INSTITUTO FEDERAL
Minas Gerais
Campus Bambuí

Organização de Computadores

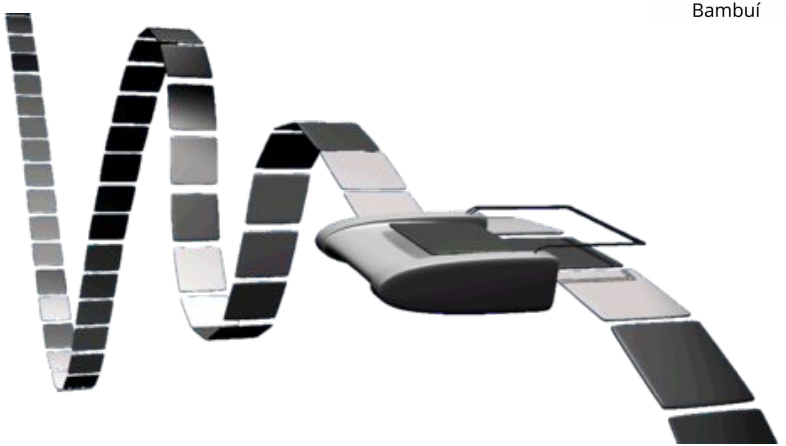
Professor Álvaro Antônio Fonseca de Souza

Aula de hoje

Máquina de Alan Turing

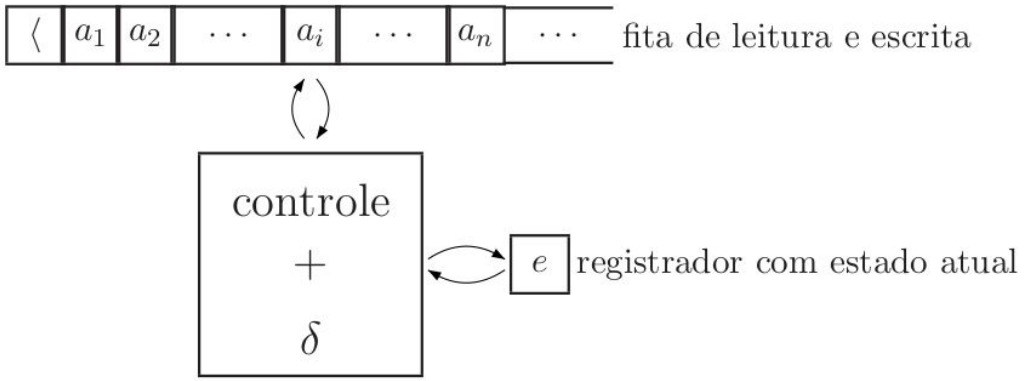
Máquina de Turing

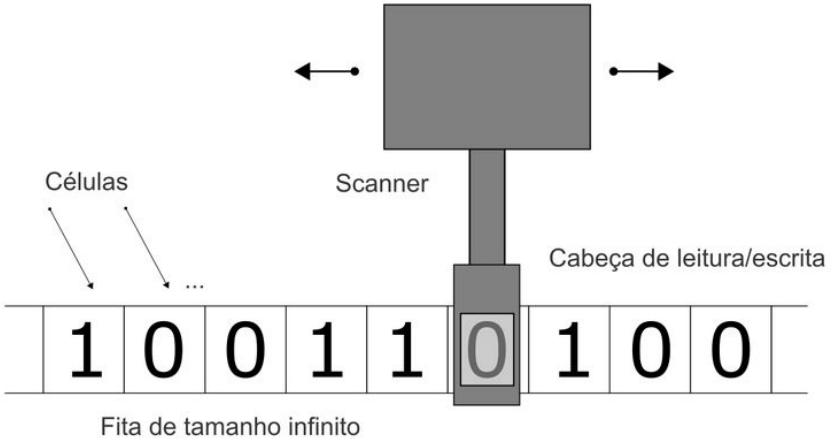
- A máquina de Turing é um modelo abstrato de um computador
- Contém os aspectos lógicos do funcionamento do computador.



Máquina de Turing

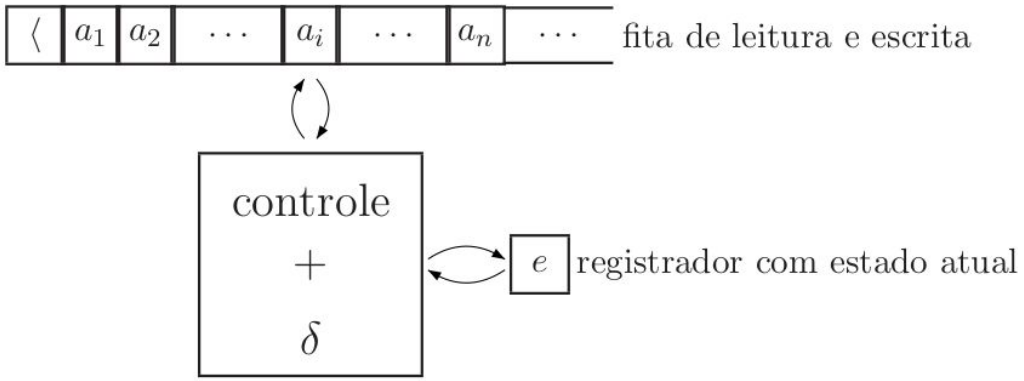
- Uma máquina de Turing (MT) pode ser vista como uma máquina que opera com uma fita com leitura e escrita na fita
- O cabeçote de leitura pode movimentar-se para a direita e para a esquerda.
- A fita é dividida em células
- Cada célula comporta apenas um símbolo
- A fita é ilimitada à direita.

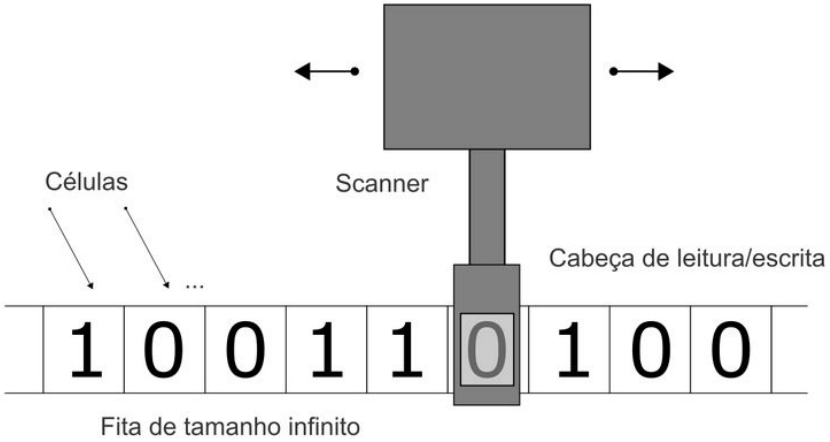




Máquina de Turing

- A máquina possui um registrador para conter o estado atual,
- um conjunto de instruções função de transição da máquina,
- uma unidade de controle



