Laboratório de Informática

Leopoldo Teixeira
leo@leopoldomt.com

Trabalho

- Prepare um resumo com o restante da história da computação de 1993 aos dias atuais, colocando a sua opinião.

- Prepare um arquivo PDF nomeado com seu nome e sobrenome (NomeSobrenome.pdf) e envie por e-mail

- **Entrega: 8/5 (hoje!)**

Bits

- Abreviação de *binary digits*
- Codificam informação nos computadores com padrões de 0s e 1s
- Geralmente associados com valores numéricos
- Símbolos cujo significado depende da aplicação
  - valores numéricos, caracteres, imagens, sons…

Operações Booleanas

- George Boole (1815-1864)
- É conveniente pensar que o bit 0 representa o valor *false* e o bit 1 representa o valor *true*
  - permite visualizar a manipulação de bits como manipulação de valores *true*/*false*
- Operações que manipulam estes tipos de valores são chamadas operações booleanas

Álgebra Booleana

- Lógica proposicional: sistema para formalizar argumentos

- Uma proposição é uma declaração que pode ser ***TRUE*** ou ***FALSE***

- Podemos combinar proposições por meio de operadores, exemplos?

Operações

```
    0
AND 0
-----
    0
```

```
    0
AND 1
-----
    0
```

```
    1
AND 0
-----
    0
```

```
    1
AND 1
-----
    1
```

Operações

```
    0
AND 0
-----
    0
```

```
    0
AND 1
-----
    0
```

```
    1
AND 0
-----
    0
```

```
    1
AND 1
-----
    1
```

```
   0
OR 0
----
   0
```

```
   0
OR 1
----
   1
```

```
   1
OR 0
----
   1
```

```
   1
OR 1
----
   1
```

Operações

```
    0
AND 0
-----
    0
```

```
    0
AND 1
-----
    0
```

```
    1
AND 0
-----
    0
```

```
    1
AND 1
-----
    1
```

```
    0
OR  0
-----
    0
```

```
    0
OR  1
-----
    1
```

```
    1
OR  0
-----
    1
```

```
    1
OR  1
-----
    1
```

```
    0
XOR 0
-----
    0
```

```
    0
XOR 1
-----
    1
```

```
    1
XOR 0
-----
    1
```

```
    1
XOR 1
-----
    0
```

Operações

```
    0
AND 0
-----
    0
```

```
    0
AND 1
-----
    0
```

```
    1
AND 0
-----
    0
```

```
    1
AND 1
-----
    1
```

```
    0
OR  0
-----
    0
```

```
    0
OR  1
-----
    1
```

```
    1
OR  0
-----
    1
```

```
    1
OR  1
-----
    1
```

```
    0
XOR 0
-----
    0
```

```
    0
XOR 1
-----
    1
```

```
    1
XOR 0
-----
    1
```

```
    1
XOR 1
-----
    0
```

```
NOT 0
-----
    1
```

```
NOT 1
-----
    0
```

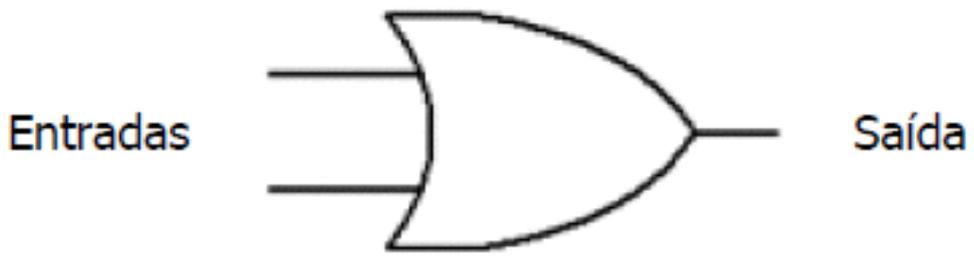
Portas Lógicas (*gates*)

- Um dispositivo que produz a saída de uma operação booleana ao receber valores de entrada é chamado de *gate* (porta lógica)

- Implementados como pequenos circuitos eletrônicos em que bits (0 e 1) são representados como níveis de tensão

