



DANIEL MARTINS GUSMAI

# CALCULADORA DAS CLASSES RESIDUAIS

**Santo André, 2018**



UNIVERSIDADE FEDERAL DO ABC

CENTRO DE MATEMÁTICA, COMPUTAÇÃO E COGNIÇÃO

DANIEL MARTINS GUSMAI

# CALCULADORA DAS CLASSES RESIDUAIS

**Orientador: Prof. Dr. Eduardo Guéron**

Dissertação de mestrado apresentada ao Centro de Matemática, Computação e Cognição para obtenção do título de Mestre

ESTE EXEMPLAR CORRESPONDE A VERSÃO FINAL DA DISSERTAÇÃO
DEFENDIDA PELO ALUNO DANIEL MARTINS GUSMAI,
E ORIENTADA PELO PROF. DR. EDUARDO GUÉRON.

**SANTO ANDRÉ, 2018**

Sistema de Bibliotecas da Universidade Federal do ABC
Elaborada pelo Sistema de Geração de Ficha Catalográfica da UFABC
com os dados fornecidos pelo(a) autor(a).

Gusmai, Daniel Martins
Calculadora das classes residuais / Daniel Martins Gusmai. — 2018.

131 fls. : il.

Orientador: Eduardo Guéron

Dissertação (Mestrado) — Universidade Federal do ABC, Mestrado Profissional em Matemática em Rede Nacional - PROFMAT, Santo André, 2018.

1. Congruências. 2. Álgebra booleana. 3. Calculadora. I. Guéron, Eduardo. II. Mestrado Profissional em Matemática em Rede Nacional - PROFMAT, 2018. III. Título.

**Este exemplar foi revisado e alterado em relação à versão original, de acordo com as observações levantadas pela banca no dia da defesa, sob responsabilidade única do autor e com a anuência de seu orientador.**

**Santo André,** 22 **de** maio **de 20**18**.**

**Assinatura do autor:** Daniel Martins Gusmai

**Assinatura do orientador:**

**MINISTÉRIO DA EDUCAÇÃO**
**Fundação Universidade Federal do ABC**
**Programa de Pós-Graduação em Mestrado Profissional em Matemática em Rede Nacional**
Avenida dos Estados, 5001 – Bairro Santa Terezinha – Santo André – SP
CEP 09210-580 · Fone: (11) 4996-0017
profmat@ufabc.edu.br

## FOLHA DE ASSINATURAS

Assinaturas dos membros da Banca Examinadora que avaliou e aprovou a Defesa de Dissertação de Mestrado do candidato Daniel Martins Gusmai, realizada em 15 de março de 2018:

Prof.(a) Dr.(a) **Eduardo Guéron** (Universidade Federal do ABC) – Presidente

Prof.(a) Dr.(a) **Jeferson Cassiano** (Universidade Federal do ABC) – Membro Titular

Prof.(a) Dr.(a) **Mariana Pelissari Monteiro Aguiar Baroni** (Instituto Federal de São Paulo) – Membro Titular

Prof.(a) Dr.(a) **Rafael Ribeiro Dias Vilela de Oliveira** (Universidade Federal do ABC) – Membro Suplente

Prof.(a) Dr.(a) **Leonardo Paulo Maia** (Universidade de São Paulo) – Membro Suplente

Dedico este trabalho a todos os professores que insistem, persistem e perseveram, que buscam a todo momento o contínuo aprimoramento, a inovação do ensino, a diversificação do currículo e a excelência na prática docente apesar de todas as condições perniciosas que assolam o atual quadro educacional.

# AGRADECIMENTOS

Agradeço primeiramente a Deus, sem o qual nada se concretizaria, à minha família, em especial aos meus pais, Antônio e Doracy, que se sacrificaram em prol da minha formação, à minha esposa Miriã, que sempre acreditou em mim até quando eu mesmo duvidei! Aos professores do *PROFMAT*, em especial ao meu orientador Eduardo, que prontamente estendeu-me a mão e a todos que direta ou indiretamente contribuíram para a desenlace deste ciclo.

*“Qualquer coisa pode se tornar tão simples quanto possível, mas não mais simples do que são.”*

(Albert Einstein)

# RESUMO

Calculadoras são aparelhos comuns no cotidiano do homem moderno, contudo, os conceitos matemáticos envolvidos em sua concepção ainda são conhecidos por poucos. Durante séculos, a obstinação da humanidade em construir máquinas capazes de computar de forma autônoma resultou tanto no surgimento dos atuais computadores, como também em um magnífico legado de conhecimentos matemáticos agregados a tal conquista. Conteúdos tais como congruências e álgebra booleana suscitaram a revolução dos sistemas informatizados e tem sido amplamente explorados por meio de inúmeras aplicações, nossa trajetória perpassou pela aritmética modular, o teorema de Euler-Fermat e as classes residuais, além de bases numéricas, tópicos de eletrônica digital e funções booleanas, com foco no desenvolvimento de circuitos lógicos e o engendrar de componentes eletrônicos, que configuram a base para idealização e construção de calculadoras que efetuem as operações aritméticas em bases arbitrárias, objetivo preponderante deste trabalho. O esmiuçar das etapas de construção das calculadoras, viabiliza o aprofundamento dos conceitos matemáticos que a fomentaram. A abordagem dos temas supracitados culmina para aprimorar e evidenciar a aplicabilidade da matemática à essência da era moderna.

**Palavras-chave**: Congruências; Álgebra booleana; Calculadora.

# ABSTRACT

Calculators are common apparatuses in the everyday of modern man, however, the mathematical concepts involved in its conception are still known by few. For centuries, mankind's obstinacy in building machines capable of computing autonomously resulted in both the emergence of current computers and a magnificent legacy of mathematical knowledge added to such achievement. Contents such as congruences and Boolean algebra have aroused the revolution of computerized systems and it has been extensively explored through numerous applications, our trajectory ran through modular arithmetic, Euler-Fermat's theorem and residual classes, as well as numerical bases, topics of digital electronics and Boolean functions, focusing on the development of logic circuits and the generation of electronic components, which form the basis for the design and construction of calculators that perform arithmetic operations on arbitrary bases, a preponderant objective of this work. The to detail of the construction steps of the calculators, enables the deepening of the mathematical concepts that fomented it. The approach to the aforementioned themes culminates in improving and evidencing the applicability of mathematics to the essence of the modern era.

**Keywords**: Congruences; Boolean algebra; Calculator.

# CONTEÚDO

# INTRODUÇÃO

O mundo se transforma constantemente e a passos largos, especialmente no que diz respeito aos avanços tecnológicos, ao ponto de que a própria sociedade tem encontrado dificuldades em acompanhar tal ritmo. Essencialmente, a referida dissertação visa amalgamar o estudo das congruências e aritmética modular com a álgebra booleana aplicada a concepção de projetos relativos aos circuitos lógicos. Ambas apresentam inúmeras e profundas aplicações em várias áreas da matemática, configurando-se na base de praticamente todos os procedimentos de cálculo e programação de computadores, além de perpassar em muitas aplicações tecnológicas.

A compreensão dos conceitos matemáticos aqui explanados, contribuem significativamente para o desenvolvimento cognitivo do aprendiz, aprimorando sua percepção na interpretação de problemas, além de fornecer subsídios ensejadores da estruturação do raciocínio, auxiliando a preferir decisões e ações. Tal compreensão sustenta estruturas que transcendem tanto o âmbito escolar como o da própria matemática, e são cruciais para o pleno exercício da cidadania na era do conhecimento, pois se configuram em ferramentas suporte para a estruturação do pensamento.

Neste âmbito, esquadrinham-se novas estratégias, ferramentas e alternativas para professores que buscam o constante aperfeiçoamento de sua prática, a fim de despertar em seus alunos o interesse por aprender esta disciplina fascinante. Partimos do seguinte pressuposto para a criação de calculadoras das classes residuais: aquele que quiser criar um dispositivo/máquina que resolva os problemas de matemática, invariavelmente deverá desenvolver uma série de conhecimentos inerentes à matemática.

No primeiro momento, realizamos a introdução de conceitos que são a base da teoria elementar dos números no capítulo intitulado: *Investigações aritméticas*, com uma abordagem contextualizada do algoritmo da divisão euclidiana e a recognição dos restos, que são fundamentais para a compreensão do conceito de congruência, trataremos da aritmética modular, na qual explanamos algumas de suas proposições e propriedades além de notáveis aplicações. Prosseguimos com o sistema completo de resíduos e as classes residuais, que transformam a congruência em uma igualdade de classes, o que simplifica substancialmente as operações aritméticas e são vitais na concepção

de algoritmos computacionais que, por sua vez, serão a base para sistematização das calculadoras por nós, a serem construídas.

O advento das novas tecnologias possibilitou ao homem o aperfeiçoamento e automação de diversas áreas por meio dos computadores. Para compreendermos o seu funcionamento, é crucial que apreendamos as linguagens implementadas em tais máquinas e, de que forma as informações são processadas. No capítulo denominado: *A álgebra das leis do pensamento*, explanamos sobre o conceito de base numérica, em especial, a base binária, que configura a linguagem utilizada para modelar instruções a serem inseridas nos computadores. Expendermos as operações aritméticas com números binários, bem com os métodos utilizados pelos computadores para efetuá-las.

Apresentamos a álgebra booleana que foi concebida com o propósito de algebrizar o pensamento humano e que se adéqua perfeitamente ao sistema operacional dos computadores, cujos desígnios de sua criação pressupunham a capacidade de imitar a mente humana. Desta forma, contemplaremos as nuances desta inusitada álgebra que fomenta toda a lógica computacional e incorpora aspectos tanto da lógica proposicional como da teoria dos conjuntos. Reportaremos aos componentes essenciais da eletrônica digital, tais como as portas lógicas, que são a tradução eletrônica dos conectivos concernentes à lógica clássica, além dos circuitos lógicos e o desenvolvimento de técnicas para a obtenção dos mesmos.

Trabalharemos meticulosamente com aparatos essenciais no desenvolver de projetos eletrônicos e na construção de calculadoras, que remetem ao tema central deste trabalho, no capítulo: *O cerne da calculadora*, no qual é feito um apanhado histórico a cerca da idealização e evolução das calculadoras. Seguimos com a análise referente às peculiaridades de uma curiosa calculadora binária mecânica, que foi o elemento impulsionador desta pesquisa e, prosseguimos com a elaboração de circuitos lógicos para somadores em diferentes bases numéricas, subtratores, multiplicadores, divisores, além de componentes e procedimentos vitais para a criação de calculadoras em uma base qualquer. Tais construções ficam disponíveis para *download* de forma que o leitor possa testá-las empiricamente e criar os próprios projetos por meio de simuladores.

Calculadoras são tão comuns e corriqueiras que poucos indagam como se dá seu funcionamento, projeto ou construção. Tal premissa consolida o principal objetivo desta obra, que consiste na construção de calculadoras em consonância com a aritmética das classes residuais, mais especificamente, calculadoras capazes de efetuar as operações aritméticas em uma base arbitrária.

Com a finalidade de despertar e instigar a curiosidade do leitor por esta envolvente disciplina, a matemática, os conteúdos são apresentados de forma a evidenciar sua relevância e aplicabilidade nas práticas sociais, que afiguram fator preponderante para sua inserção nos currículos da educação básica. A possibilidade de criar um minicomputador para efetuar determinadas tarefas, possibilita aos estudantes tonarem-se protagonistas, de fato, da própria aprendizagem. Os capítulos contam com inúmeros *hiperlinks* e referências para aprofundamento que proporcionam subsídios norteadores para os profissionais da educação que buscam tanto o contínuo aperfeiçoamento, como a retomada do ato de realizar pesquisas e produzir novos conhecimentos.

# 1

# INVESTIGAÇÕES ARITMÉTICAS

Neste capítulo, faremos a abordagem de tópicos fundamentais da teoria dos números. Trataremos do algoritmo da divisão euclidiana a fim de evidenciar a importância dos restos e suas propriedades, com o objetivo de introduzir o conceito de congruência, no qual, destacar-se-á sua relevância, bem como, algumas aplicações e resultados importantes. Por fim, versaremos sobre as classes residuais, suas propriedades e operações e, por meio destas, construiremos as tabuadas para a adição, que será a matriz para a "*programação*" das calculadoras a serem construídas.

Todos os conteúdos abordados e resultados explicitados podem ser apreciados com minucioso detalhamento nas referências que são indicadas no discorrer do trabalho.

## 1.1 O ADVENTO DA DIVISÃO

Suponha que um grupo de amigos, amantes de futebol e matemática foram ao estádio assistir à grande final. O grupo decide fazer uma arrecadação para comprar refrigerantes e salgadinhos para todos, o membro mais novo foi nomeado pelo clamor popular como o responsável em ir buscá-los. Como incentivo, todos concordam que ele pode ficar com o troco.

A quantia arrecadada soma R$80,00 entre notas de R$5,00 e R$10,00, isto posto, temos três possibilidades para esta quantia em relação ao valor que será gasto com suprimentos futebolísticos: insuficiente, suficiente e mais que suficiente. Este exemplo aparentemente trivial ilustra o teorema de Eudoxius de Cnido (408 a 355 a.C.), equivocadamente designado por princípio de Arquimedes[1] que nos diz:

1 Para saber mais leia [14]

**Teorema 1.1.** *Dados $a$ e $b$ números inteiros positivos, $a > b$, então $a$ é um múltiplo de $b$ ou $a$ está entre dois múltiplos consecutivos de $b$. Para $b$ positivo*[2]*, existe sempre um número inteiro $q$ em que: $q\cdot b \leq a < (q+1)\cdot b$*

*Demonstração:* Seja $\mathbb{A} = \{x\cdot b / x\cdot b > a,\ x \in \mathbb{N}\}$. Temos que $(a+1)\cdot b > a$, logo, o conjunto $\mathbb{A}$ é não-vazio, o *princípio da boa ordenação* nos garante a existência do elemento mínimo em todo conjunto não-vazio de inteiros positivos, dado $a > b$, segue que $b \notin \mathbb{A}$ o que acarreta em $x > 1$, ou seja, $x$ é sucessor de algum número natural. Tomemos o elemento mínimo de $\mathbb{A}$ (que será maior do que $a$) por $(q+1)\cdot b$, que, por sua vez, é maior que $q\cdot b$, ainda pelo *princípio da boa ordenação* $q\cdot b \notin \mathbb{A}$, do contrário, infringiria-se a minimalidade, segue, pela tricotomia que $q\cdot b \leq a$, e portanto:
$q\cdot b \leq a < (q+1)\cdot b$ □

Euclides de Alexandria (300 a.C.) utilizava este resultado tal como ele é apresentado no ensino fundamental, isto é, para realizar a divisão de um número $a$ por certo número $b$ *menor ou igual a* $a$ é preciso encontrar o *maior* múltiplo de $b$ que não ultrapasse o valor de $a$. Uma vez familiarizado com este resultado podemos enunciar o *Algoritmo da Divisão Euclidiana*[3]:

**Teorema 1.2** (Divisão Euclidiana)**.** *Dados $a$ e $b$ números inteiros, com $b$ positivo, existem inteiros $q$ e $r$ unicamente determinados tais que: $a = qb + r$ com $0 \leq r < b$*

*Demonstração:* A existência de $q$, denominado quociente, segue diretamente do teorema de Eudoxius (1.1), enquanto que para existência de $r$, o resto, basta subtrair $q\cdot b$ dos membros de sua desigualdade, na qual obtemos $0 \leq a - q\cdot b < b$, tomamos $a - q\cdot b$ por $r$, ou seja, a diferença entre o números a ser dividido e o maior múltiplo de $b$ menor que o referido número.

Quanto à unicidade, suponhamos a existência de $q_1$, $q_2$, $r_1$ e $r_2$ tais que: $a = q_1\cdot b + r_1$ e $a = q_2\cdot b + r_2$. Posto isto, temos $0 = a - a = (q_1\cdot b + r_1) - (q_2\cdot b + r_2) \Rightarrow (q_1 - q_2)\cdot b = (r_2 - r_1)$, assim, $(r_2 - r_1)$ é múltiplo de $b$ e dado que $r_2 < b$ e $r_1 < b$, seque que $r_2 - r_1 = 0$, o que resulta em $r_2 = r_1$, que por sua vez, suscita em $q_2 = q_1$, tendo em vista que $b \neq 0$ □

Assim, para realizarmos a divisão euclidiana, podemos efetuá-la por meio de subtrações sucessivas em busca do maior múltiplo de $b$ que seja menor ou igual a $a$, ou até

2 Para $b$ negativo, temos: $q\cdot b \leq a < (q-1)\cdot b$
3 Este e outros resultados podem ser apreciados em [34] e [35]

que o resto seja menor que $b$. Se o resto for nulo, então diremos que $a$ é múltiplo de $b$, ou ainda que $b$ divide $a$, cuja notação será $b|a$.

**Exemplo 1.1.** Caso os refrigerantes custem R$7,00, para determinar a maior quantidade de unidades possível de ser adquirida, basta subtrair sucessivamente 7 de 80 até que o resto seja menor que 7. Nestas condições temos $80 = 7 \cdot 11 + 3$, onze refrigerantes e R$3,00 de troco.

## 1.2 A IGUALDADE DAS DIFERENÇAS E A MÁQUINA DE REFRIGERANTES

Nas dependências do estádio há inúmeras máquinas de refrigerantes, uma delas contém as seguintes especificações:

ACEITAM-SE SOMENTE NOTAS DE:
R$5,00, R$10,00, R$20,00, R$50,00 E R$100,00
TROCO MÁXIMO: R$4,00

Suponha que os refrigerantes custem R$5,00 e o total de R$80,00 arrecadados em diferentes notas. A cada nota inserida na máquina, ela informa a quantidade máxima de refrigerantes que podem ser retirados da mesma, evidentemente, a máquina foi programada com o algoritmo da divisão por cinco e, como a mesma aceita somente valores múltiplos de cinco, ela jamais emitirá troco, contudo, mais adiante você avista outra máquina cujo preço de cada produto é R$4,00 e as especificações são as mesmas da máquina anterior. Novamente, para cada nota inserida a máquina informa a quantidade máxima de refrigerantes a retirar, bem como, o referido troco.

Desta vez ficamos intrigados, com quais algoritmos a máquina foi programada? Para facilitar nossa investigação, vamos reunir os dados em uma tabela:

| **valor** | **troco** | **valor** | **troco** | **valor** | **troco** | **valor** | **troco** |
|---|---|---|---|---|---|---|---|
| R$5,00 | R$ 1,00 | R$10,00 | R$ 2,00 | R$15,00 | R$ 3,00 | R$20,00 | R$ 0,00 |
| R$25,00 | R$ 1,00 | R$30,00 | R$ 2,00 | R$35,00 | R$ 3,00 | R$40,00 | R$ 0,00 |
| R$45,00 | R$ 1,00 | R$50,00 | R$ 2,00 | R$55,00 | R$ 3,00 | R$60,00 | R$ 0,00 |

Na coluna dos valores não há nada de mais, salvo progressões aritméticas de razão 20. Já na coluna do troco, identificamos um padrão, aliás, se considerarmos as linhas, surge algo interessante, pois seus valores obedecem a um ciclo que se renova a cada 4

elementos. Embora os valores de R$15,00, R$35,00 e R$55,00 sejam completamente distintos, a máquina de refrigerantes os considera como iguais, já que para tais quantias ela sempre fornecerá o mesmo troco, isto é, estes números sempre apresentam o mesmo resto na divisão por 4. O mesmo acontece para os demais valores. Como a máquina fornece troco máximo de R$4,00, podemos relacionar os valores a serem inseridos com os possíveis trocos a serem emitidos, que são R$1,00, R$2,00, R$3,00 e R$0,00, sendo que este último ocorre somente quando o valor inserido é múltiplo de R$4,00.

A relação matemática que considera essencialmente iguais os números que apresentam o mesmo resto na divisão por um número dado recebe um nome especial: ***congruência***.
Assim, podemos afirmar que os números $5, 25, 45$ são congruentes, pois todos apresentam resto 1 na divisão por 4.

Tal máquina de refrigerantes vai além. Imagine que alguém resolva desafiá-la a fornecer o troco rapidamente quando várias notas são inseridas. Suponha que o desafiante insira dezenove notas de R$10,00 e seis notas de R$50,00. Para sua infelicidade e prejuízo, a máquina informa prontamente o troco: R$2,00. A máquina, muito esperta por sinal, não calculará a soma dos produtos para depois efetuar a divisão, ela simplesmente substituirá cada valor pelo respectivo resto que este apresenta na divisão por 4, veja:

$$19 \cdot 10 \ + \ 6 \cdot 50 \ \textit{dividido por } 4 \tag{1.1}$$

$$3 \cdot 2 \ + \ 2 \cdot 2 \ \textit{dividido por } 4 \tag{1.2}$$

O resultado da *expressão (1.1)* é 10, que tem resto 2 quando dividido por 4. O fascinante é que *expressão (1.2)* fornece o mesmo resto 2, isto ocorre devido a máquina estar munida do ***Lema dos restos***, que nos diz:

**Lema 1.3** (Lema dos restos)**.** *A soma e/ou produto de números naturais quaisquer fornece o mesmo resto que a soma e/ou produto de seus respectivos restos.*

*Demonstração:* Sejam $m$ e $n \in \mathbb{Z}$, pela divisão euclidiana temos:
$m = a \cdot q_1 + r_1$ e $n = a \cdot q_2 + r_2$. Com $r_1$ e $r_2$ inteiros positivos menores que $a$. Então:

$$\begin{aligned} m+n &= a\cdot q_1 + r_1 + a\cdot q_2 + r_2 \\ &= a\cdot (q_1+q_2) + (r_1+r_2) \end{aligned} \tag{1.3}$$

$$\begin{aligned} m\cdot n &= (a\cdot q_1 + r_1)\cdot (a\cdot q_2 + r_2) \\ &= a^2\cdot q_1\cdot q_2 + a\cdot q_1\cdot r_2 + a\cdot q_2\cdot r_1 + r_1\cdot r_2 \\ &= a(a\cdot q_1\cdot q_2 + q_1\cdot r_2 + q_2\cdot r_1) + (r_1\cdot r_2) \end{aligned} \tag{1.4}$$

Em ambos os casos, o resto proveniente da divisão por $a$ depende exclusivamente de $r_1$ e $r_2$, logo o resto de $m+n$ será igual ao resto de $r_1+r_2$ e o resto de $m\cdot n$ será igual ao resto de $r_1\cdot r_2$. □

Temos agora em mãos uma ferramenta simples, porém poderosa, já que para problemas com valores altos podemos substituir e operar com os seus respectivos restos, cujos valores são menores e confortáveis de trabalhar.

### 1.2.1 *Calendários e Congruências*

Para ilustrar a versatilidade dos restos e das congruências, versaremos sobre uma aplicação interessante que encontra-se no artigo **Sexta-feira 13**[4] popularmente conhecida como o dia do azar. Esta data é repleta de crendices e lendas impregnadas de "maus agouros", como se não fosse o bastante, tal data ficou imortalizada com a sequência de filmes de terror "*Sexta-feira 13*", tendo como personagem principal o implacável *serial killer: Jason Voorhees*, que sempre ressuscita e ataca nesta data e por fim, torna a morrer! Será que Jason está fadado a este ciclo ininterrupto, em outras palavras, teria algum ano em que não há sextas-feiras 13 para que Jason tenha enfim uma folga?

Primeiramente, tomemos todos os dias "13" de cada mês, e determinaremos em que dia do ano ocorrerá cada um deles, bem como, seu respectivo resto decorrente da divisão euclidiana destes por 7.

4 Disponível em [17]

| **13/jan** | **13/fev** | **13/mar** | **13/abr** | **13/mai** | **13/jun** |
|---|---|---|---|---|---|
| 13°dia | 44°dia | 72°dia | 103°dia | 133°dia | 164°dia |
| resto:6 | resto:2 | resto:2 | resto:5 | resto:0 | resto:3 |
| **13/jul** | **13/ago** | **13/set** | **13/out** | **13/nov** | **13/dez** |
| 194°dia | 225°dia | 256°dia | 286°dia | 317°dia | 347°dia |
| resto:5 | resto:1 | resto:4 | resto:6 | resto:2 | resto:4 |

Observe que no ano de 2017, Jason terá trabalho dobrado, já que há duas sextas-feiras (janeiro e outubro), que apresentam resto 6. Como se não fosse o bastante, Jason jamais terá folga, pois todos os doze dias "13" abrangem os sete possíveis restos na divisão por 7 e, consequentemente, em todos os anos haverá ao menos uma sexta-feira 13 e desta forma, ele nunca descansará em paz. Não obstante, os anos em que o dia 13 de janeiro for uma terça-feira, serão o "*ano do Jason*", já que o mesmo contará com três sextas-feiras 13, pobre Jason!

## 1.3 O LEGADO DO PRÍNCIPE À RAINHA DAS CIÊNCIAS

Johann Carl Friedrich Gauss[5] (1777 - 1855) foi astrônomo e físico alemão mais conhecido como *príncipe dos matemáticos*. Uma de suas célebres frases era: "*a matemática é a rainha das ciências e a aritmética, a rainha da matemática*".

Em seu livro *Disquisitiones Arithmeticæ*[6], Gauss estudou a fundo os restos da divisão euclidiana sob um ponto de vista estrutural. O primeiro capítulo de sua obra introduz a definição de congruência, juntamente com a notação a seguir, que é utilizada até os dias atuais:

**Definição 1.4.** Seja $m$ um número inteiro, $m \neq 0$, dois números $a$ e $b$ são ditos *congruentes módulo m* se apresentam o mesmo resto na divisão por $m$ e, denotamos por:

$$a \equiv b \ mod \ m$$

Quando $a$ e $b$ apresentarem restos distintos na divisão por $m$, diremos que os mesmo são incongruentes módulo $m$, o que se denota por:

$$a \not\equiv b \ mod \ m$$

5 Para saber mais, recomendamos [14]

6 Investigações Aritméticas em latim

Contudo, dados $a$ e $b$ números inteiros, não é preciso realizar a divisão euclidiana e comparar seus restos para verificar se tais números são ou não congruentes. Para tanto podemos utilizar a proposição que segue:

**Proposição 1.5.** *Tem-se que $a \equiv b \ mod \ m$, se, e somente se, $m \mid (a-b)$, isto é, se $m$ divide a diferença de $a$ por $b$.*

*Demonstração:* Se $a \equiv b \ mod \ m$, pela divisão euclidiana existem $r, q_1$ e $q_2$, tais que $a = mq_1 + r$ e $b = mq_2 + r$, cuja diferença $a - b = m(q_1 - q_2)$, como $a - b$ é múltiplo de $m$, segue que $m \mid (a-b)$.

Reciprocamente, se $m \mid (a-b)$, ainda pela divisão euclidiana, existem $q_1, q_2, r_1$ e $r_2$ tais que $a = mq_1 + r_1$ e $b = mq_2 + r_2$, com $0 \leq r_1 < m$ e $0 \leq r_2 < m$. Segue que $a - b = m(q_1 - q_2) + (r_1 - r_2)$, Daí $m \mid (q_1 - q_2)$ e $m \mid (r_1 - r_2)$, por hipótese, como os restos são menores do que $m$, a única possibilidade é $m \mid 0$, logo $r_1 = r_2$ e portanto $a \equiv b \ mod \ m$. □

De modo equivalente podemos conceber a congruência $a \equiv b \ mod \ m$ como:

- a menos de um múltiplo de $m$, $a$ é igual a $b$, isto é, $a = b + mq$;
- A diferença $(a - b)$ é múltipla de $m$;
- $m$ é um divisor da diferença de $a$ por $b$, isto é, $m \mid (a - b)$
- $a$ e $b$ apresentam o mesmo resto na divisão por $m$.

A utilização da congruência é extremamente útil, pois, além de ser compatível com as operações de adição e multiplicação, trata-se de uma relação de equivalência, na qual, elementos distintos são iguais naquilo que valem, por isso "*equivalem*".

Na relação de equivalência (*onde $\sim$ é o símbolo genérico para a relação entre elementos quaisquer de um conjunto arbitrário*), três propriedades devem estar satisfeitas:
Dados $x, y$ e $z \in \mathbb{A}$ temos que:
**Reflexiva:** $x \sim x$;
**Simétrica:** se $x \sim y$, então $y \sim x$;
**Transitiva:** se $x \sim y$ e $y \sim z$ então $x \sim z$.
Verificaremos a seguir que a relação de congruência, $a \equiv b \ mod \ m$, é de fato, uma relação de equivalência.

*Reflexiva:* $a \equiv a \ mod \ m$, pois $a - a = 0$ e $m|0$. □

*Simétrica:* Se $a \equiv b \ mod \ m$, então $m|(a-b)$, consequentemente, $m|(-(a-b))$. Logo $m|(b-a)$ e $b \equiv a \ mod \ m$. □

*Transitiva:* Se $a \equiv b \ mod \ m$ e $b \equiv c \ mod \ m$, então $m|(a-b)$ e $m|(b-c)$. Logo $m|(a-b)+(b-c)$ o que implica em $m|(a-c)$ e portanto $a \equiv c \ mod \ m$. □

### 1.3.1 *A aritmética dos restos*

Gauss desenvolveu a álgebra da relação de congruência que nada mais é do que a *aritmética dos restos* da divisão euclidiana. Veremos alguns resultados[7] importantes inerentes à congruência.

Primeiramente, como todo e qualquer número apresenta *resto nulo* na divisão euclidiana por 1, iremos considerar $m > 1$ e, cabe ressaltar que todo e qualquer número é congruente ao próprio resto na divisão por $m$.
Sejam $a, b, c, d$ e $m$ inteiros, $m > 1$ temos:

**Proposição 1.6** (Adição). *Se $a \equiv b \ mod \ m$ e $c \equiv d \ mod \ m$, então $a+c \equiv b+d \ mod \ m$.*
"*A soma de dois números e, a soma de seus respectivos restos módulo m, apresentam o mesmo resto módulo m*".

*Demonstração:* Se $a \equiv b \ mod \ m$ e $c \equiv d \ mod \ m$, então $m \mid (a-b)$ e $m \mid (c-d)$, consequentemente $m|(a-b)+(c-d)$. Logo $m|(a+c)-(b+d)$, segue que $a+c \equiv b+d \ mod \ m$. □

**Proposição 1.7** (Subtração). *Se $a \equiv b \ mod \ m$ e $c \equiv d \ mod \ m$, então $a-c \equiv b-d \ mod \ m$.*
"*A diferença de dois números e, a diferença de seus respectivos restos módulo m, apresentam o mesmo resto módulo m*".

*Demonstração:* Se $a \equiv b \ mod \ m$ e $c \equiv d \ mod \ m$, então $m \mid (a-b)$ e $m \mid (c-d)$, consequentemente $m|(a-b)-(c-d)$. Logo $m|(a-c)-(b-d)$, segue que $a-c \equiv b-d \ mod \ m$. □

**Proposição 1.8** (Multiplicação). *Se $a \equiv b \ mod \ m$ e $c \equiv d \ mod \ m$, então $ac \equiv bd \ mod \ m$.*
"*O produto de dois números e, o produto de seus respectivos restos módulo m, apresentam o mesmo resto módulo m*".

7 Esta e outras preposições e suas demonstrações podem ser apreciadas em [11], [21].

*Demonstração:* Se $a \equiv b \ mod \ m$ e $c \equiv d \ mod \ m$, então $m \mid (a-b)$ e $m \mid (c-d)$. Dado que $ac - bd = a(c-d) + d(a-b)$ temos $m|(ac-bd)$ e portanto $ac \equiv bd \ mod \ m$. □

**Proposição 1.9** (Exponencial)**.** *Se $a \equiv b \ mod \ m$ e $n$ é um número natural qualquer, então $a^n \equiv b^n \ mod \ m$.*
*"A potência de um número por $n$ e, a potência de seu respectivo resto módulo $m$ por $n$, apresentam o mesmo resto módulo $m$".*

*Demonstração:* Dado que $a^n - b^n = (a-b)\cdot(a^{n-1} + a^{n-2}b + a^{n-3}b^2 + \ldots + ab^{n-2} + b^{n-1})$, como $m \mid (a-b)$ segue que $m \mid (a^n - b^n)$ e portanto $a^n \equiv b^n \ mod \ m$. □

E quanto à divisão? Será que a lei do cancelamento é válida para as congruências? O resultado a seguir nos diz quando é possível realizar o cancelamento multiplicativo.

**Proposição 1.10** (Cancelamento)**.** *Se $ac \equiv bc \ mod \ m$ e $(c,m) = 1$, então $a \equiv b \ mod \ m$, onde $(c,m)$ é o máximo divisor comum entre $c$ e $m$, ou seja, o cancelamento somente é válido se $c$ e $m$ são primos entre si.*

*Demonstração:* Se $ac \equiv bc \ mod \ m$, então $m|(ac-cb)$, isto é, $m|c(a-b)$. Como $mdc(c,m) = 1$, segue que $m$ e $c$ são primos entre si, e obrigatoriamente, $m|(a-b)$ e portanto $a \equiv b \ mod \ m$. □

#### 1.3.1.1 *Dígito verificador*

Amplamente empregado e difundido, o *dígito verificador* é uma importante ferramenta para validar e autenticar documentos importantes baseados em sequências numéricas. O dígito verificador é obtido por meio de algoritmos específicos aplicados aos demais dígitos do referido código, vamos conhecer a matemática que cinge os dígitos verificadores:

##### *1.3.1.1.1 O cartão de crédito*

No Brasil, os cartões de crédito costumam apresentar 16 dígitos[8], sendo que os primeiros designam a bandeira emissora, os dígitos subsequentes definem o cliente e o último dígito é o verificador. Sem mais delongas, vejamos os procedimentos para

8 Em outros países há cartões que apresentam de 14 a 19 dígitos.

determinar o dígito verificador de um cartão cujo número é constituído da sequência $a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, a_{14}, a_{15}, a_{16}$, sendo $a_{16}$ o dígito verificador.

Desta sequência, tomamos os dígitos de índice ímpar e calculamos $q_i$ como segue:

- $q_i = 2 \cdot a_i$ se $2 \cdot a_i \leq 9$;
- $q_i = 2 \cdot a_i - 9$ se $2 \cdot a_i > 9$.

A soma $S$ que buscamos é o somatório destes $q_i$ juntamente com o somatório dos $a_i$ de índice par, ou seja:

$$S = \sum_{i=1}^{8} q_{2i-1} + \sum_{i=1}^{8} a_{2i}$$

Por fim, o dígito verificador é obtido da seguinte forma: $(S + a_{16}) \equiv 0 \ mod\ 10$, ou ainda $a_{16} \equiv -S \ mod\ 10$.

**Exemplo 1.2.** Vamos determinar o dígito verificador de um cartão de crédito fictício, cujo 15 primeiros dígitos são: *1234 5678 9012 345*$a_{16}$. Comecemos pelos $q_i$.

$q_1 = 2 \cdot a_1 = 2 \cdot 1 \leq 9 \Rightarrow q_1 = 2$
$q_3 = 2 \cdot a_3 = 2 \cdot 3 \leq 9 \Rightarrow q_3 = 6$
$q_5 = 2 \cdot a_5 = 2 \cdot 5 > 9 \Rightarrow q_5 = 2 \cdot 5 - 9 = 1$
$q_7 = 2 \cdot a_7 = 2 \cdot 7 > 9 \Rightarrow q_7 = 2 \cdot 7 - 9 = 5$
$q_9 = 2 \cdot a_9 = 2 \cdot 9 > 9 \Rightarrow q_9 = 2 \cdot 9 - 9 = 9$
$q_{11} = 2 \cdot a_{11} = 2 \cdot 1 \leq 9 \Rightarrow q_{11} = 2$
$q_{13} = 2 \cdot a_{13} = 2 \cdot 3 \leq 9 \Rightarrow q_{13} = 6$
$q_{15} = 2 \cdot a_{15} = 2 \cdot 5 > 9 \Rightarrow q_{15} = 2 \cdot 5 - 9 = 1$

Agora podemos aplicar o algoritmo citado anteriormente:

$$S = \sum_{i=1}^{8} q_{2i-1} + \sum_{i=1}^{8} a_{2i} = (2+6+1+5+9+2+6+1) + (2+4+6+8+0+2+4) = 58$$

Segue que $(58 + a_{16}) \equiv 0 \ mod\ 10$ ou $a_{16} \equiv -58 \ mod\ 10$, e como $-58 = (-6) \cdot 10 + 2$, temos $a_{16} \equiv 2 \ mod\ 10$, donde segue que o dígito verificador $a_{16} = 2$.

O que aconteceria se digitássemos o número errado, trocando a ordem de dois dígitos consecutivos, por exemplo, referente aos números do cartão anterior, caso fosse digitado *(**21**34 5678 9012 3452)* teríamos:
$S = (4+6+1+5+9+2+6+1) + (1+4+6+8+0+2+4) = 59$, logo $(S + a_{16}) = (59 + 2) = 61 \not\equiv 0 \ mod\ 10$. O que torna tal cartão inválido.