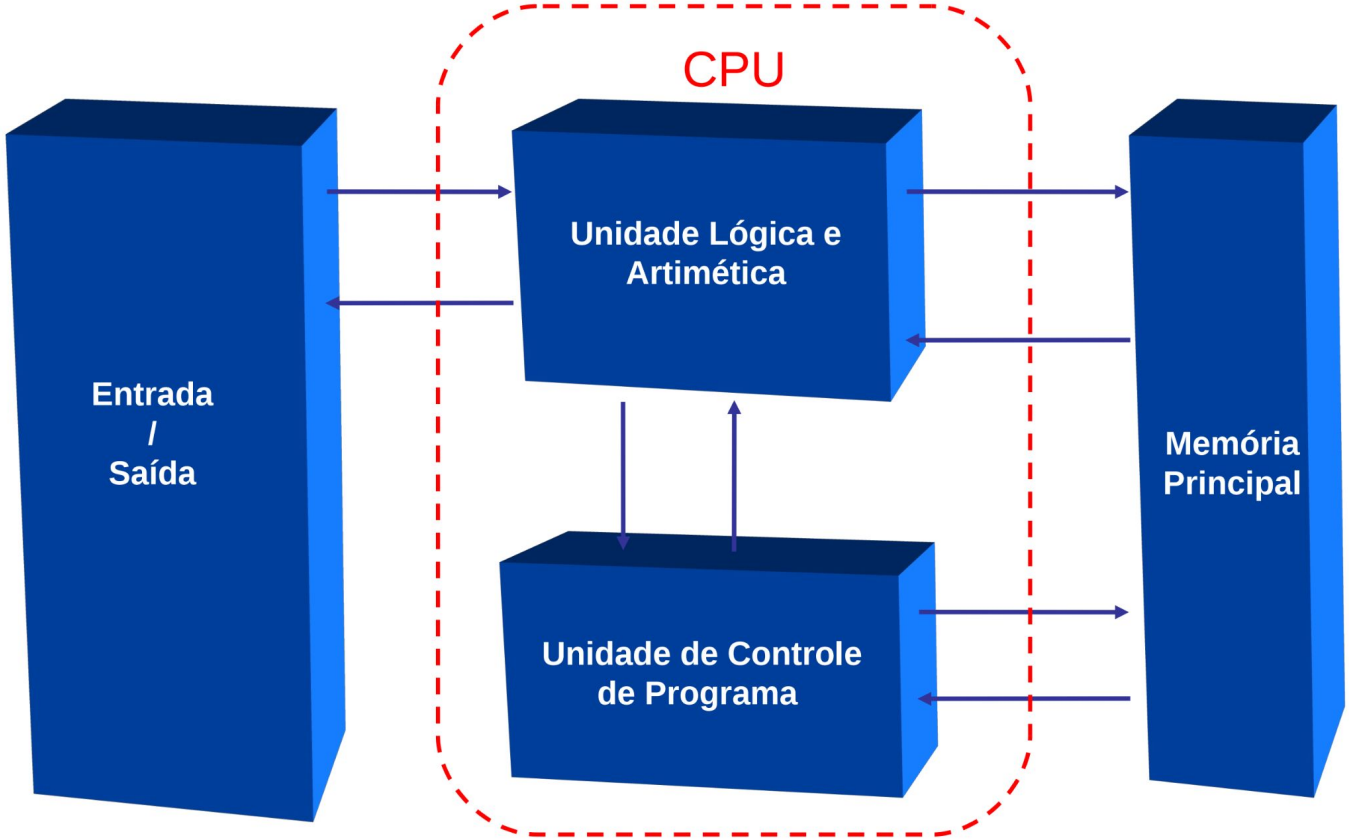


INSTITUTO FEDERAL
Minas Gerais
Campus Bambuí

Aula de hoje

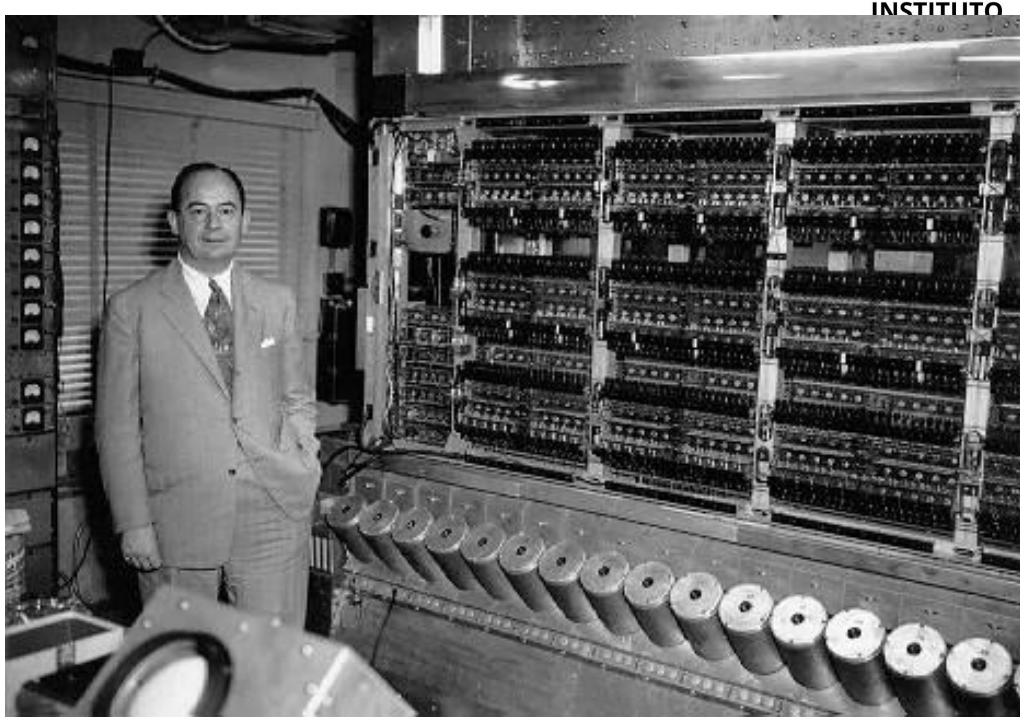
Arquitetura de Von neumann

Máquina de Von Neumann (1945)



Arquitetura de Von Neumann

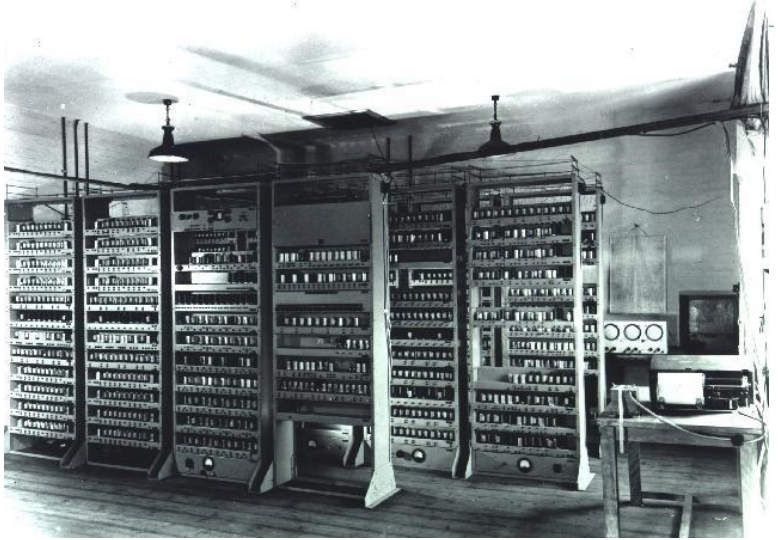
- John von Neumann participou do projeto do ENIAC.
- Desenvolveu a máquina IAS no Institute of Advanced Studies de Princeton.
- Aplicação do conceito de **programa armazenado** na IAS.
- Alan Turing desenvolveu a ideia na mesma época.



IAS e John Von Neumann. Fonte: https://terminaldeinformacao.com/2012/10/19/a-evolucao-dos-sistemas-operacionais-parte-1/, Acessado em 22/11/2024

Arquitetura de Von Neumann

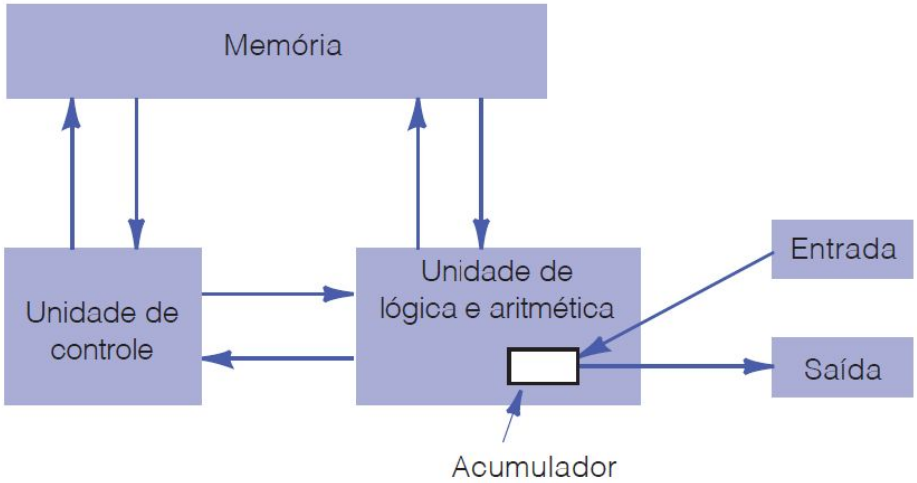
- Programar computadores era uma tarefa lenta.
- O programa na memória do computador, **junto** com os dados.
- A aritmética decimal substituída por **aritmética binária.**
- Usada no EDSAC (Electronic Delay Storage Automatic Calculator), o primeiro computador de programa armazenado.
- É a base de quase todos os computadores digitais modernos.