- Cada operação booleana tem uma porta lógica representada simbolicamente de forma distinta

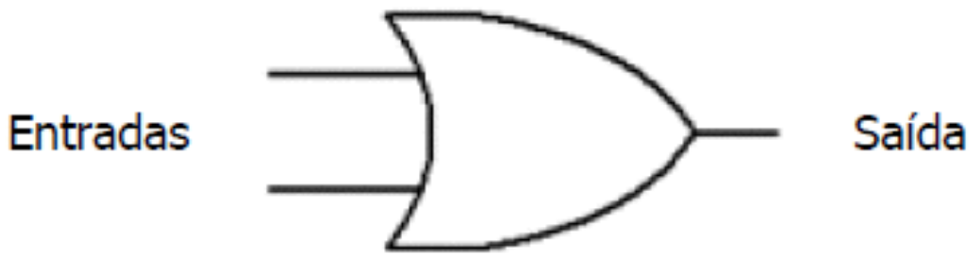
Portas Lógicas (*gates*)



Entradas		Saída
0	0	0
0	1	1
1	0	1
1	1	1

Portas Lógicas (*gates*)

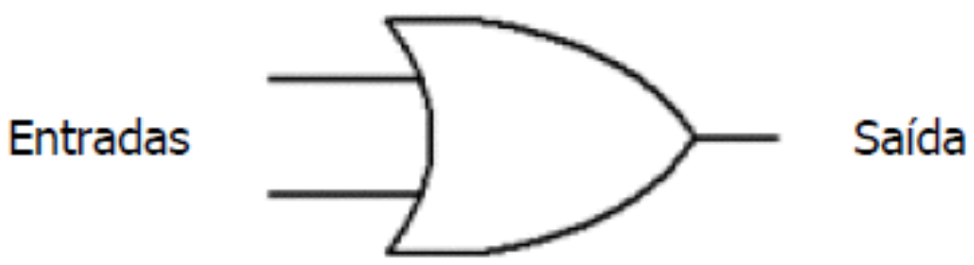
OR

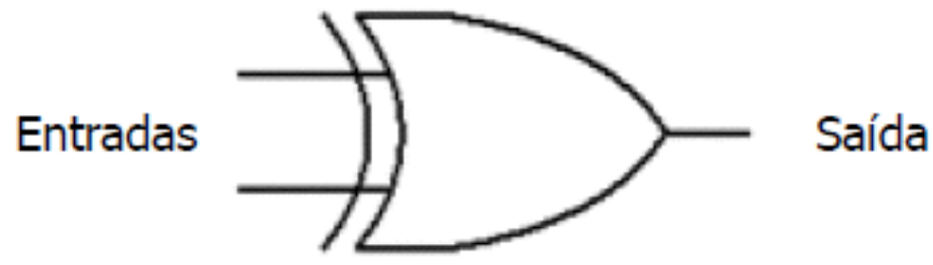


Entradas		Saída
0	0	0
0	1	1
1	0	1
1	1	1

Portas Lógicas (*gates*)