*1.3.1.1.2 CPF*

Todos os cidadãos do Brasil possuem ou, ao menos, deveriam possuir o *cadastro de pessoa física (CPF)* que retém informações do contribuinte no sistema da receita federal. O número do *(CPF)* é constituído de nove dígitos seguidos de dois dígitos verificadores, sendo que o nono dígito identifica a região fiscal de emissão do documento.

| **9º dígito** | **Região Fiscal** |
|---:|---|
| 0 | RS |
| 1 | DF, GO, MS, MT, TO |
| 2 | AC, AM, AP, PA, RO, RR |
| 3 | CE, MA, PI |
| 4 | AL, PB, PE, RN |
| 5 | BA, SE |
| 6 | MG |
| 7 | ES, RJ |
| 8 | SP |
| 9 | PR, SC |

Dada a sequência de dígitos que compõe o *(CPF)*: $a_1a_2a_3a_4a_5a_6a_7a_8a_9 - a_{10}a_{11}$, para determinar o valor do primeiro dígito verificador $a_{10}$, devemos efetuar a seguinte soma $S$:

$$S = \sum_{i=1}^{9} i \cdot a_i,$$

**Exemplo 1.3.** Considere o seguinte número fictício de *(CPF)*: **987.321.456**, vamos determinar o primeiro dígito verificador:

$$S = \sum_{i=1}^{9} i \cdot a_i = 1 \cdot 9 + 2 \cdot 8 + 3 \cdot 7 + 4 \cdot 3 + 5 \cdot 2 + 6 \cdot 1 + 7 \cdot 4 + 8 \cdot 5 + 9 \cdot 4 = 196$$

Temos $S - a_{10}$ deve ser divisível por 11, ou de modo equivalente $11|(S - a_{10})$ ou ainda $a_{10} \equiv S \ mod\ 11$.
Assim $a_{10} \equiv 196 \ mod\ 11$ e como $196 = 17 \cdot 11 + 9$ segue que $a_{10} = 9$.

Para determinar o segundo dígito verificador $a_{11}$ vamos proceder da seguinte forma, calculamos uma segunda soma $\overline{S} = \sum_{i=1}^{10}(i-1) \cdot a_i$, e de modo análogo, uma vez efetuados os cálculos, deveremos ter $a_{11} \equiv \overline{S} \ mod\ 11$. Logo:

$$\overline{S} = \sum_{i=1}^{10}(i-1) \cdot a_i = 0 \cdot 9 + 1 \cdot 8 + 2 \cdot 7 + 3 \cdot 3 + 4 \cdot 2 + 5 \cdot 1 + 6 \cdot 4 + 7 \cdot 5 + 8 \cdot 6 + 9 \cdot 9 = 232$$

Donde, temos $a_{11} \equiv 232 \ mod\ 11$ e como $232 = 21 \cdot 11 + 1$ segue que $a_{11} = 1$ e consequentemente, o número de *CPF* que buscamos é **987.321.456-91**.

Todavia, há uma pequena ressalva, caso o resto da divisão de algum dos dígitos verificadores for 10, então o mesmo será designado por 0.

Em caso de alguma das congruências não se verificarem, teremos então um o número de *CPF*[9] inválido.

## 1.4 SISTEMA COMPLETO DE RESÍDUOS

Passaremos a designar resto $r$ como sendo um *resíduo módulo* $m$. Na "congruência do calendário" vimos que ao dividir um número inteiro $a$ por 7, há ao todo sete possíveis restos: $0, 1, 2, 3, 4, 5, 6$, em que cada um deles corresponde a um único dia da semana:

| **domingo** | **segunda** | **terça** | **quarta** | **quinta** | **sexta** | **sábado** |
|---|---|---|---|---|---|---|
| 1/jan | 2/jan | 3/jan | 4/jan | 5/jan | 6/jan | 7/jan |
| 8/jan | 9/jan | 10/jan | 11/jan | 12/jan | 13/jan | 14/jan |
| 15/jan | 16/jan | 17/jan | 18/jan | 19/jan | 20/jan | 21/jan |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

Todos os dias que caem em uma terça-feira, por exemplo, apresentam resíduo 3 módulo 7, ou ainda são todos os números da forma $7n + 3$, com $n$ inteiro, os demais dias comportam-se de maneira análoga. Desta forma, cada um dos dias do ano pertencerá a uma das sete colunas, isto é, cada um dos números inteiros[10] são congruentes a um único resíduo [11] módulo 7.

**Definição 1.11** (Sistema completo de resíduos)**.** Conjunto de todos os possíveis restos da divisão euclidiana de um inteiro $a$ por $m$, com $m > 1$, cujos $m$ elementos: $\{0, 1, 2, \cdots m - 1\}$, em uma ordem qualquer, são dois a dois, incongruentes módulo $m$ será denominado por *Sistema Completo de Resíduos* módulo $m$.

Devido à congruência podemos substituir qualquer um dos elementos do sistema completo de resíduos por um de seus representantes, além disto, qualquer conjunto

9 Para uma leitura mais detalhada, indicamos [26]

10 Podemos representar os números negativos da seguinte forma: $-1 = 7 \cdot (-1) + 6$; $-18 = 7 \cdot (-3) + 3$.

11 Note que os elementos de cada coluna formam uma progressão aritmética de razão $m$, infinita para ambos os lados e, o resíduo em questão será um de seus termos.

de $m$ elementos da forma $\{r_1, r_2, r_3, \cdots r_m\}$, também será um sistema completo de resíduos desde que satisfaça:

- $r_i \not\equiv r_j \ mod\ m$ *para* $i \neq j$;
- *Para todo inteiro* $a$*, existe um* $r_i$*, tal que,* $a \equiv r_i \ mod\ m$.

### 1.4.1 *Guardando o infinito em gavetas*

**Definição 1.12** (Classes residuais)**.** Consideremos a classe de restos $r$ módulo $m$, que será representado por $[r]$ como sendo o conjunto de todos os inteiros que apresentam o mesmo resto $r$ na divisão euclidiana por $m$, isto é:

$$[r] = \{a \in \mathbb{Z} : a \equiv r \ mod\ m\} = \{a \in \mathbb{Z} : m \mid a - r\} = \{r + km : k \in \mathbb{Z}\}$$

Uma prerrogativa de se trabalhar com as classes residuais reside em que as mesmas transformam a congruência $a \equiv b \ mod\ m$ na igualdade $[a] = [b]$, assim concebemos as classes residuais módulo $m$ de um elemento $r \in \mathbb{Z}$ como sendo a classe de equivalência de $r$ segundo a relação de equivalência dada pela congruência módulo $m$, que devido a isto, goza das seguintes propriedades:

- A classe $[a]$ é igual à classe $[b]$ tão somente se $a \equiv b \ mod\ m$;
- As classes residuais módulo $m$ são conjuntos disjuntos;
- A reunião de todas as classes residuais módulo $m$ é igual a $\mathbb{Z}$ .

Segue que o conjunto dos inteiros $\{0, 1, 2, 3, \cdots, m-1\}$ é um sistema completo de resíduos se, e somente se $[0], [1], [2], [3], \cdots, [m-1]$ são as $m$ classes residuais módulo $m$, que atuam tal como um "*ficheiro*" com $m$ "*gavetas*" nas quais organizaremos cada um dos elementos de $\mathbb{Z}$, que por sua vez, fica particionado em $m$ subconjuntos formados por inteiros mutualmente congruentes módulo $m$.

**Exemplo 1.4.** Para $m = 2$ temos duas classes residuais para $\mathbb{Z}_2$:
$[0] = \{x \in \mathbb{Z} : x \equiv 0 \ mod\ 2\} = x$ *é par* e;
$[1] = \{x \in \mathbb{Z} : x \equiv 1 \ mod\ 2\} = x$ *é ímpar*.
Destarte, duas classes residuais, disjuntas e cuja reunião é o próprio $\mathbb{Z}$.

#### 1.4.1.1 *Casas, pombos e mais pombos...*

Como dito anteriormente, um conjunto de $m$ elementos, dois a dois, incongruentes módulo $m$ formam um *Sistema Completo de Resíduos* módulo $m$, pois bem, mas o que aconteceria se tomássemos $m+1$ elementos? Certamente haveria ao menos dois elementos que apresentariam o mesmo resto na divisão euclidiana por $m$, ou seja, seriam congruentes módulo $m$, ou ainda, pertenceriam à mesma classe de restos, e por que não, ficariam guardados na mesma gaveta.

Embora pareça elementar, este interessante fato mostra-se extremamente útil para garantir a existência de elementos em conjuntos, desde que satisfaça determinadas exigências. Proposto por Dirichlet[12] como *Princípio das Gavetas*, o mesmo ficou conhecido como *Princípio da Casa dos Pombos*.

Repleto de sutilezas, o *Princípio da Casa dos Pombos* possui inúmeras aplicações na Geometria, Combinatória e é claro, na Teoria dos Números, tal princípio é bem simples:

**Teorema 1.13** (Princípio da casa dos pombos)**.** *Suponha que você tenha $p$ pombos e $m$ casas, de forma que $p$ seja maior que $m$, então certamente, ao menos uma casa será ocupada por mais de um pombo.*

*Demonstração:* Temos mais pombos do que casas, ($p > m$). Suponha, pelo contrário, que em cada casa não tenha mais do que um pombo, então, ao contarmos todos os pombos das $m$ casas não teremos mais do que $m$ pombos, o que contradiz o fato de termos $p$ pombos distribuídos em $m$ casas. □

Pode parecer trivial, ou mesmo ingênuo, dado que qualquer criança é capaz de compreender isto, contudo, este princípio é um importante aliado para atacar inúmeros problemas matemáticos nem um pouco triviais.

Por meio do versátil *princípio da casa dos pombos* vamos demonstrar o teorema de Bachet-Bézout:

**Teorema 1.14** (Teorema de Bachet-Bézout)**.** *Se $d$ é o $mdc$ de $a$ e $b$, então existem números inteiros $x$ e $y$ tais que, $d = mdc(a,b) = ax + by$.*

*Demonstração:* Tome sem perda de generalidade que $d = mdc(a,b) = 1$. Considere o conjunto $\mathbb{A} = \{a, 2a, 3a, \cdots ba\}$ deve existir ao menos um dos elementos de $\mathbb{A}$ que

12 Considerações adicionais e material para aprofundar-se sobre o tema pode ser encontrado em [11], [14], [15] e [34]

apresente resto 1 quando dividido por $b$. Pelo princípio da casa dos pombos suponha que haja dois elementos $ai$ e $aj$ com $b > j > i \geqslant 1$ que deixem o mesmo resto na divisão por $b$, dado que $aj \equiv ai \ mod \ b$, segue da proposição 1.10 que $b \mid (aj - ai) = a(j - i)$ e como $mdc(a, b) = 1$ temos que $b$ divide $(j - i) > 0$, todavia $b > (j - i)$, absurdo!

Claramente, os elementos de $\mathbb{A}$ formam um sistema completo de resíduos módulo $m$ e desta forma, apenas um de seus elementos apresenta resto 1 quando dividido por $b$, seja $ax$ tal elemento, segue que $ax - 1$ é múltiplo de $b$, ou seja, $ax - 1 = by \Rightarrow ax - by = 1$, o que conclui a demonstração. □

### 1.4.2 *O conjunto das classes residuais*

O conjunto formado pela reunião de todas as classes residuais módulo $m$ será denotado por $\mathbb{Z}_m$, que por sua vez é uma partição de $\mathbb{Z}$, para encontrar a qual classe residual pertence um inteiro $a$ qualquer, basta determinar o resto $r$ da divisão euclidiana de $a$ por $m$.

A figura 1 ilustra o comportamento de $\mathbb{Z}_m$, no qual imaginamos o enrolar da reta numérica $\mathbb{Z}$ sobre um círculo com $m$ pontos marcados, de tal forma que o ponto $m$ coincida com o ponto 0, o ponto $m+1$ coincida com o ponto 1, o ponto $m+2$ coincida com o ponto 2 e assim sucessivamente. Como $\mathbb{Z}_m$ permanece "intacto" ele preserva as operações usuais.

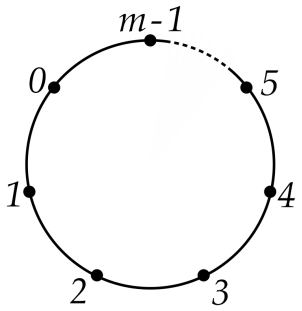


Figura 1: Comportamento de $\mathbb{Z}_m$

#### 1.4.2.1 *Operações em $\mathbb{Z}_m$*

A partição de $\mathbb{Z}$ em $m$ classes residuais é vantajosa não só por transformar a congruência em uma igualdade, mas também por ser compatível com as operações de adição e multiplicação.

**Exemplo 1.5.** Vamos tomar como exemplo as classe de congruência módulo 5 e alguns de seus representantes:

$$[0] = \{\cdots, -10, -5, 0, 5, 10, \cdots\}$$
$$[1] = \{\cdots, -9, -4, 1, 6, 11, \cdots\}$$
$$[2] = \{\cdots, -8, -3, 2, 7, 12, \cdots\}$$
$$[3] = \{\cdots, -7, -2, 3, 8, 13, \cdots\}$$
$$[4] = \{\cdots, -6, -1, 4, 9, 14, \cdots\}$$

Ao adicionar um elemento de [2] com um elemento de [4] a soma será $[2+4] = [6] = [1]$. A multiplicação é análoga, $[2]\cdot[4]$ obtemos como produto $[2 \cdot 4] = [8] = [3]$.

Por definição, se $[a] = [b]$ e $[c] = [d]$, então $a \equiv b \ mod \ m$ e $c \equiv d \ mod \ m$, e pelas proposições (1.6) e (1.8) temos que $[a+b] = [c+d]$ e $[a\cdot b] = [c\cdot d]$ e, desta forma, podemos definir as operações de adição e multiplicação em $\mathbb{Z}_m$, que devido à relação de congruência será válida, independentemente da escolha do representante da classe.

##### *1.4.2.1.1 Tabuadas em $\mathbb{Z}_m$*

Podemos sintetizar as operações binárias em tabuadas.

Tabuadas da adição e multiplicação em $\mathbb{Z}_2$

| + | [0] | [1] |
|---|---|---|
| [0] | [0] | [1] |
| [1] | [1] | [0] |

| · | [0] | [1] |
|---|---|---|
| [0] | [0] | [0] |
| [1] | [0] | [1] |

Tabuadas da adição e multiplicação em $\mathbb{Z}_3$

| + | [0] | [1] | [2] |
|---|---|---|---|
| [0] | [0] | [1] | [2] |
| [1] | [1] | [2] | [0] |
| [2] | [2] | [0] | [1] |

| · | [0] | [1] | [2] |
|---|---|---|---|
| [0] | [0] | [0] | [0] |
| [1] | [0] | [1] | [2] |
| [2] | [0] | [2] | [1] |

Tabuadas da adição e multiplicação em $\mathbb{Z}_4$

| + | [0] | [1] | [2] | [3] |
|---|---|---|---|---|
| [0] | [0] | [1] | [2] | [3] |
| [1] | [1] | [2] | [3] | [0] |
| [2] | [2] | [3] | [0] | [1] |
| [3] | [3] | [0] | [1] | [2] |

| · | [0] | [1] | [2] | [3] |
|---|---|---|---|---|
| [0] | [0] | [0] | [0] | [0] |
| [1] | [0] | [1] | [2] | [3] |
| [2] | [0] | [2] | [0] | [2] |
| [3] | [0] | [3] | [2] | [1] |

Tais tabuadas são de extrema importância para este trabalho, já que serão o ponto de partida para elaboração dos circuitos lógicos que devem necessariamente apresentar os resultados em consonância com o exposto acima, tal procedimento será de suma importância para a construção das calculadoras.

#### *1.4.2.1.2 Algumas propriedades*

As operações citadas na seção anterior gozam das seguintes propriedades[13]:

*Adição*

**A1**-*Comutativa*: $[a]+[b]=[b]+[a]$.
**A2**-*Associativa*: $([a]+[b])+[c]=[a]+([b]+[c])$.
**A3**-*Elemento neutro*: $[a]+[0]=[0]+[a]=[a]$.
**A1**-*Simétrico*: $[a]+[-a]=[0]$.

*Multiplicação*

**M1**-*Comutativa*: $[a]\cdot[b]=[b]\cdot[a]$.
**M2**-*Associativa*: $([a]\cdot[b])\cdot[c]=[a]\cdot([b]\cdot[c])$.
**M3**-*Elemento neutro*: $[a]\cdot[1]=[1]\cdot[a]=[a]$.

*Adição / Multiplicação*

**D1**-*Distributiva*: $[a]\cdot([b]+[c])=[a]\cdot[b]+[a]\cdot[c]$.

Um conjunto $G$ munido de uma operação binária $*$ que satisfaça as propriedades **A1**, **A2** e **A3** possui a estrutura algébrica de grupo. Caso **A4** também se verifique, então $G$ é denominado grupo comutativo ou abeliano.

13 As demonstrações serão omitidas e estão disponíveis em [19].

**Definição 1.15.** Um elemento $a \in G$ é dito gerador do grupo $(G, *)$ se, para qualquer $a \in G$ existe $\overline{m} \in G$, tal que $\overline{m} = a * m$.

**Definição 1.16.** Todo grupo $(G, *)$ gerado por um único elemento $a \in G$ será denominado grupo cíclico, e o designaremos por $< a >$.

**Exemplo 1.6.** $(\mathbb{Z}_2, +)$ é um grupo cíclico gerado por [1], pois para qualquer $[a] \in \mathbb{Z}_2$, $(a \in \mathbb{Z})$, $[a] = a \cdot [1]$.

**Definição 1.17.** A ordem de um grupo $G$ gerado por $a \in G$ é o menor inteiro positivo $k$ tal que $a^k = m$. A ordem de $G$ é dita finita se existirem inteiros positivos distintos $r$, $s$ tais que $a^r = a^s$.

Para as classes residuais, a definição 1.17 remete à equivalência de classes, trataremos agora de um teorema que será de grande valia mais adiante.

**Teorema 1.18** (Teorema de Cauchy para grupos abelianos[14])**.** *Seja* $(G, *)$ *um grupo abeliano finito. Se* $p$ *é primo e divide a ordem de* $G$*, então existe* $a \in G$*,* $a \neq e$*, tal que* $a^p = e$*,(*$e$ *consiste no elemento neutro de* $G$*).*

Lembramos que $a^p = a \cdot a \cdot a \cdot$ ..., para um grupo multiplicativo e $a^p = a + a + a +$ ..., para um grupo aditivo. Além disto, no conjunto das classes residuais, $e = [0] = [m]$.

Segue que, $\mathbb{Z}_m$ com as operações binárias $(+, \cdot)$ satisfazendo as propriedades **A1, A2, A3, A4, M1, M2, M3** e **D1** possui a estrutura algébrica de um anel[15], denominado ***anel das classes residuais módulo m***.

As próximas propriedades tratam do comportamento do elemento neutro da adição na multiplicação, e do inverso multiplicativo:

**D0**-*Divisor de zero*: $[a] \cdot [b] = [0]$, então $[a] = [0]$ ou $[b] = [0]$.
**M4**-*Inverso multiplicativo*: Existe $[a] \in \mathbb{Z}_m$ tal que, $[a] \cdot [b] = [1]$.

Em $\mathbb{Z}_m$, $[a]$ *(não nulo)* será um divisor de zero se existir $[b]$ *(não nulo)* tal que $[a] \cdot [b] = [0]$. O fato é que um divisor de zero nunca é invertível. Vejamos: suponha que $[a] \in \mathbb{Z}_m$ seja simultaneamente um divisor de zero e invertível, logo existiriam $[b]$ e $[c] \in \mathbb{Z}_m$ tais que, $[a] \cdot [b] = [0]$ e $[a] \cdot [c] = [1]$, entretanto $[0] = [c] \cdot [0] = [c] \cdot ([a] \cdot [b]) = ([c] \cdot [a]) \cdot [b] = [1] \cdot [b] = [b]$ o que é um absurdo!

14 Considerações adicionais sobre estas definições, teoremas e demonstrações podem ser encontrados em [8]

15 Recomendamos [22] para aprofundamento acerca deste tema.

O Anel em que se verifiquem as propriedades de **A1** à **M3**, e também **D0** será tratado como um ***domínio de integridade***. Note que $\mathbb{Z}_4$ não é um domínio de integridade por apresentar divisores de zero, uma vez em que $[2] \neq [0]$, entretanto, $[2] \cdot [2] = [0]$, o mesmo vale para $\mathbb{Z}_6$, pois, $[2], [3]$ e $[4] \neq [0]$, entretanto, $[2] \cdot [3] = [3] \cdot [4] = [0]$.

**Teorema 1.19.** *Temos que $[a] \in \mathbb{Z}_m$ será invertível se, e somente se, $mdc(a, m) = 1$, isto é, quando $a$ e $m$ forem primos entre si.*

*Demonstração:* Se $mdc(a, m) = 1 \Leftrightarrow$ Existem $x$ e $y$ inteiros, tais que $ax + my = 1$ (Relação de Bachet-Bézout) $\Leftrightarrow ax \equiv 1 \ mod \ m \Leftrightarrow [a \cdot x] = [1] \Leftrightarrow [a]$ é invertível. □

Todavia, $\mathbb{Z}_2$, $\mathbb{Z}_3$ e $\mathbb{Z}_5$ são domínios de integridade, uma vez em que **M4** se verifica, e desta forma, os mesmos se enquadram na estrutura algébrica de ***Corpo***.[16]

**Corolário 1.20.** *$\mathbb{Z}_m$ é um corpo se, e somente se, $m$ é primo.*

*Demonstração:* Suponha, por absurdo, que $\mathbb{Z}_m$ é um corpo e $m$ não é primo, logo, temos que $m = m_1 \cdot m_2$, com $1 < m_1 < m$ e $1 < m_2 < m$. Contudo, $[0] = [m] = [m_1] \cdot [m_2]$, contradição!
Reciprocamente, suponha $m$ primo, e $mdc(k, m) = 1$ para $k = 1, 2, 3, \cdots, m - 1$, segue do teorema 1.19 que $[1], [2], [3], \cdots, [m - 1]$ são invertíveis e, consequentemente $\mathbb{Z}_m$ é um corpo. □

Disto, verificamos que há certas restrições que devem ser consideradas na elaboração de uma "máquina" programada para efetuar multiplicações, especialmente no que diz respeito quanto à utilização do inverso multiplicativo, o mesmo já não ocorre com a operação de adição, que será o foco deste trabalho mais adiante.

## 1.5 O GIGANTESCO PEQUENO TEOREMA DE FERMAT

Este é um teorema dos mais notáveis e úteis na resolução de problemas na Aritmética, Pierre de Fermat[17] (1607 - 1665) foi um entusiasta matemático e cientista francês e, apenas enunciou este teorema em uma de suas cartas, sua demonstração ocorreu quase cem anos depois e coube a Leonard Euler (1707-1783) matemático, físico, astrônomo, lógico e engenheiro suíço:

16 Para analisar estes e outros resultados, indicamos [22]
17 Para uma leitura aprofundada recomendamos [11], [21] e [40]

**Teorema 1.21** (Pequeno Teorema de Fermat)**.** *Seja $p$ um número primo, se $p \nmid a$ então $a^{p-1} \equiv 1 \ mod \ p$.*

*Demonstração:* Primeiramente, observe que ao dividir um número inteiro qualquer por um número primo $p$ há ao todo $p$ possíveis restos: $0, 1, 2, 3, \cdots, (p-1)$, sendo que $0$ ocorre somente quanto tal número é múltiplo de $p$, e portanto, vamos desconsiderá-lo. Neste conjunto de $p-1$ elementos, nenhum deles é divisível por $p$ e todos são, dois a dois, incongruentes módulo $p$. Tome $a$ um número natural tal que $p$ não divida $a$, e considere o conjunto formado pelos elementos: $a, 2a, 3a, \cdots, (p-1)a$, tais números também não são divisíveis por $p$ e todos, incongruentes módulo $p$. Desta forma, cada elemento do primeiro conjunto é congruente a um único elemento do segundo conjunto e, ao multiplicarmos tais congruências obtemos:

$$a \cdot 2a \cdot 3a \cdot \cdots \cdot (p-1)a \equiv 1 \cdot 2 \cdot 3 \cdot \cdots \cdot (p-1) \ mod \ p$$
$$a^{p-1}(p-1)! \equiv (p-1)! \ mod \ p$$

Como $mdc((p-1)!, p) = 1$, pela proposição 1.10 podemos cancelar o fator $(p-1)!$, logo: $a^{p-1} \equiv 1 \ mod \ p$ e o resultado segue. □

Assim, dado um número $p$, primo e, $a$ um número não divisível por $p$, temos que *$a$ elevado ao antecessor de $p$ é congruente a $1$ módulo $p$*.

**Corolário 1.22.** *Seja $p$ um número primo e $a$ um número inteiro positivo, então $a^p \equiv a \ mod \ p$.*

*Demonstração:* Note que de $a^p \equiv a \ mod \ p$ segue que $p \mid a^p - a$ e, consequentemente, $p \mid a(a^{p-1} - 1)$. Se $p$ divide $a$ o resultado é imediato, se $p$ não divide $a$, pelo teorema 1.21 $p$ divide $a^{p-1} - 1$ e o resultado segue. □

Em $\mathbb{Z}_m$, para determinar a qual das $m$ classes residuais[18] pertence certo inteiro $a$ muito elevado, recorreremos ao *Pequeno Teorema de Fermat* e o *Teorema de Euler* (que abordaremos a seguir) em busca de um *trunfo*, um expoente muito peculiar, o que torna verdadeira a congruência $a^n \equiv 1 \ mod \ m$, uma vez encontrado o expoente $n$ poderemos utilizar a proposição 1.9: $a^{k \cdot n} \equiv 1^k \equiv 1 \ mod \ m$.

18 Abordaremos o referido tema no capítulo seguinte.

## 1.6 O PODER DE "QUEBRA" DE UM NÚMERO

Euler investigava as propriedades dos números, em particular a distribuição dos números primos, e definiu uma função aritmética, a *φ de Euler*, denotada por $\varphi(m)$.

**Definição 1.23** (A função *φ de Euler*)**.** Para $m$ inteiro positivo, $\varphi(m)$ configura a quantidade de inteiros positivos menores ou iguais a $m$, relativamente primos com $m$.

**Definição 1.24.** O conjunto com $\varphi(m)$ elementos: $\{r_1, r_2, \cdots, r_{\varphi(m)}\}$, relativamente primos com $m$ e dois a dois, incongruentes módulo $m$ formam um **Sistema Reduzido de Resíduos**[19] módulo $m$.

Temos ainda que dado $\{r_1, r_2, \cdots, r_{\varphi(m)}\}$ um sistema reduzido de resíduos módulo $m$, e $mdc(a, m) = 1$ então $\{ar_1, ar_2, \cdots, ar_{\varphi(m)}\}$ também será um sistema reduzido de resíduos[20]

### 1.6.1 *O cálculo de φ(m)*

Temos três casos a considerar:

• Se $p$ é um número primo, $\varphi(p) = p - 1$, e não poderia ser diferente, os $''p''$ números $1, 2, 3, \cdots, p$, com exceção do próprio $p$, são todos primos com $p$.

• Se $p$ é um número primo e $k$ um número natural, então $\varphi(p^k) = p^k - p^{k-1}$. De fato, temos $p^k$ números naturais e, destes, devemos excluir todos os que não são primos com $p^k$, isto é, todos os múltiplos de $p$, a saber: $p, 2p, 3p, \cdots, p^{k-1}p$ que são ao todo $p^{k-1}$.

• Se $m$ não é primo, então $m = p_1^{k_1} \cdot p_2^{k_2} \cdot \ldots \cdot p_n^{k_n}$, que nada mais é do que a decomposição canônica de $m$ em potências de primos distintos, segue que:

$$\varphi(m) = \varphi(p_1^{k_1} \cdot p_2^{k_2} \cdot \ldots \cdot p_n^{k_n}) = \varphi(p_1^{k_1}) \cdot \varphi(p_2^{k_2}) \cdot \ldots \cdot \varphi(p_n^{k_n}) =$$

19 O sistema reduzido de resíduos possui a propriedade de que todos os seus elementos são invertíveis em $\mathbb{Z}_m$

20 Nesta demonstração, basta mostrar que há ao todo $\varphi(m)$ elementos, que os mesmos são primos com $m$ e, dois a dois, incongruentes módulo $m$. Utilize o fato que tais elementos são extraídos de um Sistema Completo de Resíduos.

$$(p_1^{k_1} - p_1^{k_1-1})\cdot(p_2^{k_2} - p_2^{k_2-1})\cdot\ldots\cdot(p_n^{k_n} - p_n^{k_n-1}) =$$
$$p_1^{k_1-1}\cdot p_2^{k_2-1}\cdot\ldots\cdot p_n^{k_n-1}\cdot(p_1-1)\cdot(p_2-1)\cdot\ldots\cdot(p_n-1).$$

Se $m$ é composto, $\varphi(m)$ é o produtório da função $\varphi$ *de Euler* aplicada a cada uma das potências de seus respectivos fatores primos[21]. Desta forma, fica claro que a função $\varphi$ *de Euler* é multiplicativa[22], isto é, para $mdc(m,n) = 1$, a função $\varphi(m\cdot n) = \varphi(m)\cdot\varphi(n)$.

**Exemplo 1.7.** Para elucidar o que foi acima exposto, vamos calcular $\varphi(36)$.
$\varphi(36) = \varphi(2^2\cdot 3^2) = 2^{2-1}\cdot 3^{2-1}\cdot(2-1)\cdot(3-1) = 2\cdot 3\cdot 1\cdot 2 = 12$.
A saber $\{1,5,7,11,13,17,19,23,25,29,31,35\}$ são os doze números menores que 36 e primos com 36.

#### 1.6.1.1 *O teorema de Euler*

**Teorema 1.25** (Euler)**.** *Dado $m$ um inteiro positivo e $a$ um inteiro com $mdc(a,m) = 1$, então:*

$$a^{\varphi(m)} \equiv 1 \ mod\ m$$

Equivalentemente: Se $[a] \in \mathbb{Z}_m$, então

$$[a]^{\varphi(m)} = [1]$$

*Demonstração:* Se $mdc(a,m) = 1$, os $\varphi(m)$ elementos $r_1, r_2, \cdots, r_{\varphi(m)}$ e $ar_1, ar_2, \cdots, ar_{\varphi(m)}$, ambos formam um *sistema reduzido de resíduos módulo $m$* e, logo, existe uma correspondência biunívoca, isto é, cada $ar_i$ é congruente a um único $r_j$, donde o produtório de $ar_i$ é congruente ao produtório de $r_j$ módulo $m$, assim:
$ar_1\cdot ar_2\cdot\ldots\cdot ar_{\varphi(m)} \equiv r_1\cdot r_2\cdot\ldots\cdot r_{\varphi(m)}\ mod\ m$. Seque que, $a^{\varphi(m)}\cdot r_1\cdot r_2\cdot\ldots\cdot r_{\varphi(m)} \equiv r_1\cdot r_2\cdot\ldots\cdot r_{\varphi(m)}\ mod\ m$. Como $mdc(r_1\cdot r_2\cdot\ldots\cdot r_{\varphi(m)}, m) = 1$, pela proposição1.10 podemos cancelar o termo $r_1\cdot r_2\cdot\ldots\cdot r_{\varphi(m)}$ e, assim obtemos: $a^{\varphi(m)} \equiv 1\ mod\ m$. □

Note que para $p$ primo, $\varphi(p) = p-1$ e, o teorema de Euler aplicado a um número primo é $a^{\varphi(p)} \equiv 1\ mod\ p$, substituindo o valor de $\varphi(p)$ obtemos $a^{p-1} \equiv 1\ mod\ p$. O "*gigantesco*" pequeno teorema de Fermat é, na verdade, apenas um caso particular do teorema de Euler, o que justificaria o antenome "pequeno".

21 Ver teorema fundamental da aritmética em *Apêndice A*
22 Para este e outros resultados recomendamos a leitura de [11], [21]

Pois bem, procurávamos desde o início, o precioso expoente que torna verdadeira a congruência $a^n \equiv 1 \ mod \ m$, recorreremos a ambos os teoremas para encontrá-lo. O teorema de Euler[23] possibilita a resolução de equações do tipo: $x^k \equiv b \ mod \ m$.

Para tanto, devemos ter $mdc(b, m) = 1$ e $mdc(k, \varphi(m)) = 1$. Pelo teorema de Bachet-Bézout existem inteiros $t$ e $u$ tais que $k \cdot t - \varphi(m) \cdot u = 1$, o que implica em $k \cdot t = 1 + \varphi(m) \cdot u$. Segue que:

$$x^k \equiv b \ mod \ m \Rightarrow x^{k \cdot t} \equiv b^t \ mod \ m \Rightarrow x^{1+\varphi(m) \cdot u} \equiv b^t \ mod \ m \Rightarrow x \cdot x^{\varphi(m)^u} \equiv b^t \ mod \ m$$

Para que $x$ seja solução da equação, devemos ter $mdc(x, m) = 1$, satisfeito isso, podemos aplicar o Teorema de Euler e, assim, obtemos:

$$x^k \equiv b \ mod \ m \Rightarrow x \equiv b^t \ mod \ m$$

De fato, $b^t$ é solução da equação pois:

$$(b^t)^k = x^{1+\varphi(m) \cdot u} \equiv b \ mod \ m$$

E, desta forma, a equação possui solução única $b^t$, onde $t$ é um inteiro tal que $k \cdot t \equiv 1 \ mod \ \varphi(m)$

**Exemplo 1.8.** Vejamos como aplicar o resultado acima encontrando a solução de $x^{43} \equiv 2 \ mod \ 99$. Primeiramente, vamos calcular $\varphi(99) = \varphi(11) \cdot \varphi(3^2) = (11 - 1) \cdot (3^2 - 3^1) = 10 \cdot 6 = 60$.
Como $mdc(2, 99) = 1$ e $mdc(43, 60) = 1$, a solução é da forma $2^t$, onde $43 \cdot t \equiv 1 \ mod \varphi(99)$, isto é, $43 \cdot t - 60 \cdot u = 1$, ao aplicar o algoritmo de Euclides[24] , encontramos $43 \cdot 7 - 60 \cdot 5 = 1$ e, portanto, a solução da equação será $2^7 = 128$.

Este tipo de equação, assim como o teorema de Euler, possui aplicação em um ramo muito peculiar na Matemática, que abordaremos a seguir.

#### *1.6.1.1.1 Criptografia RSA*

A Criptografia é a arte de ocultar o significado de informações, ainda que públicas, de forma que somente quem as ocultou e a quem se destina possam acessar o seu real conteúdo. Este recurso amplamente utilizado no ramo da informática e do mercado financeiro, dentre os métodos disponíveis para criptografar dados, o *RSA*[25] é um dos

23 Para uma leitura detalhada sugerimos [11], [21] e [40]
24 Indicamos [15] para aprofundamento de estudos
25 RSA provém do nome de seus criadores: Ronald **R**ivest, Adi **S**hamir e Leonard **A**dleman, fundadores a atual empresa *RSA Data Security*.

mais utilizados e confiáveis por atender um princípio básico, tal método apresenta facilidade em codificar e extrema dificuldade em decodificar.[26]

Primeiramente, vejamos como criptografar mensagens via RSA.

**CODIFICAÇÃO**

*(Passo 1)* - *Pré-codificação*, vamos associar nossa mensagem ao quadro abaixo:

| A | B | C | D | E | F | G | H | I | J | K | L | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
| N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 |

Escolhemos os números de 10 a 35, mas poderíamos ter utilizado de 42 a 67, por exemplo. Utilizam-se números de dois dígitos para evitar ambiguidades, caso tivéssemos A = 1 e B = 2, a cifra 12 poderia corresponder tanto a AB como ao L.

Nossa mensagem será: $DNA \longrightarrow 132310$

*(Passo 2)* - Vamos escolher dois números primos, que tal $p = 3$ e $q = 17$. Com estes, determinamos a *chave de codificação*: $n = p \cdot q = 3 \cdot 17 = 51$.

**Observação 1.6.1.** *A chave* $n = 51$ *pode vir a ser um dado público, conhecido por todos. Entretanto,* $p$ *e* $q$ *precisam permanecer secretos, pois é isto que garante a eficiência do método.*

*(Passo 3)* - Vamos quebrar a mensagem em blocos, cujos dígitos devem representar, necessariamente, números menores que $n$: **13-23-10** e aplicar a cada um destes blocos a *regra de codificação* $a = C(b) = b^e \equiv a \ mod \ n, \ 0 < C(b) < n$, a escolha de $e$ é livre, contanto que $mdc(e, \varphi(n)) = 1$, utilizaremos $e = 5$.

- $13^5 \equiv a_1 \ mod 51 \Rightarrow a_1 = 13$
- $23^5 \equiv a_2 \ mod 51 \Rightarrow a_2 = 41$
- $10^5 \equiv a_3 \ mod 51 \Rightarrow a_3 = 40$

26 Vamos discorrer sobre o tema de maneira sucinta de tal forma que indicamos a leitura de [32] e [39] para apurar-se sobre o tema.

Assim, nossa mensagem codificada é ***13-41-40*** e deve ser enviada desta forma, caso os dígitos sejam convertidos em um único bloco, a decodificação será impossível.

**Observação 1.6.2.** *A mensagem foi quebrada em blocos de dois dígitos, contudo, poderíamos tomar 1-32-3-10 para aplicar a regra de codificação. A ausência de padrão na composição dos blocos evita a decodificação por contagem de frequência dos caracteres.*

**DECODIFICAÇÃO**

*(Passo 1)* - É imprescindível determinar a chave de decodificação $(n, d)$, onde $n = 51$ é público, e $d$ é o inverso de $e$ módulo $\varphi(n)$ ou seja, $e \cdot d \equiv 1 \ mod \varphi(n) \Rightarrow 5 \cdot d \equiv 1 \ mod \ (p-1) \cdot (q-1) \Rightarrow 5 \cdot d \equiv 1 \ mod \ (2 \cdot 16) \Rightarrow 5 \cdot d \equiv 1 \ mod \ (32) \Rightarrow d = 13$.

*(Passo 2)* - Aplicaremos a cada um dos novos blocos, a *chave de decodificação* $b = D(a) = a^d \equiv b \ mod \ n, \ 0 \leq D(a) < n$.

- $13^{13} \equiv b_1 \ mod 51 \Rightarrow b_1 = 13$
- $41^{13} \equiv b_2 \ mod 51 \Rightarrow b_2 = 23$
- $40^{13} \equiv b_3 \ mod 51 \Rightarrow b_3 = 10$

E assim retornamos para ***13-23-10*** e, desta forma, *acabamos de decodificar o DNA*!