EDSAC I, Junho de 1948. Fonte: https://pt.wikipedia.org/wiki/Ficheiro:EDSAC_(19).jpg Acessado em 25/11/2024

Componentes da Arquitetura de Von Neumann

- **Memória principal:** armazena dados e instruções.
- **Unidade lógica e aritmética (ALU):** capaz de operar dados binários.
- **Unidade de controle:** interpreta instruções que estão na memória e faz com que sejam executadas.
- **Equipamento de entrada/saída (E/S):** controlado pela unidade de controle.

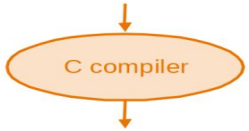


Esquema da máquina de Von Neumann Fonte: [2]

Abstração

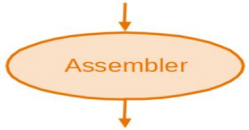
High-level language program (in C)

```
swap(int v[], int k)
{int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```



Assembly language program (for MIPS)

```
swap:
    muli $2, $5,4
    add  $2, $4,$2
    lw   $15, 0($2)
    lw   $16, 4($2)
    sw   $16, 0($2)
    sw   $15, 4($2)
    jr   $31
```



Binary machine language program (for MIPS)

```
00000000101000010000000000011000
00000000100011100001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000000
```

- Maior aprofundamento revela mais informações.
- A abstração omite detalhes desnecessários.
- Lida com a complexidade.

INSTITUTO FEDERAL
Minas Gerais
Campus Bambuí

Aula de hoje

Sistemas numéricos

Introdução

- No dia a dia, usamos um sistema baseado em 10 dígitos (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) para a representação de números.
- Nos referimos a esse sistema como sistema decimal.
- o que o número 83 significa. Ele quer dizer oito dezenas mais três:

$$83 = (8 \times 10) + 3$$

- O número 4.728 significa quatro milhares, sete centenas, duas dezenas, mais oito:

$$4.728 = (4 \times 1.000) + (7 \times 100) + (2 \times 10) + 8$$

Sistema decimal

INSTITUTO FEDERAL
Minas Gerais
Campus Bambuí

- o sistema decimal tem uma base, ou raiz, de 10.
- cada dígito no número é multiplicado por 10, elevado a uma potência que corresponde à posição do dígito:

$$83 = (8 \times 10^1) + (3 \times 10^0)$$

$$4.728 = (4 \times 10^3) + (7 \times 10^2) + (2 \times 10^1) + (8 \times 10^0)$$

Sistema decimal

- O mesmo princípio é mantido para frações decimais,
- potências negativas de 10 são usadas.
- A fração decimal 0,256 corresponde a 2 décimos mais 5 centésimos mais 6 milésimos:

$$0{,}256 = (2 \times 10^{-1}) + (5 \times 10^{-2}) + (6 \times 10^{-3})$$

Sistema decimal

- Um número com uma parte inteira e fracionária
- tem dígitos elevados à potência positiva e à negativa de 10.

$$442{,}256 = (4 \times 10^{2}) + (4 + 10^{1}) + (2 \times 10^{0}) + (2 \times 10^{-1}) + (5 \times 10^{-2}) + (6 \times 10^{-3})$$

Sistema decimal

- o dígito mais à esquerda é conhecido como dígito mais significativo
  - contém o valor mais alto.
- O dígito mais à direita é chamado de dígito menos significativo.
  - contém o valor mais baixo
- No número decimal **442,256**,
  - o 4 à esquerda é o dígito mais significativo
  - o 6 à direita é o dígito menos significativo.

A representação decimal de $X = \{\ldots d_2 d_1 d_0, d_{-1} d_{-2} d_{-3} \ldots\}$, o valor de X é

$$X = \sum_i (d_i \times 10^i)$$

Sistema numérico posicional

sistema numérico posicional

- cada número é representado por uma cadeia de dígitos
- cada posição i do dígito tem um peso associado r^i
- r é a raiz, ou base, do sistema numérico.
- A forma geral de um número nesse sistema com raiz r é

$$(\ldots a_3 a_2 a_1 a_0, a_{-1} a_{-2} a_{-3} \ldots)_r$$

- onde o valor de qualquer dígito a_i é um inteiro no intervalo $0 \leq a_i < r$.
- A vírgula entre a_0 e a_{-1} é chamada de vírgula de raiz.

Sistema numérico posicional