OR



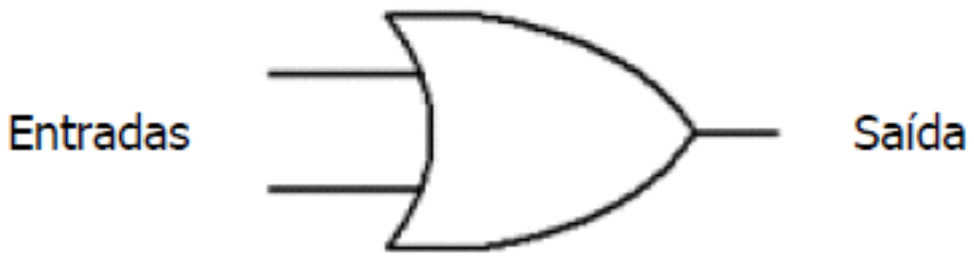


Entradas		Saída
0	0	0
0	1	1
1	0	1
1	1	1

Entradas		Saída
0	0	0
0	1	1
1	0	1
1	1	0

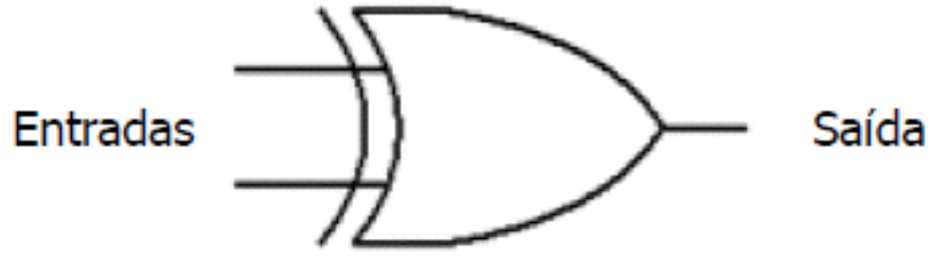
Portas Lógicas (*gates*)

OR



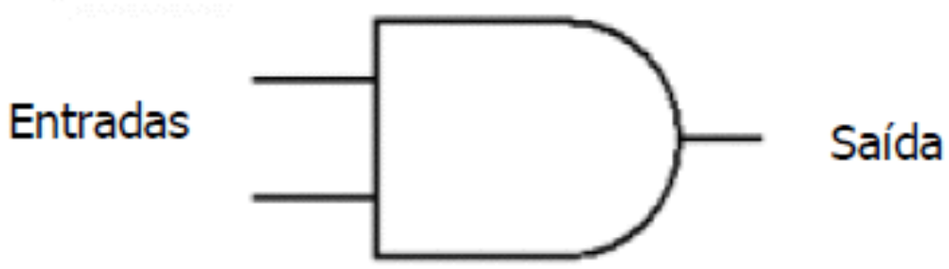
Entradas		Saída
0	0	0
0	1	1
1	0	1
1	1	1

XOR



Entradas		Saída
0	0	0
0	1	1
1	0	1
1	1	0

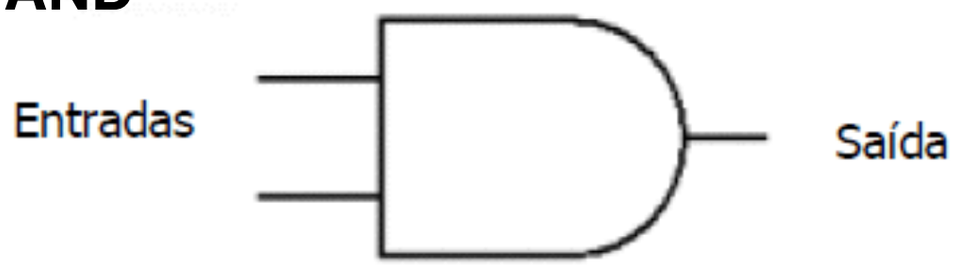
Portas Lógicas (*gates*)



Entradas		Saída
0	0	0
0	1	0
1	0	0
1	1	1

Portas Lógicas (*gates*)