Contudo, para nos certificarmos de que o método é efetivamente válido, precisamos mostrar que, uma vez aplicado, o resultado obtido após a decodificação é, de fato, igual à mensagem original que foi codificada, o que nos remete ao seguinte teorema:

**Teorema 1.26.** $D(C(b)) = b$

*Demonstração.* : $D(C(b)) = C(b)^d \equiv b^{e \cdot d} \ mod \ n$. dado que $d$ é o inverso de $e$ módulo $\varphi(n)$, isto é, $e \cdot d \equiv 1 \ mod \varphi(n)$ ou ainda $e \cdot d = 1 + k \cdot \varphi(n)$. Segue que:

$$D(C(b)) \equiv b^{1+k \cdot \varphi(n)} \equiv b \cdot (b^{\varphi(n)})^k \equiv \ mod \ n$$

Como $n = p \cdot q$ temos que $\varphi(n) = (p-1) \cdot (q-1)$, o que conduz a:

$$D(C(b)) \equiv b \cdot (b^{p-1})^{(q-1) \cdot k} \ mod \ p.$$

Se $p$ não divide $b$, então pelo *Pequeno teorema de Fermat1.21* temos:

$$D(C(b)) \equiv b \cdot (b^{p-1})^{(q-1) \cdot k} \equiv b \cdot (1^{p-1})^{(q-1) \cdot k} \equiv b \ mod \ p.$$

Caso $p$ divida $b$, então $b \equiv 0 \ mod \ n$, donde temos:

$$D(C(b)) \equiv b \cdot (b^{p-1})^{(q-1)\cdot k} \equiv 0 \ mod \ p.$$

De modo análogo, obtemos $D(C(b)) \equiv b \ mod \ q$ e, uma vez em que $p$ e $q$ são primos, $D(C(b)) \equiv b \ mod \ (p \cdot q)$, isto é, $D(C(b)) \equiv b \ mod \ n$.

A congruência em si não assegura necessariamente a igualdade, todavia, o fato de tanto $C(b)$ como $D(a)$ serem menores do que $n$ acarreta em que a congruência implica na igualdade e o resultado segue. Tal fato justifica a obrigatoriedade referente ao "*tamanho*" dos blocos, que devem representar números menores que $n$. □

A criptografia RSA é explorada em larga escala na informatização do setor financeiro: transações com cartões de crédito, segurança de *e-mails*, autenticações de chamadas telefônicas, assinaturas digitais, garantia da segurança das criptomoedas, como os *bitcoins*, entre outros.

Tal método utiliza a chave pública, $(e, n)$ que pode ser conhecida por todos, sendo utilizada apenas para codificar, contudo, $d$, $p$ e $q$ devem ser mantidos em absoluto sigilo. Este método de criptografar é o que designamos por assimétrico,[27] uma vez que a chave de decodificação não é o oposto da chave de codificação. Toda sua credibilidade sustenta-se na dificuldade em determinar $p$ e $q$ a partir de $n$. Neste caso estamos nos referindo a números primos muito grandes, na ordem de mais de 150 dígitos, cujo produto $n$ será absurdamente maior, de tal forma que mesmo os supercomputadores de posse de métodos sofisticados levariam anos para fatorar $n$.

O RSA configura-se no criptossistema de chave pública mais confiável da atualidade, tendo resistido a mais de 30 anos de incessantes ataques que as maiores mentes do mundo puderam conceber, até este momento.[28]

Fatorar $n$ significa quebrar o código de fato, caso a fatoração não seja efetuada, não se descobre quem são $p$ e $q$ e consequentemente, não se descobre $\varphi(n) = (p -$

27 Durante muito tempo acreditou-se que este método era impossível. Whitfield Diffie, Martin Hellman e Ralph Merkle (1976) idealizam e descrevem o método de criptografia de chave pública, tal como um cadeado, que, quando aberto, pode ser trancado por qualquer pessoa, porém somente o detentor da chave pode abri-lo. Contudo faltava criar uma função que contemplasse tais requisitos.

28 Indicamos [43] que apresenta a história da arte de escrever segredos, bem como a luta entre criadores e decifradores de códigos, desde a antiguidade até os dias atuais, onde a criptografia é usada na internet e em operações bancárias.

$1)\cdot(q-1)$ e tampouco se descobre o valor de $d$, e sem tais elementos, a decodificação é impossível.

A principal desvantagem, se não a única, consiste na atual capacidade das máquinas, pois ainda que se conheçam números primos extremamente grandes, o que aumentaria a segurança do sistema, sua implementação é impraticável. Isso ocorre porque o processamento computacional de cálculos desta magnitude demandaria máquinas exponencialmente mais potentes se comparadas às atuais.

# 2

# A ÁLGEBRA DAS LEIS DO PENSAMENTO

Este capítulo versa sobre elementos essenciais para a concepção de computadores e calculadoras. A princípio expõe conceitos que cingem as bases numéricas, em especial a base binária, que é o sistema de numeração utilizado na programação das máquinas acima citadas, além de explanar suas operações, propriedades e aplicações.

Na sequência, introduzimos a álgebra booleana, idealizada para formalizar e manipular as leis do pensamento e, que suscitou o surgimento dos computadores atuais, suas propriedades, axiomas, teoremas e aplicações são amplamente explorados e esmiuçados. Apresentar-se-ão as portas lógicas, que são a base para construções de circuitos lógicos eletrônicos, além de evidenciar a interligação entre este, a tabela verdade e as expressões booleanas, cujo objetivo maior é aprimorar a destreza em traduzir problemas para a linguagem computacional, manipulá-los algebricamente e, desta forma, viabilizar a tomada de decisões e obtenção de soluções.

Todos os conteúdos e conceitos aqui explicitados podem ser contemplados e aprofundados com riqueza de detalhes nas referências citadas ao longo deste trabalho.

## 2.1 UM TRUQUE BINÁRIO

Apresentaremos a técnica/truque que consiste em adivinhar o número que foi pensado[1]. O (*mágico*) pedirá ao espectador que pense em um número entre 10 e 100 e procederá como segue:

- Perguntará ao espectador se o número é par ou ímpar, e irá proferir as instruções: se par, divida o número por 2, se ímpar, subtraia 1 e então di-

1 Disponível em [42]

vida por 2;

• Novamente, perguntará se o resultado obtido é par ou ímpar e retomará as instruções do item anterior;

• O procedimento continua com cada resultado obtido, até que o quociente seja 1 o que cessa os cálculos;

• Informado de que o resultado é 1, o mágico (*você*) anunciará prontamente o número pensado inicialmente pelo espectador.

**Como funciona o truque**: Suponha que o espectador pensou no número 57, enquanto ele realiza as etapas da coluna à esquerda o mágico (*secretamente*) faz as anotações da coluna à direita:

| **espectador** | **mágico** |
|---|---|
| 57 | 1 |
| 28 | 2 |
| 14 | 4 |
| 7 | 8 |
| 3 | 16 |
| 1 | 32 |

Para cada número ímpar informado, o mágico deverá sublinhar a potência de 2 correspondente, ao final do processo, basta que o mágico adicione as mesmas para informar o número pensado:

$$1 + 8 + 16 + 32 = 57$$

**Por que o truque funciona**: a técnica utilizada foi o método das divisões sucessivas por 2, que nada mais é do que a forma para representar um inteiro positivo na base binária, ou seja, como soma de potências distintas de 2, para tanto realizamos o processo abaixo:

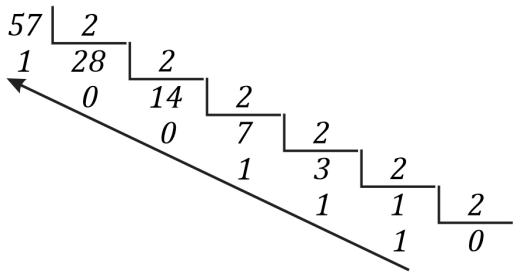


Figura 2: Divisões sucessivas por 2

Lemos da direita para a esquerda todos os restos e, assim, obtemos a representação do 57 (*base decimal*) no sistema de numeração de base 2 (*base binária*).

$$(57)_{10} = (111001)_2 = 1\cdot 2^5 + 1\cdot 2^4 + 1\cdot 2^3 + 0\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 = 2^5 + 2^4 + 2^3 + 2^0 = 32 + 16 + 8 + 1.$$

Na sequência das divisões, todos os números pares deixarão resto 0, enquanto que todos os números ímpares deixarão resto 1, por isso o mágico toma nota apenas das potências de 2 correspondentes aos números ímpares.

### 2.1.1 *Frações binárias*

Podemos representar também números fracionários na base binária por meio de potências de 2, porém com expoentes negativos:

$$\begin{aligned}(0,1101)_2 &= 1\cdot 2^{-1} + 1\cdot 2^{-2} + 0\cdot 2^{-3} + 1\cdot 2^{-4} = \\ &= 1\cdot 0,5 + 1\cdot 0,25 + 1\cdot 0,0625 = (0,8125)_{10}\end{aligned}$$

O processo de multiplicar um número da base decimal por 10 acarreta no deslocamento da vírgula *(que distingue o inteiro do fracionário)* uma posição à direita. De forma análoga, vamos multiplicar o número fracionário por 2 sucessivas vezes e tomar o dígito à esquerda da vírgula:

$$\begin{aligned}0,8125\cdot 2 &= \mathbf{1},625\\ 0,625\cdot 2 &= \mathbf{1},25\\ 0,25\cdot 2 &= \mathbf{0},5\\ 0,5\cdot 2 &= \mathbf{1}\\ (0,8125)_{10} &= (0,1101)_2\end{aligned}$$

## 2.2 BASES

Embora estejamos abordando o sistema de numeração binária (base 2), podemos aplicar tais procedimentos a qualquer outra base, uma vez que as propriedades dos números existirão independentemente da escolha do sistema de numeração utilizado, cuja escolha é meramente convencional. Caso tivéssemos apenas 2 dedos em ambas as mãos, muito provavelmente utilizaríamos a base quaternária em vez da base decimal.

Existem muitos sistemas de numeração, a maioria conta com uma quantidade finita de símbolos, denominados *algarismos*, para representar os números. Qualquer sistema de numeração baseia-se no emprego de unidades de diversas ordens, isto é, a quantidade de unidades que devemos ter para compor uma unidade na ordem seguinte. Tal quantidade é que denominados de *base*, que deve ser sempre maior ou igual a dois[2], que podem ser identificados pela quantidade de símbolos distintos utilizados para a representação os números.

Estes e outros procedimentos/resultados fundamentam-se no seguinte teorema:[3]

**Teorema 2.1** (*Teorema fundamental da numeração*)**.** *Dados $a, b \in \mathbb{N}$, $b > 1$, existem números naturais $c_0, c_1, \ldots, c_n$ menores que $b$ e univocamente determinados tais que:*

$$a = \sum_{i=0}^{n} c_i b^i = c_n b^n + \ldots + c_2 b^2 + c_1 b^1 + c_0 b^0$$

*Onde $0 \leq c_i < b$ para $0 \leq i \leq n$.*

*Demonstração:* A expressão $a = c_i b^i = c_n b^n + \ldots + c_2 b^2 + c_1 b^1 + c_0 b^0$, com $0 \leq c_i < b$ para $i = 0, \ldots, n$ é chamada de *expressão b-ádica* do inteiro $a$. A mesma pode ser obtida por meio de divisões euclidianas aplicadas sucessivamente:

$$\begin{aligned}
a &= b\cdot\ q_0 + c_0, && (2.1)\\
q_0 &= b\cdot\ q_1 + c_1, && (2.2)\\
q_1 &= b\cdot\ q_2 + c_2, && (2.3)\\
&\vdots\\
q_{n-2} &= b\cdot\ q_{n-1} + c_{n-1}, && (2.4)\\
q_{n-1} &= b\cdot 0 + c_n && (2.5)
\end{aligned}$$

Onde $0 \leq c_j < b$ para $0 \leq i \leq n$

Substituímos o valor de $q_0$ da equação (2.2) na equação (2.1), na sequência, substituímos nesta nova equação o valor de $q_1$ da equação (2.3) e assim por diante:

2 Do contrário, as unidades de diversas ordens seriam iguais entre si e as ordens deixariam de existir.

3 Para analisar outras formas de demonstração deste teorema, recomendamos [21], [22] e [34].

$$\begin{aligned}
a &= b\cdot q_0 + c_0 \\
&= b\cdot(b\cdot q_1 + c_1) + c_0 = b^2\cdot q_1 + c_1\cdot b + c_0 \\
&= b^2\cdot(b\cdot q_2 + c_2) + c_1\cdot b + c_0 = b^3\cdot q_2 + c_2\cdot b^2 + c_1\cdot b + c_0 \\
&\vdots \\
&= b^n\cdot q_{n-1} + c_{n-1}\cdot b^{n-1} + \cdots + c_2\cdot b^2 + c_1\cdot b + c_0 \\
&= c_n\cdot b^n + c_{n-1}\cdot b^{n-1} + \cdots + c_2\cdot b^2 + c_1\cdot b + c_0 \\
&= \sum_{k=0}^{n} c_k b^k
\end{aligned}$$

Quanto à unicidade de $c_0, c_1, \ldots, c_n$, suponha, pelo contrário, que exista mais de uma forma de representar o mesmo número, isto é: $a = c_n b^n + \ldots + c_2 b^2 + c_1 b^1 + c_0 b^0$ e $a = d_n b^n + \ldots + d_2 b^2 + d_1 b^1 + d_0 b^0$, temos que:

$$\begin{gathered}
a - a = (c_n b^n + \ldots + c_2 b^2 + c_1 b^1 + c_0 b^0) - (d_n b^n + \ldots + d_2 b^2 + d_1 b^1 + d_0 b^0) = 0 \iff \\
(c_n - d_n)b^n + \ldots + (c_2 - d_2)b^2 + (c_1 - d_1)b^1 + (c_0 - d_0)b^0 = 0 \iff \\
c_n = d_n;\ \ldots\ ;\ c_2 = d_2;\ c_1 = d_1;\ c_0 = d_0
\end{gathered}$$

□

Designamos o resultado acima por representação de $a$ na base $b$, o qual denotaremos por $(c_n c_{n-1} \cdots c_1 c_0)_b$. Tal resultado justifica a utilização do método das divisões sucessivas para a conversão de números em uma base qualquer para outra, e acarreta em muitos dos critérios de divisibilidade.

### 2.2.1 *Alguns critérios de divisibilidade*

Os critérios de divisibilidade são regras/técnicas, geralmente simples, que possibilitam a verificação da divisibilidade de um número natural por outro sem a necessidade de efetuar os cálculos da divisão euclidiana.

**Definição 2.2** (Divisibilidade)**.** Dados $a$ e $b$ números naturais, $b \neq 0$, diremos que $a$ é divisível por $b$, se existir $k \in \mathbb{N}$ tal que $a = b\cdot k$.

Os testes de divisibilidade por 2, 4, 5, 10 são corriqueiros a qualquer estudante, por serem trabalhados no ensino fundamental. Todavia, por meio das congruências

podemos correlacionar propriedades da divisibilidade de um número com sua representação *b-ádica* e, desta forma, formalizar resultados mais sofisticados.

Do teorema 2.6, temos que: $(c_k c_{k-1} \cdots c_1 c_0)_b \overset{def}{=} \sum_{k=0}^{n} c_k b^k$

Correspondente à representação do número natural $a$ na base $b$, tendo $c_k$, $0 \leq k \leq n$, por seus algarismos. Desta representação decorrem muitos dos critérios de divisibilidade, cuja utilidade de alguns é controversa, como veremos a seguir. Verifiquemos inicialmente, na base decimal, a divisibilidade por 9.

Seja $a = (c_n c_{n-1} \cdots c_1 c_0)_{10}$ um inteiro não negativo, dado que $10 \equiv 1 \ mod\ 9$ temos pela proposição 1.9 que $10^k \equiv 1 \ mod\ 9$, segue que:

$$a = \sum_{k=0}^{n} c_k b^k \equiv \sum_{k=0}^{n} c_k \mod 9$$

De modo conciso, $a$ e a soma de seus algarismos na base decimal apresentam o mesmo resto na divisão por 9, ou seja, $a$ é divisível por 9, somente se $c_n + c_{n-1} + \cdots + c_0$ também for.

A divisibilidade por 11 ocorre de forma semelhante, note que $10 \equiv -1 \ mod\ 11$, logo:

$$a = \sum_{k=0}^{n} c_k b^k \equiv \sum_{k=0}^{n} (-1)^k c_k \mod 11$$

Decorre disto que a divisibilidade por 11 verifica-se somente quando a diferença entre a soma dos algarismos de posição par e soma dos algarismos de posição ímpar for divisível por 11.

Tradicionalmente, diversas bibliografias optam por omitir o critério de divisibilidade por 7, teria algum motivo especial?

Primeiramente, temos que $10^3 \equiv (-1) \ mod\ 7$, vamos decompor $a = (c_n c_{n-1} \cdots c_1 c_0)_{10}$ em blocos de classe, isto é, $m_0 = c_2 c_1 c_0, m_1 = c_5 c_4 c_3, m_2 = c_8 c_7 c_6, \cdots, m_k = c_{3k+2} c_{3k+1} c_{3k}$, obtidos pela separação do número $a$ em grupos de três algarismos da direita para a esquerda. Isto posto, segue que:

$$a = \sum_{k=0}^{n} c_k b^k \equiv \sum_{k=0}^{n} m_k (10^3)^k \equiv \sum_{k=0}^{n} (-1)^k m_k \equiv \sum_{k=0}^{n} (-1)^k (c_{3k+2} c_{3k+1} c_{3k}) \ mod\ 7$$

Deste modo, temos que $a \equiv (\cdots - c_{11} c_{10} c_9 + c_8 c_7 c_6 - c_5 c_4 c_3 + c_2 c_1 c_0) \ mod\ 7$, disto advém que se a soma dos blocos de classe ímpar diminuídos da soma dos blocos de classe par for divisível por 7, então a divisibilidade de $a$ por 7 também se verifica.

Este é apenas um dos muitos critérios de divisibilidade por 7. Tantos critérios provém justamente de tentativas frustradas de torná-los o mais simples possível, afinal que utilidade teria um critério de divisibilidade que demanda mais recursos e esforços que a própria divisão.

E quanto às bases diferentes da decimal, será que é possível estender tais critérios de divisibilidade[4]?

Para exemplificar, tomemos o número natural $a = c_n 10^n + c_{n-1}10^{n-1} + \ldots + c_1 10^1 + c_0 10^0$. A divisibilidade por 10 se verifica caso $c_0 \cdot b^0 = 0$, já que todas as demais parcelas são divisíveis por 10, reciprocamente, o referido número terminará em 0 caso seja divisível por 10.

Tal resultado pode estender-se a uma base numérica arbitrária, a representação de um número na base $n$ terminará em 0 se, e apenas se, o mesmo for divisível por $n$.

Podemos expandir esta ideia para outros critérios de divisibilidade para bases arbitrárias, analisemos se o número $(43524030513)_6$ é divisível por 5?

Para tanto, faremos uma analogia ao critério de divisibilidade por 9, que em uma base $n$ qualquer, o designamos por $n-1$. De fato, temos que $n \equiv 1 \ mod\ (n-1)$, pela proposição 1.9 segue que $n^k \equiv 1 \ mod\ (n-1)$. Logo:

$$a = \sum_{k=0}^{n} c_k b^k \equiv \sum_{k=0}^{n} c_k \mod (n-1)$$

Sucintamente, a soma dos algarismos de um inteiro $a$ representado na base $n$ é divisível por $n-1$ somente quando $a$ é divisível por $n-1$.

Retomando a questão inicial, dado que a soma dos algarismos é $(30)_{10}$, e que a mesma é divisível por 5, então $(43524030513)_6$ também será divisível por 5.

De modo similar, estendemos o critério de divisibilidade por 11, que em uma base $n$ qualquer indicamos por $n+1$. De fato, $n \equiv -1 \ mod\ (n+1)$, novamente, pela proposição 1.9 temos $n^k \equiv (-1)^k \ mod\ (n+1)$. Segue que:

$$a = \sum_{k=0}^{n} c_k b^k \equiv \sum_{k=0}^{n} (-1)^k \cdot c_k \mod (n+1)$$

De forma concisa, a soma dos algarismos de um inteiro $a$ representado na base $n$ é divisível por $n+1$ somente quando a diferença entre a soma dos algarismos de índice par e soma dos algarismos de índice ímpar for divisível por $n+1$.

4 Para explorar o tema, indicamos [15]

## 2.3 ARITMÉTICA BINÁRIA

A vantagem da utilização do sistema de numeração binário consiste na simplificação das operações aritméticas, embora sejam inegáveis os avanços tecnológicos na área computacional, com máquinas cada vez mais velozes e com incrível capacidade de processamento, os computadores.

Mesmo os atuais, reconhecem em seu sistema interno apenas dois estados: tensão baixa e alta representados pelos algarismos 0 e 1 respectivamente, Ou seja, os computadores utilizam o sistema de numeração binária em sua configuração, todos os dados do computador são representados de maneira única, com os algarismos (0, 1) que denotaremos por *BITS (BInary digiTS)*. Além de reconhecer apenas dois dígitos, o computador é capaz de fazer uma única operação, a *adição*:

**Adição:** por trabalhar com apenas dois dígitos, a adição é realizada através das somas: $0 + 0$, $0 + 1$, $1 + 0$ e $1 + 1$ que serão repetidas quantas vezes for necessário, os resultados de tais adições são os mesmo que encontramos nas tabuadas de $\mathbb{Z}_2$.

Note que $1 + 1 = 2$, que na base binária se escreve como 10, ou seja, $1 + 1 = 0$ e ocorre *(carry)*, vulgo "*vai 1*" que deverá ser adicionado na próxima coluna *(ordem)* à esquerda.

**Exemplo 2.1.**

$$\begin{array}{rr} & 1\,1\,0\,0\,1 \\ & +1\,0\,1\,0\,1 \\ \hline \textit{resultado} & 1\,0\,1\,1\,1\,0 \\ \text{"}\textit{vai um}\text{"} & 1\,0\,0\,0\,0\,1 \end{array}$$

### 2.3.1 *Inversão e deslocamento de bits, um truque de mestre.*

Dado que o computador é capaz de somar "*apenas*", precisamos fazer uso de certos "*truques*" para converter as demais operações em adições[5]:

5 Vamos converter as operações aritméticas em adições para conhecer como o computador opera, contudo as mesmas podem ser realizadas de forma convencional, tal como ocorrem no sistema de numeração decimal.

**A Operação de Inversão de *bits*:** conhecida também como complementação de 2, consiste em inverter os *bits* 0 por 1 e vice-versa[6], por exemplo: o inverso de 10010110 é 01101001.

**A Operação de Deslocamento de *bits*:** consiste em acrescentar *bits* 0 no final do número binário[7], isto equivale a multiplicar o número por 2, por exemplo: dado 1001011, após dois deslocamentos temos 100101100, após três deslocamentos, 1001011000.

#### 2.3.1.1 *Subtração*

Para entendermos o processo, primeiramente vamos realizar uma subtração com números na base decimal por meio da complementação por 9. Vejamos: $876 - 234 = 642$, por outro lado, temos que a complementação por 9 de 234 é 765, daí realizamos a adição: $876 + 765 = 1641$, somamos 1 ao resultado e descartamos o algarismo mais significativo *(neste caso o do milhar)* e obtemos novamente: 642.

Considere a seguinte subtração: $xyz - abc$ que podemos representar por $(100x + 10y + z) - (100a + 10b + c)$, cuja diferença será $100(x - a) + 10(y - b) + (z - c)$. Contudo, ao somarmos com a complementação de 9 do subtraendo $abc$, temos: $100(x + (9 - a)) + 10(y + (9 - b)) + (z + (9 - c))$. Por fim somamos 1 ao resultado, o que criará um efeito "*cascata*" eliminando todos os "9" e alocando 1 na próxima casa decimal, por isso, para obter o resultado da diferença, precisamos desconsiderar o algarismo mais significativo.

Utilizaremos a mesma técnica para realizar subtrações de binários: vamos somar ao minuendo o subtraendo com os *bits* invertidos e, na sequência, vamos somar 1 ao resultado e desconsiderar o dígito mais significativo, observe:

**Exemplo 2.2.** $11101 - 10110 = 11101 + \mathbf{01001} = 100110$, somamos 1 a este resultado e descartando o algarismo mais significativo, obtemos: **1**00111 e por fim: 111 que na base decimal corresponde a $29 - 22 = 7$.

#### 2.3.1.2 *Multiplicação*

Poderíamos utilizar o fato de que efetuar uma multiplicação consiste em uma soma reiterada de parcelas iguais, mas seria demasiadamente exaustivo, até para os compu-

6 Na base decimal, esta operação consiste na complementação por 9, na qual cada um dos $n$ algarismos é substituído por $9 - n$.

7 Na base decimal, esta operação equivale a multiplicar um número por 10, na qual deslocamos a vírgula para a direita.

tadores, por isso, nos deteremos ao fato de que a multiplicação de números binários pode ser realizada por uma soma, cujas parcelas tiveram seus *bits* deslocados para a esquerda. Tal descolamento é definido pela posição de cada *bits* 1 do multiplicador:

**Exemplo 2.3.** Segue que, ao multiplicar certo número binário por 1101, cuja decomposição é 1000 + 100 + 1, o produto será a soma do multiplicando *(sem deslocamento)*, mais o multiplicando *(com dois deslocamentos)*, mais o multiplicando *(com três deslocamentos)*.

| | |
|---|---|
| 1 1 0 0 1 | *multiplicando* $(25)_{10}$ |
| $\times$ 1 1 0 1 | *multiplicador* $(13)_{10}$ |
| 1 1 0 0 1 | *sem deslocamento* |
| 1 1 0 0 1 0 0 | *com 2 deslocamentos* |
| + 1 1 0 0 1 0 0 0 | *com 3 deslocamentos* |
| 1 0 1 0 0 0 1 0 1 | *produto* $(325)_{10}$ |

#### 2.3.1.3 *Divisão*

Quanto à divisão já não temos tantas opções como na multiplicação, todavia o quociente pode ser obtido por meio de sucessivas subtrações *(aplicáveis em qualquer base)*, por exemplo, $23 \div 5$ *(na base decimal)*: $23-5=18, 18-5=13, 13-5=8$ e $8-5=3<5$.

Assim, o quociente é 4 *(4 subtrações efetuadas)* e o resto é 3. Na base binária teríamos: $10111-101=10010, 10010-101=1101, 1101-101=1000, 1000-101=11<101$. Logo, o quociente é 100 *(4 subtrações realizadas)* e o resto é 11, ou seja, $10111 = 101 \cdot 100 + 11$[8].

##### *2.3.1.3.1 Código de barras*

Tornou-se difícil pensar em alguma atividade ou setor na era moderna que não faça uso dos computadores e consequentemente dos números binários, ainda que indiretamente. Em um futuro próximo será de suma importância para o cidadão moderno conhecer e manipular os números binários.

8 Para o leitor que queira ampliar sua visão sobre o tema, recomendamos [3], [16], [24] e [37]

Com a industrialização em constante expansão, tornou-se corriqueiro identificar produtos por meio de um código numérico, o código de barras[9] que converte um código numérico em barras e possibilita ao computador realizar uma leitura rápida e precisa. Dado que o computador consegue interpretar tal código, certamente a conversão é feita utilizando números binários. Contudo, como a leitora óptica consegue ler o código de barras de "*trás para frente*", e de "*cabeça para baixo*" dado que suas barras são assimétricas?

Devido à grande variedade de tipos de códigos de barras, restringiremos nossa abordagem ao *ENA-13 (European Article Nunbering systen)* amplamente utilizado no Brasil contendo 13 dígitos. Para facilitar a leitura óptica, devemos utilizar a linguagem dos


Figura 3: Código de Barras ENA-13

computadores e converter o código numérico em uma sequência de **0** (barras brancas) e **1** (barras pretas). Além da cor, há outro fator utilizado em tal conversão, a espessura das barras que podem ser: finas, médias, grossas e muito grossas. As diferentes espessuras decorrem, na verdade, da junção de listras. Tomando as listras pretas, temos: listra fina *(1)*, listra média *(11)*, listra grossa *(111)* e listra muito grossa *(1111)*, de modo análogo, temos *(0)*, *(00)*, *(000)* e *(0000)* respectivamente para as listras brancas.

Assim uma sequência de listra branca fina, listra preta média, listra branca fina e listra preta grossa, para a leitora óptica equivalem à sequência 0110111 que, por sua vez, corresponderá ao dígito 8. Desta forma, cada algarismo será representado por uma sequência de sete dígitos sendo estes **0** ou **1**. Sabendo que a leitura do código de barras pode ser feita tanto pela esquerda como pela direita, os dígitos são codificados de maneiras distintas para ambos os sentidos.

9 Indicamos [29] para uma leitura mais detalhada.

| Dígito | Esquerdo | Direito |
|---|---|---|
| 0 | 0001101 | 1110010 |
| 1 | 0011001 | 1100110 |
| 2 | 0010011 | 1101100 |
| 3 | 0111101 | 1000010 |
| 4 | 0100011 | 1011100 |
| 5 | 0110001 | 1001110 |
| 6 | 0101111 | 1010000 |
| 7 | 0111011 | 1000100 |
| 8 | 0110111 | 1001000 |
| 9 | 0001011 | 1110100 |

Quando a leitura muda de sentido, observa-se a utilização do complementar de 2, ou simplesmente, o inverso dos números, mas como a máquina reconhece e determina se a leitura deve ser feita pela direita ou pela esquerda? A resposta é simples e sagaz, e justifica a escolha do 7 para a quantidade de dígitos nas representações acima. Cada sequência de 7 dígitos do lado esquerdo possui uma quantidade ímpar de "*1*" enquanto que a sequência de 7 dígitos do lado direito possui uma quantidade par de "*1*", ou seja, a máquina verifica a paridade dos dígitos e determina prontamente em qual dos lados a leitura deve ser realizada.

#### *2.3.1.3.2 O décimo terceiro dígito*

No código de barras *ENA-13*, os três primeiros dígitos indicam o país de origem do produto, o código para produtos brasileiros é **789**, os quatro dígitos seguintes reportam ao fabricante do produto, os cinco dígitos na sequência especificam o produto e o último dígito é nada menos que o *dígito verificador*. A máquina recorre a tal dígito para identificar erros de digitação e impedir que o número digitado erroneamente remeta a um outro código barras qualquer.

Como visto anteriormente, é preciso aplicar um algoritmo específico para determinar o dígito verificador. Primeiramente devemos efetuar uma soma $S$, que é inerente a cada propósito. Para este código de barras, o método consiste em multiplicar os dígitos de índice ímpar por 1, multiplicar os dígitos de índice par por 3 e então, somá-los:

$$S = \sum_{i=1}^{6} a_{(2\cdot i+1)} \quad + \quad 3\cdot \sum_{i=1}^{6} a_{(2\cdot i)}$$

Feito isto, temos que o dígito verificador é dado por:

$$S + dv \equiv 0 \ mod \ 10$$

**Exemplo 2.4.** Considere o código de barras da figura 3, para determinar o seu dígito verificador, os doze primeiros dígitos são *789012345678dv*:

$$(7+9+1+3+5+7)+3\cdot(8+0+2+4+6+8)+dv \equiv 0 \ mod \ 10$$

$$32+3\cdot 28+dv \equiv 0 \ mod \ 10$$

$$116+dv \equiv 0 \ mod \ 10$$

Logo, 4 é o dígito verificador procurado.

Caso ocorresse algum erro de digitação, seja ele a troca de dígitos, ou a troca de ordem de dígitos adjacentes, uma vez efetuado o procedimento, o erro seria identificado já que a congruência não se verificaria.

## 2.4 UM POUCO DE LÓGICA, É LÓGICO!

Os modernos computadores [10] reconhecem apenas dois símbolos/estados (0, 1) que podemos associar: sim ou não; verdadeiro ou falso, aberto e fechado; vivo ou morto. Isso os torna capazes de tomar apenas duas decisões (0, 1): acende ou apaga; direita ou esquerda; liga ou desliga, imprima ou não imprima.

O computador poderia funcionar com uma base numérica qualquer. Entretanto, os projetos de tais máquinas, que são programadas com o princípio de imitar o pensamento humano, tornam-se mais simples, concisos e eficientes com apenas dois estados para analisar (0, 1) e apenas duas decisões a tomar (0, 1).

O pensar binário surgiu muito antes das primeiras máquinas, e ele pode auxiliar as pessoas, em especial as impulsivas, a tomar decisões sem a influência de aspectos emocionais.

As atividades do dia-a-dia estão repletas de situações nas quais podemos aplicar o pensamento binário. Tomemos, por exemplo, um casal que planeja adquirir um automóvel, vamos chamá-los de ***R*** e ***S***. Recentemente o casal recebeu um telefonema do gerente do banco informando-lhes que caso desejassem dinheiro extra o crédito consignado estava pré-aprovado.

10 Recomendamos [18] para uma leitura descontraída e bem-humorada sobre este e outros temas.

A possibilidade de adquirir, enfim, o automóvel deixou ***R*** entusiasmado, afinal a aquisição de tal bem é um excelente negócio, seria o fim das longas esperas para utilizar o transporte público e, os gastos com combustível seriam irrisórios se comparados com o gasto do casal com passagens. Rapidamente ***S*** intervém dizendo que muitos economistas não consideram o automóvel um bem, e sim uma despesa, já que o mesmo desvaloriza rapidamente e demanda uma série de encargos, tais como a manutenção, combustível e impostos.

Temos um impasse, ***R*** considera a compra do automóvel um ótimo investimento, enquanto ***S*** acredita ser um péssimo negócio. Note que ***R*** e ***S*** possuem pontos de vistas complementares ou opostos, representados na tabela verdade que segue. Nas colunas temos o posicionamento de ***R*** e ***S***, (1) remete a compra do automóvel e (0) o contrário.

| **R** | **S** |
|---|---|
| 1 | 0 |
| 0 | 1 |

A opinião de ***R*** nega/barra a de ***S*** e vice-versa, uma decisão parece inatingível, ***S*** então, intervém novamente e faz uso dos seguintes argumentos: "*Todas as contas já foram pagas?*" "*Temos dinheiro sobrando?*" Denominando os argumentos de ***P*** e ***D***, respectivamente e, ***C*** para "*compra do automóvel*" temos a seguinte tabela verdade, segundo a análise de ***S***:

| **P** | **D** | **C** |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

Já ***R***, de posse dos mesmos argumentos, raciocinaria de acordo com a seguinte tabela verdade:

| **P** | **D** | **C** |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |

Repare que por ***S***, a compra do automóvel somente se concretizaria caso "*todas as contas estivessem pagas*" ***E*** "*houvesse dinheiro sobrando*", isto é, caso alguma das sen-

tenças não se verificasse, a compra não seria feita. Em contra partida, ***R*** compraria o automóvel caso "*todas as contas estiverem pagas*" ***OU*** "*houver dinheiro sobrando*", ou seja, se ao menos uma das condições for satisfeita então existe margem para adquirir o automóvel.

Doravante, em ambos os casos, tanto ***S*** quanto ***R*** não comprariam o automóvel caso não houvesse dinheiro sobrando e com contas ainda para serem pagas. Assim, além de facilitar a tomada de decisões de forma racional, quem utiliza tal recurso passa a enxergar com maior clareza todas as opções disponíveis. Cabe ressaltar que atribuímos o valor (1) quando a sentença se verifica e (0) quando a mesma não se verifica. Todavia, poderíamos ter invertidos os valores, (0) para a primeira e (1) para a segunda, já que esta é apenas uma questão de ponto de vista, pois como vimos, aquilo que é desfavorável para um, pode ser altamente favorável para o outro e vice-versa.

### 2.4.1 *A álgebra de Boole*

Em meados do século XIX, uma estrutura algébrica apropriada para a formulação, manipulação e otimização das funções lógicas foi proposta pelo matemático e filósofo britânico George Boole (1815-1864)[11], na qual, associou-se as leis do pensamento às da álgebra com o propósito de sistematizar a lógica do cálculo proposicional.

A álgebra booleana instaurou um conjunto universal de regras para combinar proposições que podem assumir valores verdadeiros ou falsos, de maneira a permitir a tomada de decisões, isto é, avaliar a sua veracidade ou falsidade. Tais proposições são combinadas utilizando três operações básicas: ***E***, ***OU*** e ***NÃO***, como visto nas tabelas acima.

Boole queria criar a "*álgebra das leis do pensamento*", para tanto, atribuía valores binários para as questões morais de sua época e, por meio de manipulações algébricas peculiares chegava a conclusões/decisões que eram precedidas de argumentos falsos e verdadeiros. A álgebra de Boole se aplica a qualquer sistema dual, isto é, cujas variáveis podem assumir apenas dois valores, discretos e mutuamente exclusivos, assim tal álgebra se aplica perfeitamente para resolver problemas do sistema de produção fabril, e em projetos de circuitos digitais utilizados nos modernos computadores. Curiosamente, a concepção da obra de Boole se deu em uma época em que não havia

11 Para saber mais, recomendamos a leitura de [31], [41]

eletrônica. Demorou quase um século para que Boole fosse compreendido, mas por fim, sua obra lhe rendeu o título de um dos pais da *Lógica Moderna*.

Tal como ocorre nos demais sistemas matemáticos dedutíveis, a álgebra booleana pode ser definida como um conjunto de elementos, um conjunto de operações e certa quantidade de axiomas.

**Definição 2.3** (*Álgebra booleana*)**.** Uma álgebra booleana é uma anel $\langle B, +, \cdot, 0, 1\rangle$ no qual cada um de seus elementos são idempotentes, isto é, $x^2 = x$ para todo $x \in B$.

