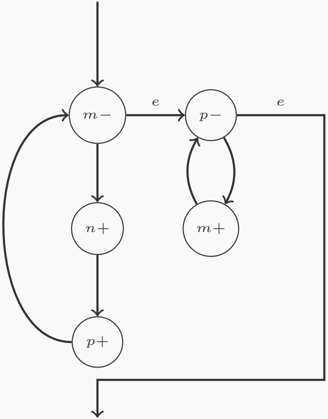


Exemplo: Esvaziar R_m em R_n

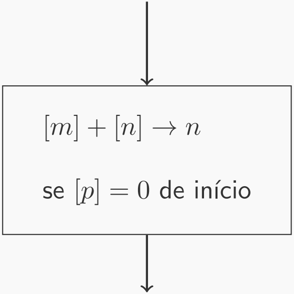
Exemplo: Adicionar R_m a R_n, sem perda de R_m

Para adicionar o conteúdo de R_m em R_n, sem perda de R_m, é preciso um registrador auxiliar R_p, inicialmente vazio.

(a) Fluxograma



(b) Diagrama de Blocos

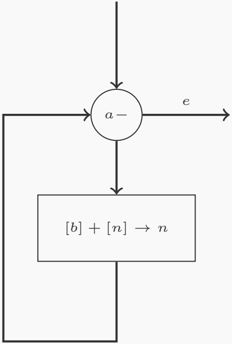


Exemplo: Adicionar R_m a R_n, sem perda de R_m

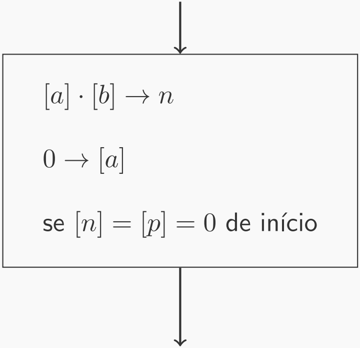
Exemplo: Multiplicação

O ábaco abaixo computa o produto dos números armazenados em R_a e R_b. O resultado ficará armazenado em R_n e, inicialmente, tanto R_n quanto R_p devem estar vazios.

(a) Fluxograma abreviado



(b) Diagrama de Blocos



Exemplo: Multiplicar R_a e R_b

Exemplo: Multiplicação

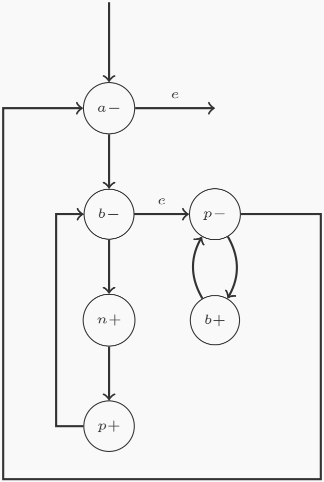


Figura: Fluxograma completo

Equivalência entre ábacos e máquinas de Turing

Teorema

Toda função computável por ábaco é Turing computável.

Referências

1. **BOOLOS**, George S.; **BURGESS**, John P.; **JEFFREY**, Richard C. *Computabilidade e Lógica*, Editora Unesp, 2012.