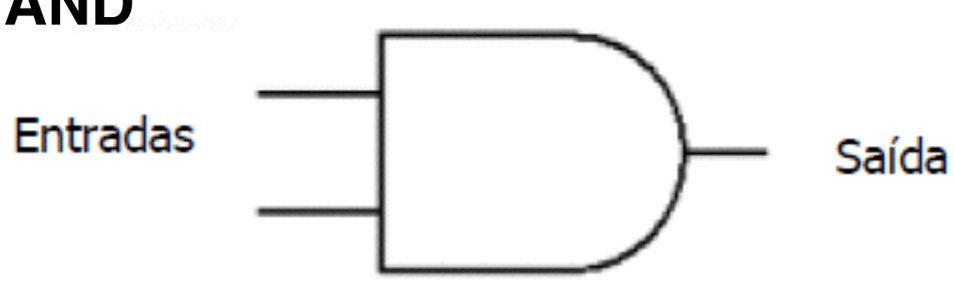
AND



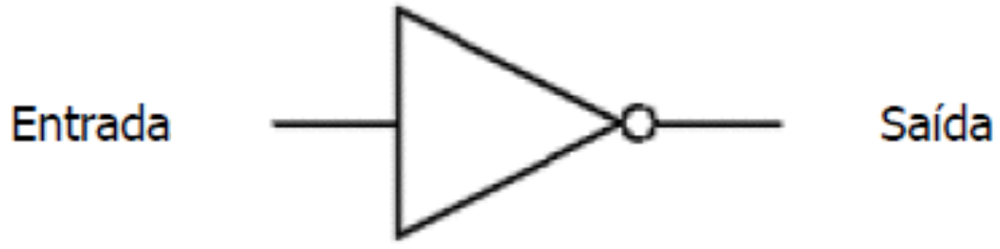
Entradas		Saída
0	0	0
0	1	0
1	0	0
1	1	1

Portas Lógicas (*gates*)

AND



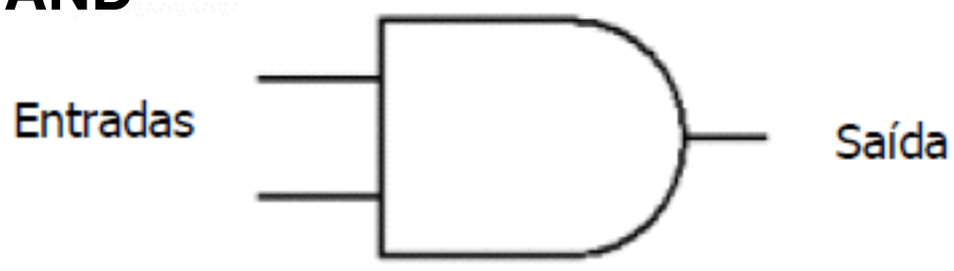
Entradas		Saída
0	0	0
0	1	0
1	0	0
1	1	1



Entradas	Saída
0	1
1	0

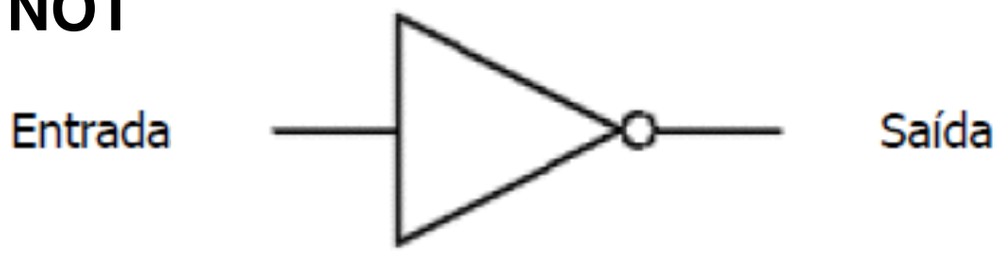
Portas Lógicas (*gates*)

AND



Entradas		Saída
0	0	0
0	1	0
1	0	0
1	1	1

NOT



Entradas	Saída
0	1
1	0

Portas Lógicas - Exemplo

Adaptado dos slides do Prof. Cleber Zanchettin (CIn-UFPE)

Portas Lógicas - Exemplo

- Podemos pensar em uma proposição para decidir se devemos levar o guarda-chuva ou não.

Portas Lógicas - Exemplo

- Podemos pensar em uma proposição para decidir se devemos levar o guarda-chuva ou não.

- Exemplo?

Portas Lógicas - Exemplo

- Podemos pensar em uma proposição para decidir se devemos levar o guarda-chuva ou não.

- Exemplo?
  - Está chovendo **OR** previsão do tempo indica chuva

Portas Lógicas - Exemplo

- Podemos pensar em uma proposição para decidir se devemos levar o guarda-chuva ou não.