- O número é definido para ter o valor

$$\ldots + a_3 r^3 + a_2 r^2 + a_1 r^1 + a_0 r^0 + a_{-1} r^{-1} + a_{-2} r^{-2} + a_{-3} r^{-3} + \ldots = \sum_i (a_i \times b^i)$$

- O sistema decimal é um caso especial de um sistema numérico posicional
  - com raiz 10 e com dígitos no intervalo 0 a 9.

Sistema numérico posicional

INSTITUTO FEDERAL
Minas Gerais
Campus Bambuí

- Um exemplo de sistema posicional, considere o sistema com base 7.
- A tabela mostra o valor do peso para as posições −1 a 4.
- O valor de dígito varia de 0 a 6

Posição	4	3	2	1	0	−1
Valor na forma exponencial	7^4	7^3	7^2	7^1	7^0	7^{-1}
Valor decimal	2.401	343	49	7	1	1/7

Sistema binário

INSTITUTO FEDERAL
Minas Gerais
Campus Bambuí

- O sistema decimal usa 10 dígitos para representar números em base de 10.
- O sistema binário usa dois dígitos, 1 e 0.
- Os números no sistema binário são representados na base 2.
- Geralmente, utiliza-se um subscrito com um número para indicar a base do número para evitar confusão.

$$0_2 = 0_{10}$$

$$1_2 = 1_{10}$$

Sistema binário

- Para representar números maiores, como a notação decimal,
- Cada dígito em número binário tem um valor que depende de sua posição:

$$10_2 = (1 \times 2^1) + (0 \times 2^0) = 2_{10}$$

$$11_2 = (1 \times 2^1) + (1 \times 2^0) = 3_{10}$$

$$100_2 = (1 \times 2^2) + (0 \times 2^1) + (0 \times 2^0) = 4_{10}$$

Sistema binário

- os valores fracionários são representados com as potências negativas da raiz:

$$1001{,}101 = 2^3 + 2^0 + 2^{-1} + 2^{-3} = 9{,}625_{10}$$

- A representação binária do valor de Y = {... $b_2 b_1 b_0 , b_{-1} b_{-2} b_{-3}$ ... }, valor de Y é

$$Y = \sum_i (b_i \times 2^i)$$

Conversão entre binário e decimal

- Para converter um número de uma notação binária em uma notação decimal.
- Multiplica cada dígito binário pela potência de 2 e adiciona os resultados.
- Para converter de decimal para binário, as partes inteiras e fracionárias são consideradas em separado.

Conversão entre binário e decimal

Inteiro

Na notação binária um inteiro representado por

$$b_{m-1} b_{m-2} \ldots b_2 b_1 b_0 \quad b_i = 0 \text{ ou } 1$$

tem o valor decimal

$$(b_{m-1} \times 2^{m-1}) + (b_{m-2} \times 2^{m-2}) + \ldots + (b_1 \times 2^1) + b_0$$

Conversão entre binário e decimal

- Para converter um inteiro decimal N para a forma binária.
- Se dividirmos N por 2, no sistema decimal, e obtivermos um quociente N_1 e um resto R_0, poderemos escrever

$$N = 2 \times N_1 + R_0 \; R_0 = 0 \; ou \; 1$$

- Em seguida, dividimos o quociente N_1 por 2. Suponha que um novo quociente seja N_2 e o novo resto R_1.

Conversão entre binário e decimal

- Então

$$N_1 = 2 \times N_2 + R_1 \; R_1 = 0 \; ou \; 1$$

- de modo que

$$N = 2(2N_2 + R_1) + R_0 = (N_2 \times 2^2) + (R_1 \times 2^1) + R_0$$

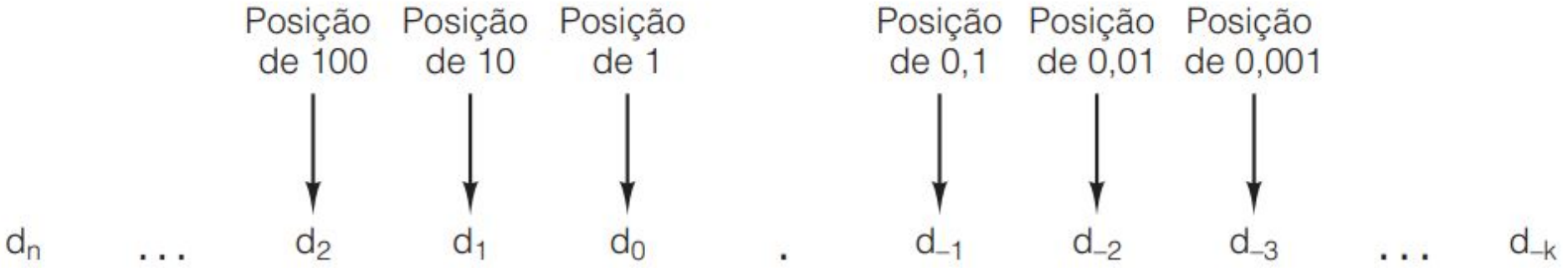
- em seguida,

$$N_2 = 2N_3 + R_2$$

- temos

$$N = (N_3 \times 2^3) + (R_2 \times 2^2) + (R_1 \times 2^1) + R_0$$

Forma geral do número decimal

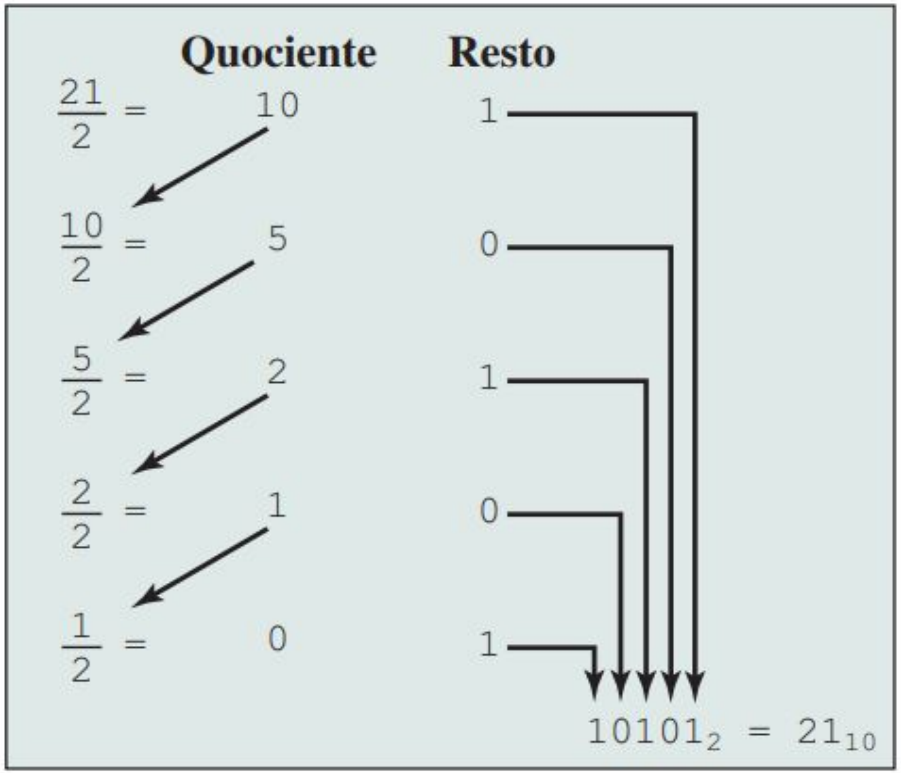


$$\text{Número} = \sum_{i=-k}^{n} d_i \times 10^i$$

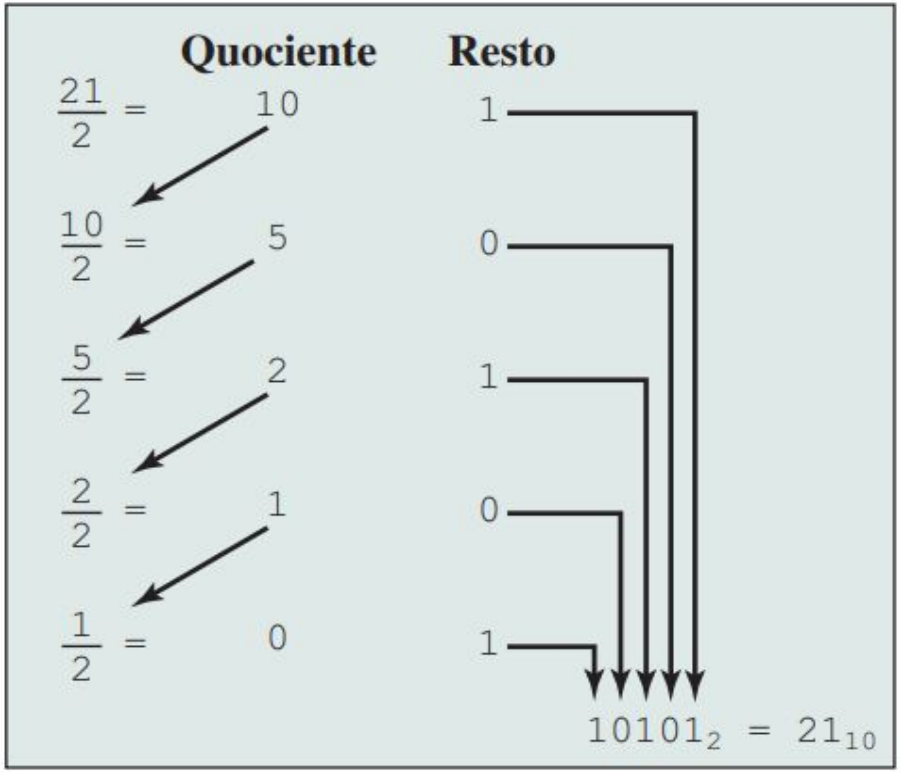
Exercício Conversão entre binário e decimal

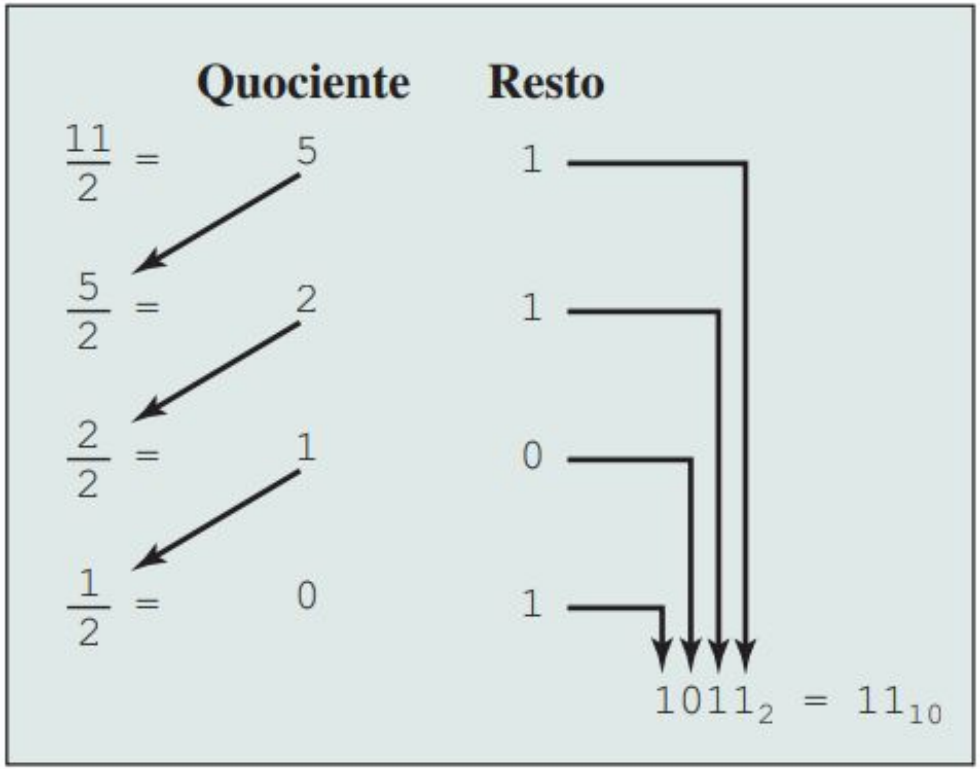
1. Converta 21_{10} para binário usando as divisões sucessivas
2. Converta 11_{10} para binário usando as divisões sucessivas

Conversão entre binário e decimal



Conversão entre binário e decimal



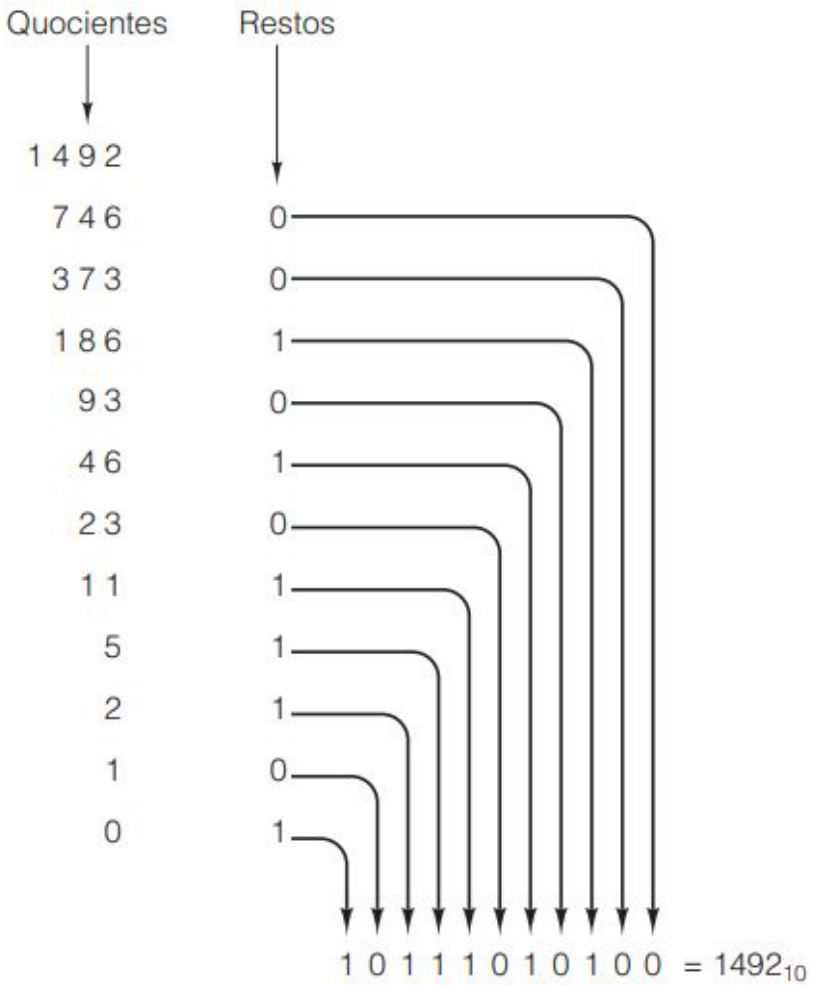


Exercício

Faça a conversão de 1.492_{10} para binário utilizando divisões sucessivas.

Exercício

Faça a conversão de 1.492_{10} para binário utilizando divisões sucessivas.



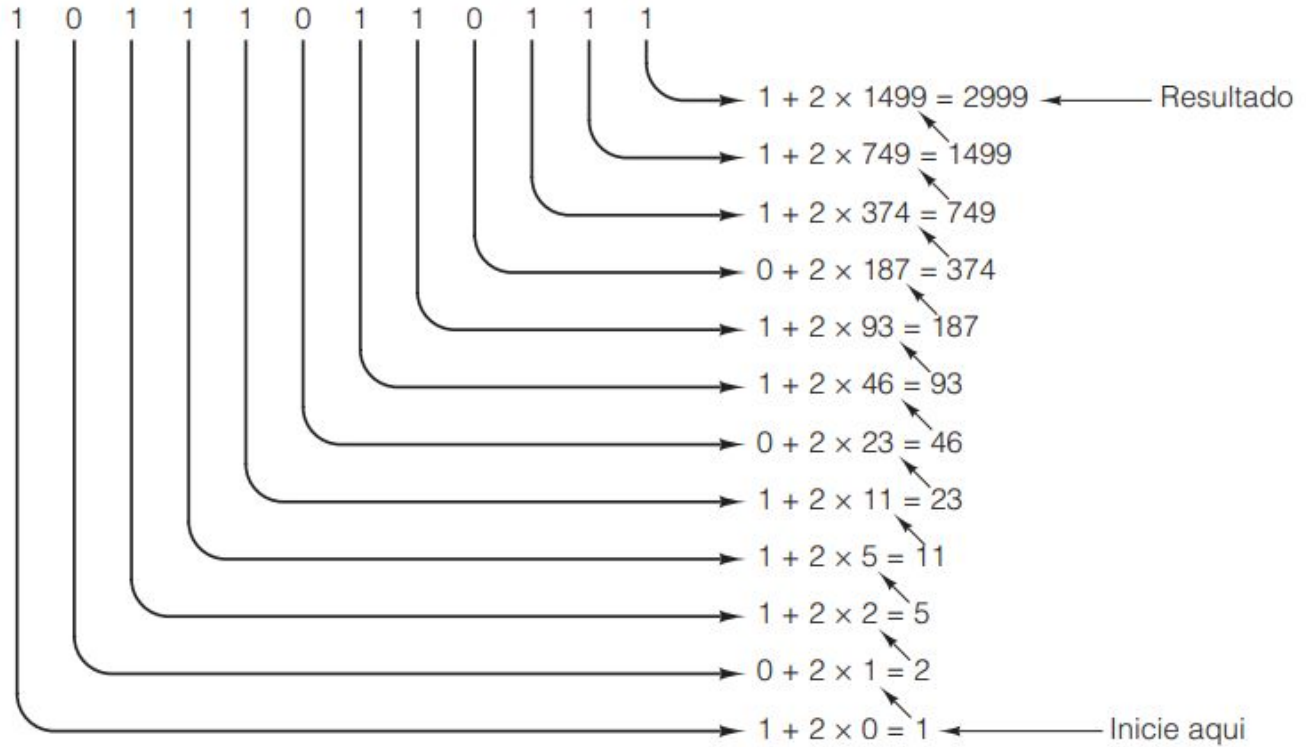