**Teorema 2.4.** *Toda álgebra booleana é um anel booleano e vice-versa.*

*Demonstração.* Seja $(B, \wedge, \vee)$ uma álgebra booleana, munida das seguintes operações. Para todo $x, y \in B$ : $x + y = (x \wedge \overline{y}) \vee (\overline{x} \wedge y) = x \veebar y$ e $x \cdot y = x \wedge y$. Dado que o *encontro*($\vee$) e a *junção*($\wedge$) são comutativas, associativas e distributivas, $x + y$ e $x \cdot y$ também são. Além disto, tanto o elemento *zero* como o elemento *unidade* do anel são os mesmos da álgebra booleana.

Reciprocamente, todo anel booleano com elemento *unidade* é uma álgebra booleana com as seguintes operações: $x \vee y = x + y + x \cdot y$, $x \wedge y = x \cdot y$ e $\overline{x} = 1 + x$. □

Dado que ambas as construções são inversas, uma da outra, conclui-se que cada anel booleano surge de uma álgebra booleana e vice-versa. Assim, a classe de anéis booleanos é equivalente, em definição, à classe da álgebra booleana.

**Exemplo 2.5.** Dado o conjunto $A$, o conjunto de todos os subconjuntos de $A$, ou simplesmente conjunto das partes de $A$, denotado e definido por $\mathscr{P}(A) = \{X : X \subseteq A\}$. Segue que $\langle \mathscr{P}(A), \cup, \cap, ^c, \emptyset, A\rangle$ é uma álgebra booleana.

**Exemplo 2.6.** O anel $\mathbb{Z}_2 = \{[0], [1]\}$ é uma álgebra booleana.

**Exemplo 2.7.** O anel $\mathbb{Z}_2 \times \mathbb{Z}_2$ com a soma e produto coordenada a coordenada é uma álgebra booleana.

Todos os exemplos supracitados são anéis finitos, quanto ao exemplo 2.7 tomando os produtos cartesianos de $\mathbb{Z}_2$ e mantidas as operações coordenada a coordenada obtemos uma infinidade de anéis booleanos da forma: $\mathbb{Z}_2 \times \mathbb{Z}_2 \times \cdots \times \mathbb{Z}_2$.

**Proposição 2.5.** *$\mathbb{Z}_2$ é o único anel booleano que é corpo (salvo isomorfismo[12]).*

12 Sistemas modelados segundo a mesma estrutura algébrica são ditos isomorfos, as operações de cada sistema são equivalentes e relacionam-se entre si. Para tanto, basta realizar um mapeamento adequado entre os sistemas.

*Demonstração.* De fato, seja $B$ uma álgebra booleana que possui a estrutura algébrica de corpo, consequentemente, $B$ é um domínio de integridade. Temos para todo $x \in B$, $x^2 = x \Rightarrow x \cdot (x-1) = 0$. Isto posto, segue que $x = 0$ ou $x = 1$, ou seja, $B$ é isomorfo a $\mathbb{Z}_2$. □

A luz do que foi exposto, um questionamento é latente: A quantidade de elementos de um anel booleano finito é sempre da forma $2^k$, para $k \in \mathbb{N}$? Para responder tal questionamento, veremos alguns resultados intermediários, porém, não menos importantes:

**Lema 2.6.** *Seja $B$ um anel booleano, então $car(B) = 2$ (característica de $B$).*

*Demonstração.* Seja $x \in B$, segue da definição de anel booleano e de suas propriedades básicas que $(-x) = (-x)^2 = x^2 = x$. Logo, $x + x = 2x = x + (-x) = 0$. O que acarreta em $car(B) = 2$. □

**Lema 2.7.** *Todo anel booleano é comutativo.*

*Demonstração.* Seja $B$ um anel booleano, $x, y \in B$, segue que $x + y = (x+y)^2 = x^2 + x \cdot y + y \cdot x + y^2 = x + x \cdot y + y \cdot x + y$, o que implica em $x \cdot y + y \cdot x = 0$. Pelo lema 2.6 temos $x \cdot y = -y \cdot x = y \cdot x$ □

A abordagem de tais resultados nos permite avançar rumo à conclusão quanto a quantidade $m$ de elementos de um anel booleano finito, denotaremos a referida quantidade por $|B| = m$, com $m \in \mathbb{N} - \{0\}$.

**Teorema 2.8.** *Se $B$ é um anel booleano finito, então $|B| = 2^k$ para algum $k \in \mathbb{N}$.*

*Demonstração.* Dado $m$ a quantidade de elementos de $B$. Suponha pelo contrário, que apareça na decomposição de $m$ algum fator primo $p \neq 2$. Segue do lema 2.7 que $B$ é um grupo abeliano *(aditivo e comutativo)*. Vamos recorrer ao teorema 1.18 de Cauchy[13] para grupos abelianos finitos que nos garante a existência de um elemento $a \neq 0$ em $(B, +)$ tal que:

$$p \cdot a = \underbrace{a + a + \cdots + a}_{p \text{ vezes}} = 0$$

Evidentemente, $p = 2 \cdot n + 1$, para algum $n \in \mathbb{N}$. Logo $(2 \cdot n + 1) \cdot a = 0 \Rightarrow 2 \cdot n \cdot a + a = 0$. Do lema 2.6, $car(B) = 2$, segue que $2 \cdot n \cdot a = 0$ e portanto, $a = 0$. Absurdo. □

13 Este e outros resultados importantes estão disponíveis em [8] e [23]

Essencialmente, a álgebra booleana emprega conceitos da álgebra na teoria dos conjuntos e à lógica proposicional. Por assemelhar-se a álgebra dos números reais, adotou-se na álgebra booleana os símbolos + e · para os operadores designados por "*soma*" e "*produto*" lógicos e, 0 e 1 para os elementos. Com o intuito de algebrizar o cálculo proposicional define-se a seguinte estrutura algébrica:

*i*. $B = \{0, 1\}$

*ii*. $0 = \overline{1}$ e $1 = \overline{0}$

*iii*. $(1 \cdot 1) = (1 + 1) = (1 + 0) = (0 + 1) = 1$

*iv*. $(0 + 0) = (0 \cdot 0) = (0 \cdot 1) = (1 \cdot 0) = 0$

Abaixo segue o quadro que retrata a associação da álgebra de Boole à teoria dos conjuntos e à lógica proposicional:

| Teoria dos conjuntos | Lógica proposicional | Álgebra de Boole |
|---|---|---|
| $\cap$ | $\wedge$ | $\cdot$ |
| $\cup$ | $\vee$ | $+$ |
| $\emptyset$ | $F$ | 0 |
| $U$ | $V$ | 1 |
| $^c$ | $\neg$ | $^-$ |

Assim, o "1" expressa o conceito lógico verdadeiro ou o conjunto universo, enquanto que o "0" remete ao conceito lógico falso ou ao conjunto vazio. A soma "+" equivale ao *OU* lógico ou à união de conjuntos e, a multiplicação "·" consiste no *E* lógico ou a intersecção de conjuntos. Os axiomas a seguir visam garantir tais equivalências, bem como, conferir à álgebra booleana a estrutura algébrica de anel[14].

#### 2.4.1.1 *Axiomas, teoremas e manipulação algébrica*

Axiomas e postulados constituem um conjunto de definições que normatizam as regras das operações algébricas. Os axiomas constituem um conjunto mínimo de regras aceito como verdade, isto é, que não podem ser demonstrados. Já os teoremas são relações que viabilizam a manipulação algébrica e, são passíveis de demonstração via axiomática, ou por meio de outros teoremas demonstrados anteriormente[15].

14 Estas e outras demonstrações são aprofundadas em [27] e [45]
15 Indicamos [9] e [13] para posterior aprofundamento

• ***Axiomas:*** Seja $B$ um conjunto com 2 ou mais valores, Boole define os axiomas sobre as variáveis booleanas (0, 1) e sobre os operadores elementares ***E***: *produto lógico* $( \cdot )$, ***OU***: *soma lógica* $( + )$ e ***NÃO***: *inversor, negação ou complementar*, de modo que para todo $X, Y$ e $Z \in B$ temos:

**A1 - Operações:** *Fechamento em relação aos operadores* "·" *e* "+".

$$X + Y \in B$$
$$X \cdot Y \in B$$

**A2 - Elemento identidade:** *Existem valores 0 e 1 tais que:*

$$X + 0 = X$$
$$X \cdot 1 = X$$

**A3 - Comutativa:**

$$X + Y = Y + X$$
$$X \cdot Y = Y \cdot X$$

**A4 - Associativa:**

$$X + (Y + Z) = (X + Y) + Z$$
$$X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z$$

**A5 - Distributiva:**

$$(X \cdot Y) + (X \cdot Z) = X \cdot (Y + Z) =$$
$$(X + Y) \cdot (X + Z) = X + Y \cdot Z$$

**A6 - Complemento:** *Para todo* $X \in B$ *existe um elemento* $\overline{X} \in B$*, denominado complemento de X, tal que:*

$$X + \overline{X} = 1$$
$$X \cdot \overline{X} = 0$$

Notoriamente, qualquer proposição na álgebra booleana possui sua *dual*. A dualidade é característica de sistemas compostos por duas unidade ou dois elementos.

Os axiomas foram apresentados aos pares e, cada axioma pode ser obtido por meio do outro simplesmente efetuando a substituição de 0 por 1 e das operações de "+" por "·" e vice-versa.

**Definição 2.9** (Princípio da dualidade)**.** O dual de qualquer proposição de uma álgebra booleana $(B, +, \cdot, 0, 1)$ é uma proposição derivada da troca de "+" e "·", de seus elementos identidades "1" e "0" na proposição original.

O princípio da dualidade também está presente nos teoremas da álgebra booleana, dada sua demonstração via axiomática, se aplicarmos o dual de cada um dos axiomas utilizados, então teremos demonstrado o dual do teorema inicial.

**Teorema 2.10** (Teorema da dualidade)**.** *O dual de qualquer teorema na álgebra booleana é também um teorema.*

Diante do exposto, para os teoremas que apresentaremos a seguir, para aqueles que possuírem expressões duais, será demonstrada apenas uma delas, tendo em vista que a veracidade de teoremas duais seguem de forma automática.

• ***Teoremas:*** Um conjunto básico de teoremas possibilitará a manipulação algébrica, afinal não é viável embasar todas as transformações por meio de axiomas. Os teoremas aplicar-se-ão nas simplificações de expressões booleanas, uma vez que é extremamente útil transformar expressões booleanas complexas em expressões equivalentes, porém mais simples[16]. Para todo $X, Y$ e $Z \in B$ temos:
**T1** - *Idempotência*

$$X + X = X$$
$$X \cdot X = X$$

*Demonstração:* $X \overset{A2}{=} X + 0 \overset{A6}{=} X + X \cdot \overline{X} \overset{A5}{=} (X + X) \cdot (X + \overline{X}) \overset{A6}{=} (X + X) \cdot 1 \overset{A2}{=} X + X$ □

**T2** - *Elemento identidade*

$$X + 1 = 1$$
$$X \cdot 0 = 0$$

*Demonstração:* $X + 1 \overset{A2}{=} X + (X + \overline{X}) \overset{A4}{=} (X + X) + \overline{X} \overset{T2}{=} X + \overline{X} \overset{A6}{=} 1$ □

**T3** - *Involução*

$$\overline{\overline{X}} = X$$

16 Mais simples pode ser entendido como menor.

*Demonstração: Seja* $\overline{X} = Y$*, temos:*
$Y \cdot X \overset{A3}{=} X \cdot Y \overset{def}{=} X \cdot \overline{X} \overset{A6}{=} 0 \quad e \quad Y + X \overset{A3}{=} X + Y \overset{def}{=} X + \overline{X} \overset{A6}{=} 1$
*Logo,* $X = \overline{Y} = \overline{\overline{X}}$ □

**T4** - *Absorção I*

$$X + X \cdot Y = X$$
$$X \cdot (X + Y) = X$$

*Demonstração:* $X + X \cdot Y \overset{A2}{=} X \cdot 1 + X \cdot Y \overset{A6}{=} X \cdot (Y + \overline{Y}) + X \cdot Y \overset{A5}{=} X \cdot Y + X \cdot \overline{Y} + X \cdot Y \overset{A3}{=}$
$X \cdot Y + X \cdot Y + X \cdot \overline{Y} \overset{T2}{=} X \cdot Y + X \cdot \overline{Y} \overset{A5}{=} X \cdot (Y + \overline{Y}) \overset{A6}{=} X \cdot 1 \overset{A2}{=} 1$ □

**T5** - *Absorção II*

$$X + \overline{X} \cdot Y = X + Y$$
$$X \cdot (\overline{X} + Y) = X \cdot Y$$

*Demonstração:* $X + \overline{X} \cdot Y \overset{T4}{=} (X + X \cdot Y) + \overline{X} \cdot Y \overset{A5}{=} X + Y \cdot (X + \overline{X}) \overset{A6}{=} X + Y \cdot 1 \overset{A2}{=} X + Y$ □

**T6** - *Adjacência*

$$X \cdot Y + X \cdot \overline{Y} = X$$
$$(X + Y) \cdot (X + \overline{Y}) = X$$

*Demonstração:* $X \cdot Y + X \cdot \overline{Y} \overset{A5}{=} X \cdot (Y + \overline{Y}) \overset{A6}{=} X \cdot 1 \overset{A2}{=} X$ □

**T7** - *Inversão "Teorema de De Morgan"*

$$\overline{X + Y} = \overline{X} \cdot \overline{Y}$$
$$\overline{X \cdot Y} = \overline{X} + \overline{Y}$$

*Demonstração: Devemos mostrar que* $\overline{X} \cdot \overline{Y}$ *é o complemento de* $X + Y$, temos:
$(X + Y) \cdot \overline{X} \cdot \overline{Y} \overset{A2}{=} X \cdot \overline{X} \cdot \overline{Y} + Y \cdot \overline{X} \cdot \overline{Y} \overset{A6}{=} 0 \cdot \overline{Y} + 0 \cdot \overline{X} \overset{def}{=} 0 + 0 = 0$ *e,*
$(X + Y) + \overline{X} \cdot \overline{Y} \overset{A2}{=} (X + Y + \overline{X}) \cdot (X + Y + \overline{Y}) \overset{A6}{=} (1 + \overline{Y}) \cdot (1 + \overline{X}) \overset{def}{=} 1 \cdot 1 = 1$
*Logo:* $\overline{X + Y} = \overline{X} \cdot \overline{Y}$ □

**T8** - *Consenso*

$$X\cdot Y + \overline{X}\cdot Z + Y\cdot Z = X\cdot Y + \overline{X}\cdot Z$$
$$(X+Y)\cdot(\overline{X}+Z)\cdot(Y+Z) = (X+Y)\cdot(\overline{X}+Z)$$

*Demonstração:* $X\cdot Y + \overline{X}\cdot Z + Y\cdot Z \overset{A6}{=} X\cdot Y + \overline{X}\cdot Z + Y\cdot Z\cdot(X+\overline{X}) \overset{A5}{=} X\cdot Y + \overline{X}\cdot Z + Y\cdot Z\cdot X + Y\cdot Z\cdot\overline{X} \overset{A3}{=} X\cdot Y + X\cdot Y\cdot Z + \overline{X}\cdot Z + \overline{X}\cdot Z\cdot Y \overset{A5}{=} X\cdot Y\cdot(1+Z) + \overline{X}\cdot Z\cdot(1+Y) \overset{T2}{=} X\cdot Y\cdot 1 + \overline{X}\cdot Z\cdot 1 \overset{A2}{=} X\cdot Y + \overline{X}\cdot Z$ □

A validade dos axiomas e a veracidade dos teoremas e afirmações podem ser obtidas também por meio da construção de tabelas verdade. Duas expressões lógicas são ditas equivalentes quando apresentam a mesma tabela verdade. Construiremos a tabela verdade com todas as possíveis combinações de valores para $X, Y$ e $Z$ com o intuito de verificar a validade da distributividade em relação a soma, ou seja, $(X+Y)\cdot(X+Z) = X + Y\cdot Z$:

| $X$ | $Y$ | $Z$ | $Y\cdot Z$ | $X+Y$ | $X+Z$ | $\mathbf{X+Y\cdot Z}$ | $\mathbf{(X+Y)\cdot(X+Z)}$ |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | **0** | **0** |
| 0 | 0 | 1 | 0 | 0 | 1 | **0** | **0** |
| 0 | 1 | 0 | 0 | 1 | 0 | **0** | **0** |
| 0 | 1 | 1 | 1 | 1 | 1 | **1** | **1** |
| 1 | 0 | 0 | 0 | 1 | 1 | **1** | **1** |
| 1 | 0 | 1 | 0 | 1 | 1 | **1** | **1** |
| 1 | 1 | 0 | 1 | 1 | 1 | **1** | **1** |
| 1 | 1 | 1 | 1 | 1 | 1 | **1** | **1** |

Repare que para a mesma combinação de valores para $X, Y$ e $Z$, as expressões $X + Y\cdot Z$ e $(X+Y)\cdot(X+Z)$ assumem os mesmos valores lógicos, e desta forma, tais expressões são ditas equivalentes o que confere à álgebra booleana a propriedade distributiva.

Em eletrônica digital, os teoremas serão utilizados em técnicas de simplificação algébrica das expressões lógicas, tais procedimentos são relevantes por viabilizar a implementação de circuitos digitais mais compactos, reduzindo seu custo.

## 2.5 PORTAS LÓGICAS E SUAS FUNÇÕES

Em meados do século *XX*, o matemático, engenheiro eletrônico e criptógrafo norte americano, Claude Shannon (1916 - 2001) adaptou a álgebra de Boole para os circuitos elétricos, as variáveis lógicas (0, 1) por meio de interruptores (desligado, ligado)

e *relés* para as operações. Seus esforços lhe renderam o título de "*pai da teoria da informação*".

Com o avanço da tecnologia surge a eletrônica digital, empregada em computadores, calculadoras, codificadores, decodificadores, sistemas de controle de automação, entre muitos outros. Os circuitos digitais utilizam um apanhado de funções lógicas, que associam a $n$ variáveis independentes (entradas) uma única variável dependente (saída), de acordo como uma ou mais regras associadas, (portas lógicas/conectores lógicos).

Um computador, ou mesmo, uma simples calculadora é composta por inúmeros blocos integrados, as portas lógicas. São responsáveis por fornecer a resposta dada pelo circuito lógico que, por sua vez, dependerá das variáveis lógicas que adentram nas portas lógicas e de quais regras booleanas as regem.

Torna-se interessantíssimo observar que, com uma pequena quantidade de operações, podemos obter uma infinidade de operações mais complexas que se interligam e aumentam exponencialmente a capacidade das máquinas em resolver problemas e, associando isto à velocidade de execução, nos faz crer que tais máquinas são realmente dotadas de inteligência.

Isto posto, para compreender como as máquinas realizam tantas operações, com os mais variados níveis de complexidade, primeiramente temos que conhecer e compreender as operações mais básicas, denominadas por portas lógicas[17]. Vamos a elas:

### 2.5.1 *Operação NOT*

***Operação NOT** (Não)*: é a operação inversora, isto é, para qualquer entrada A, a mesma define-se como:

$$f(A) = \overline{A}$$

Ou seja, é a entrada negada (barrada).

17 Passaremos a adotar os termos em inglês para denotarmos as portas e funções lógicas, bem como, a representação padrão das mesmas, tal como aparecem nas literaturas. Para aprofundar-se, recomendamos [12] e [36]

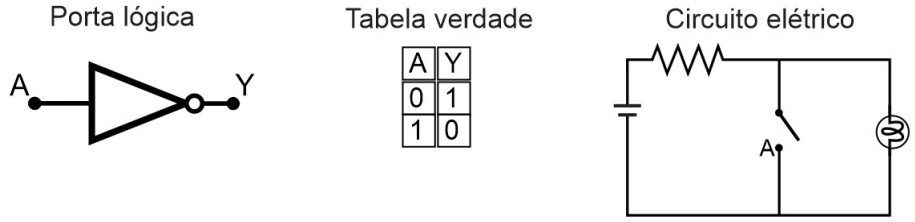


Figura 4: Operação NOT

### 2.5.2 *Operação AND*

***Operação AND*** *(E)*: operação produto lógico, para as entradas $A_1, \ldots, A_n$. Tal operação define-se como:

$$f(A_1, \ldots, A_n) = \prod_{i=1}^{n} A_i$$

Esta operação assume valor 1 apenas quando todas as entradas forem 1, para duas entradas, **A** e **B** temos:


Figura 5: Operação AND

### 2.5.3 *Operação OR*

***Operação OR*** *(OU)*: operação soma lógica, para as entradas $A_1, \ldots, A_n$. Tal operação define-se como:

$$f(A_1, \ldots, A_n) = \sum_{i=1}^{n} A_i$$

Esta operação assume valor 1 quando qualquer uma das entradas forem 1, para duas entradas, **A** e **B** temos:


Figura 6: Operação OR

### 2.5.4 *Operação NAND*

***Operação NAND*** *(NÃO-E)*: é a operação *AND* negada, para duas entradas, **A** e **B**, Tal operação define-se como:

$$f(A, B) = \overline{A \cdot B}$$

Esta operação assume valor 0 apenas quando todas as entradas forem 1, temos:


Figura 7: operação NAND

### 2.5.5 *Operação NOR*

***Operação NOR*** *(NÃO-OU)*: é a operação *OU* negada, para duas entradas, **A** e **B**. Tal operação define-se como:

$$f(A, B) = \overline{A + B}$$

Esta operação assume valor 1 apenas quando todas as entradas forem 0, temos:


Figura 8: Operação NOR

### 2.5.6 *Operação XOR*

***Operação XOR** (OU EXCLUSIVO)*: definida para duas entradas apenas, **A** e **B** como sendo:

$$f(A, B) = A \oplus B = \overline{A} \cdot B + A \cdot \overline{B}$$

Esta operação assume valor 1 apenas quando as entradas forem distintas, para ilustrar esta operação suponha que dois atletas $R$ e $S$ disputem os 100 metros rasos e que ambos completem a prova. Isto posto, como afirmações verdadeiras temos: ou $R$ "vence a prova" (1, 0), ou $S$ "vence a prova" (0, 1). Como afirmações falsas: Nenhum dos dois "vence a prova" (0, 0) ou ambos "vencem" (1, 1).


Figura 9: Operação XOR

### 2.5.7 *Operação XNOR*

***Operação XNOR** (NÃO-OU EXCLUSIVO)*: é a operação inversa da *XOR*, definida para duas entradas apenas, **A** e **B** como sendo:

$$f(A, B) = A \odot B = \overline{A} \cdot \overline{B} + A \cdot B$$

Esta operação assume valor 1 apenas quando as entradas forem idênticas, por este motivo, tal operação também é conhecida como COMPARAÇÃO.


Figura 10: Operação XNOR

## 2.6 FUNÇÕES E SUAS PORTAS LÓGICAS

Uma *expressão booleana* é qualquer combinação das $n$ variáveis lógicas de acordo com as operações elementares: *soma, produto e complemento*. Considere a seguinte expressão booleana:

$$A \cdot (\overline{A} \oplus B)$$

A expressão acima utiliza três portas lógicas: *NOT*, *AND* e *XOR*. Como visto anteriormente, a mesma pode ser simplificada via álgebra booleana:

$$A \cdot (\overline{A} \oplus B) \overset{A5}{=} (A \cdot \overline{A}) \oplus (A \cdot B) \overset{A6}{=} 0 \ \oplus \ (A \cdot B) \overset{A2}{=} A \cdot B$$

A simplificação representa a mesma função utilizando apenas uma porta lógica *AND*, quando uma expressão for derivada de outra, diremos que elas são *equivalentes* e, logicamente, opta-se pela mais simples.

**Definição 2.11** (*Função booleana*)**.** Seja $\mathbb{B}$ um conjunto munido de três operações (*soma, produto, complemento*). Uma função booleana de $n$ variáveis $X_1, X_2, \ldots, X_n$ é uma função $f : \mathbb{B}^n \to \mathbb{B}$ de tal modo que $f(X_1, X_2, \ldots, X_n)$ é uma expressão booleana[18].

As $n$ variáveis *(independentes)*, determinam um conjunto discreto de $2^n$ possíveis valores *(domínio da função booleana)* para as $n$ entradas, que de acordo com as expressões booleanas *(lei de formação da função booleana)* fornecem como saída um dos elementos do conjunto $\{0, 1\}$ *(contradomínio da função booleana)*.

18 Para aprofundar-se sobre o tema, recomendamos [9]

Para verificar a equivalência de expressões booleanas podemos utilizar também as ***tabelas verdade*** nas quais constam os valores da função booleana para cada uma das $2^n$ combinações das $n$ variáveis independentes. Tal tabela é única para cada função, possui $2^n$ linhas, cuja ordem obedece à combinação de níveis lógicos das variáveis independentes correspondendo ao equivalente decimal dos números binários por elas formados. Vamos constatar a veracidade da simplificação acima via tabela verdade:

| **A** | **B** | $\overline{A}$ | $\overline{A} \oplus B$ | $A \cdot (\overline{A} \oplus B)$ | $A \cdot B$ |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | **0** | **0** |
| 0 | 1 | 1 | 0 | **0** | **0** |
| 1 | 0 | 0 | 0 | **0** | **0** |
| 1 | 1 | 0 | 1 | **1** | **1** |

Como $A \cdot (\overline{A} \oplus B)$ e $A \cdot B$ apresentam valores lógicos idênticos na tabela verdade, segue que, de fato, as expressões são equivalentes.

De posse das expressões booleanas podemos representá-las poe meio dos ***circuitos lógicos*** que configuram na representação gráfica das expressões booleanas formadas pela interligação de símbolos (portas lógicas) que representam os operadores elementares. Por convenção, as entradas ficam à esquerda e a saída à direita do circuito. Por mais complexo que sejam os circuitos lógicos, todos, sem exceção, são apenas combinações das portas lógicas. Consta na figura 11 a representação gráfica das funções abordadas anteriormente utilizando os símbolos padrão das portas lógicas: No circuito

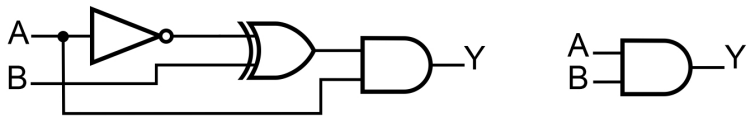


Figura 11: Circuito de $Y = A \cdot (\overline{A} \oplus B)$ e de seu equivalente $Y = A \cdot B$

lógico acima, a porta ***NOT*** pode também aparecer apenas com um pequeno círculo antes da entrada da porta lógica seguinte. Como ilustrado na figura 12.

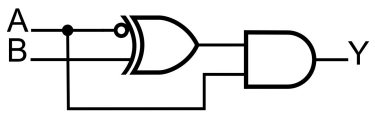


Figura 12: Circuito de $Y = A \cdot (\overline{A} \oplus B)$

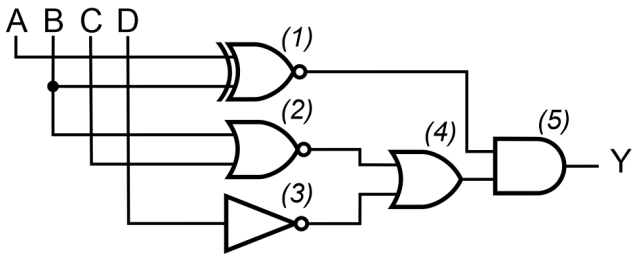


Figura 13: Circuito lógico com expressão a determinar

**Exemplo 2.8.** Vamos obter a expressão booleana a partir do circuito lógico da figura 13 logo acima. Neste circuito temos ao todo quatro literais (*variáveis independentes*) sendo que em *(1)* os literais $A$ e $B$ estão conectados a uma porta ***XNOR***, logo teremos o termo: $A \odot B$. Em seguida, temos $B$ e $C$ conectados a uma porta ***NOR***, que dará origem ao termo: $\overline{(A+B)}$ em *(2)*, já em *(3)* temos o literal **D** conectado a uma porta ***NOT***, ou seja: $\overline{D}$. Em *(4)* temos uma porta ***OR*** cujas entradas são as saídas de *(2)* e *(3)*, isto é: $\overline{(A+B)}+\overline{D}$ e, por fim, em *(5)* temos uma porta ***AND*** cujas entradas são as saídas de *(4)* e *(1)*, logo, a nossa expressão é: $(\overline{(A+B)}+\overline{D})\cdot(A \odot B)$.

**Exemplo 2.9.** Vejamos a configuração de alguns teoremas da álgebra de Boole via portas lógicas:

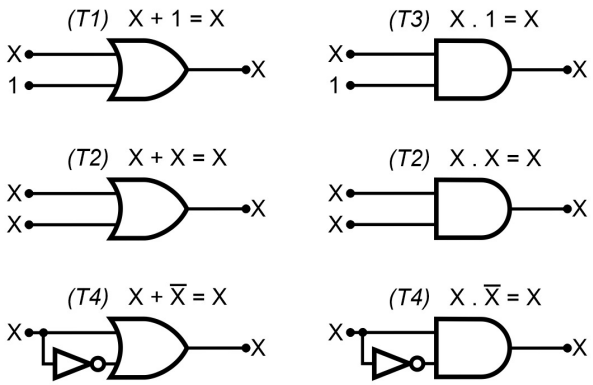


Figura 14: Teoremas de uma única variável

Podemos ler a tabela verdade e dela, deduzir a expressão booleana e desta, projetar o circuito lógico, não necessariamente nesta ordem. A representação de uma função booleana por qualquer uma das três formas e as mesmas se interligam de forma intrínseca como veremos a seguir.

### 2.6.1 *Formas Canônicas*

Dada uma função booleana representada em uma tabela verdade, podemos obter a expressão booleana de forma relativamente simples, realizando a soma lógica dos termos mínimos (**MINTERM's**), isto é, os termos para os quais a função booleana vale 1. Para tanto, devemos numerar as linhas e, assim, identificar quais são os termos que devem ser somados, ou seja, vamos "somar" os produtos lógicos dos termos mínimos. Tal expressão será denominada *soma canônica* ou *soma-de-produtos*, e será nosso ponto de partida para a construção de circuitos lógicos. Vamos explorar a tabela verdade da função ***XOR***.

| *d* | **A** | **B** | **Y** | **MINTERM's** |
|---|---|---|---|---|
| *0* | 0 | 0 | 0 | $\overline{A}\cdot\overline{B}$ |
| ***1*** | 0 | 1 | 1 | $\overline{A}\cdot B*$ |
| ***2*** | 1 | 0 | 1 | $A\cdot\overline{B}*$ |
| *3* | 1 | 1 | 0 | $A\cdot B$ |

A função é verdadeira (*assume valor 1*) nas linhas *1* e *2*, assim, devemos "somar" os termos mínimos *1* e *2*:

$$Y = f(A,B) = \sum m(1,2) = \overline{A}\cdot B + A\cdot\overline{B}$$

Pelo princípio da dualidade, podemos estender os procedimentos anteriores para os **MAXTERM's** ou *produto-de-somas*. De forma dual vamos tomar as linhas para as quais a função vale *0*, e "multiplicar" as somas lógicas dos termos máximos.

| *d* | **A** | **B** | **Y** | **MAXTERM's** |
|---|---|---|---|---|
| ***0*** | 0 | 0 | 0 | $\overline{A}+\overline{B}*$ |
| *1* | 0 | 1 | 1 | $\overline{A}+B$ |
| *2* | 1 | 0 | 1 | $A+\overline{B}$ |
| ***3*** | 1 | 1 | 0 | $A+B*$ |

A função é falsa (*assume valor 0*) nas linhas *0* e *3*, assim, devemos "multiplicar" os termos máximos *0* e *3*:

$$Y = f(A,B) = \prod M(0,3) = (A+B)\cdot(\overline{A}+\overline{B})$$

Naturalmente, ambas as expressões obtidas para a função ***XOR*** são equivalentes, veja:

$$\prod M(0,3) = (A+B)\cdot(\overline{A}+\overline{B}) \overset{T7}{=} A\cdot\overline{A} + A\cdot\overline{B} + \overline{A}\cdot B + B\cdot\overline{B} \overset{T4}{=}$$
$$0 + A\cdot\overline{B} + \overline{A}\cdot B + 0 \overset{A4,A5}{=} A\cdot\overline{B} + \overline{A}\cdot B = \sum m(1,2)$$

De forma sucinta, temos que $Y = f(A,B) = \sum m(1,2) = \prod M(0,3)$.

Abaixo segue os circuitos lógicos de ambas as expressões:

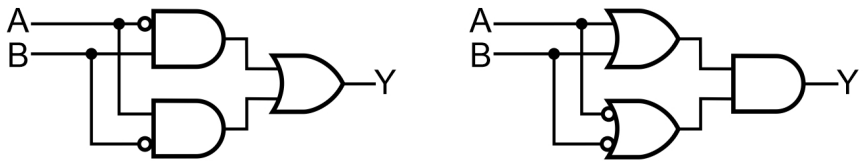


Figura 15: circuito lógico de $\overline{A}\cdot B + A\cdot\overline{B}$ e circuito lógico de $(A+B)\cdot(\overline{A}+\overline{B})$

Vamos obter a expressão booleana de uma tabela verdade com três literais:

| *d* | **A** | **B** | **C** | **Y** |
|---|---|---|---|---|
| *0* | 0 | 0 | 0 | 0 |
| *1* | 0 | 0 | 1 | 0 |
| ***2*** | 0 | 1 | 0 | 1 |
| *3* | 0 | 1 | 1 | 0 |
| *4* | 1 | 0 | 0 | 0 |
| *5* | 1 | 0 | 1 | 0 |
| ***6*** | 1 | 1 | 0 | 1 |
| ***7*** | 1 | 1 | 1 | 1 |

A função assume valor lógico **1** para as linhas *2* , *6* e *7*, e desta forma temos:

$$Y = f(A,B,C) = \sum m(2,6,7) = \overline{A}\cdot B\cdot\overline{C} + A\cdot B\cdot\overline{C} + A\cdot B\cdot C$$

Tal expressão pode ser simplificada, observe:

$$\overline{A}\cdot B\cdot\overline{C} + A\cdot B\cdot\overline{C} + A\cdot B\cdot C \overset{T2}{=} \overline{A}\cdot B\cdot\overline{C} + (A\cdot B\cdot\overline{C} + A\cdot B\cdot\overline{C}) + A\cdot B\cdot C \overset{T7}{=}$$

$$B\cdot\overline{C}\cdot(\overline{A}+A) + A\cdot B\cdot(\overline{C}+C) \overset{T4}{=} B\cdot\overline{C} + A\cdot B$$

Contudo, o processo de simplificação poderia se dar diretamente na tabela verdade. Note que o termo $A\cdot B$ é proveniente da junção dos *minterm's* $A\cdot B\cdot C$ e $A\cdot B\cdot\overline{C}$ que

correspondem às linhas *6* e *7*, respectivamente. Em tais linhas, a função apresenta o valor lógico **1** quando $A$ e $B$ valem 1 simultaneamente. Diferem apenas no valor de $C$, logo, podemos inferir que a função assumirá valor lógico **1** independentemente do valor que $C$ assumirá e, desta forma, podemos descartá-lo, donde surge $Y = f(A,B,C) = A \cdot B + \ldots$

O mesmo raciocínio pode ser empregado para $B \cdot \overline{C}$, oriundo da junção de $\overline{A} \cdot B \cdot \overline{C}$ e $A \cdot B \cdot \overline{C}$ referentes às linhas *2* e *6*, respectivamente. A função assume valor lógico **1** em ambas as linhas quando $B$ vale 1 e $C$ vale 0, estas linhas diferem apenas no valor de A, cujo valor não é determinante para o valor da função, assim, $Y = f(A,B,C) = B \cdot \overline{C} + \ldots$

Esgotados os "*1*"*s*, concluímos que $Y = f(A,B,C) = A \cdot B + B \cdot \overline{C}$ tal como mostramos anteriormente.

Resumidamente, podemos simplificar as expressões booleanas diretamente da tabela verdade, por meio de associações de linhas para as quais a função assume valor lógico **1** e diferem no valor de apenas uma variável. Designaremos tais linhas como ***adjacentes***.

Evidentemente, quando as linhas não forem subsequentes, como ocorrido com as linhas *2* e *6*, o processo pode demandar cuidados e atenção redobrada, e pode tornar-se árduo, principalmente para funções com mais de três variáveis. Para facilitar e agilizar este processo, idealizou-se uma nova forma de organização da tabela verdade, que abordaremos a seguir.

#### 2.6.1.1 *Mapa de Karnaugh*

O mapa de Karnaugh é um método prático para transformar uma tabela verdade idealizada em um circuito implementável, simplificado e otimizado. Contudo, se tivermos somente dez minutos para fornecer tal circuito a partir de uma tabela verdade, é aconselhável que passemos nove minutos preparando e organizando o *mapa*.

Como exemplo, voltemos para a tabela verdade acima abordada, que forneceu como expressão simplificada $Y = f(A,B,C) = A \cdot B + B \cdot \overline{C}$. Vamos aprimorá-la via mapa de Karnaugh. Para tanto, buscaremos linhas adjacentes (linhas que diferem pelo valor de apenas uma das variáveis). Logo para a plotagem do mapa, de uma coluna/linha para outra, devemos ter a variação de apenas *1 bit*, caso a ordem original seja mantida, de (0 1) para (1 0) teríamos a variação de ambos os *bits*.

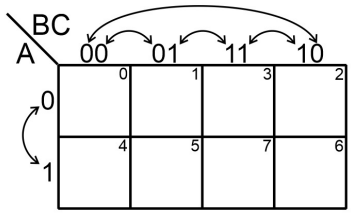


Figura 16: Adjacências no mapa de Karnaugh

Os arcos da figura 16 indicam as adjacências das linhas da tabela verdade. O arco maior unindo o primeiro 0 ao último nos diz que há adjacência entre os extremos do mapa, tal como ocorre com a versão planificada do mapa múndi. Na sequência, vamos numerar as células tal como numeramos as linhas da tabela verdade, isto é, a numeração se dá de acordo com o correspondente decimal dos números binários associados às linhas da tabela verdade.