Exercício

Faça a conversão binária do número 101110110111_2 para decimal utilizando multiplicações sucessivas.

Exercício

Faça a conversão binária do número 101110110111_2 para decimal utilizando multiplicações sucessivas.

Você consegue pensar em uma forma mais prática de realizar essa conversão?



Conversão entre binário e decimal - Frações

INSTITUTO FEDERAL
Minas Gerais
Campus Bambuí

Frações

- na notação binária um número com um valor entre 0 e 1 é representado por

$$0,b_{-1}b_{-2}b_{-3}\ldots \qquad b_i = 0 \text{ ou } 1$$

- e tem o valor

$$(b_{-1} \times 2^{-1}) + (b_{-2} \times 2^{-2}) + (b_{-3} \times 2^{-3}) \ldots$$

Conversão entre binário e decimal - Frações

- pode ser reescrito como

$$2^{-1} \times (b_{-1} + 2^{-1} \times (b_{-2} + 2^{-1} \times (b_{-3} + ...) ...))$$

- Essa expressão sugere uma técnica para conversão.
- Suponha que se queira converter um número F(0 < F < 1) da notação **decimal** para a **binária**.
- Sabemos que F_{10} pode ser expresso na forma

$$F = 2^{-1} \times (b_{-1} + 2^{-1} \times (b_{-2} + 2^{-1} \times (b_{-3} + ...) ...))$$

Conversão entre binário e decimal - Frações

- Se multiplicarmos F por 2, obteremos,

$$2 \times F = b_{-1} + 2^{-1} \times (b_{-2} + 2^{-1} \times (b_{-3} + ...) ...)$$

- A partir dessa equação, vimos que a parte inteira de (2 × F), que deve ser 0 ou 1 porque 0 < F < 1, é simplesmente b_{-1}.
- Então, pode-se dizer (2 × F) = $b_{-1} + F_1$, em que $0 < F_1 < 1$ e onde

$$F_1 = 2^{-1} \times (b_{-2} + 2^{-1} \times (b_{-3} + 2^{-1} \times (b_{-4} + ...) ...))$$

Para encontrar b_{-2}, repetimos o processo.

Conversão entre binário e decimal - Frações

- Para encontrar b_{-2}, repetimos o processo.
- O algoritmo de conversão envolve multiplicação por 2.
- Em cada etapa, a parte fracionária do número a partir da etapa anterior é multiplicada por 2.
- O dígito à esquerda da vírgula decimal no produto será 0 ou 1.
- A parte fracionária do produto é usada como o multiplicando na etapa seguinte.
- Exemplo: $0,3_{10} \approx 0,010011\ldots_{2}$

Exemplo

$0,3_{10} \simeq 0,010011\ldots_{2}$

2xF = **0** + F1=0,10011

2xF1 = **1** + F2 = 0,0011

2xF2 = **0** + F3=0,011

2xF3 = **0** + F4=0,11

2xF4 = **1** + F5=0,1

2xF5 = **1** + 0

Exemplo

$0,3_{10} \simeq 0,010011\ldots_2$

2xF = **0** + F1=0,10011

2xF1 = **1** + F2 = 0,0011

2xF2 = **0** + F3=0,011

2xF3 = **0** + F4=0,11

2xF4 = **1** + F5=0,1

2xF5 = **1** + 0

$0,3_{10} \simeq 0,010011\ldots_2$

2x0,3 = **0**,6

2x0,6 = **1**,2

2x0,2 = **0**,4

2x0,4 = **0**,8

2x0,8 = **1**,6

2x0,6 = **1**,2

Conversão entre binário e decimal - Frações

- No sistema decimal, temos dígitos de 0 a 9, e
- cada posição após a vírgula representa potências de 10 negativas que representam os valores
- (1/10, 1/100, 1/1000, etc.).
- No sistema binário, só existem dois dígitos: 0 e 1.
- Cada posição após a vírgula representa as potências de 2 negativas que representam os valores
- (1/2, 1/4, 1/8, 1/16, etc.).

Conversão entre binário e decimal - Frações

- O número decimal 0,1 não tem uma representação finita em binário.
- Sua conversão em binário resulta em uma sequência infinita:

0,0001100110011001...

  - (a parte "0011" se repete infinitamente).

- O número 0,5 é facilmente representado em binário como

0,1,

- pois 1/2 é uma potência de 2.

Conversão entre binário e decimal - Frações

- A representação binária de uma fração decimal só é exata se o denominador da fração irredutível for uma **potência de 2**.
- Caso contrário, será uma **dízima periódica em binário**,
- levando a **aproximações** quando armazenada **em computadores.**

```
0.1 + 0.2 == 0.3  # Retorna False em muitas linguagens!
```

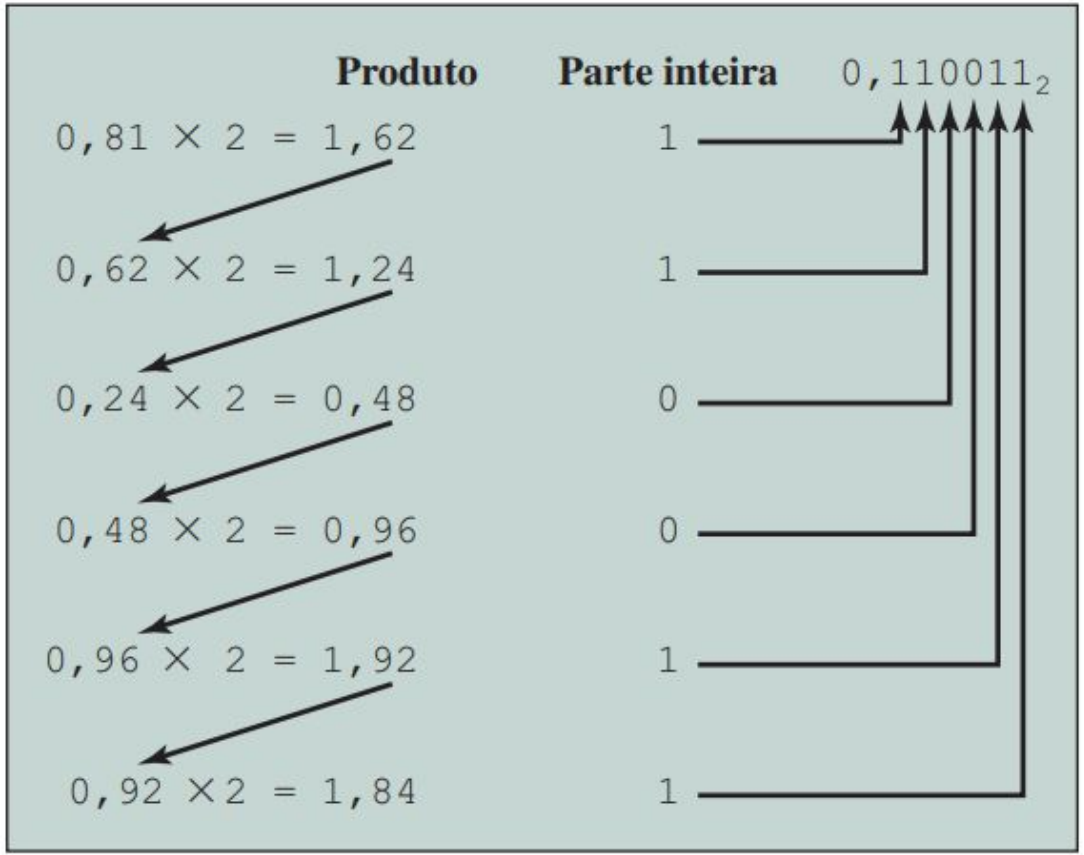
- O resultado é `False` porque `0.1` e `0.2` têm representações binárias aproximadas, e a soma acumula um erro pequeno.

Conversão entre binário e decimal - Frações

INSTITUTO FEDERAL
Minas Gerais
Campus Bambuí

Faça a conversão de $0,81_{10}$ para a forma binária

Conversão entre binário e decimal - Frações



(a) $0,81_{10} = 0,110011_2$ (aproximadamente)

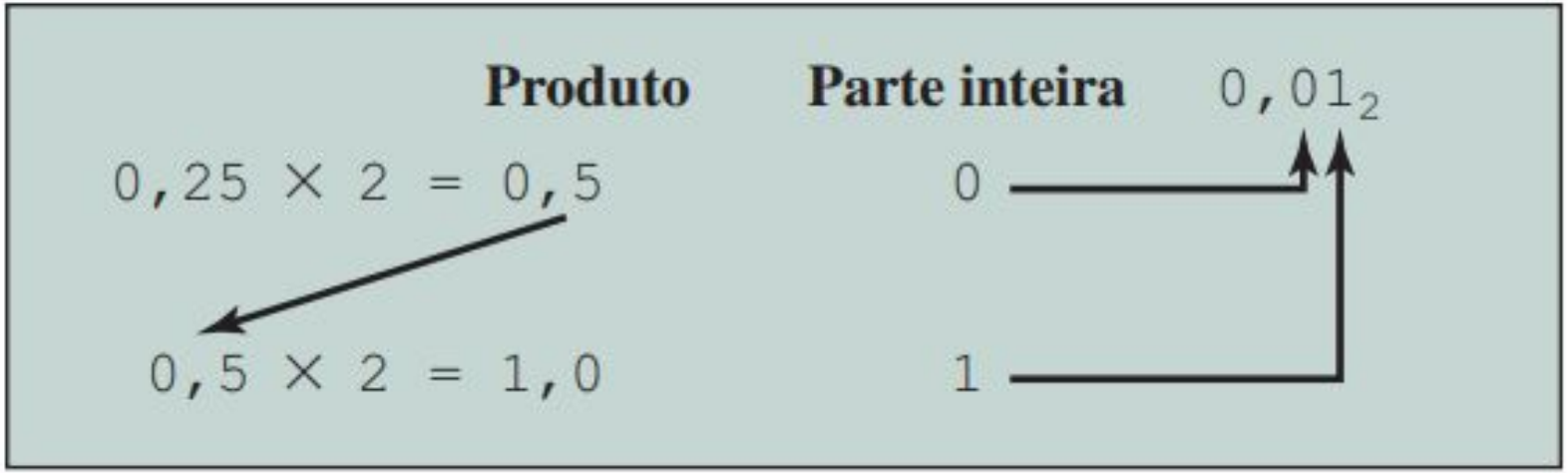
Faça a conversão de $0,81_{10}$ para a forma binária

Conversão entre binário e decimal - Frações

Faça a conversão de $0{,}25_{10}$ para binário

Conversão entre binário e decimal - Frações

Faça a conversão de $0{,}25_{10}$ para binário



(b) $0{,}25_{10} = 0{,}01_2$ (exatamente)

Exercício

Converta o número $0{,}01001101_2$ para decimal.

Exercício

Converta o número $0,01001101_2$ para decimal.

$0x2^{-1}+1x2^{-2}+0x2^{-3}+0x2^{-4}+1x2^{-5}+1x2^{-6}+0x2^{-7}+1x2^{-8}$

$=0x0,5 +1x0,25 + 0x0,125 + 0x0,0625 + 1x0,03125 + 1x0,015625 +0x0,0078125$

$+1x0,00390625 = 0,30078125_{10}$

Base Hexadecimal

INSTITUTO FEDERAL
Minas Gerais
Campus Bambuí

- Notação Hexadecimal
- natureza binária inerente dos componentes de computador digital
- dados dentro dos computadores são representadas por diversos códigos binários
- Necessidade de uma notação mais compacta
- Notação decimal é inadequada devido a dificuldade de conversão entre as bases 2 e 10.
- A notação hexadecimal é adotada por ser mais adequada e compacta.

Base Hexadecimal

As razões para o uso de notação hexadecimal são as seguintes:

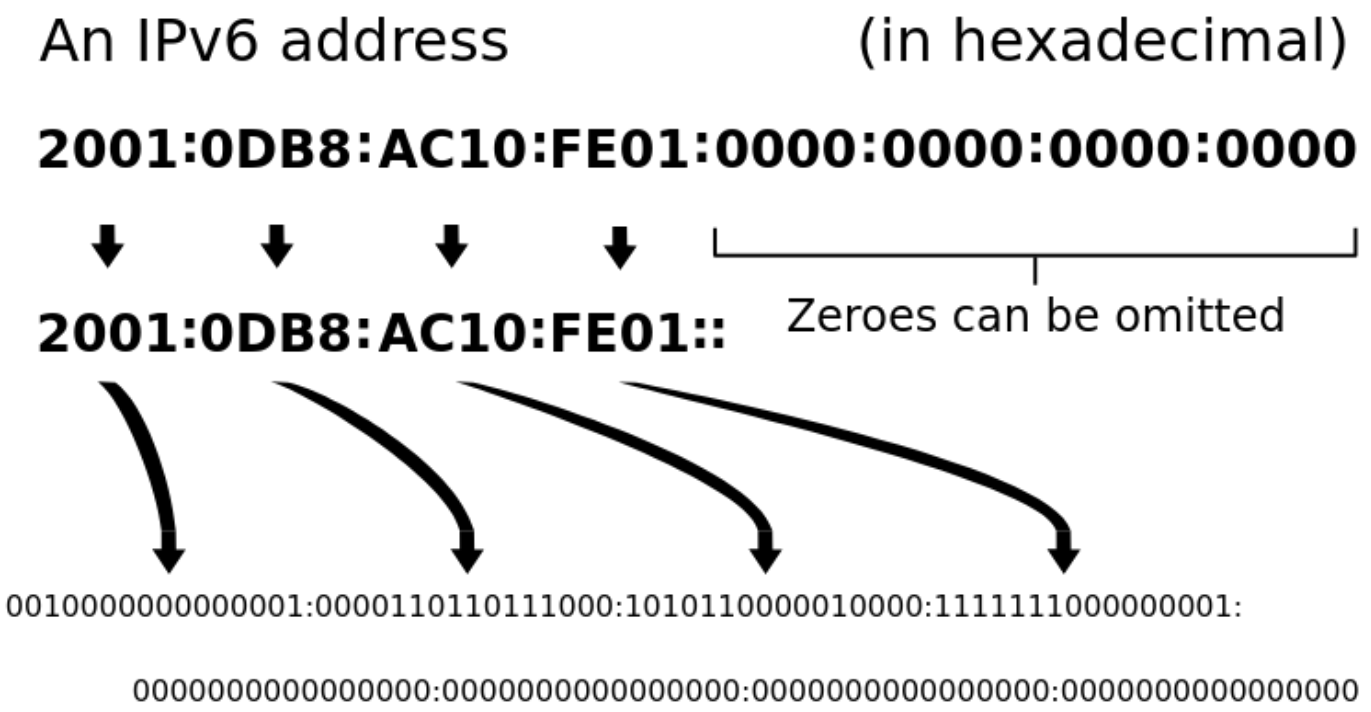
1. É mais compacta que uma notação binária.
2. Na maioria dos computadores, os dados binários ocupam alguns múltiplos de 4 bits
    a. consequentemente, alguns múltiplos de um único dígito hexadecimal.
3. É bem fácil converter entre a notação binária e a hexadecimal.

Base Hexadecimal

Aplicação da base hexadecimal no sistema de cores

Base Hexadecimal

Aplicação da base hexadecimal em redes de computadores.



Base Hexadecimal

- Os dígitos binários são agrupados em conjuntos de quatro bits, chamados de ***nibble***.
- Cada combinação possível de dígitos binários é dada por um símbolo, do seguinte modo:

0000 = 0	0100 = 4	1000 = 8	1100 = C
0001 = 1	0101 = 5	1001 = 9	1101 = D
0010 = 2	0110 = 6	1010 = A	1110 = E
0011 = 3	0111 = 7	1011 = B	1111 = F

A notação é chamada de **hexadecimal** por utilizar 16 símbolos para representação.

Base Hexadecimal

- Uma sequência de dígitos hexadecimais pode ser pensada como uma representação de um inteiro na base 16. Assim,

$$2C_{16} = (2_{16} \times 16^{1}) + (C_{16} \times 16^{0}) =$$

$$(2_{10} \times 16^{1}) + (12_{10} \times 16^{0}) = 44$$

Base Hexadecimal

- Uma sequência de dígitos hexadecimais pode ser pensada como uma representação de um inteiro na base 16. Assim,

$$2C_{16} = (2_{16} \times 16^{1}) + (C_{16} \times 16^{0}) =$$

$$(2_{10} \times 16^{1}) + (12_{10} \times 16^{0}) = 44$$

- os números hexadecimais em sistema numérico posicional com base 16, são

$$Z = \sum_{i} (h_i \times 16^{i})$$

- 16 é a base, o dígito hexadecimal h_i está em um intervalo decimal de $0 \leq h_i \leq 15$.
- equivalente ao intervalo hexadecimal de $0 \leq h_i \leq F$.

Base Hexadecimal

Passo a passo:

1. Divida o número decimal por 16.
2. Anote o resto (que será um dígito hexadecimal).
3. Repita o processo com o quociente até que ele seja 0.
4. O número hexadecimal é a sequência dos restos em ordem inversa.

Base Hexadecimal

1. Exemplo: Converter 255 para hexadecimal

Divisão	Quociente	Resto	Dígito Hex
255 ÷ 16	15	15	F
15 ÷ 16	0	15	F

2. Resultado: FF (lê-se de baixo para cima).