Nesta nova organização, as linhas da tabela verdade passam a ser literalmente adjacentes. Para o próximo passo, vamos preencher somente as células (linhas na tabela) para as quais a função assume valor lógico 1, isto é, dado que $Y = f(A,B,C) = \sum m(2,6,7)$, vamos tomar apenas as células *2, 6* e *7*.

Passemos agora para a simplificação da expressão booleana, no mapa de Karnaugh podemos agrupar os termos adjacentes ("*1*"*s*) associando-os sempre em conjuntos de $2^n$ elementos, ou seja, conjuntos com 1 elemento, com 2, com 4, com 8, etc.

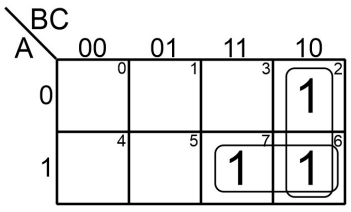


Figura 17: Agrupamentos possíveis para termos adjacentes

Em tais agrupamentos é facultado o reaproveitamento de termos de grupos já formados. A sobreposição de grupos é válida tal como exposto na figura 17, entretanto, se determinado grupo contiver todos os seus elementos também pertencentes a outros grupos, então o mesmo deverá ser desconsiderado, por ser obsoleto.

Uma vez formados os grupos tais como os descritos na figura 18, vamos tomar para compor a expressão booleana da função, apenas os literais que se mantêm constantes

em tal grupo. No agrupamento formado pelos *minterm's (6, 7)*, observe que tanto $A$, quanto $B$ são constantes e valem 1, enquanto que $C$ deverá ser descartado por variar de estado lógico, logo, $C$ não é determinante para que a função assuma valor lógico 1, donde resulta em $A \cdot B$.

Analogamente, no grupo formado pelos *minterm's (2, 6)*, enquanto que $B$ e $C$ são constantes, valendo *1* e *0* respectivamente, o literal $A$ varia, logo a função vale 1 nestas linhas independentemente do valor lógico de $A$ e, portanto, o mesmo pode e será descartado, resultando $B \cdot \overline{C}$.

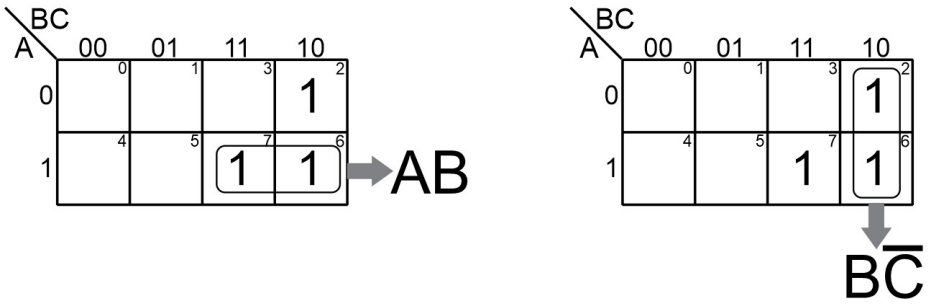


Figura 18: Agrupamentos dos *minterm's (6, 7)* e *minterm's (2, 6)*

Pelo *principio da dualidade*, poderíamos realizar o *produto-de-somas* com os *maxterm's*. Para tanto, preencheríamos as células do mapa de Karnaugh onde a função assume valor *0*. Os demais procedimentos seguem de forma análoga.

**Exemplo 2.10.** Vejamos como proceder em uma tabela verdade que apresenta ao todo quatro variáveis:

| *d* | **A** | **B** | **C** | **D** | **Y** |
|---|---|---|---|---|---|
| *0* | 0 | 0 | 0 | 0 | 0 |
| ***1*** | 0 | 0 | 0 | 1 | 1 |
| *2* | 0 | 0 | 1 | 0 | 0 |
| *3* | 0 | 0 | 1 | 1 | 0 |
| ***4*** | 0 | 1 | 0 | 0 | 1 |
| *5* | 0 | 1 | 0 | 1 | 0 |
| ***6*** | 0 | 1 | 1 | 0 | 1 |
| *7* | 0 | 1 | 1 | 1 | 0 |
| *8* | 1 | 0 | 0 | 0 | 0 |
| *9* | 1 | 0 | 0 | 1 | 0 |
| *10* | 1 | 0 | 1 | 0 | 0 |
| ***11*** | 1 | 0 | 1 | 1 | 1 |
| ***12*** | 1 | 1 | 0 | 0 | 1 |
| *13* | 1 | 1 | 0 | 1 | 0 |
| ***14*** | 1 | 1 | 1 | 0 | 1 |
| ***15*** | 1 | 1 | 1 | 1 | 1 |

A função assume valor lógico **1** para as linhas *1*, *4*, *6*, *11*, *12*, *14* e *15*, donde segue: $Y = f(A, B, C) = \sum m(1, 4, 6, 11, 12, 14, 15)$.

Abaixo segue o quadro com as adjacências das linhas da tabela verdade para quatro variáveis, 19, bem como o quadro com as células preenchidas para as linhas em que a função vale **1**.

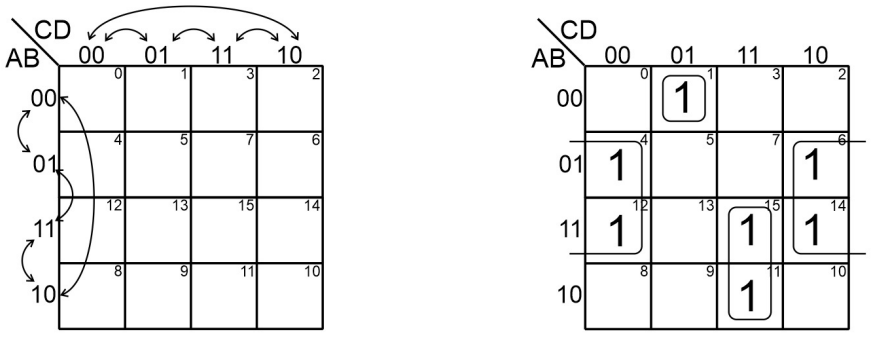


Figura 19: Adjacências para quatro variáveis no mapa de Karnaugh à esquerda e os agrupamentos possíveis para adjacentes iguais à direita

Recorremos à figura 20 para analisar os agrupamentos no *minterm (1)*, temos todos os literais: $\overline{A}\cdot\overline{B}\cdot\overline{C}\cdot D$. Logo, nenhum literal será descartado quando o grupo contiver apenas um único elemento.

Já no agrupamento dos *minterm (4,6,12,14)* os únicos literais que se mantêm constantes são: $B$ com o valor 1 e D com o valor 0, daí temos $B\cdot\overline{D}$. Por fim, no agrupamento dos *minterm (11,15)*, os literais $A, C$ e $D$ permanecem constantes dentro grupo, todos valendo 1, enquanto que o literal $B$ varia e, por este motivo, será descartado, assim temos $A\cdot C\cdot D$.

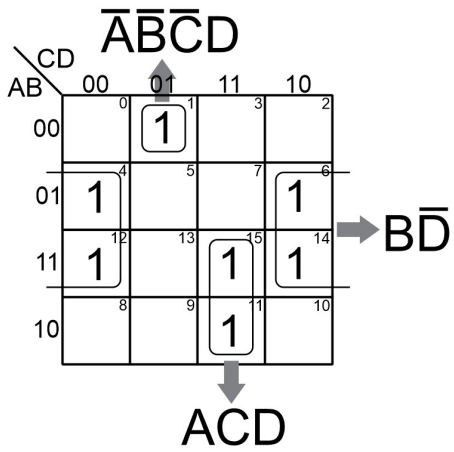


Figura 20: Termos resultantes dos agrupamentos das linhas adjacentes

Findadas todas as linhas em que a função assume valor lógico "*1*", concluímos que $Y = f(A,B,C) = \overline{A}\cdot\overline{B}\cdot\overline{C}\cdot D + B\cdot\overline{D} + A\cdot C\cdot D$.

Ainda que seja possível manipular a expressão, o mapa de Karnaugh costuma fornecer o circuito mais otimizado possível. Doravante, efetuada a "*simplificação*", obteremos uma expressão equivalente cuja quantidade de portas lógicas utilizadas permanecerá constante.

Segue abaixo, a estrutura dos mapas de Karnaugh para cinco e seis variáveis, figuras 21 e 22 respectivamente.

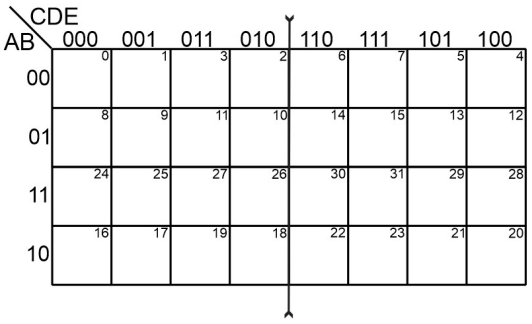


Figura 21: Estrutura do mapa de Karnaugh de 5 variáveis, sendo $A$ a variável mais significativa

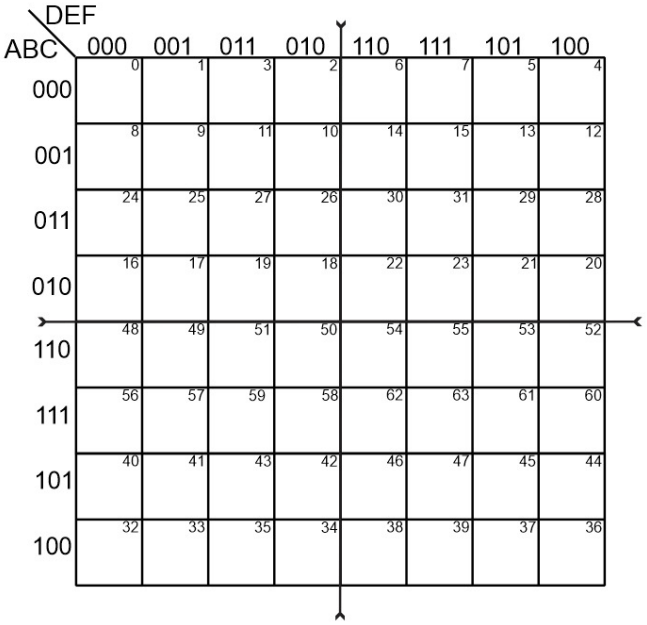


Figura 22: Estrutura do mapa de Karnaugh de 6 variáveis, sendo $A$ a variável mais significativa

Para mais de seis variáveis, a simplificação via mapa de Karnaugh torna-se inviável. Para tanto, existem outros métodos [19] cuja abordagem pode ser conferida em outras bibliografias.

19 Caso o leitor queira aprofundar-se, indicamos [31] e [44]

# 3

## O CERNE DA CALCULADORA

A história da calculadora está profundamente ligada à história dos computadores. A concepção de ambos, desde os *primórdios* dos tempos, foi impulsionado pela fixação do homem em construir um instrumento/máquina capaz de realizar as operações fundamentais de forma autônoma. Doravante, trataremos a calculadora como um pequeno microcomputador.

Historicamente, os períodos de guerra configuram-se no maior catalisador de esforços de mentes do mundo inteiro para fomentar os avanços tecnológicos. Os resultados das guerras estavam estritamente relacionados com a eficiência dos tiros, cuja tentativa de cálculos eram feitas com calculadoras mecânicas. Assim, o lado que desenvolvesse uma máquina capaz de efetuar operações de forma autônoma, teria em mãos uma grande vantagem, e é neste ambiente, que surgem os primeiros dispositivos analógicos de cálculo, que por sua vez, foram substituídos por modelos digitais, cujo aprimoramento deu origem aos "modernos" computadores que ocupavam inteiramente salas enormes e tinham massa na ordem das toneladas.

Com o aperfeiçoamento constante de seus componentes, computadores e calculadoras ficaram menores, melhores e consideravelmente mais baratos, o que, por sua vez, veio possibilitar o acesso, se não um microcomputador ao menos uma calculadora.

Abordaremos alguns componentes específicos provenientes da hábil associação das portas lógicas, tais como os codificadores, decodificadores e a memória, a fim de fornecer subsídios tanto para a compreensão do funcionamento interno destas máquinas, como também para a idealização e construção das mesmas. Por fim, temos o desfecho deste trabalho. Primeiramente, investigaremos uma intrigante calculadora mecânica binária e, na sequência, engajar-nos-emos na criação de circuitos lógicos em conformidade com as tabuadas das classes residuais. Construiremos somadores em diferentes

bases, além de subtratores, multiplicadores e divisores. Apresentaremos ainda alguns componentes inerentes aos computadores e que são de suma importância para seu desempenho.

Este capítulo conta com uma série de referências e *hiperlinks*, tanto dos arquivos com tais construções, bem como, com indicações de *softwares* gratuitos, simuladores *online* e aplicativos, que serão amplamente explorados com a finalidade de viabilizar o desenvolvimento de novos projetos.

## 3.1 PRIMÓRDIOS

Dado que computar significa contar, fazer cálculos, efetuar operações aritméticas, o computador seria uma técnica, mecanismo ou máquina que auxilia nesta tarefa, visando à praticidade, agilidade e precisão.

Partindo desta perspectiva, as mãos configuram-se no mais antigo e difundido acessório/técnica de contagem. Nossa primeira máquina de contar já era digital[1] desde o princípio. Todavia, este "computador" era deveras limitado, mesmo para civilizações antigas que, em certas circunstâncias, necessitavam de muitas pessoas para realizar contagens com valores elevados.

A necessidade do homem de facilitar e potencializar a contagem e os cálculos aritméticos deu origem ao ábaco.

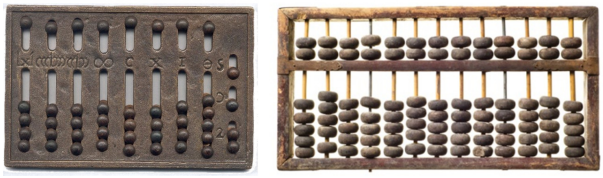

Figura 23: Ábacos

Os primeiros ábacos, figura 23, datam de *3500 a.C.* e eram constituídos de pequenas bolinhas de mármore que deslizavam em sulcos feitos na terra ou em placas de bronze. As bolinhas em latim eram denominadas de *cálculus* e efetuar cálculos aritméticos era *calculare*. A versão mais "atual" do ábaco possui uma moldura de madeira dividida em

1 A palavra dígito provém do latim *digitus* que significa dedo, que por sua vez provém do ato de utilizar os dedos na contagem e, que deu origem ao sistema de numeração na base 10.

duas partes cujas contas (do latim, *computo*) assumem valor quando são deslocadas em direção à haste interna. O ábaco foi o primeiro dispositivo manual de cálculo e perdurou por muitos séculos.

Somente no século *XVII* surge um novo dispositivo para computar: os ossos de Naiper[2], figura 24 que nada mais eram do que as tabuadas de multiplicação gravadas em bastões de marfim. Abaixo segue um exemplo de um dos ossos de Naiper bem como o procedimento para realizar a multiplicação de 579 por 7. Para tanto, tomamos e alinhamos os bastões 5, 7 e 9, em seguida, evidenciamos os resultados da linha 7.

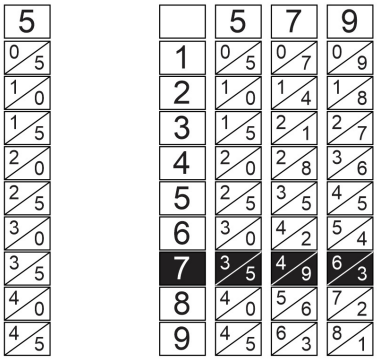


Figura 24: Multiplicação com os ossos de Naiper

Dos resultados em destaque, devemos somar os dígitos das diagonais, da direita para a esquerda:

- o algarismo da unidade é 3;
- o algarismo da dezena é $9 + 6 = 15$, e portanto 5;
- o algarismo da centena é $5 + 4 = 9$ acrescido de 1 proveniente da soma anterior, 10 e portanto 0;
- o algarismo da milhar é 3 acrescido de 1 proveniente da soma anterior, 4;
- por fim, segue que $579 \times 7 = 4053$.

A evolução dos ossos de Naiper deu origem às réguas de cálculo, largamente utilizadas nos cursos de engenharia e arquitetura até meados da década de 1970. No século *XVII* florescem as invenções de máquinas de calcular[3] (*mecânicas*).

2 John Naiper também é conhecido como o inventor dos logaritmos.
3 Para o leitor que deseje aprofundar-se, recomendamos a leitura de [10], [18] e [33]

A primeira que se tem registros trata-se de uma calculadora mecânica apelidada de *relógio calculador*, atribuída a Wilhelm Schickard. A criação de tal máquina fomentou o surgimento de uma série de outras, tais como a *pascaline*, desenvolvida por Blaise Pascal, a Rechenmaschine de Leibniz, a *arithmometer* (séc. *XVIII*); a *máquina diferencial* e o *calculador analítico* ambos projetados por Charles Babbage e amplamente esmiuçados por Ada Lovelace, cujas contribuições render-lhes-iam o título de "pai e mãe dos computadores"[4].

A primeira máquina totalmente eletrônica, entretanto, surge em 1943 no auge da II guerra mundial, projetada pelo matemático, lógico, criptoanalista e cientista da computação britânico Allan Turing (1912 - 1954) e foi batizada de *COLOSSUS*. Tal máquina utilizava os cartões perfurados e processava informações com velocidade de 25000 caracteres por segundo. A mesma foi concebida com a finalidade de decriptar mensagens alemãs criptografadas pela *ENIGMA*, que era considerada uma máquina invencível na época. Turing propôs a criação de uma máquina para vencer outra, esta história foi retratada de forma brilhante no filme: "*The imitation game*".

Allan Turing criou a primeira máquina "*pensante*" do mundo que ficou conhecida como "*Máquina de Turing*", e ele tinha apenas 24 anos. Tal máquina é composta de um fita suficientemente grande composta de lacunas preenchidas por símbolos, 0 e 1, e um cabeçote que poderia avançar ou retroceder sobre a fita, escrever ou apagar símbolos além de iniciar e parar obviamente. A máquina de Turing é o que podemos considerar como sendo o computador primitivo cuja funcionalidade conta com a ideia matriz de um computador programável, isto é, por mais rudimentar que pareça, esta máquina tem tanta capacidade quanto os computadores modernos.

O princípio geral da computação é fornecer instruções que serão seguidas sistematicamente pela máquina. O conjunto de instruções que alimentará a máquina é o que chamamos de *programa*. Turing demonstrou que toda e qualquer tarefa que pode ser representada pelos procedimentos descritos anteriormente poderá ser mecanizada, ou seja, realizada por uma máquina. A obra de Turing é a base de toda a teoria da computação moderna e, por este motivo, Allan Turing é considerado o *Pai da Computação*.

A popularização dos computadores tem início com a exploração do vale do silício, e a invenção do transistor, um substituto para as medonhas válvulas, eram cerca de 100 vezes menores, assim o tamanho dos computadores passou de gigantes colossais para gigantes apenas.

4 Para inteirar-se sobre o tema, recomendamos [18]

Posteriormente o transistor foi substituído pela tecnologia dos circuitos integrados (*CIs*), constituídos por dezenas de componentes eletrônicos montados em uma placa de silício em miniatura, os populares *CHIPs*, que possibilitaram uma evolução nos computadores, que se tornaram "*menos gigantes*", "*menos caros*", *menos lentos*, devido a redução da distância entre seus componentes.

Os avanços tecnológicos não cessam. Surgem os microprocessadores e, com eles, os microcomputadores ou computador pessoal e também os *softwares integrados* tais como editores de texto, planilhas eletrônicas, gerenciamento de banco de dados, de comunicação, gráficos e a internet. Houve uma explosão no mercado, e os microcomputadores foram fabricados em larga escala e vendidos a preços acessíveis se comparados com os preços de seus antecessores, e é aqui que dissociamos a calculadora do computador.

Nos anos 70, surgem as calculadoras parecidas com as que conhecemos atualmente, porém muito maiores e com custo elevado. Contudo, com a evolução constante dos circuitos integrados, a calculadora passa literalmente a caber no bolso, já que diminui consideravelmente de tamanho e de preço, além de expandir suas funções.

Com a criação do visor de cristal líquido (*LCD*) surge a calculadora de bolso tal como a que conhecemos hoje, que oferece confiabilidade e rapidez nos cálculos diários ou complexos com um simples apertar de tecla. Na figura 25[5].

Figura 25: Calculadoras

## 3.2 CALCULADORA BINÁRIA MECÂNICA

Nossa primeira calculadora binária é mecânica e feita de madeira. Caso o leitor queira confeccionar uma calculadora dessas, o projeto com as devidas especificações técnicas se encontra no *apêndice B*, caso a curiosidade para testar o que

5 A evolução da calculadora ao longo da história pode ser apreciada no *apêndice B*.

for dito seja tamanha que não consiga esperar pela construção da mesma, existe um simulador gratuito on-line desta calculadora, segue o seu endereço eletrônico: https://leloctai.tk/game/mechanicalAdder/

Lembra-se da tabuada de $\mathbb{Z}_2$? Nossa calculadora comporta-se de acordo com a mesma, sendo capaz de somar números na base binária. Vamos nos familiarizar com seus componentes explorando a figura 26.

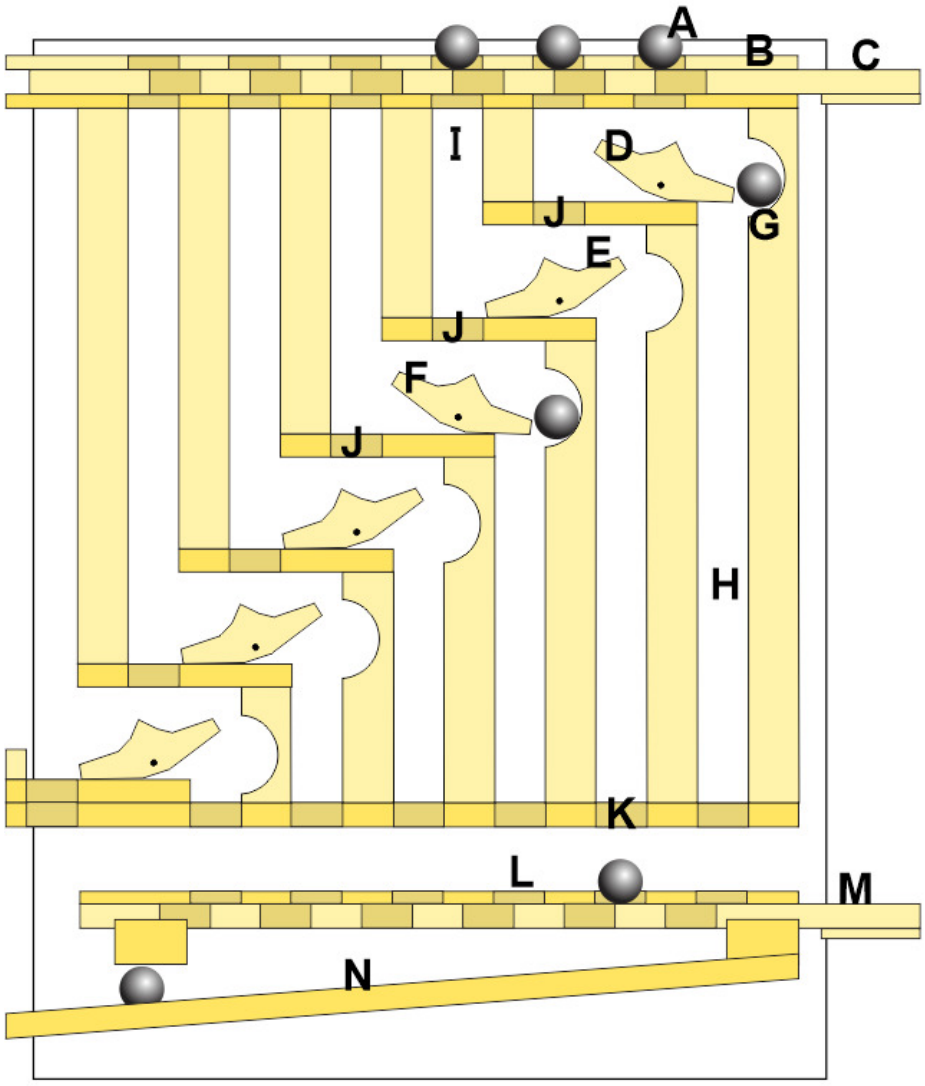


Figura 26: Máquina de adição de bolinha de gude

Primeiramente, temos em *(A)* os dados de entrada, os bits, que neste caso são representados por bolinhas de gude, em inglês *marble*. É interessante notar que *marble* designa tanto bolinha de gude quanto mármore, matéria prima das bolinhas utilizadas nos primeiros ábacos. Por isso designaremos os dados de entrada por *marbits*: uma contração de *marble* e *digits*.

Em *(B)* temos as entradas superiores com orifícios por onde entram os *marbits* que são barrados pela placa deslizante *(C)*, esta faz o papel de "chave" e garante que todos os *marbits* caiam ao mesmo tempo e assim evita erros de computação. *(D)*, *(E)* e *(F)* são pequenas gangorras que ora obrigam o *marbit* a pender para direita, ora para a esquerda. Portanto tais gangorras que denotaremos por alternadores são as portas lógicas.

Quando o *marbit* pende para a direita ele fica cativo em *(G)*, uma cavidade que funcionará tal como a memória da calculadora, designada por *flip-flop*[6]. *(I)* e *(H)* são dutos, sendo o primeiro de entrada e o segundo de saída.

Um *marbit* pode chegar ao alternador *(F)* tanto pelo duto *(I)* como vindo de transportes do alternador *(E)*, mas não há erros de computação quanto a este fato, já que os *marbits* oriundos de transporte são mais lentos que os lançados diretamente no alternador. Quando um *marbit* é liberado de *(G)*, ele cai em *(H)* e passa pelo orifício *(K)* que direciona para o orifício *(L)* que deve estar aberto, isto é, alinhado com a chave *(M)* para que o *marbit* seja descartado e siga por *(N)* até um recipiente.

A chave *(M)* é o botão de "*igual*" da calculadora, ao acioná-la, ao mesmo tempo em que reinicia a calculadora, move todos os alternadores para a esquerda e assim libera todos os *marbits* cativos. Tal chave fica desalinhada com (L) que será o nosso visor de resultados, já que os *marbits* ficaram temporariamente retidos e são liberados ao voltarmos com a chave *(M)* e, a calculadora está pronta para novos cálculos.

Cada uma das entradas está associada a uma potência de 2, assim, o primeiro orifício da direita é o menos significativo e vale $2^0 = 1$, seguido de 2, 4, 8, 16, 32. Vamos analisar agora o comportamento das portas lógicas desta curiosa calculadora.

A figura 27 ilustra o funcionamento dos alternadores. Curiosamente a porta lógica combina dois itens, o primeiro muito claro, **0**. Se o *marbit* está preso em *(B)* e **1** caso ele seja liberado, o outro item é justamente a posição do alternador, que é **0** caso ele pender para esquerda (*posição inicial*) e **1** se pender à direita.

6 Adiante, esmiuçaremos os *flip-flops*, bem como outros circuitos interessantes.

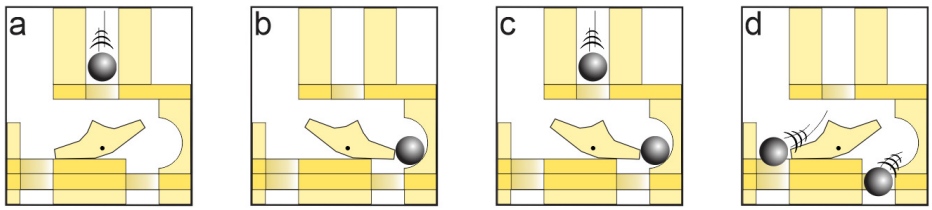


Figura 27: Etapas de funcionamento

Em (a), a princípio temos *0* do *marbit* que está em repouso em (B) e *0* do alternador pendendo para esquerda *(0 + 0 = 0)*. Quando o *marbit* é liberado, ele assume valor **1** e é desviado para a direita onde fica cativo no *flip-flop*, como se pode observar em (b), temos neste caso *(1 + 0 = 1)*. Enquanto o próximo *marbit* ficar retido em (B) seu valor será **0**, já o alternador pendendo à direita assume valor **1**, já que o *marbit* continua cativo *(0 + 1 = 1)*.

Agora atente para (c) e (d), o alternador possui valor **1**, ao liberarmos o *marbit* ele também assume valor **1**. O alternador então o desvia para a esquerda ao mesmo tempo em que libera e descarta o *marbit* que estava cativo, concomitantemente o *marbit* desviado é direcionado para o sistema de alternador seguinte, que o condicionará no próximo *flip-flop*. Em outras palavras, o "*sistema*" atua efetivamente com o *carry* realizando o descarte e adicionando um *marbit* na próxima posição mais significativa, isto é: *(1 + 1 = 0, e "vai um")*. Diante do exposto, os alternadores se comportam tal como a porta *XOR*.

Esta calculadora apresenta resultados até $2^6 - 1$, isto é, 63. O que aconteceria caso efetuássemos a adição $63 + 1$? Acreditamos que este seja o movimento dos mais singulares desta calculadora, ao depositar um *marbit* em cada uma das entradas todos ficarão cativos, $(63)_2$ e, em seguida adicionamos *1 marbit* tal como na figura 28, uma vez liberado desarmará todos os alternadores criando o "*efeito cascata*", todos os *marbits* serão descartados, incluindo último *marbit*, e o visor sem dado algum a informar. Tal desfeche é o que denominamos por ***overflow***.

Para evitar o *overflow*, basta adicionar números de utilizem no máximo 5 *marbits*, afinal uma adição de dois números de $n$ dígitos apresentam soma de no máximo $n + 1$ dígitos. Nossa calculadora é capaz de realizar adições. Contudo, vimos anteriormente como efetuar subtrações, multiplicações e divisões por meio das operações de *inversão* e de *deslocamento*, assim com um pouco de habilidade, esta calculadora é capaz de realizar as quatro operações fundamentais.

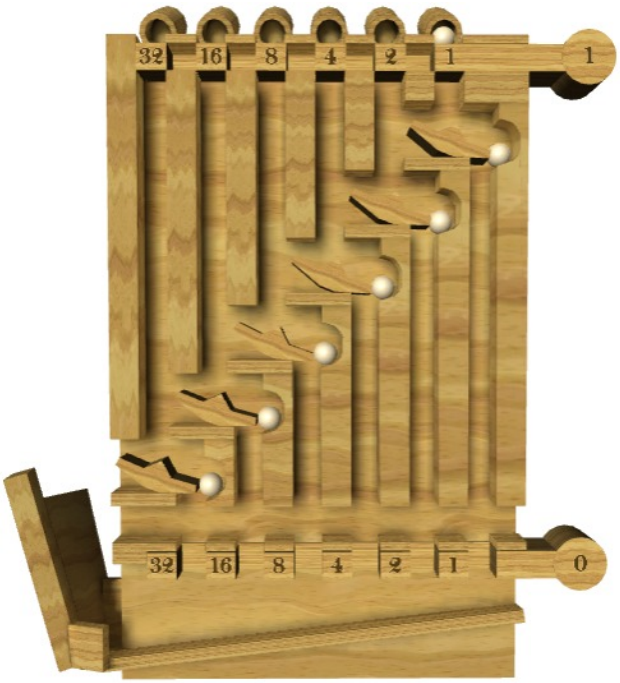


Figura 28: Simulador on-line calculadora binária - *overflow*

## 3.3 A ENGENHOSIDADE DOS CIRCUITOS

Passemos, de fato, para a construção de circuitos lógicos implementáveis, oriundos de engenhosas associações das portas lógicas e a explorar os componentes específicos, essenciais para a concepção de calculadoras e computadores.

Os computadores convencionais baseiam-se no modelo idealizado por Jhonn Von Neumann (1903-1957), matemático húngaro e um dos construtores do gigantesco ENIAC (Eletronoic Numéric Integrador And Calculador) desenvolvido durante a II guerra mundial e concluído em 1946. Atualmente, o modelo de Von Neumann foi aperfeiçoado para outro tipo de barramento de sistemas, constituído de:

- **ENTRADAS/SAÍDAS**, composto pelas unidades de entradas e saídas além de codificadores e decodificadores;
- **CPU** *(Central Processing Unit)* é o processador, composto por unidade de controle, unidade lógica e aritmética (**ULA**), registradores e contadores;
- **MEMÓRIA**, armazena instruções e dados;
- **BARRAMENTO DE DADOS**, transporta informações, movendo dados entre os componentes do sistema.

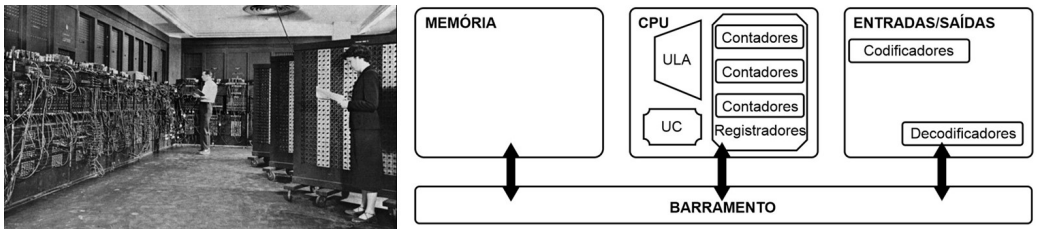


Figura 29: ENIAC a esquerda, modelo aperfeiçoado de Von Neumann a direita

Doravante, analisaremos cada um dos componentes supracitados.

### 3.3.1 *Codificadores e decodificadores*

Um circuito codificador/decodificador é capaz converter determinados dados em outros. Tal circuito segue o princípio das funções, estabelecendo uma correspondência entre o domínio e o contradomínio por meio de uma regra, além de muito utilizada na criptografia, os codificadores/decodificadores são empregados tanto na informática como na matemática para a conversão de números de uma determinada base para outra. Um dos mais utilizados é o circuito que codifica os números decimais em binários, conhecidos como ***BCD*** (*Binary-Codec-Decimal*) dado pelo quadro a seguir:

| *d* | **A** | **B** | **C** | **D** | **Y** |
|---|---|---|---|---|---|
| *0* | 0 | 0 | 0 | 0 | $\overline{A}.\overline{B}.\overline{C}.\overline{D}$ |
| *1* | 0 | 0 | 0 | 1 | $\overline{A}.\overline{B}.\overline{C}.D$ |
| *2* | 0 | 0 | 1 | 0 | $\overline{A}.\overline{B}.C.\overline{D}$ |
| *3* | 0 | 0 | 1 | 1 | $\overline{A}.\overline{B}.C.D$ |
| *4* | 0 | 1 | 0 | 0 | $\overline{A}.B.\overline{C}.\overline{D}$ |
| *5* | 0 | 1 | 0 | 1 | $\overline{A}.B.\overline{C}.D$ |
| *6* | 0 | 1 | 1 | 0 | $\overline{A}.B.C.\overline{D}$ |
| *7* | 0 | 1 | 1 | 1 | $\overline{A}.B.C.D$ |
| *8* | 1 | 0 | 0 | 0 | $A.\overline{B}.\overline{C}.\overline{D}$ |
| *9* | 1 | 0 | 0 | 1 | $A.\overline{B}.\overline{C}.D$ |

O ***BCD*** estabelece uma correspondência biunívoca entre os dígitos[7] (*utilizados no sistema decimal*) e a sua representação na base binária. Por meio da tabela verdade acima,

7 Para bases que utilizam mais de nove símbolos, como a base hexadecimal, adota-se as letras: A, B, C, D, E e F que representam os valores 10, 11, 12, 13, 14 e 15 respectivamente.

gera-se o circuito lógico e, por meio deste podemos converter os números na base decimal para a base binária e operar os mesmos via álgebra booleana e, como as *saídas* são fornecidas em números binários, faz-se necessário a utilização de um decodificador que, por sua vez, comporta-se como a função inversa do codificador.

Para construir um um circuito lógico decodificador que estabeleça correspondência entre os números binários (*de 4 bits utilizados há pouco*") e os dígitos do sistema decimal (decimal - binário) basta realizar o processo dual, alocamos interruptores nas saídas do circuito representado na figura 30 além de substituir as portas lógicas OR por portas AND em cada um dos números decimais.

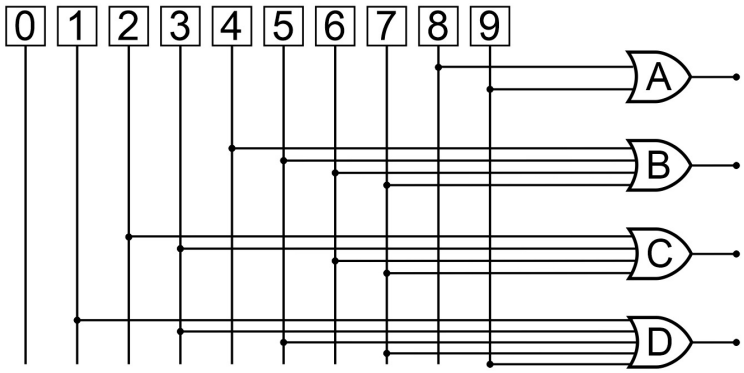


Figura 30: Codificador (BCD)

O *software Logic ly* é um simulador de circuitos lógicos e será utilizado para as construções e posterior validação. A versão *tryal* do simulador está disponível em: https://logic.ly/download/, existe também a versão *online* deste *software*, segue seu endereço eletrônico: https://logic.ly/demo/. Todas as construções estão disponíveis para *download* e todos os *links* se encontram no *apêndice B*.