Base Hexadecimal

- Considere a cadeia binária

  110111100001.

- Ela é equivalente a

  1101 1110 0001 = $DE1_{16}$

  D E 1

Decimal (base 10)	Binário (base 2)	Hexadecimal (base 16)
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	0001 0000	10
17	0001 0001	11
18	0001 0010	12
31	0001 1111	1F
100	0110 0100	64

Exercício

INSTITUTO FEDERAL
Minas Gerais
Campus Bambuí

1. Faça a conversão do número $0001\ 0000\ 0000_2$ para a notação hexadecimal.

Exercício

1. Faça a conversão do número $0001\ 0000\ 0000_2$ para a notação hexadecimal.

   0001 = 1
   0000 = 0
   0000 = 0
   = 100

Exercício

2. Faça a conversão do número 256_{10} para a notação hexadecimal.

Exercício

2. Faça a conversão do número 256_{10} para a notação hexadecimal.

Divisão	Quociente	Resto	Dígito Hex
256 ÷ 16	16	0	0
16 ÷ 16	1	0	0
1 ÷ 16	0	1	1

Números negativos

INSTITUTO FEDERAL

4 formas básicas de representar número binário negativo.

1. Sinal magnitude
2. Complemento de 1
3. Complemento de 2
4. excesso 128

N decimal	N binaria	–N magnitude com sinal	–N complemento de 1	–N complemento de 2	–N excesso 128
1	00000001	10000001	11111110	11111111	01111111
2	00000010	10000010	11111101	11111110	01111110
3	00000011	10000011	11111100	11111101	01111101
4	00000100	10000100	11111011	11111100	01111100
5	00000101	10000101	11111010	11111011	01111011
6	00000110	10000110	11111001	11111010	01111010
7	00000111	10000111	11111000	11111001	01111001
8	00001000	10001000	11110111	11111000	01111000
9	00001001	10001001	11110110	11110111	01110111
10	00001010	10001010	11110101	11110110	01110110
20	00010100	10010100	11101011	11101100	01101100
30	00011110	10011110	11100001	11100010	01100010
40	00101000	10101000	11010111	11011000	01011000
50	00110010	10110010	11001101	11001110	01001110
60	00111100	10111100	11000011	11000100	01000100
70	01000110	11000110	10111001	10111010	00111010
80	01010000	11010000	10101111	10110000	00110000
90	01011010	11011010	10100101	10100110	00100110
100	01100100	11100100	10011011	10011100	00011100
127	01111111	11111111	10000000	10000001	00000001
128	Não existe	Não existe	Não existe	10000000	00000000

Sinal magnitude

- o bit da extrema esquerda é o bit de sinal (0 é + e 1 é –)
- os restantes contêm a magnitude absoluta do número

- 00000110 = (+6)
- 10000110 = (-6)

N decimal	N binaria	–N magnitude com sinal
1	00000001	10000001
2	00000010	10000010
3	00000011	10000011
4	00000100	10000100
5	00000101	10000101
6	00000110	10000110
7	00000111	10000111
8	00001000	10001000
9	00001001	10001001
10	00001010	10001010
20	00010100	10010100
30	00011110	10011110
40	00101000	10101000
50	00110010	10110010
60	00111100	10111100
70	01000110	11000110
80	01010000	11010000
90	01011010	11011010
100	01100100	11100100
127	01111111	11111111
128	Não existe	Não existe

Complemento de 1

- também tem um bit de sinal, que é 0 para mais e 1 para menos.
- Para tornar um número negativo, substitua cada 1 por 0 e cada 0 por 1.
- Isso vale também para o bit de sinal.
- O complemento de 1 é obsoleto.

1. 00000110 = (+6)

inverte os 1s e os 0s

2. 11111001 = (–6 em complemento de um)

N decimal	N binaria	–N complemento de 1
1	00000001	11111110
2	00000010	11111101
3	00000011	11111100
4	00000100	11111011
5	00000101	11111010
6	00000110	11111001
7	00000111	11111000
8	00001000	11110111
9	00001001	11110110
10	00001010	11110101
20	00010100	11101011
30	00011110	11100001
40	00101000	11010111
50	00110010	11001101
60	00111100	11000011
70	01000110	10111001
80	01010000	10101111
90	01011010	10100101
100	01100100	10011011
127	01111111	10000000
128	Não existe	Não existe

Complemento de 1

- Também tem um bit de sinal que é 0 para mais e 1 para menos.
- Negar um número é um processo em duas etapas.