### 3.3.2 *Flip-Flops*

Existem alguns circuitos lógicos peculiares e de grande importância que passamos a abordar. As combinações lógicas (circuitos) que vimos até então apenas apresentam a saída enquanto são mantidas as entradas. Em outras palavras, tais circuitos não possuem memória e desta forma a saída depende única e exclusivamente das entradas e das combinações lógicas.

A memória é capaz de armazenar dados e instruções e, configura-se em um dos itens essenciais para o funcionamento dos computadores. Todavia há uma maneira de construir circuitos lógicos capazes manter a saída, por tempo indeterminado, ainda que variem as entradas, analisemos o circuito lógico do "*Flip-Flop*"[8] representado na figura 31:

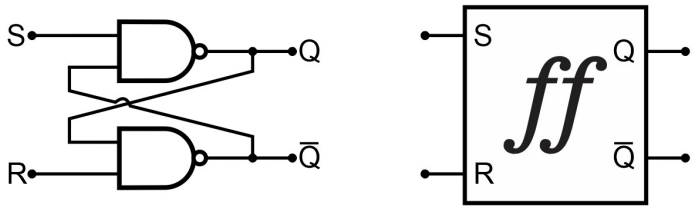


Figura 31: Flip-Flop

Temos que para $S = 1$ e $R = 0$ o $\overline{Q} = 1$ já que a porta $\boldsymbol{NAND}_{inferior}$ produz a saída 1 que alimenta a porta $\boldsymbol{NAND}_{superior}$ que produz $Q = 0$ que por sua vez realimenta a porta $\boldsymbol{NAND}_{inferior}$ que torna a gerar a saída 1 que torna a realimentar $\boldsymbol{NAND}_{superior}$ e assim por diante. Em síntese, o dado ou *bit* fica confinado em um ciclo.

O segundo caso, em que $S = 0$ e $R = 1$ é o dual do primeiro.

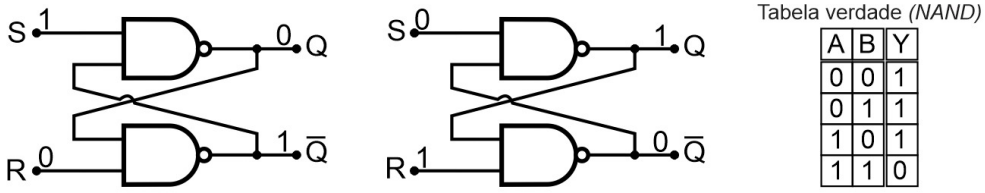


Figura 32: Flip-Flops, entradas e saídas

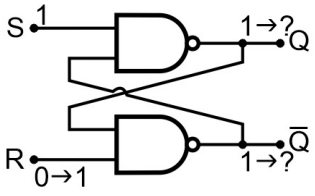


Figura 33: Flip-Flops, entradas iguais

8 Para saber mais recomendamos [31] e [44]

Sabem quais são as mudanças que ocorreriam nas saídas caso mudássemos uma das entradas?

NENHUMA! Isso mesmo, não há mudanças nas saídas caso alteremos a entrada $R$. Note que a porta $\boldsymbol{NAND}_{inferior}$ permanecerá *(1, 0)* ou melhor, alterna de *(0, 1)* para *(1, 0)* e consequentemente $\overline{Q} = 1$. Isso significa que entradas diferentes podem produzir a mesma saída, e a mesma entrada poderá produzir saídas distintas. Lembre-se de que os *Flip-Flops* se lembram das entradas anteriores, e que estas são consideradas nas combinações das novas saídas. Por isso os *Flip-Flops* também são chamados de "*Latchs*" (*trincos*), pois são capazes de prender os dados ou bits, e é desta forma que se engendra a memória dos computadores.

Para ativar os *Flip-Flops*, mantemos as entradas $S = R = 1$ cujas saídas são um verdadeiro mistério, e damos um *pulso* no fio $S$, ou seja, $S$ alterna de 1 para 0 e retorna para 1. Quando $S$ muda para 0, temos como saídas $Q = 1$ e $\overline{Q} = 0$ (*que ativa o Flip-Flop*) $S$ então, retorna a valer 1, porém, as saídas não se alteram.

Para desativar fazemos o processo dual, ou seja, damos um pulso em $R$ que nos fornecerá $Q = 0$ e $\overline{Q} = 1$ (*que desativa o Flip-Flop*). Por fim, temos um problema, as entradas $S = R = 0$ produziram saídas $Q = \overline{Q} = 1$, ao alternamos as entradas para $S = R = 1$ (*para ativar ou desativar o Flip-Flop*) o resultado é imprevisível, isto é, a saída dependerá de qual mudança ocorrerá primeiro, e não sabemos se acontecerá em $Q$ ou em $\overline{Q}$. Devido a tal aleatoriedade, a entrada $S = R = 0$ é estritamente proibida.

| **S** | **R** | **Q** | $\boldsymbol{\overline{Q}}$ |
|---|---|---|---|
| 0 | 0 | proibido! | |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | não altera! | |

Logicamente, os circuitos são construídos de maneira a impedir que o caso proibido ocorra. Os *Flip-Flops* também podem ser construídos com portas ***NOR***. Neste caso o funcionamento e o acionamento se dá de forma dual. As entradas proibidas passam a ser $S = R = 1$. Segue a tabela verdade deste *Flip-Flop*.

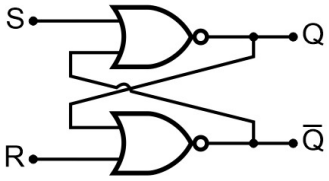


Tabela verdade *(NOR)*

| A | B | Y |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

Figura 34: Flip-Flop com portas NOR

| **S** | **R** | **Q** | $\overline{\mathbf{Q}}$ |
|---|---|---|---|
| 0 | 0 | não altera! | |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | proibido! | |

O arquivo dos *Flip-Flops* estão disponíveis para download em https://drive.google.com/open?id=1gcr9K1w_sa74CN0R6lXRS6XSKW5K-k07 e podem ser abertos no simulador **Logic ly**.

#### 3.3.2.1 *Registradores e Contadores*

Precisamente, o *registrador* é uma associação de *flip-flops* para o armazenamento de vários *bits*, todavia, devido à entrada proibida, sua simples junção poderia gerar resultados imprecisos, para tanto acrescentamos um "*circuito de porta*":

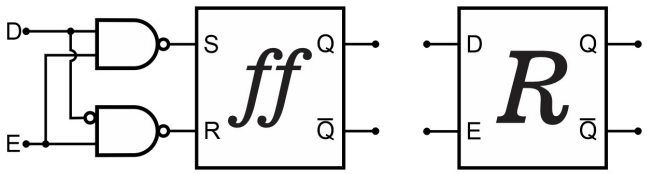


Figura 35: Registrador

Na figura 35 a entrada de dados $D$, e $E$ *(enable)* refere-se à "*habilitação/seleção*". Quando $E = 1$; $R = Q = D$, que habilita o *flip-flop*, que assume a função de um fio condutor, já para $E = 0$; $R = S = 1$, o *flip-flop* é desativado, e impede a entrada de novos dados, ou seja, o valor de $D$ fica registrado e não se altera, ainda que mudemos o valor da entrada. Note que o *circuito de porta* impede que ocorra o caso proibido

$R = S = 0$, por isso o incorporamos aos registradores, que podem ser associados em série tal como descrito na figura 36.

Tais arquivos foram construídos no **Logic-Ly** e estão disponíveis para download em https://drive.google.com/open?id=1fuYYZISrRoRM4nx8s7u1T1sMEB1t3ivK

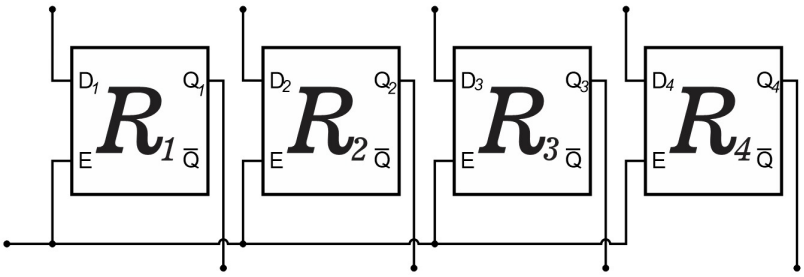


Figura 36: Registradores em série

Para agregar todos os *bits* registrados em cada *flip-flop*, utilizamos um tipo especial de registrador: o *contador*, figura 37, que soma 1 aos dados retidos quando acionado, e evidentemente é capaz de contar! Para construir tal circuito, faremos uso do *flip-flop mestre-escravo*:

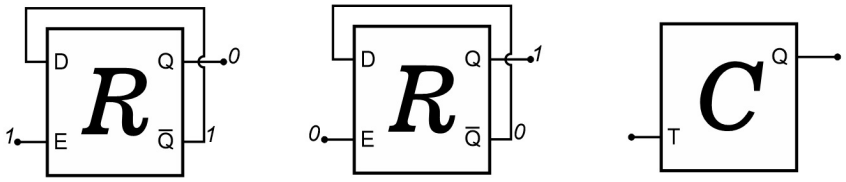


Figura 37: Contador

Observe que $\overline{Q}$ realimenta $D$, assim, para $E = 1, \overline{Q} = D$, caso $E = 0, Q = D$ e a saída se inverte. Tal inversão, ou giro *(toggles)* de $Q$ ocorre sempre em que o valor de $T$ muda de 0 para 1. Os contadores podem ser associados em série tal como descrito na figura 38, cuja contagem limita-se a $2^n$ números para $n$ contadores:[9]

Quatro contadores associados podem contar até $(1111)_2$, conforme a tabela abaixo. Este arquivo foi construído no **Logic-Ly** e está disponível para download em: https://drive.google.com/open?id=1mEnqfrXAYZ5NeP5FNWy4_yaN22OJ3ds-

9 Para saber mais indicamos [31] e [44]

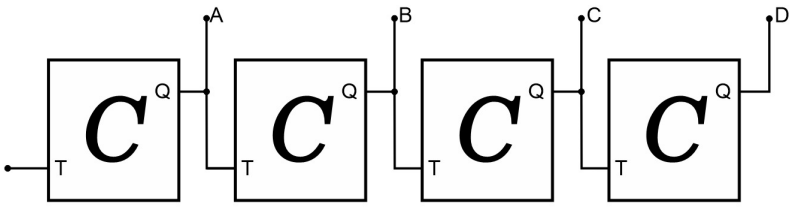


Figura 38: Contadores em série

| Entradas de contagem | D | C | B | A |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 |
| ⋮ | ⋮ | | | |
| 15 | 1 | 1 | 1 | 1 |

## 3.4 UNIDADE LÓGICA E ARITMÉTICA

A unidade lógica e aritmética *(ULA)* é qualquer circuito integrado *(CI)* que efetue as operações aritméticas ou realize outras funções: adicionar, subtrair, multiplicar, dividir, comparar, deslocar, desviar, testar, negar, empilhar, chamar e interromper sub-rotinas, enfim, qualquer ação que possa ser executada por combinações de circuitos lógicos. A *(ULA)* apenas não se configura como o cerne de todo o processamento computacional por não ser capaz de armazenar dados.

Como visto anteriormente, as operações aritméticas podem ser efetuadas por meio de adições e combinações de funções lógicas, quando reunidas em um único *(CI)* designado por *(ULA)* e constituem no bloco primordial presente nos microprocessadores.

### 3.4.1 *Somadores*

A *SOMA* é a função mínima que se espera de um computador/calculadora, pois como visto, todas as operações aritméticas podem ser efetuadas por meio de adições. Realizar-se-á um estudo detalhado destes circuitos que são cruciais para a confecção de calculadoras em uma base qualquer.

#### 3.4.1.1 *Somador binário*

Sem mais delongas, vamos diretamente para o meio somador binário, isto é, o circuito lógico que traduz a soma binária para apenas duas entradas $A$ e $B$. $S$ será a soma e $C_{out}$ o "*carry*" (*vulgo* "*vai-um*"). Eis a tabela verdade para a soma:

| **A** | **B** | **S** | **C**$_{out}$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

Temos que a soma $S$ vale 1, quando as entradas são distintas e, vale 0 caso contrário, S fica determinada por meio de uma porta lógica *XOR*, já o $C_{out}$ assume valor lógico 1 somente quando ambas as entradas forem iguais a 1, ou seja, ele é obtido por meio de uma porta lógica *AND*.

Assim, o circuito lógico do meio somador é o que segue:

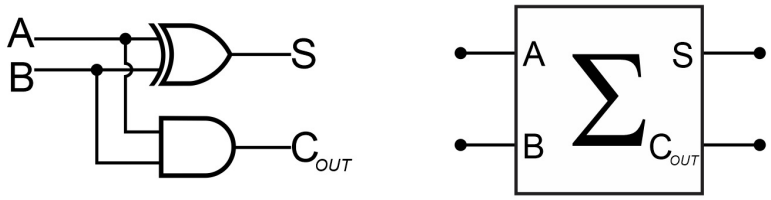


Figura 39: Meio somador

Este circuito recebe o nome de *meio somador* ou *somador simples* devido às suas limitações, já que ele é capaz de somar apenas dois números binários de um *bit* cada, em outras palavras, esta é a unidade mais simples na calculadora.

Apresentamos o *somador completo*, que admite três entradas: $A$, $B$ e $C_{in}$ e fornecem duas saídas, a soma $S$ e um novo *carry* de saída, o $C_{out}$, observe a seguinte tabela verdade:

| *d* | **A** | **B** | $\mathbf{C}_{in}$ | **S** | $\mathbf{C}_{out}$ |
|---|---|---|---|---|---|
| *0* | 0 | 0 | 0 | 0 | 0 |
| *1* | 0 | 0 | 1 | 1 | 0 |
| *2* | 0 | 1 | 0 | 1 | 0 |
| *3* | 0 | 1 | 1 | 0 | 1 |
| *4* | 1 | 0 | 0 | 1 | 0 |
| *5* | 1 | 0 | 1 | 0 | 1 |
| *6* | 1 | 1 | 0 | 0 | 1 |
| *7* | 1 | 1 | 1 | 1 | 1 |

Utilizando os *minterm's (soma-de-produtos)* temos:

$$f(A,B,C_{in}) = S = \sum m(1,2,4,7)$$

$$\overline{A}\cdot\overline{B}\cdot C_{in} + \overline{A}\cdot B\cdot\overline{C_{in}} + A\cdot\overline{B}\cdot\overline{C_{in}} + A\cdot B\cdot C_{in} \overset{T5}{=}$$

$$\overline{A}\cdot\overline{B}\cdot C_{in} + A\cdot B\cdot C_{in} + A\cdot\overline{B}\cdot\overline{C_{in}} + \overline{A}\cdot B\cdot\overline{C_{in}} \overset{T7}{=}$$

$$(\overline{A}\cdot\overline{B} + A\cdot B)\cdot C_{in} + (A\cdot\overline{B} + \overline{A}\cdot B)\cdot\overline{C_{in}} \overset{def}{=}$$

$$(A \odot B)\cdot C_{in} + (A \oplus B)\cdot\overline{C_{in}} \overset{def}{=}$$

$$(\overline{A \oplus B})\cdot C_{in} + (A \oplus B)\cdot\overline{C_{in}} \overset{def}{=}$$

$$A \oplus B \oplus C_{in}$$

Já para o *carry*, temos a seguinte expressão via *(soma-de-produtos)*:

$$f(A,B,C_{in}) = C_{out} = \sum m(3,5,6,7)$$

$$\overline{A}\cdot B\cdot C_{in} + A\cdot\overline{B}\cdot C_{in} + A\cdot B\cdot\overline{C_{in}} + A\cdot B\cdot C_{in} \overset{T7}{=}$$

$$(\overline{A}\cdot B + A\cdot\overline{B})\cdot C_{in} + A\cdot B\cdot(\overline{C_{in}}\cdot C_{in}) \overset{T4}{=}$$

$$(A \oplus B)\cdot C_{in} + A\cdot B$$

Seu circuito lógico é apresentado na figura 40. Tomando a segunda representação, alguns diriam que diferem apenas por um pequeno fio a mais com um "$C_{in}$" na ponta. O que não deixa de ser verdade. Poderíamos ainda, obter o somador completo por meio da combinação de dois meios somadores, como observado na figura 41.

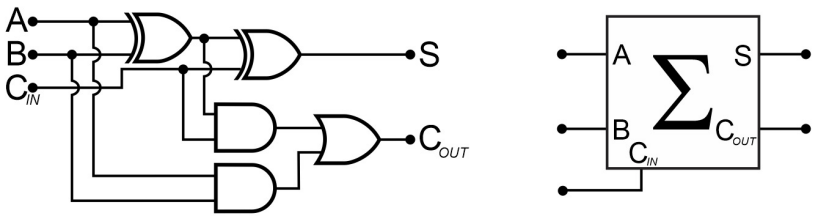


Figura 40: Somador completo

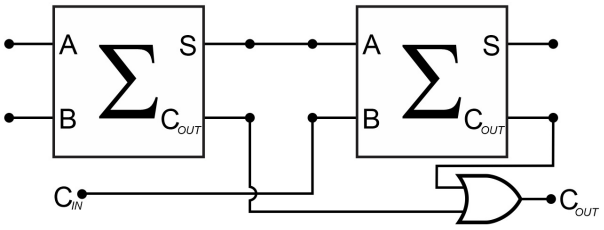


Figura 41: Somador completo alternativo

A vantagem do somador completo, além de considerar o "*carry*" na entrada, é a possibilidade de combiná-los em série a fim de criar um somador de quantos *bits* se queira, um *bit* para cada um dos somadores evidentemente. Assim podemos criar um somador de 5 *bits* capaz de realizar adição com as características abaixo:

$$\begin{array}{cccccc} & A_5 & A_4 & A_3 & A_2 & A_1 \\ + & B_5 & B_4 & B_3 & B_2 & B_1 \\ \hline C_5 & S_5 & S_4 & S_3 & S_2 & S_1 \end{array}$$

Nosso ponto de partida para criar calculadoras em outras bases será o de determinar o meio somador por meio da tabela verdade e na sequência o somador completo e, associá-los em série. Segue na figura 42 o circuito somador (*5 bits*):

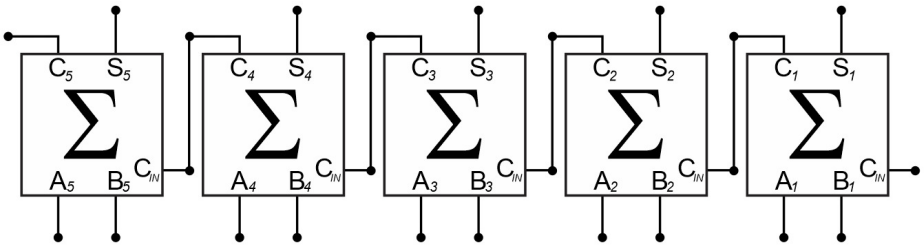


Figura 42: Somadores em série

Este arquivos foi construído no **Logic-Ly** e está disponível para download em: https://drive.google.com/open?id=1SIBgsnPwkKaEwAu04ZoFP8Bc28RekwfM

#### 3.4.1.2 *Somador ternário, por que não?*

Será que é vantajoso criar uma calculadora ternária? Para responder a tal questionamento, primeiro precisamos criá-la!

Vamos construir uma calculadora ternária de *leds*, ou melhor, com um visor de *leds*. Cada *led* terá uma função específica, isto é, o primeiro *led* corresponderá ao **0**, o segundo, **1**, o terceiro, **2** e o último indicará a presença ou não do "*carry*". Por exemplo, na calculadora de *5 bits* que vimos anteriormente, bastava associar a cada entrada de $A_n$ e $B_n$ um interruptor comum, e um *led* ao $C_5$ e a cada soma $S_n$, e teríamos a nossa calculadora , Convidamos o leitor a testar tal calculadora no simulador online https://logic.ly/demo/, ou se preferir salvar e imprimir seus projetos ou abrir outros arquivos, baixe a versão *tryal* em: https://logic.ly/download/.

Também vamos utilizar um interruptor de três pinos, que possui três estados (I, 0, II): desligado **0**, ligado **1** e ligado **2**. Por se tratar de um interruptor, os estados são mutuamente excludentes de forma a garantir que o mesmo não possa estar ligado em **1** e **2** ao mesmo tempo. Para construímos tal calculadora, precisaremos considerar algumas adaptações.

A primeira é a que o **0**, embora represente a ausência de sinal, o mesmo deverá existir, do contrário, o led não acenderia, mas resolvemos isto com inversores. Outra adaptação é a de que somente o **1** representava a presença de sinal, contudo, o **2** também representará, e devemos distingui-los. Para tanto, na álgebra booleana, utilizávamos para determinada entrada, uma letra maiúscula para **1** e a mesma letra com uma barra superior para **0**, acrescentaremos a mesma letra com uma barra inferior para **2**. Queremos aqui deixar claro que tais adaptações não pertencem à álgebra booleana, embora, nos pautaremos na mesma.

Comecemos pela tabela verdade (adaptada) contendo todos os possíveis resultados onde $A$ e $B$ são as entradas, $S$ é a soma e $C$ o "*carry*":

| *d* | **A** | **B** | **S** | **C** |
|---|---|---|---|---|
| *0* | 0 | 0 | 0 | 0 |
| *1* | 0 | 1 | 1 | 0 |
| *2* | 0 | 2 | 2 | 0 |
| *3* | 1 | 0 | 1 | 0 |
| *4* | 1 | 1 | 2 | 0 |
| *5* | 1 | 2 | 0 | 1 |
| *6* | 2 | 0 | 2 | 0 |
| *7* | 2 | 1 | 0 | 1 |
| *8* | 2 | 2 | 1 | 1 |

Lembre-se de que estamos lidando com uma adaptação da álgebra booleana, o número **2** presente na tabela é apenas uma adequação para distinguir os sinais de **1** e **2**. Nova adaptação, ao longo da nossa trajetória, utilizamos a soma-de-produtos das linhas nas quais a função vale **1**. Pois bem, vamos continuar utilizando o mesmo procedimento, contudo, repetiremos o mesmo para as linhas nas quais a função assume valor **0** e **2**.

Assim a função assume valor **0** nas linhas *(0,5,7)*, valor **1** nas linhas *(1,3,8)*, valor **2** nas linhas *(2,4,6)*, e há a presença de *carry* nas linhas *(0,7,8)*.

Determinemos pela *(soma-de-produtos)* seus **minterm´s**:

- (0) $\overline{A} \cdot B + A \cdot \underline{B} + A \cdot B$
- (1) $\overline{A} \cdot B + A \cdot \overline{B} + A \cdot B$
- (2) $\overline{A} \cdot \underline{B} + \underline{A} \cdot \overline{B} + A \cdot B$
- (*CARRY*) $A \cdot \underline{B} + \underline{A} \cdot B + A \cdot B$

A obtenção do circuito lógico se dará de forma convencional, afinal a "*barra inferior*" designa apenas se a entrada provém ou não de (II).

Poderíamos ter realizado algumas simplificações inserindo algumas portas lógicas *XOR*, por exemplo. Todavia, precisaríamos dispor de quatro entradas e, para tanto, seria necessário associá-las às duas portas *OR*, isto é, na prática não haveria simplificação já que a quantidade de portas lógicas utilizadas permaneceria a mesma. Assim temos:

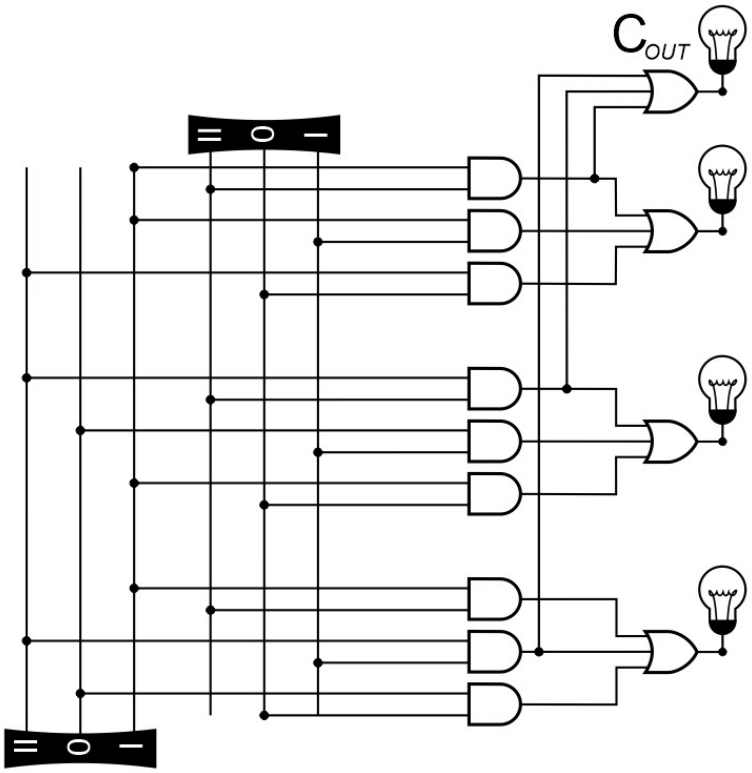


Figura 43: Meio somador ternário

O circuito do meio-somador apresentado na figura 43 foi construído no **Logic-Ly** e seu arquivo está disponível para download em: https://drive.google.com/open?id=1nWJ1DZAcKHMFAY7yw_gyka2eatpCVQZS

Prosseguindo de forma análoga em busca do somador completo binário, vamos construir a tabela verdade para três entradas: $A$, $B$ e $C_{in}$ e duas saídas, a soma S e $C_{out}$:

| *d* | **A** | **B** | $\mathbf{C}_{in}$ | $S$ | $C_{out}$ |
|---|---|---|---|---|---|
| *0* | 0 | 0 | 0 | 0 | 0 |
| *1* | 0 | 0 | 1 | 1 | 0 |
| *2* | 0 | 1 | 0 | 1 | 0 |
| *3* | 0 | 1 | 1 | 2 | 0 |
| *4* | 0 | 2 | 0 | 2 | 0 |
| *5* | 0 | 2 | 1 | 0 | 1 |
| *6* | 1 | 0 | 0 | 1 | 0 |
| *7* | 1 | 0 | 1 | 2 | 0 |
| *8* | 1 | 1 | 0 | 2 | 0 |
| *9* | 1 | 1 | 1 | 0 | 1 |
| *10* | 1 | 2 | 0 | 0 | 1 |
| *11* | 1 | 2 | 1 | 1 | 1 |
| *12* | 2 | 0 | 0 | 2 | 0 |
| *13* | 2 | 0 | 1 | 0 | 1 |
| *14* | 2 | 1 | 0 | 0 | 1 |
| *15* | 2 | 1 | 1 | 1 | 1 |
| *16* | 2 | 2 | 0 | 1 | 1 |
| *17* | 2 | 2 | 1 | 2 | 1 |

A função assume valor **0** nas linhas *(0,5,9,10,13,14)*, valor **1** nas linhas *(1,2,6,11,15, 16)*, valor **2** nas linhas *(3,4,7,8,12,17)*, e há a presença de *carry* nas linhas *(5,9,10,11,13, 14,15,16,17)*. Logo, por *(soma-de-produtos)*, temos:

- (0) $\overline{A}\cdot\overline{B}\cdot\overline{C_{in}} + \overline{A}\cdot\underline{B}\cdot C_{in} + A\cdot B\cdot C_{in} + A\cdot\underline{B}\cdot\overline{C_{in}} + \underline{A}\cdot\overline{B}\cdot C_{in} + \underline{A}\cdot B\cdot\overline{C_{in}}$
- (1) $\overline{A}\cdot\overline{B}\cdot C_{in} + \overline{A}\cdot B\cdot\overline{C_{in}} + A\cdot\overline{B}\cdot\overline{C_{in}} + A\cdot\underline{B}\cdot C_{in} + \underline{A}\cdot B\cdot C_{in} + \underline{A}\cdot\underline{B}\cdot\overline{C_{in}}$
- (2) $\overline{A}\cdot B\cdot C_{in} + \overline{A}\cdot\underline{B}\cdot\overline{C_{in}} + A\cdot\overline{B}\cdot C_{in} + A\cdot B\cdot\overline{C_{in}} + \underline{A}\cdot\overline{B}\cdot\overline{C_{in}} + \underline{A}\cdot\underline{B}\cdot\overline{C_{in}}$
- ($CARRY_{OUT}$) $\overline{A}\cdot\underline{B}\cdot C_{in} + A\cdot B\cdot C_{in} + A\cdot\underline{B}\cdot\overline{C_{in}} + \underline{A}\cdot\overline{B}\cdot C_{in} + \underline{A}\cdot B\cdot\overline{C_{in}}$ $+ A\cdot\underline{B}\cdot C_{in} + \underline{A}\cdot B\cdot C_{in} + \underline{A}\cdot\underline{B}\cdot\overline{C_{in}} + \underline{A}\cdot\underline{B}\cdot C_{in}$

Apresentamos na figura 44 o circuito lógico do somador completo ternário:

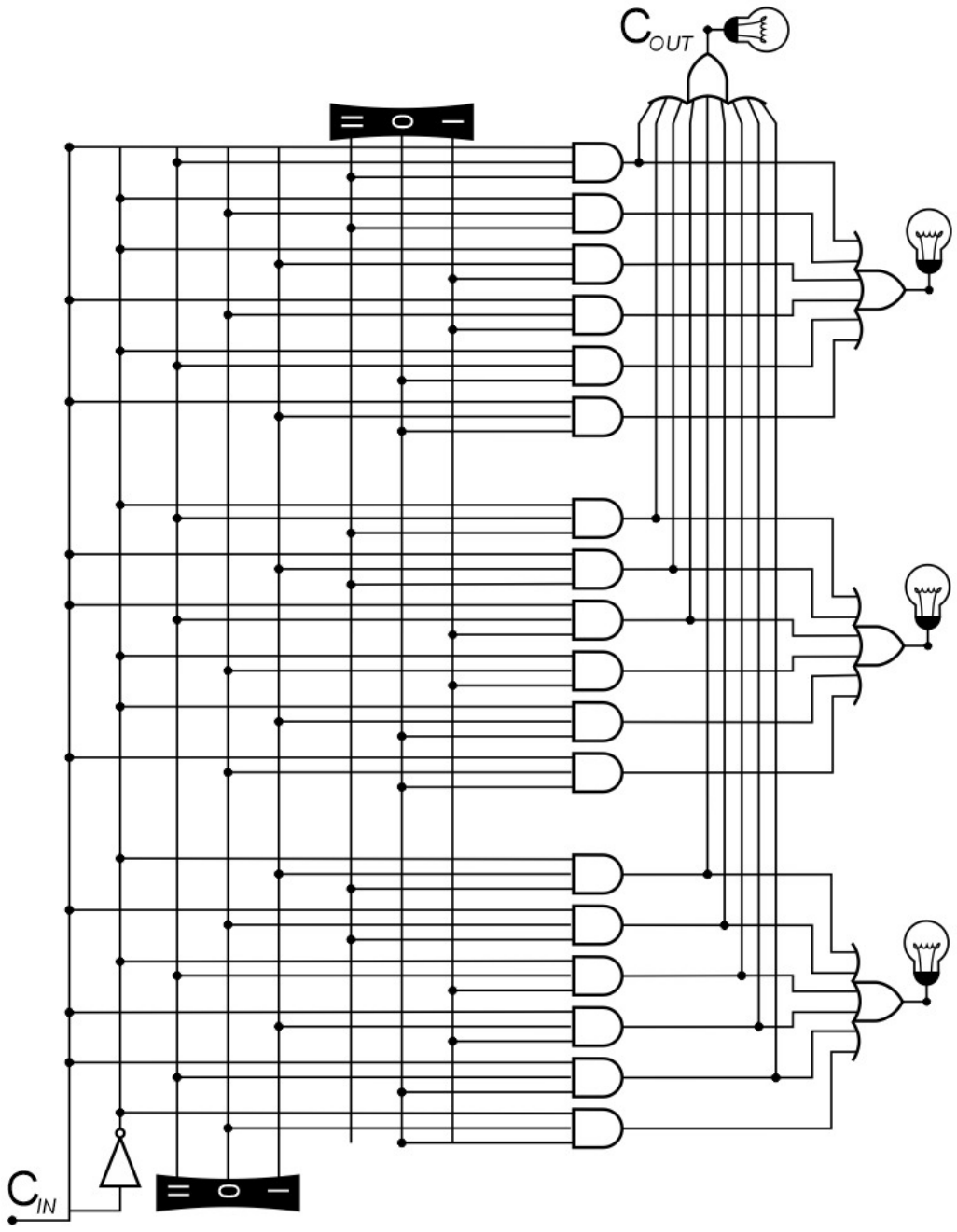


Figura 44: Somador completo ternário

O circuito do somador apresentado na figura 44 foi construído no **Logic-Ly** e seu arquivo está disponível para download em:
https://drive.google.com/open?id=12zoAauSn2v4Vcj9FWxLlUKBZHi9wfRcV

No início indagamos: Somador ternário, por que não? E agora que já construímos o seu circuito lógico temos uma resposta que é deveras simples: obviamente impraticável. O circuito lógico é desmedido, utiliza muitas portas lógicas e realiza a mesma função que o somador binário.

A implantação da lógica ternária na programação de calculadoras e computadores implica em "ensinar" a máquina a lidar com o incerto. Dado que temos três valores lógicos: verdadeiro; falso e **desconhecido**, que poderá ser considerado como verdadeiro; falso; verdadeiro e falso; provável; neutro; entre outros, sua função dependerá da lógica adotada.

Aderir à lógica multivalorada e/ou nebulosa[10]*(que admite mais de dois valores verdade)* corresponde a abandonar a lógica no domínio booleano. Para exemplificar, denotemos por $\times$ o valor lógico desconhecido, ao associarmos tanto $(1, 0)$ como $(1, \times)$ por meio de uma porta *OR*, obteríamos 1 como saída em ambos os casos, o que pode gerar ambiguidades.

Contudo, a lógica multivalorada possui inúmeras aplicações, inclusive em funções booleanas, sua implementação possibilitaria o aprimorar das máquinas por meio da ampliação de suas funções e expansão da densidade de sua memória sem necessariamente aumentar o espaço físico ocupado. Porém, tais vantagens são apenas teóricas, pois como acabamos de ver, o motivo que inviabiliza sua adesão é a criação de circuitos compatíveis e que possam competir com os já existentes. Entretanto, isto nos remete a um outro questionamento:

Se seguirmos nesta linha, o somador quaternário ficará ainda maior, e nem queremos imaginar como seria o circuito lógico de uma calculadora convencional que utiliza a base decimal. E agora, será que existe alguma alternativa ou realmente é mais prático efetuar os cálculos à mão?

#### 3.4.1.3 *Somador quaternário e além!*

Se fôssemos construir a tabela verdade para um meio somador quaternário, teríamos ao menos $2^4 = 16$ *linhas* isto claro, se considerarmos que cada uma das entradas possa assumir valores **0** ou **1** apenas.
Felizmente existe uma alternativa, deveras elegante. De fato, o somador binário é o mais simples, por isso o utilizaremos. Para tanto, vamos nos valer dos codificadores e decodificadores. Segue na figura 45 o circuito lógico de um somador completo quater-

10 Indicamos [6] para aprofundar-se sobre o tema.

nário:

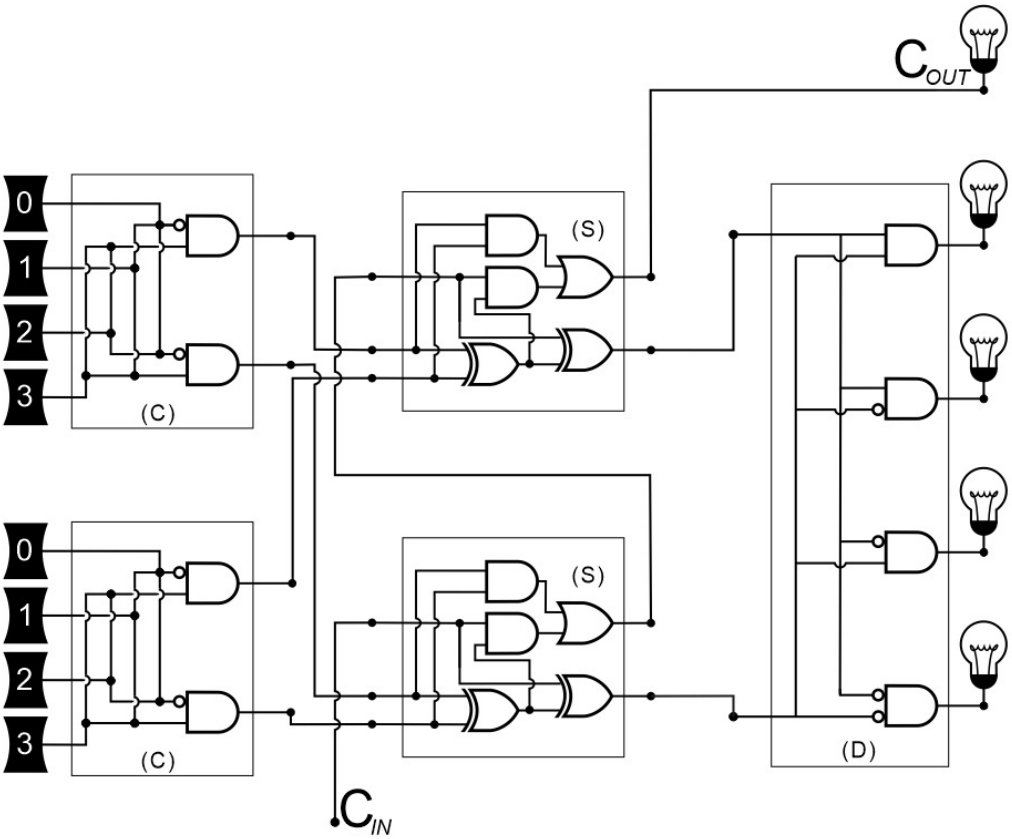


Figura 45: Somador completo quartenário

O circuito do somador completo quartenário foi construído no **Logic-Ly** e seu arquivo está disponível para download em:
https://drive.google.com/open?id=1yc-4y0BB1xk1YQovHZ2cda4zdj34E5LJ

Primeiramente temos as entradas *(0,1,2,3)* que serão convertidas em números binários *(00,01,10,11)* por meio dos CODIFICADORES *(C)*, na sequência, uma vez convertidos, tais números serão somados por meio de SOMADORES COMPLETOS *(S)* que já contam com o $carry_{in}$ e $carry_{out}$. Contudo, as saídas serão números binários que devem ser convertidos novamente em números quaternários. Para tanto, utilizaremos o DECODIFICADOR *(D)*.

Acabamos por auferir um resultado expressivo com a criação de um procedimento otimizador para projetar somadores completos em bases diversas. Primeiramente cons-

truímos um codificador de uma base qualquer para a base binária. Em seguida, utilizamos tantos somadores binários completos quantos forem os *bits* na saída da codificação. Por fim, construímos um decodificador para voltar à base original ou não, pois com este procedimento, o decodificador pode apresentar a soma em uma base arbitrária.

### 3.4.2 *Subtratores*

Análogo ao somador, o subtrator binário utilizará o recurso da *propagação de empréstimo*. Difere apenas no *borrow* (empréstimo por transporte), que designaremos por $T$ e possui a mesma função do *carry*. $D$ será a diferença ($A - B$) o $T$ inicial igual a 1 é somado ao complemento de dois do subtraendo e com o minuendo, o $T$ de saída é propagado para a soma seguinte de forma sucessiva, o último $T$ é desconsiderado. Analisemos sua tabela verdade:

| **A** | **B** | **D** | $\mathbf{T}_{out}$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

Temos que a diferença $D$ fica determinada por uma porta lógica *XOR*. Já o $T_{out}$ por meio de uma porta lógica *AND*, com a entrada $A$ barrada. Segue na figura 46 o circuito lógico do meio subtrator:

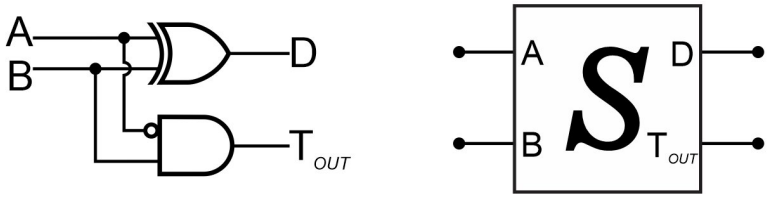


Figura 46: Meio subtrator

O subtrator completo admite três entradas: $A$, $B$ e $T_{in}$ e fornecem duas saídas, a diferença $D$ e um novo *borrow* de saída, o $T_{out}$, observe a seguinte tabela verdade:

| *d* | **A** | **B** | $\mathbf{T}_{in}$ | **D** | $\mathbf{T}_{out}$ |
|---|---|---|---|---|---|
| *0* | 0 | 0 | 0 | 0 | 0 |
| *1* | 0 | 0 | 1 | 1 | 1 |
| *2* | 0 | 1 | 0 | 1 | 1 |
| *3* | 0 | 1 | 1 | 1 | 0 |
| *4* | 1 | 0 | 0 | 0 | 1 |
| *5* | 1 | 0 | 1 | 0 | 0 |
| *6* | 1 | 1 | 0 | 0 | 0 |
| *7* | 1 | 1 | 1 | 1 | 1 |

Utilizando os *minterm's (soma-de-produtos)* temos:

$$f(A, B, T_{in}) = D = \sum m(1, 2, 4, 7)$$

$$\overline{A}\cdot\overline{B}\cdot T_{in} + \overline{A}\cdot B\cdot \overline{T_{in}} + A\cdot\overline{B}\cdot\overline{T_{in}} + A\cdot B\cdot T_{in} \overset{A3}{=}$$

$$\overline{A}\cdot\overline{B}\cdot T_{in} + A\cdot B\cdot T_{in} + A\cdot\overline{B}\cdot\overline{T_{in}} + \overline{A}\cdot B\cdot\overline{T_{in}} \overset{A5}{=}$$

$$(\overline{A}\cdot\overline{B} + A\cdot B)\cdot T_{in} + (A\cdot\overline{B} + \overline{A}\cdot B)\cdot\overline{T_{in}} \overset{def}{=}$$

$$(A \odot B)\cdot T_{in} + (A \oplus B)\cdot\overline{T_{in}} \overset{def}{=}$$

$$(\overline{A \oplus B})\cdot T_{in} + (A \oplus B)\cdot\overline{T_{in}} \overset{def}{=}$$

$$A \oplus B \oplus T_{in}$$

Já para o *borrow*, temos a seguinte expressão via *(soma-de-produtos)*:

$$f(A, B, C_{in}) = T_{out} = \sum m(1, 2, 4, 7)$$

$$\overline{A}\cdot\overline{B}\cdot T_{in} + \overline{A}\cdot B\cdot\overline{T_{in}} + \overline{A}\cdot B\cdot T_{in} + A\cdot B\cdot T_{in} \overset{A3}{=}$$

$$\overline{A}\cdot B\cdot\overline{T_{in}} + \overline{A}\cdot B\cdot T_{in} + \overline{A}\cdot\overline{B}\cdot T_{in} + A\cdot B\cdot T_{in} \overset{A5}{=}$$

$$\overline{A}\cdot B\cdot(\overline{T_{in}} + T_{in}) + (\overline{A}\cdot\overline{B} + A\cdot B)\cdot T_{in} \overset{A6}{=}$$

$$\overline{A}\cdot B + (\overline{A}\cdot\overline{B} + A\cdot B)\cdot T_{in} \overset{def}{=}$$

$$\overline{A}\cdot B + (\overline{A \oplus B})\cdot T_{in}$$

Abaixo apresentamos o subtrator completo na figura 47 e o circuito subtrator (*5 bits*) obtido por meio de associações em série como consta na figura 48:

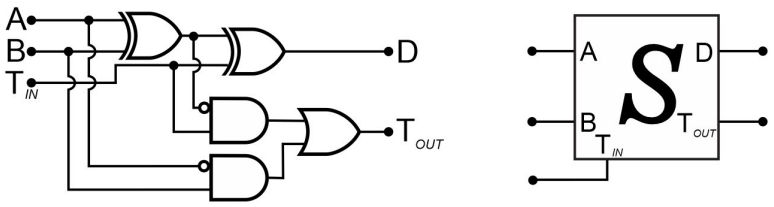


Figura 47: Subtrator completo

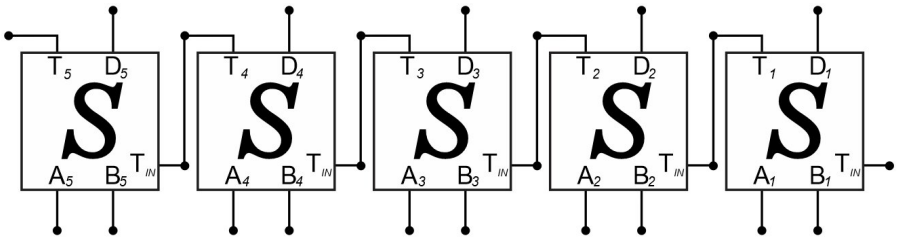


Figura 48: Subtrator em série

O circuito do subtrator em série foi construído no **Logic-Ly** e seu arquivo está disponível para download em:

https://drive.google.com/open?id=1uQ2W49asPmRA4o-mZwd9vn1luGnMh5Zm

#### 3.4.2.1 *Números negativos*

Vejamos como se dá a representação de números negativos, ou melhor, como o computador os reconhece. Voltemos à aritmética binária, para realizar a subtração, o subtraendo era substituído por seu complemento de 2. Este é o método mais prático para representar números negativos.

Em tal procedimento, invertemos os *bits* de um número binário e adicionamos a este, 1 *bit*, isto o torna negativo, por propagação, este *bit* extra, aloca-se no algarismo mais significativo. Assim, caso este último *bit* seja *1*, a máquina o tem como negativo. Já números positivos que não sofreram modificações, possuem *0* em seu *bit* mais significativo.

Tal fato justifica tanto a adição de *1 bit*, quanto o descarte do último *bit* mais significativo, já que este nada mais é do que o sinal do número binário. Assim, o computador efetua a subtração através da soma com o oposto do subtraendo. A vantagem deste tipo de representação consiste em que o computador consegue lidar com números positivos e negativos da mesma forma.

### 3.4.3 *Multiplicadores*

Os circuitos de multiplicação e divisão[11] raramente são empregados. Tais operações são efetuadas por meio de rotinas em linguagem de programação *Assembly*[12]. Fluxogramas são a representação gráfica do procedimento a ser executado pelo processador da máquina.

Contudo, iremos construir um multiplicador completo utilizando preceitos da multiplicação de matrizes, cuja técnica é usual e simples quando trabalhadas com números binários, basicamente, realizaremos somas. Primeiramente efetuamos o produto parcial $P_p$ das entradas $A$ e $B$, em seguida realizamos a soma entre $P_p$, o $P_{in}$ *(produto proveniente do termo anterior)* e o $carry_{in}$, veja a tabela verdade:

| *d* | **A** | **B** | $\mathbf{P}_{in}$ | $\mathbf{C}_{in}$ | $\mathbf{P}_{p}$ | $\mathbf{C}_{out}$ | $\mathbf{P}_{out}$ |
|---|---|---|---|---|---|---|---|
| *0* | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| *1* | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| *2* | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| *3* | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| *4* | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| *5* | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| *6* | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| *7* | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| *8* | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| *9* | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| *10* | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| *11* | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| *12* | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| *13* | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| *14* | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| *15* | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Por *(soma-de-produtos)* obtemos:

$$P_p = A \cdot B$$

11 Para conferir a idealização destes e de outros circuitos indicamos [5]

12 Notação legível para o programador assemelha-se a um mapa conceitual, para arquitetar a rotina de algoritmos a serem entendidos e seguidos por dispositivos computacionais, como microprocessadores e microcontroladores. O código de máquina torna-se legível pela substituição dos valores em bruto por símbolos mnemônicos, leia mais sobre em [5].

$$f(A,B,C_{in}) = P_{out} = \sum m(1,2,5,6,9,10,12,15)$$

$$\overline{P_{in}}\cdot C_{in}\cdot \overline{P_p} + P_{in}\cdot \overline{C_{in}}\cdot \overline{P_p} + \overline{P_{in}}\cdot C_{in}\cdot \overline{P_p} + P_{in}\cdot \overline{C_{in}}\cdot \overline{P_p} +$$

$$+\overline{P_{in}}\cdot C_{in}\cdot \overline{P_p} + P_{in}\cdot \overline{C_{in}}\cdot \overline{P_p} + \overline{P_{in}}\cdot \overline{C_{in}}\cdot P_p + P_{in}\cdot C_{in}\cdot P_p \overset{T1}{=}$$

$$\overline{P_{in}}\cdot C_{in}\cdot \overline{P_p} + P_{in}\cdot \overline{C_{in}}\cdot \overline{P_p} + +\overline{P_{in}}\cdot \overline{C_{in}}\cdot P_p + P_{in}\cdot C_{in}\cdot P_p \overset{A5}{=}$$

$$(\overline{P_{in}}\cdot C_{in} + P_{in}\cdot \overline{C_{in}})\cdot \overline{P_p} + (\overline{P_{in}}\cdot \overline{C_{in}} + P_{in}\cdot C_{in})\cdot P_p \overset{def}{=}$$

$$(P_{in}\oplus C_{in})\cdot \overline{P_p} + \overline{(P_{in}\oplus C_{in})}\cdot P_p \overset{def}{=}$$

$$P_{in}\oplus C_{in}\oplus P_p$$

$$f(A,B,C_{in}) = C_{out} = \sum m(3,7,11,13,14,15)$$

$$P_{in}\cdot C_{in}\cdot \overline{P_p} + P_{in}\cdot C_{in}\cdot \overline{P_p} + P_{in}\cdot C_{in}\cdot \overline{P_p} + \overline{P_{in}}\cdot C_{in}\cdot P_p + P_{in}\cdot \overline{C_{in}}\cdot P_p + P_{in}\cdot C_{in}\cdot P_p \overset{T1}{=}$$

$$P_{in}\cdot C_{in}\cdot \overline{P_p} + \overline{P_{in}}\cdot C_{in}\cdot P_p + P_{in}\cdot \overline{C_{in}}\cdot P_p + P_{in}\cdot C_{in}\cdot P_p \overset{A5}{=}$$

$$(P_{in}\cdot \overline{P_p}) + \overline{P_{in}}\cdot P_p)\cdot C_{in} + P_{in}\cdot P_p\cdot (\overline{C_{in}} + C_{in}) \overset{def}{=}$$

$$(P_{in}\oplus P_p)\cdot C_{in} + P_{in}\cdot P_p\cdot (\overline{C_{in}} + C_{in}) \overset{A6}{=}$$

$$(P_{in}\oplus P_p)\cdot C_{in} + P_{in}\cdot P_p$$

Na figura 49 apresentamos o multiplicador completo, bloco básico e essencial para a construção desta calculadora:

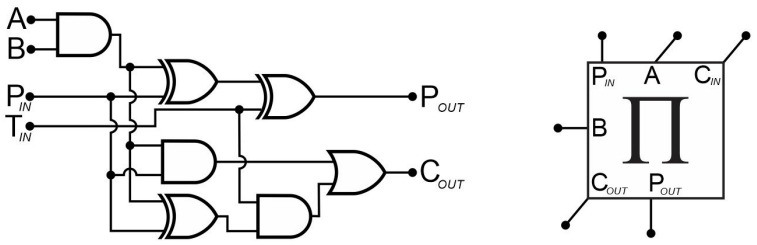


Figura 49: Multiplicador completo

Tais $células_{ij}$ serão associadas a uma matriz, o escalonamento das linhas respeita o deslocamento dos *bits*. Esta calculadora é capaz de multiplicar números de 4 *bits* e seu projeto está representado na figura 50.

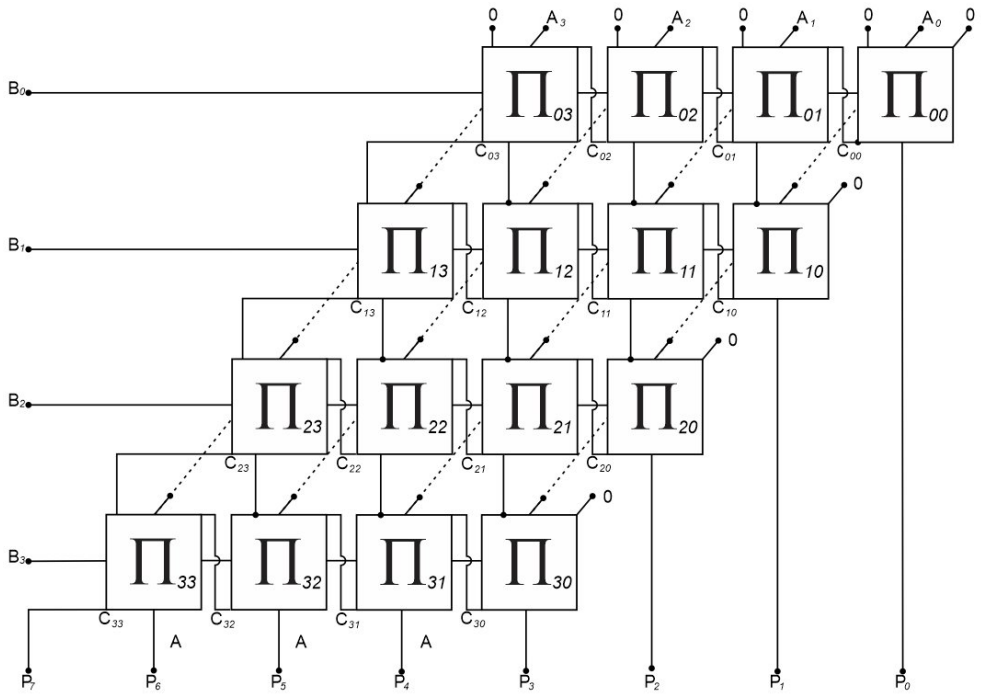


Figura 50: Matriz multiplicadora

Para ampliar a capacidade desta calculadora basta associar mais células em cada linha da matriz, bem como, acrescentar-lhe mais linhas. O circuito da matriz multiplicadora foi construído no **Logic-Ly** e seu arquivo está disponível para download em:

https://drive.google.com/open?id=1st3Zlknl68E6bl6JOZPzbdxcXzTIDTqS

### 3.4.4 *Divisores*

A divisão, de fato, é a operação que mais demanda recursos. Utilizaremos o algoritmo das subtrações sucessivas, normalmente apresentado em forma de fluxograma. Evidentemente, a simplificação da linguagem oculta a complexidade dos circuitos, no qual temos de combinar registradores e contadores aos subtratores. Contudo apresentaremos um circuito similar à matriz multiplicadora da figura 50.

Para efetuar a divisão de $X$ por $Y$, subtraímos sucessivamente o divisor $Y$ dos dígitos mais significativos de $X$. Caso a diferença seja positiva, $E = 1$, se negativa, então $E = 0$ e a subtração é suspensa. O quociente é a junção dos valores de $E$, enquanto que o resto advém do último passo de subtração.

**Exemplo 3.1** (Divisão binária por subtrações sucessivas)**.**

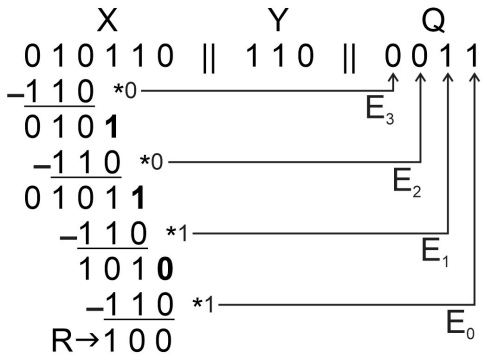


Figura 51: Divisão binária

Note que o procedimento representado pela figura 51 faz uso do condicional: **SE** $E = 1$ subtraia, **SENÃO** cancele. Para tanto, precisaremos de um circuito peculiar:

**MULTIPLEXADOR**$_{(2\ \textit{para}\ 1)}$

Tal circuito utiliza um valor de endereço $E$ para selecionar apenas uma dentre várias entradas direcionando-a para assumir o valor de saída. Abaixo segue a tabela verdade do multiplexador *(MUX)* 2 para 1:

| *d* | **E** | **A** | **B** | **Y** |
|---|---|---|---|---|
| *0* | 0 | 0 | 0 | 0 |
| *1* | 0 | 0 | 1 | 0 |
| *2* | 0 | 1 | 0 | 1 |
| *3* | 0 | 1 | 1 | 1 |
| *4* | 1 | 0 | 0 | 0 |
| *5* | 1 | 0 | 1 | 1 |
| *6* | 1 | 1 | 0 | 0 |
| *7* | 1 | 1 | 1 | 1 |

Por meio da *(soma-de-produtos)* obtemos, como esperado:

$$f(E, A, B) = Y = \sum m(2, 3, 5, 7)$$

$$\overline{E} \cdot A \cdot \overline{B} + \overline{E} \cdot A \cdot B + E \cdot \overline{A} \cdot B + E \cdot A \cdot B \overset{A5}{=}$$

$$\overline{E}\cdot A\cdot(\overline{B}+B)+E\cdot B\cdot(A+\overline{A}) \overset{A6}{=}$$

$$\overline{E}\cdot A+E\cdot B$$

Segue o circuito deste multiplexador[13], que será de suma importância para a confecção do divisor binário:

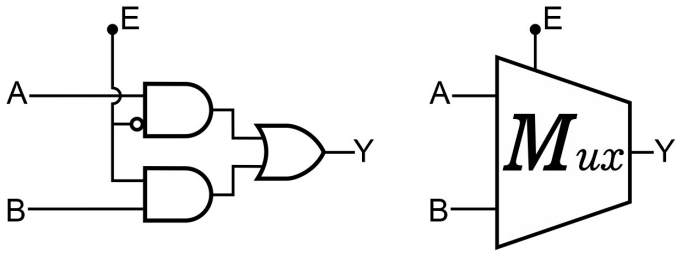


Figura 52: Multiplexador 2 para 1

Por enviar dados de inúmeras entradas através de uma única saída, os multiplexadores são utilizados em larga escala, em especial pelas companhias telefônicas. Possuem inúmeras aplicações. Uma que muito nos interessa refere-se ao multiplexador 4 para 1, posto que, uma vez criado um calculador para cada operação aritmética, poderemos conectá-los a tal circuito e, desta forma, selecionar a operação desejada nesta calculadora com um simples apertar de tecla.

Passemos para a implementação do algoritmo da divisão via combinação de funções lógicas. Utilizaremos subtratores que podem cancelar a subtração, ou seja, vamos acoplar um multiplexador a cada subtrator. O valor do endereço $E$ será o sinal da diferença $T_{out}$, onde o sinal negativo cancela a operação substituindo-a por uma subtração de zeros.

Observe, no exemplo 3.1, que houve inversão de sinais *(1 para positivo e 0 para negativo)* por isso, utilizaremos inversores nas saídas de cada *bit* do quociente. No subtrator temos $A, B$ e $T_{in}$ nas entradas e, $D = A - B$ e $T_{out}$ nas saídas, o sinal da diferença está atrelado ao *bit* de empréstimo:

- 1, subtração positiva e $D = Q$
- 0, subtração negativa e $D = A$

Abaixo segue a tabela verdade da divisão binária:

13 Para conferir a idealização destes e de outros circuitos indicamos [5]

| *d* | **E** | **A** | **B** | $\mathbf{T}_{in}$ | $\mathbf{D}_{out}$ | $\mathbf{T}_{out}$ | **Q** |
|---|---|---|---|---|---|---|---|
| *0* | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| *1* | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| *2* | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| *3* | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| *4* | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| *5* | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| *6* | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| *7* | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| *8* | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| *9* | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| *10* | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| *11* | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| *12* | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| *13* | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| *14* | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| *15* | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Através da *(soma-de-produtos)* obtemos:

$$f(A,B,T_{in}) = D = \sum m(4,5,6,7,9,10,12,15)$$

$$A\cdot B\cdot \overline{T_{in}} + \overline{A}\cdot B\cdot \overline{T_{in}} + A\cdot \overline{B}\cdot \overline{T_{in}} + A\cdot B\cdot T_{in} +$$

$$+\overline{A}\cdot \overline{B}\cdot T_{in} + \overline{A}\cdot B\cdot \overline{T_{in}} + A\cdot \overline{B}\cdot \overline{T_{in}} + A\cdot B\cdot T_{in} \overset{T1}{=}$$

$$\overline{A}\cdot \overline{B}\cdot T_{in} + \overline{A}\cdot B\cdot \overline{T_{in}} + A\cdot \overline{B}\cdot \overline{T_{in}} + A\cdot B\cdot T_{in} \overset{A3}{=}$$

$$\overline{A}\cdot \overline{B}\cdot T_{in} + A\cdot B\cdot T_{in} + A\cdot \overline{B}\cdot \overline{T_{in}} + \overline{A}\cdot B\cdot \overline{T_{in}} \overset{A5}{=}$$

$$(\overline{A}\cdot \overline{B} + A\cdot B)\cdot T_{in} + (A\cdot \overline{B} + \overline{A}\cdot B)\cdot \overline{T_{in}} \overset{def}{=}$$

$$(A \oplus B)\cdot \overline{T_{in}} + (A \odot B)\cdot T_{in} \overset{def}{=}$$

$$(A \oplus B)\cdot \overline{T_{in}} + (\overline{A \oplus B})\cdot T_{in} \overset{def}{=}$$

$$A \oplus B \oplus T_{in}$$

$$f(A,B,T_{in}) = T_{out} = \sum m(1,2,3,7,9,10,11,15)$$

$$\overline{A}\cdot \overline{B}\cdot T_{in} + \overline{A}\cdot B\cdot \overline{T_{in}} + \overline{A}\cdot B\cdot T_{in} + \overline{A}\cdot \overline{B}\cdot T_{in} +$$

$$+\overline{A}\cdot B\cdot \overline{T_{in}} + \overline{A}\cdot B\cdot T_{in} + A\cdot B\cdot T_{in} + A\cdot B\cdot T_{in} \overset{T1}{=}$$

$$\overline{A}\cdot\overline{B}\cdot T_{in} + \overline{A}\cdot B\cdot \overline{T_{in}} + \overline{A}\cdot B\cdot T_{in} + A\cdot B\cdot T_{in} \overset{A3}{=}$$

$$\overline{A}\cdot B\cdot \overline{T_{in}} + \overline{A}\cdot B\cdot T_{in} + \overline{A}\cdot\overline{B}\cdot T_{in} + A\cdot B\cdot T_{in} \overset{A5}{=}$$

$$\overline{A}\cdot B\cdot (T_{in} + \overline{T_{in}}) + (\overline{A}\cdot\overline{B}\cdot + A\cdot B)\cdot T_{in} \overset{A6}{=}$$

$$\overline{A}\cdot B + (\overline{A}\cdot\overline{B}\cdot + A\cdot B)\cdot T_{in} \overset{def}{=}$$

$$\overline{A}\cdot B + \overline{(A\oplus B)}\cdot T_{in}$$

$$f(E, A, D_{in}) = Q = \sum m(4,5,6,7,9,10,12,15)$$

$$\overline{E}\cdot A\cdot D_{in} + \overline{E}\cdot A\cdot \overline{D_{in}} + \overline{E}\cdot A\cdot \overline{D_{in}} + \overline{E}\cdot A\cdot D_{in} +$$

$$+E\cdot\overline{A}\cdot D_{in} + E\cdot\overline{A}\cdot D_{in} + E\cdot A\cdot D_{in} + E\cdot A\cdot D_{in} \overset{T1}{=}$$

$$\overline{E}\cdot A\cdot D_{in} + \overline{E}\cdot A\cdot \overline{D_{in}} + E\cdot\overline{A}\cdot D_{in} + E\cdot A\cdot D_{in} \overset{A5}{=}$$

$$\overline{E}\cdot A\cdot (D_{in} + \overline{D_{in}}) + E\cdot D_{in}\cdot (\overline{A} + A) \overset{A6}{=}$$

$$\overline{E}\cdot A + E\cdot D$$

O circuito mostrado abaixo refere-se à célula divisora, a qual, associaremos em série a fim de configurar as linhas da matriz na construção desta calculadora:

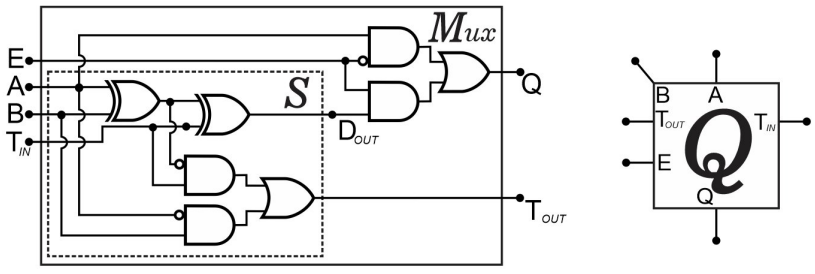


Figura 53: Divisor completo

As *células*$_{(ij)}$ serão associadas a uma matriz divisora com 4 *bits* de capacidade, cada uma executará um ciclo de "*tentativas*", os subtratores calculam a diferença entre os valores de duas entradas. Desta, o *bit* mais significativo é conduzido para o endereço $E$ do multiplexador que define se a subtração será ou não considerada. Caso a diferença seja negativa, a operação é cancelada e substituída por uma subtração de 0. Os dados seguem para a linha seguinte. Ao final temos tanto o quociente quanto o resto[14]. O

14 Esta calculadora desconsidera a divisão por 0.

projeto do circuito[15] está representado na figura 54:

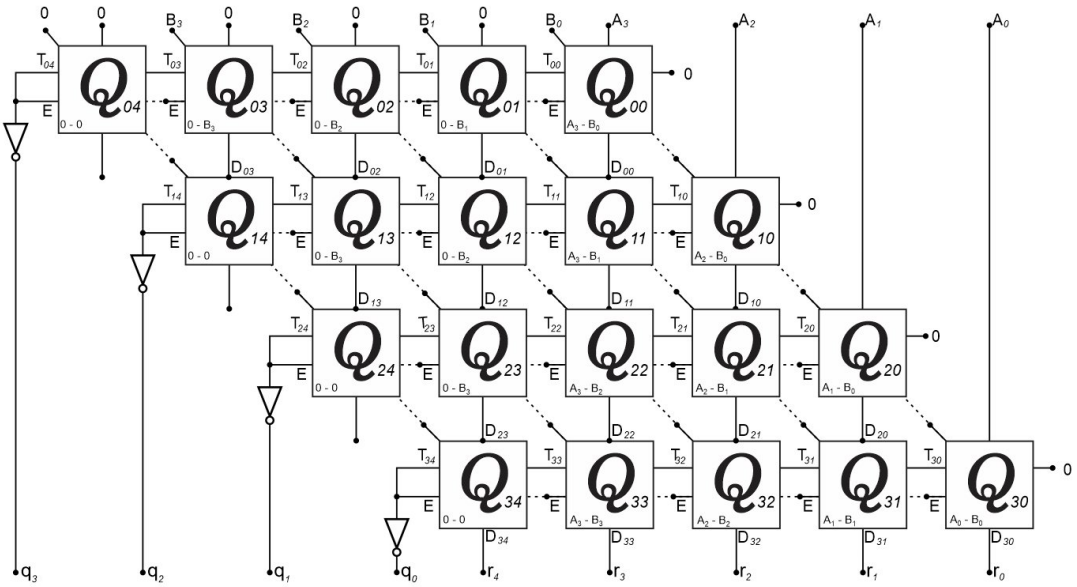

Figura 54: Matriz divisora

O circuito da matriz divisora foi construído no **Logic-Ly** e seu arquivo está disponível para download em:

https://drive.google.com/open?id=1FgbBySQzOsWXmzRz0tgtTM0DDUBcx0k3

Podemos ampliar sua capacidade para dividir números maiores. Para aumentar a quantidade de *bits* do dividendo basta inserir mais linhas à matriz divisora. Para aumentar a quantidade de *bits* do divisor, agregamos mais células divisoras em cada linha. Como esta calculadora apresenta quociente e resto, podemos explorar as congruência módulo um número $m$ dado, pela definição 1.4, para verificarmos se dois números inteiros dados, $a$ e $b$ são congruentes módulo $m$, efetuamos na calculadora a diferença $a - b$ e na sequência a dividimos por $m$ e verificamos se o resto é nulo. Já para determinar a qual número $x$ o inteiro $a$ é congruente módulo $m$, basta efetuar a divisão de $a$ por $m$ nesta calculadora e tomar seu respectivo resto por $x$.

Em face do exposto, acreditamos que o propósito inicial do trabalho foi alcançado, uma vez que exploramos resultados importantes referentes à aritmética modular, o sistema de numeração binária e a álgebra booleana em prol do engendrar de calcula-

15 Indicamos [5] para leitura detalhada.

doras somadores; subtratores; multiplicadores e divisores que, por sua vez, podem ser utilizados para explorar os conteúdos que viabilizaram a sua concepção.

Apresentamos no *apêndice B* sugestões de *softwares* e aplicativos gratuitos que possibilitaram desde a exploração do conteúdo acima explicitado, como também, a criação de projetos educacionais que visem à abordagem de tais conteúdos no âmbito educacional.

# 4

# CONCLUSÃO

Os resultados das avaliações externas[1] evidenciam o quanto é desafiador o ensino da Matemática. Há um consenso entre autores renomados[2] de que a inserção das novas tecnologias na prática educacional é crucial para melhoria significativa do quadro atual. Em contrapartida, no âmbito escolar e/ou social encontramos inúmeras limitações e empecilhos referentes a tais práticas.

A presente pesquisa explorou desde a história da matemática até mágicas matemáticas, além de aplicações que podem ser convertidas em temas geradores a serem explorados em projetos e sequências didáticas direcionadas às escolas.

As novas tecnologias fazem parte da rotina do homem moderno, por outro lado, a compreensão dos conceitos envolvidos em sua concepção, ainda é para poucos. Esta pesquisa realizou um levantamento dos elementos essenciais para a confecção de uma calculadora, os conceitos envolvidos, conhecimentos e procedimentos, cuja abordagem na educação básica seria expressivamente significativa, uma vez que demanda o tratamento de conteúdos que transcendem a própria matemática.

Tendo como objetivo cativar o aluno de graduação, futuro profissional da educação, buscou-se demonstrar importantes resultados de forma mais inteligível possível. O aluno proveniente da educação básica que nunca teve contato com temas tais como: congruências, aritmética modular, bases numéricas, álgebra booleana e circuitos lógicos, encontrará nesta dissertação, ainda que em caráter introdutório, um material acessível e fundamentado, propiciando posterior aprofundamento de estudos em áreas diversificadas.

1 Segundo [2], [4], [28] e [38].
2 Leia [30] e [46].

A presente pesquisa uniu conceitos e apresentou métodos que culminam na construção de calculadoras em diferentes bases numéricas perfeitamente praticáveis na educação básica. Por meio da utilização das calculadoras que foram construídas, pode-se explorar os conceitos que fomentaram sua confecção.

Evidentemente há inúmeras vertentes para a extensão desta pesquisa, afinal há incontáveis aplicações dos circuitos lógicos na era da modernidade. Poderíamos tê-la expandido, abordando as unidades lógicas e aritméticas em outras bases numéricas, bem como, explorar circuitos abrangendo mais de uma operação aritmética. Existe, ainda, a possibilidade de introduzir tópicos da linguagem de programação para dar continuidade aos trabalhos, cuja área, além de promissora, até este momento apresenta pouco material a respeito.

A inserção de tais conteúdos no ensino é imprescindível, devido à automação e avanços tecnológicos cerca de 85% das futuras profissões ainda não foram inventadas e, consequentemente, inúmeras profissões deixarão de existir nos próximos anos[3]. Invariavelmente, caso não ocorra a reformulação do currículo, teremos um país de desempregados devido à falta de qualificação profissional. A escola tornar-se-á obsoleta quanto ao preparo do cidadão para o mundo do trabalho, e tal meta configura-se em uma das incumbências primordiais destas instituições, prevista e assegurada por lei[4].

A presente dissertação proporcionou o acesso a subsídios para professores e futuros profissionais da educação no que diz respeito a suscitarem a abordagem dos referidos temas tratados na educação básica, uma vez em que o desenvolvimento dos mesmos abrange uma gama imensa de conceitos matemáticos, e auxiliam no desenvolvimento intelectual fomentando uma série de habilidades e competências na formação do indivíduo. Assim, dedicamos esta pesquisa a todos os profissionais da educação que nutrem a esperança de que a mudança é possível, por meio de seu contínuo aprimoramento e de seu árduo e significativo trabalho.

3 Disponível em [1].
4 Veja [25].

# A

# APÊNDICE A

## A.1 DIVISORES MÁXIMOS, MÚLTIPLOS MÍNIMOS E ALGO EM COMUM

O Máximo Divisor Comum *(MDC)* de dois números naturais $a$ e $b$ que denotamos por *mdc(a,b)* ou simplesmente *(a,b)* é, como o próprio nome sugere, o maior número natural que divide ambos os números.

Por sua vez o Mínimo Múltiplo Comum *(MMC)* de dois números naturais $a$ e $b$ que denotamos por *mmc(a,b)* ou simplesmente *[a,b]* é o menor número natural divisível por ambos os números.

Para calcular o *MDC/MMC* de dois números naturais $a$ e $b$, aprendemos no ensino fundamental a listar todos os seus *divisores/múltiplos* e buscar nestes conjuntos o *maior/menor* elemento da intersecção. Este método é exaustivo, mesmo para os computadores.

Outra forma, menos árdua e que pode vir a ser explorada, consiste em fatorar ambos os números e tomar a *intersecção/união* destas decomposições para o *MDC/MMC* respectivamente. Por exemplo:

$$\begin{aligned} 18 &= 2\cdot 3^2 \\ 30 &= 2\cdot 3\cdot 5 \end{aligned}$$

Logo $mdc(30,18) = 2\cdot 3 = 6$ e $mmc(30,18) = 2\cdot 3^2\cdot 5 = 90$. Vamos agora, nos ater ao *MDC*, que foi amplamente explorado no decorrer deste trabalho, afinal para calcular o *MDC* de números altos, por exemplo, 9876543210 e 123456789 com os procedimentos descritos acima é uma tarefa praticamente impossível.

### A.1.1 *O algoritmo de Euclides*

O algoritmo a seguir encontra respaldo no fato de que: qualquer divisor comum de $a$ e $b$ com $(a > b)$ também dividirá $a - b$, temos ainda que os divisores de $b$ e $a - b$ também são divisores de $a$. Logo $mdc(a, b) = mdc(b, b - a)$.

$$\begin{aligned} mdc(30, 18) &= mdc(18, 12) \\ &= mdc(12, 6) \\ &= mdc(6, 6) \\ &= 6 \end{aligned}$$

Este algoritmo torna-se ainda mais eficiente quando substituímos a diferença $a - b$ pelo resto $r$ da divisão euclidiana de $a$ por $b$, o que viabilizará consideravelmente sua utilização:

$$\begin{aligned} mdc(9876543210, 123456789) &= mdc(123456789, 90) \\ &= mdc(90, 9) \\ &= mdc(9, 0) \\ &= 9 \end{aligned}$$

Por meio deste procedimento, alcançamos o resultado de forma incrivelmente mais rápida. Por fim, podemos enunciar o seguinte lema:

**Lema A.1** (*Euclides*)**.** *Se $a = bq + r$, então $mdc(a, b) = mcd(b, r)$.*

*Demonstração*: Seja $D_a, D_b$ e $D_r$ o conjunto de todos os divisores positivos de $a, b$ e $r$ respectivamente, devemos mostrar que $D_a \cap D_b = D_b \cap D_r$, já que, se os conjuntos forem iguais, então o seu máximo divisor comum também será. Se $d = mdc(a, b)$ então, $d \in D_a \cap D_b$, donde temos que $d|a$ e $d|b$, logo $d|a - bq \iff d|r$, o que acarreta que $d \in D_b \cap D_r$. Analogamente, temos que se $d \in D_b \cap D_r$, então $d|b$ e $d|r$, logo $d|bq + r \iff d|a$ e assim $d \in D_a \cap D_b$. □

Desta forma, o algoritmo de Euclides configura-se na aplicação reiterada do lema acima, que habitualmente designamos por: *método das divisões sucessivas*, e dado que os restos formam uma sequência estritamente decrescente, o algoritmo cessará, de fato, ao atingirmos o resto igual a 0.