- Na primeira, cada 1 é substituído por um 0 e cada 0 por um 1, assim como no complemento de um.
- Na segunda, 1 é somado ao resultado.

1. 00000110 = (+6)
2. inverte os 0s e 1s = 11111001 (–6 em complemento de um)
3. adiciona 1 ao valor invertido = 11111010 (–6 em complemento de dois)

N decimal	N binaria	–N complemento de 2
1	00000001	11111111
2	00000010	11111110
3	00000011	11111101
4	00000100	11111100
5	00000101	11111011
6	00000110	11111010
7	00000111	11111001
8	00001000	11111000
9	00001001	11110111
10	00001010	11110110
20	00010100	11101100
30	00011110	11100010
40	00101000	11011000
50	00110010	11001110
60	00111100	11000100
70	01000110	10111010
80	01010000	10110000
90	01011010	10100110
100	01100100	10011100
127	01111111	10000001
128	Não existe	10000000

Excesso 128

- O quarto sistema é chamado excesso 2^{m-1} para números de m bits.
- representa um número armazenando-o como a soma dele mesmo com 2^{m-1}.
- Para números de 8 bits, m = 8, o sistema é denominado excesso 128.
- Um número é armazenado como seu verdadeiro valor mais 128.

N decimal	N binaria	–N complemento de 2
1	00000001	11111111
2	00000010	11111110
3	00000011	11111101
4	00000100	11111100
5	00000101	11111011
6	00000110	11111010
7	00000111	11111001
8	00001000	11111000
9	00001001	11110111
10	00001010	11110110
20	00010100	11101100
30	00011110	11100010
40	00101000	11011000
50	00110010	11001110
60	00111100	11000100
70	01000110	10111010
80	01010000	10110000
90	01011010	10100110
100	01100100	10011100
127	01111111	10000001
128	Não existe	10000000

Excesso 128

–3 se torna –3 + 128 = 125

–3 é representado pelo número binário de 8 bits para 125 (01111101).

Os números de –128 a +127 mapeiam para 0 a 255.

Os quais podem ser expressos como um inteiro positivo de 8 bits.

Esse sistema é idêntico ao complemento de dois com o bit de sinal invertido.

N decimal	N binaria	–N complemento de 2	–N excesso 128
1	00000001	11111111	01111111
2	00000010	11111110	01111110
3	00000011	11111101	01111101
4	00000100	11111100	01111100
5	00000101	11111011	01111011
6	00000110	11111010	01111010
7	00000111	11111001	01111001
8	00001000	11111000	01111000
9	00001001	11110111	01110111
10	00001010	11110110	01110110
20	00010100	11101100	01101100
30	00011110	11100010	01100010
40	00101000	11011000	01011000
50	00110010	11001110	01001110
60	00111100	11000100	01000100
70	01000110	10111010	00111010
80	01010000	10110000	00110000
90	01011010	10100110	00100110
100	01100100	10011100	00011100
127	01111111	10000001	00000001
128	Não existe	10000000	00000000

Observações

1. Magnitude com sinal e complemento de um, têm duas representações para zero: mais zero e menos zero.
   a. Essa situação é indesejável.
2. Em complemento de dois, o padrão de bit que consiste em 1 seguido por 0s é seu próprio complemento.
3. As faixas de números positivos e negativos ficam não simétricas;
4. Há um número negativo sem nenhuma contraparte positiva.

Observações

Queremos um sistema de codificação com duas propriedades:

1. Somente uma representação para zero.

2. Exatamente a mesma quantidade de números positivos e negativos.

- Qualquer conjunto de números com a mesma quantidade de números positivos e números negativos e só um zero tem um número ímpar de membros
- m bits permite um número par de padrões de bits.
- Sempre haverá um padrão de bits a mais ou um padrão de bits a menos,
- Não importando qual representação seja escolhida.

Bibliografia

- MONTEIRO, M. A. Introdução à Organização de Computadores. 5ª Edição. Rio de Janeiro: LTC Editora, 2007.
- STALLINGS, W. Arquitetura e Organização de Computadores. 5ª. Edição. São Paulo: Prentice Hall, 2002.
- TANENBAUM, A. Organização estruturada de computadores. 5ª Edição. São Paulo: Pearson Prentice Hall, 2011.