## A.2 TEOREMA FUNDAMENTAL DA ARITMÉTICA

**Teorema A.2** (Teorema Fundamental da Aritmética)**.** *Todo número natural maior que 1 pode ser representado de maneira única, salvo a ordenação, como um produto de fatores primos.*

*Demonstração*[1]: Seja $n$ um natural maior que 1, se $n$ é primo não há o que demonstrar, senão, diremos que $n$ é composto e tomemos $p_1$ menor dos divisores de $n$ *(podemos afirmar que $p_1$ é primo, pois, do contrário ele não seria o menor entre os divisores por também ser um número composto)* e assim podemos escrever $n = p_1 n_1$.
Caso $n_1$ seja primo, a prova estaria concluída. Caso contrário, tomaríamos $p_2$, primo e o menor entre os divisores de $n_1$, logo teríamos $n = p_1 p_2 n_2$.

Tal processo poderá ser repetido, porém não indefinidamente, uma vez em que, ao tomarmos os divisores de divisores, acabamos por criar uma sequência decrescente cujos elementos são todos naturais e maiores que 1, e assim, o procedimento certamente cessará. Dado que os números primos $p_1, p_2, \ldots, p_k$ não são necessariamente distintos, temos que:

$$n = p_1^{m1} \cdot p_2^{m2} \cdot \ldots p_k^{mk}.$$

Quanto à unicidade, vamos supor pelo contrário que existam duas formas distintas de representar o natural $n$.

$$n = p_1 p_2 \ldots p_s = q_1 q_2 \ldots q_r.$$

Dado que $p_1$ divide $n$, então $p_1$ divide $p_n = q_1 q_2 \ldots q_r$, segue que $p_1$ divide um determinado $q_j$ que podemos supor ser $q_1$ e, como são ambos primos, segue que $p_1 = q_1$. Repetindo-se tal argumentação para os demais fatores, concluímos que as fatorações são idênticas. □

## A.3 O PROBLEMA DO CHAPÉU MÁGICO

Proposto pelo matemático Issai Schur(1875-1941), em 1916, o problema continua em aberto. Considere os números inteiros positivos 1, 2, 3, 4, ... e dois chapéus, no quais iremos "*colocar*" os números de acordo com a seguinte regra:

1 Esta e outras demonstrações encontram-se detalhadamente em [11]

Denotaremos os chapéus mágicos por A e B, dos quais, devemos escolher um para colocar o número (1), em seguida, devemos colocar o (2) em qualquer um dos dois chapéus, depois o (3) e assim sucessivamente. Contudo, não podemos colocar determinado número em um chapéu caso este seja igual à soma de outros dois números que já estão dentro do chapéu, por exemplo:

| chapéu A | chapéu B |
|---|---|
| 1 | |
| | 2 |
| | 3 |
| | 4 |
| 5 | |

E agora? Em qual chapéu devemos colocar o 6, já que estamos impedidos de colocar no chapéu A, pois o mesmo já contém o 1 e o 5, e também estamos impedidos de colocar no chapéu B devido ao fato de o mesmo conter o 4 e o 2? Resultado: ambos os chapéus mágicos explodem.

Este é um problema interessante de partições cuja questão é, seguindo este algoritmo, qual é o maior número que conseguimos colocar em um dos chapéus antes que ele exploda? Tal problema é proposto por Gordon Hamilton para crianças do segundo ano do ensino fundamental, para que elas pratiquem adições. Ele pede às crianças que trabalhem com os números $1, 2, 3, 4, 5, 6, 7, 8$.

| chapéu A | chapéu B |
|---|---|
| 1 | |
| 2 | |
| | 3 |
| 4 | |
| | 5 |
| | 6 |
| | 7 |
| 8 | |

Bom, não temos onde colocar o 9, logo ambos os chapéus explodirão.

A variação do problema consiste em aumentar a quantidade de chapéus, uma vez que as crianças tenham descoberto a solução, propõe-se um novo problema: qual o maior número que conseguimos colocar em três chapéus antes que eles explodam? E

com quatro chapéus? Abaixo apresentaremos as soluções para até 4 chapéus, pois para cinco ou mais chapéus o problema ainda não tem solução, logo, este é um problema que ainda está em aberto na teoria dos números[2].

| Solução para três chapéus | | | | | |
|---|---|---|---|---|---|
| chapéu | | | chapéu | | |
| A | B | C | A | B | C |
| 1 | | | | | 13 |
| 2 | | | | | 14 |
| | 3 | | | | 15 |
| 4 | | | 16 | | |
| | 5 | | | | 17 |
| | 6 | | | | 18 |
| | 7 | | | 19 | |
| 8 | | | | | 20 |
| | | 9 | | 21 | |
| | | 10 | 22 | | |
| 11 | | | | 23 | |
| | | 12 | | | |

Por não termos onde colocar o número 24, todos os chapéus explodirão.

Para quatro chapéus, o número máximo é 66. A seguir, apresentamos a solução deste problema:

2 Retirado de [20]

| Solução para quatro chapéus | | | | | | | |
|---|---|---|---|---|---|---|---|
| chapéu | | | | chapéu | | | |
| A | B | C | D | A | B | C | D |
| 1 | | | | 34 | | | |
| 2 | | | | | | | 35 |
| | 3 | | | | | | 36 |
| 4 | | | | | | | 37 |
| | 5 | | | | | | 38 |
| | 6 | | | | | | 39 |
| | 7 | | | 40 | | | |
| 8 | | | | | | | 41 |
| | | 9 | | | | | 42 |
| | | 10 | | 43 | | | |
| 11 | | | | | | | 44 |
| | | 12 | | | | | 45 |
| | | 13 | | | | | 46 |
| | | 14 | | | | | 47 |
| | | 15 | | | | | 48 |
| 16 | | | | | | | 49 |
| | | 17 | | | 50 | | |
| | | 18 | | | 51 | | |
| | 19 | | | | 52 | | |
| | | 20 | | 53 | | | |
| | 21 | | | | | 54 | |
| 22 | | | | | | 55 | |
| | 23 | | | | | 56 | |
| | | | 24 | | | 57 | |
| 25 | | | | | | 58 | |
| | | | 26 | | | 59 | |
| | | | 27 | | | 60 | |
| | | | 28 | | | 61 | |
| | | | 29 | | | 62 | |
| | | | 30 | | 63 | | |
| | | | 31 | | 64 | | |
| | | | 32 | | 65 | | |
| | | | 33 | 66 | | | |

## A.4 NÚMEROS EXPLOSIVOS

Uma variação proveitosa do desafio anterior consiste no seguinte jogo: dois participantes de posse de um relógio para o jogo de xadrez[3], deverão decompor os números pares a partir do 4 como soma de dois números primos. Perde a disputa aquele cujo tempo findar antes de concluída a decomposição de seu número, que explode. Fica facultada a consulta do crivo de Erastóstenes.

Temos então uma conjectura: seria sempre possível escrever números pares maiores ou iguais a 4 como a soma de dois números primos. O jogo é realmente justo, ou existem números cuja decomposição é impossível? Se aplicado em sala de aula, os alunos poderão ser confrontados quanto sua veracidade. Após alguns testes para números maiores, alguns poderão dizer que aparentemente o resultado é verídico, contudo não poderão afirmá-lo. Muitos matemáticos já se empenharam em demonstrar a veracidade desta premissa. Tomás Oliveira e Silva[4] já testou todos os números até $4 \cdot 10^{18}$ e anunciou que continuará a verificação. Contudo, o que há de tão intrigante que mobiliza matemáticos do mundo inteiro a concentrar seus esforços em um resultado, digamos, simplório?

### A.4.1 *Conjectura de Goldbach*

O jogo acima se baseia na conjectura de Goldbach, matemático prussiano(1690-1764)[5], um problema em aberto na teoria dos números há quase trezentos anos, em cartas destinadas a Euler, Goldbach profere a seguinte convicção: *todo número natural maior que 5 pode ser escrito como soma de três números primos*.

Na época *(1742)* considerava-se o 1 como sendo um número primo e por este motivo aparece o número 1 nas decomposições em soma de primos, presente nas cartas, como mostra a figura 55. Na realidade, Goldbach escreveu que todo número par, maior que 2 pode ser escrito como soma de três números primos. A conjectura acima é uma versão moderna desta, já que posteriormente, convencionou-se que 1 não era primo, uma das justificativas está no próprio *teorema fundamental da aritmética*, pois do contrário, a unicidade das decomposições de números em fatores primos seria impossível.

3 Existem inúmeros aplicativos para *android* com esta função.
4 http://sweet.ua.pt/tos/goldbach.html
5 Para conhecer mais, recomendamos [7].

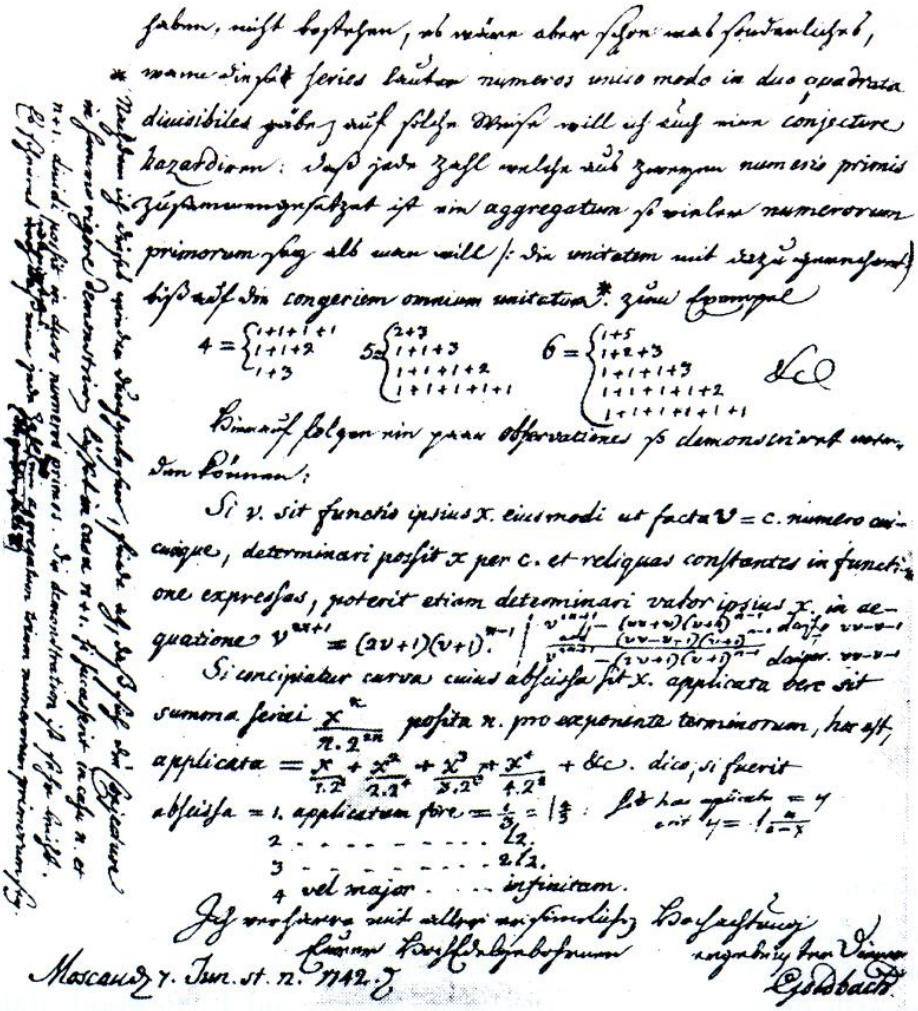

Figura 55: Carta Goldbach - Euler

Em resposta, Euler disse que a afirmação equivaleria à seguinte: *todo número par, maior que 4 (na época, maior que 2) é a soma de dois números primos*. De fato, para $n \geq 3$ e $p$, $p'$, $p''$ números primos, temos que $2n - 2 = p + p'$, então $2n = p + p' + 2$ e desta forma $2n + 1 = p + p' + 3$. Reciprocamente, seja $2n + 2$ um número par maior que 4, se $2n + 2 = p + p' + p''$ então, necessariamente, um dos números primos será 2, donde segue que $2n + 2 = p + p' + 2$, logo $2n = p + p'$.

Euler ainda reitera dizendo ter plena convicção de estar certo quanto à afirmação, porém não era capaz de prová-la. Por ser mais abrangente, a versão de Euler, é conhecida como conjectura "*forte*" de Goldbach. Esta implica ainda na conjectura "*fraca*" de

Goldbach, assim denominada por tratar-se de um corolário da primeira, esta afirma que *todo números ímpar maior ou igual a 7 é a soma de três primos*.

Mesmo em aberto, o problema continua a catalisar esforços e avanços significativos. Recentemente, no ano de *2013*, o matemático Harald Helfgott afirmou ter demonstrado a versão fraca da conjectura de Goldbach[6].

6 Em [40] o leitor encontrará as técnicas utilizadas pelos matemáticos na busca de uma demonstração, bem como os recordes impulsionados por esta célebre conjectura.

# B

# APÊNDICE B

## B.1 LINKS E APLICATIVOS

Construção detalhada da calculadora binária mecânica:
https://woodgears.ca/marbleadd/build.html *(Acesso em 19/11/2017; 16:25)*

Vídeo demonstrativo do funcionamento da calculadora binária mecânica:
https://youtu.be/md0TlSjIags *(Acesso em 19/11/2017; 16:30)*

Simulador da calculadora binária mecânica:
https://leloctai.tk/game/mechanicalAdder/ *(Acesso em 19/11/2017; 17:23)*

Simulador de circuitos lógicos: **Logic ly online**:
https://logic.ly/demo/ *(Acesso em 17/11/2017; 16:40)*

Simulador **Logic ly** versão *tryal* para *download*:
https://logic.ly/download/ *(Acesso em 17/11/2017; 20:05)*

Gerador de código de barras:
https://www.invertexto.com/codigo-barras *(Acesso em 21/11/2017; 12:10)*

Simulador de construções de circuitos digitais:
http://www.tourdigital.net/Simuladores/ConstructorVirtualDeCircuitosPt.zip *(Acesso em 08/12/2017; 14:43)*

## B.2 PROJETO: CALCULADORA MECÂNICA

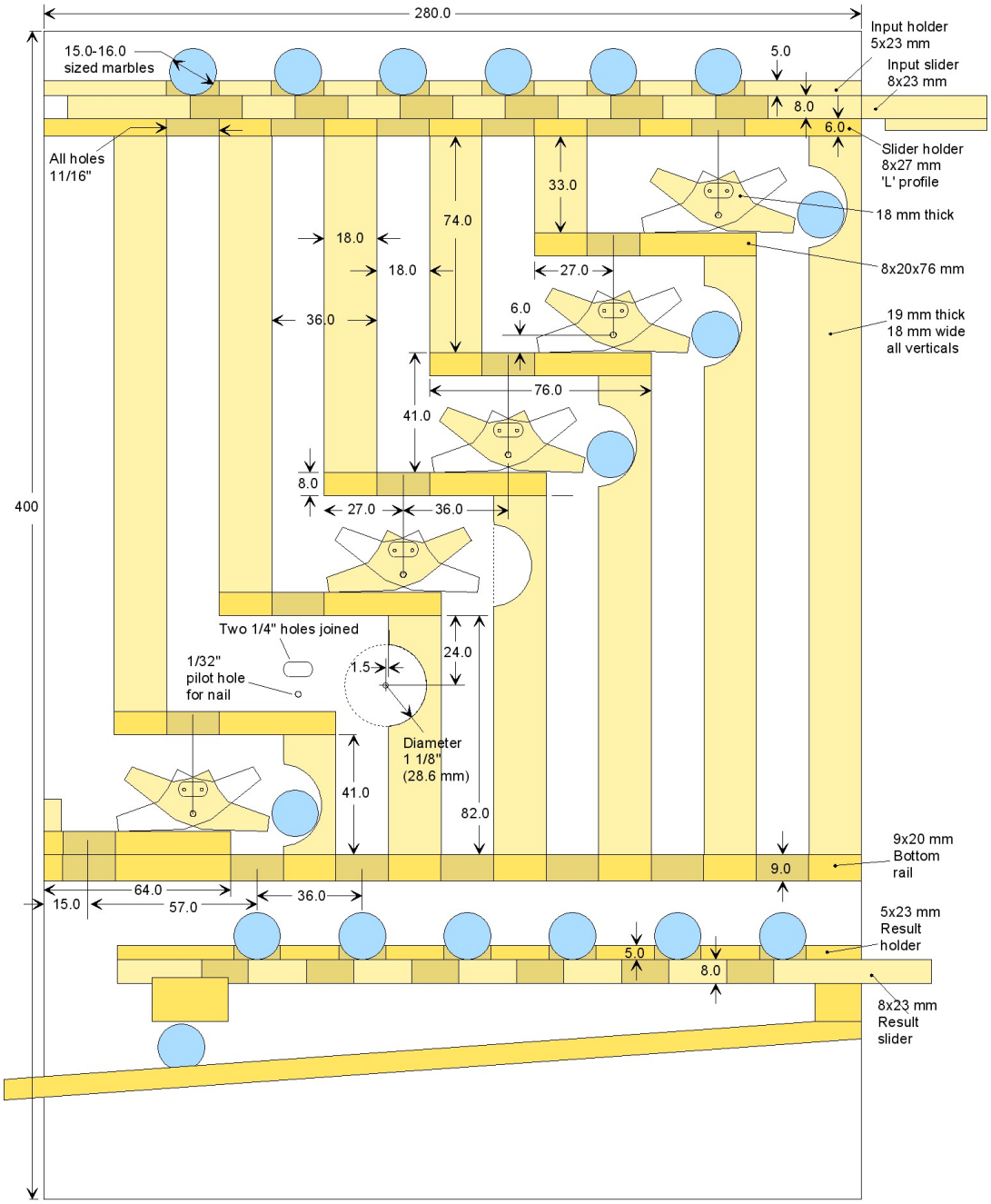


Figura 56: Projeto calculadora binária mecânica

O leitor poderá encontrar o projeto na íntegra em https://woodgears.ca/marbleadd/plans/index.html *(Acesso em 19/11/2017; 16:25)*

## B.3 CALCULADORA BINÁRIA MECÂNICA

Calculadora binária mecânica construída a partir do projeto apresentado na figura 56. Tal calculadora foi doada ao Prof. Dr. Eduardo Guéron e será usada em um projeto de extensão da UFABC voltado a alunos do ensino médio.


Figura 57: Calculadora binária mecânica

## B.4 A EVOLUÇÃO DA CALCULADORA

Do mecânico ao eletrônico, a evolução da calculadora ao longo da história da humanidade.


Figura 58: Evolução da calculadora

## B.5 CIRCUITOS E SIMULADORES

Apresentamos alguns circuitos construídos no **Logic ly**. Tal *software* possui versão online e possibilita a elaboração e validação de projetos de forma simples e intuitiva.

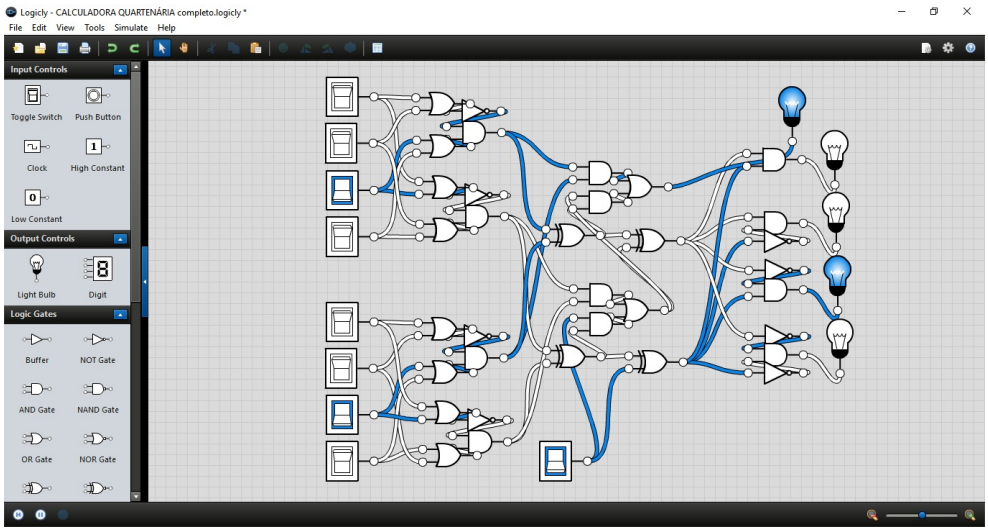


Figura 59: Circuito I projetado no Logic ly

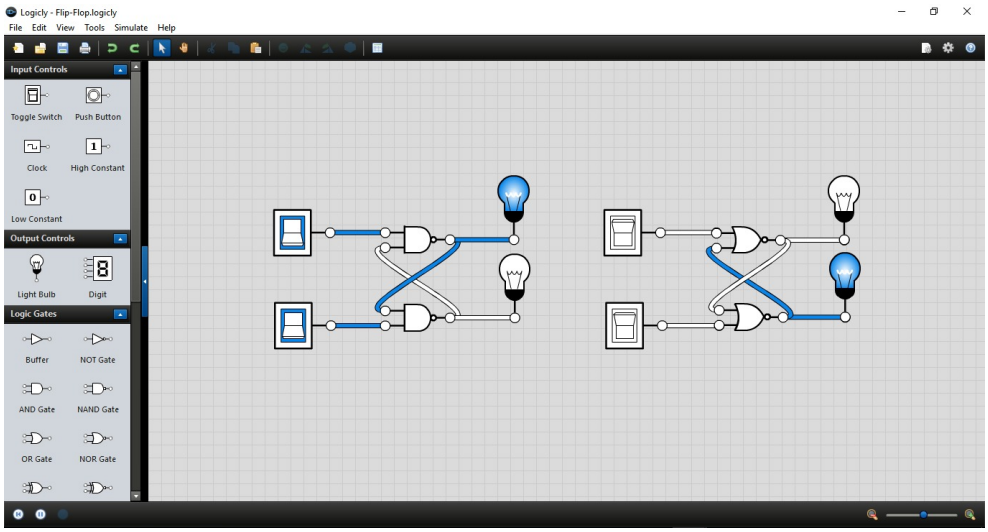


Figura 60: Circuito II projetado no Logic ly

Uma ferramenta extremamente útil para desenvolver sequências didáticas que abordem os circuitos lógicos é o aplicativo gratuito: **Logic Simulator PRO**:

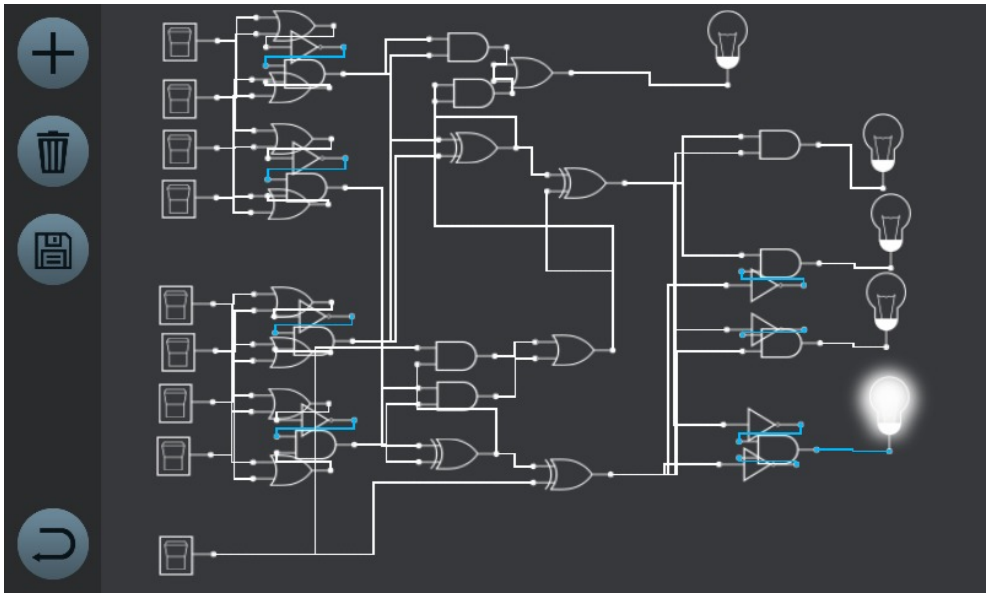

Figura 61: Circuito I projetado no lógic simulator PRO

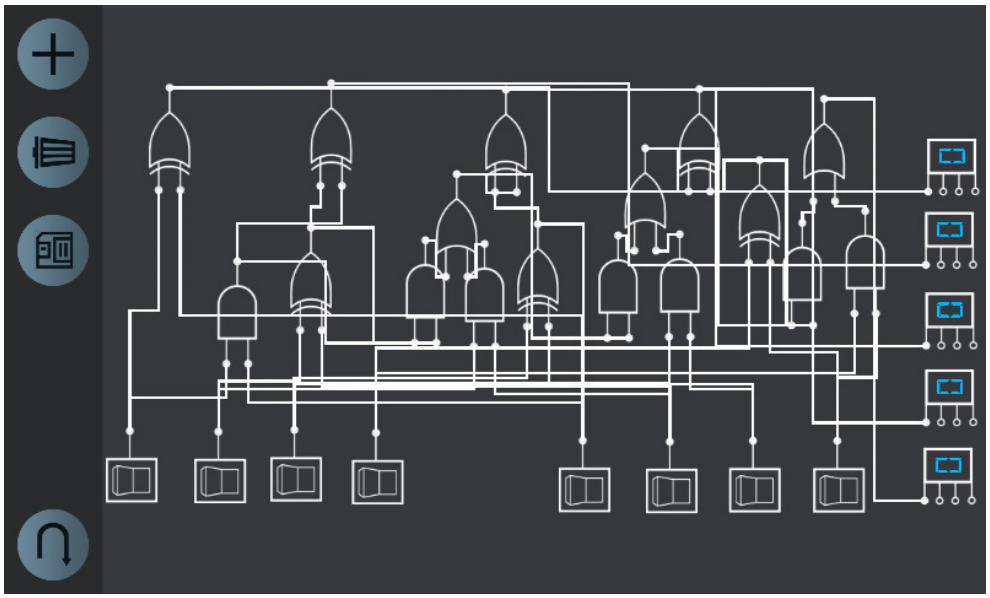

Figura 62: Circuito II projetado no lógic simulator PRO

Este *software* gratuito é um simulador de *protoboard* excelente para estudantes de eletrônica.

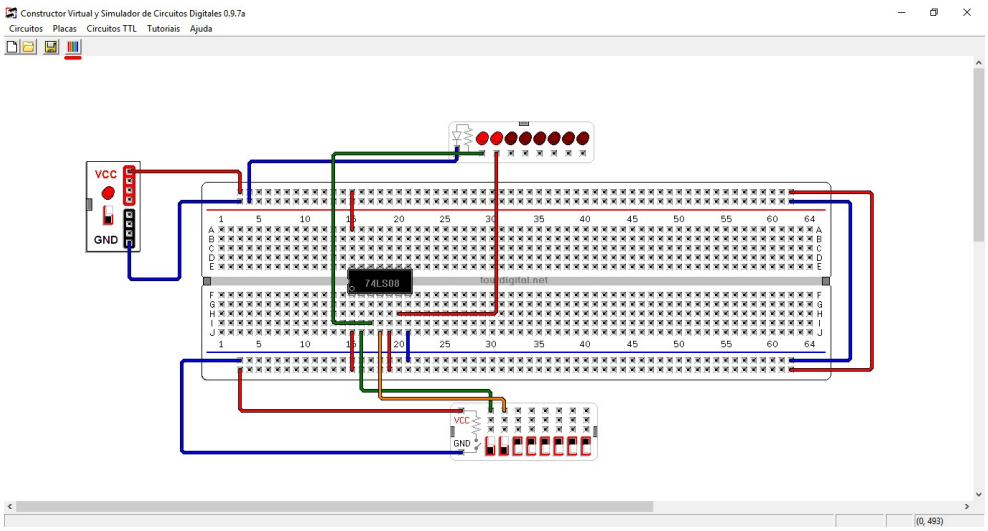


Figura 63: Porta AND - Simulador de construções de circuitos digitais

Arquivo do circuito disponível para *download*.
https://drive.google.com/open?id=1vFHV2Sua5YRFaVyXZtOQY-DsQRe_Ga_B

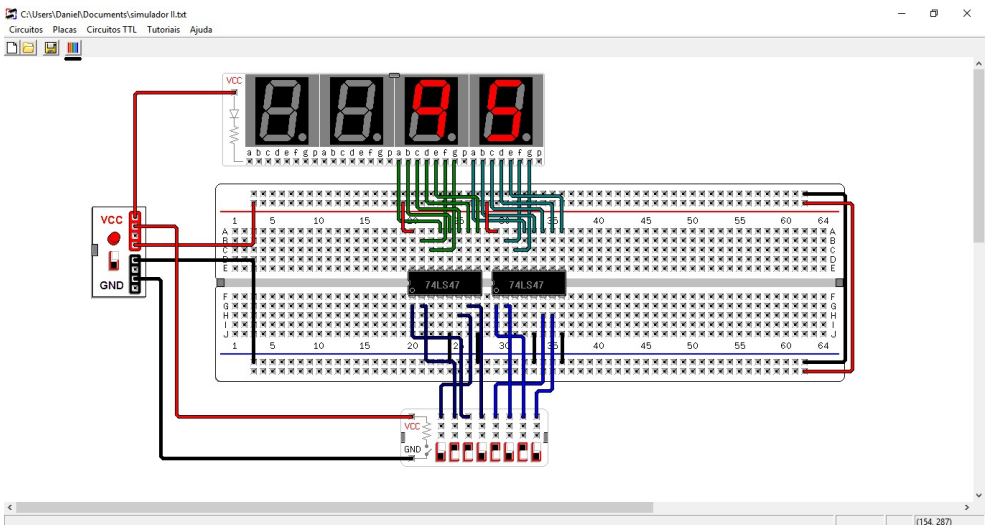


Figura 64: Decodificadores - Simulador de construções de circuitos digitais

Arquivo do circuito disponível para *download*.
https://drive.google.com/open?id=1JdqnE43xUkCOiUh8mpCPni76HOXwlDgQ

# BIBLIOGRAFIA


[1] *Estudo da Dell Technologies projeta o impacto das novas tecnologias na sociedade até 2030*, http://www.dell.com/learn/br/pt/en/press-releases/2017-07-24-dell-technologies-impact-of-new-technologies-on-society, acessado em 22/04/2018.

[2] *Resultado do Pisa de 2015 é tragédia para o futuro dos jovens brasileiros*, http://portal.mec.gov.br/component/content/article?id=42741:, acessado em 21/04/2018.

[3] José Carlos Alves, *Sistemas Digitais. 75 f. 2003. v04*, Material de apoio à disciplina. Faculdade de Engenharia da Universidade do Porto, Porto. Disponível em:< http://www. fe. up. pt/˜ jca/feup/sd>. Acesso em 13/11/2017 **30** (2003).

[4] OCDE Brasil, *Brasil no PISA 2015: análise e reflexões sobre o desempenho do estudante brasileiro*, São Paulo : Fundação Santillana (2016).

[5] Ivan Dario Castellanos, *Analysis and implementation of decimal arithmetic hardware in nanometer CMOS technology*, Oklahoma State University, 2008.

[6] Matheus Bastos de Castro, *Provas de teoremas em lógica três-valorada*, (2017).

[7] Marcelo Santos Chaves et al., *A linguagem dos números primos: uma abordagem epistemológica sobre a conjectura de Goldbach*, Cuadernos de Educación y Desarrollo (2014), nº 45.

[8] Henrique Bernardes da Silva e Esdras Teixeira Costa, *Estruturas de grupos finitos*, Revista Eletrônica de Matemática (2010), nº 2.

[9] Fábio Álvaro Dantas, *Explorando a matemática dentro da calculadora*, Tese de Mestrado, Universidade Federal do Rio Grande do Norte, 2015.

[10] Denise Alves de Araújo e Eduardo Sarquis Soares, *Calculadoras e outras geringonças na escola*, Presença Pedagógica **8** (2002), nº 47.

[11] José Plínio de Oliveira Santos, *Introdução à teoria dos números*, Instituto de Matemática Pura e Aplicada, 1998.

[12] Grace S Deaecto, *Circuitos Lógicos*.

[13] Carlos Magno Corrêa Dias, *Álgebra booleana e lógica digital uma aplicação da lógica matemática*, Revista Acadêmica: ciências agrárias e ambientais (1994).

[14] Howard Whitley Eves, *Introdução à história da matemática*, Unicamp, 1995.

[15] Dmitri Fomin, Sergey Genkin e Ilia Itenberg, *Círculos Matemáticos: A experiência russa*, Trad. Valéria de Magalhães Iório. Rio de Janeiro: IMPA (2012).

[16] Sergei Vasilovich Fomin, *Sistemas de numeração*, Mir, 1984.

[17] Carlos A. Gomes, *Sexta-feira 13*, Revista do Professor de Matemática, nº 59, SBM, Rio de Janeiro, 2006.

[18] Larry Gonick, *Introdução ilustrada à computação (com muito humor!)*, Harbra, 1984.

[19] Rafael Martins Gusmai, *Um estudo dos três problemas clássicos da geometria euclidiana*, Tese de Mestrado, Universidade de São Paulo, 2016.

[20] Gordon Hamilton, *O equilíbrio entre o fracasso e o sucesso*, Revista Cálculo: Matemática para todos, nº 43, Segmento, São Paulo, 2014.

[21] Abramo Hefez, *Elementos de aritmética*, Sociedade Brasileira de Matemática, 2006.

[22] _____, *Curso de álgebra*, Impa, 2013.

[23] Israel Nathan Herstein e Israel N Herstein, *Topics in algebra*, vol. 964, Blaisdell New York, 1964.

[24] Willian Barden Jr., *Matemática para microcomputadores*, Campus, 1985.

[25] BRASIL Lei n. 9.394, de 20 de dezembro de 1996, *Estabelece as diretrizes e bases da educação nacional.*, `http://www.planalto.gov.br/Ccivil_03/leis/L9394.htm`, acesso em 22/04/2018.

[26] Guilherme Liegel Leopold, *Congruência e aplicações*, (2015).

[27] Alex Sandro Faria Manuel, *Monografia sobre Anéis Booleanos e o Anel das partes de um conjunto*, (2013).

[28] Emerson Silva Mazulo, *Análise da proficiência em matemática por meio de regressão linear múltipla*, Revista Intersaberes **10** (2015), nº 21, 613–626.

[29] Francisco César Polcino Milies, *A matemática dos códigos de barras*, Revista do

Professor de Matemática, nº 65, SBM, Rio de Janeiro, 2008.

[30] José Manuel Moran, *Novas tecnologias e mediação pedagógica*, Papirus Editora, 2000.

[31] Jurandyr Santos Nogueira, *Eletrônica digital básica*, (2011).

[32] Mirella Kiyo Okumura, *Números primos e criptografia RSA*, Tese de Doutoramento, Universidade de São Paulo, 2014.

[33] Edvaldo Fialho de Oliveira, *A calculadora como ferramenta de aprendizagem*, (2011).

[34] Krerley Oliveira e Adán J Corcho, *Iniciação à Matemática*, Rio de Janeiro: SBM (2010).

[35] Rafael Américo de Oliveira, *Explorando o universo dos Números Primos*, (2015).

[36] Edward David Moreno Ordoñez, *Projeto, desempenho e aplicações de sistemas digitais em circuitos programáveis (FPGAs)*, Cesar Giacomini Penteado, 2003.

[37] Fabíola A Pessoa, Paulo C Oliveira, Zander P Souza e André LB Cavalcante, *Somador Binário com Decodificador Decimal*.

[38] OECD Pisa, *Draft Science Framework*, 2015, `https://www.oecd.org/pisa/PISA-2015-Brazil-PRT.pdf`, acesso em 22/04/2018.

[39] Jorge Luiz Vares Raposo e Juan Carlos Zavaleta Aguilar, *Criptografia RSA teoria e prática numa abordagem motivacional*.

[40] Paulo Ribenboim, *Números primos: velhos mistérios e novos recordes*, IMPA, 2012.

[41] Fausto Arnaud Sampaio, *Matemágica: história, aplicações e jogos matemáticos*, Papirus Editora, 2005.

[42] João C. V. Sampaio, *Mágica com números*, Revista do Professor de Matemática, nº 60, SBM, Rio de Janeiro, 2006.

[43] Simon Singh, *O livro dos códigos*, Editora Record, 2004.

[44] Ronald J Tocci, Neal S Widmer e Gregory L Moss, *Sistemas digitais: princípios e aplicações*, vol. 8, Prentice Hall, 2003.

[45] Dr Fernando Torres, *Tema: Anéis Booleanos*, (2014).

[46] José Armando Valente et al., *O computador na sociedade do conhecimento*, Campinas: Unicamp/NIED **6** (1999).