- Exemplo?
  - Está chovendo **OR** previsão do tempo indica chuva
  - Se estiver chovendo **ou** a previsão do tempo indicar chuva, então levarei o guarda-chuva

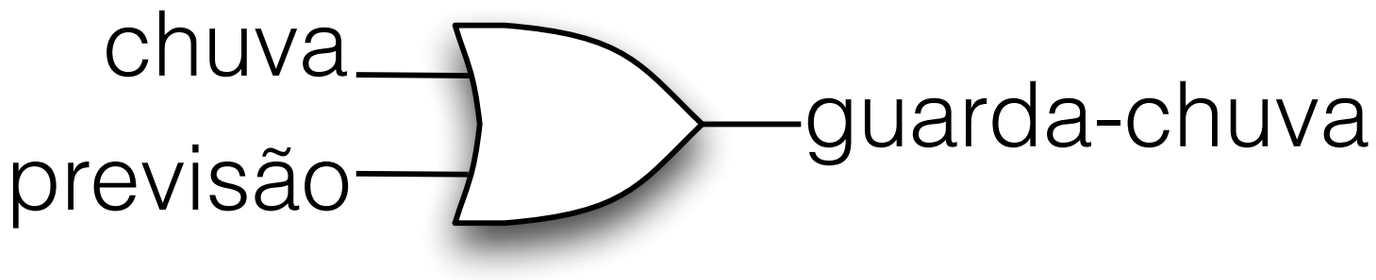
Portas Lógicas - Exemplo

- Podemos pensar em uma proposição para decidir se devemos levar o guarda-chuva ou não.

- Exemplo?

  - Está chovendo **OR** previsão do tempo indica chuva

  - Se estiver chovendo **ou** a previsão do tempo indicar chuva, então levarei o guarda-chuva

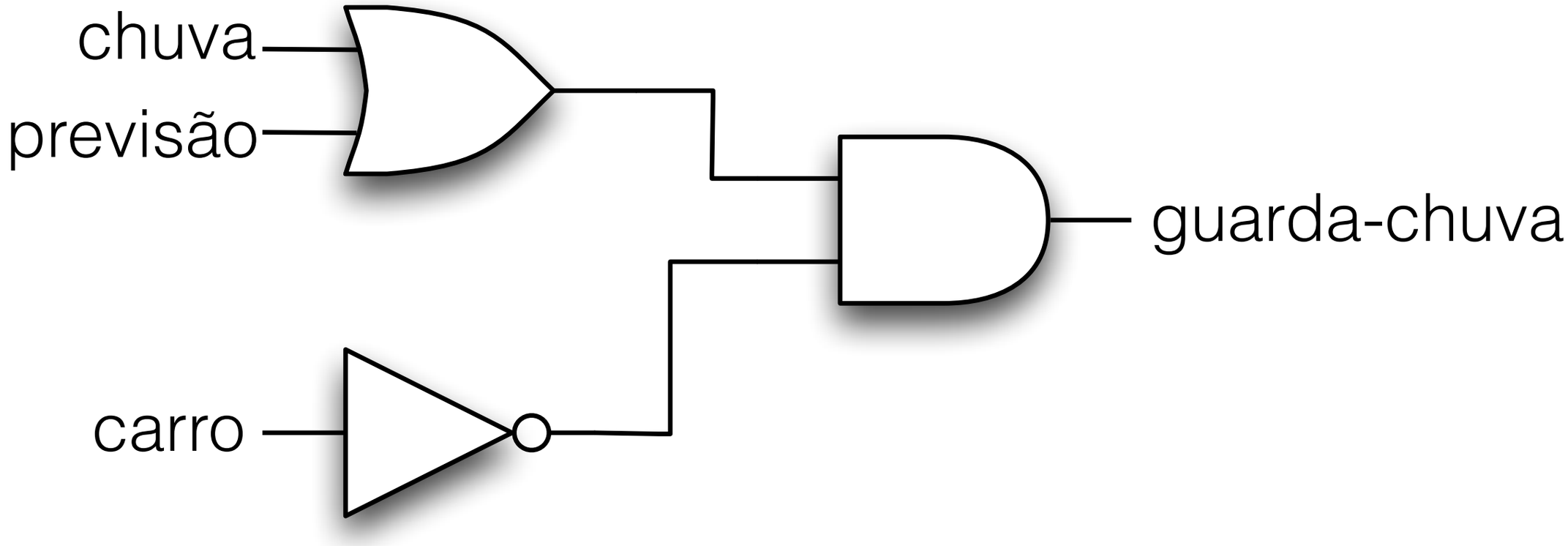


Adaptado dos slides do Prof. Cleber Zanchettin (CIn-UFPE)

Flip-Flop

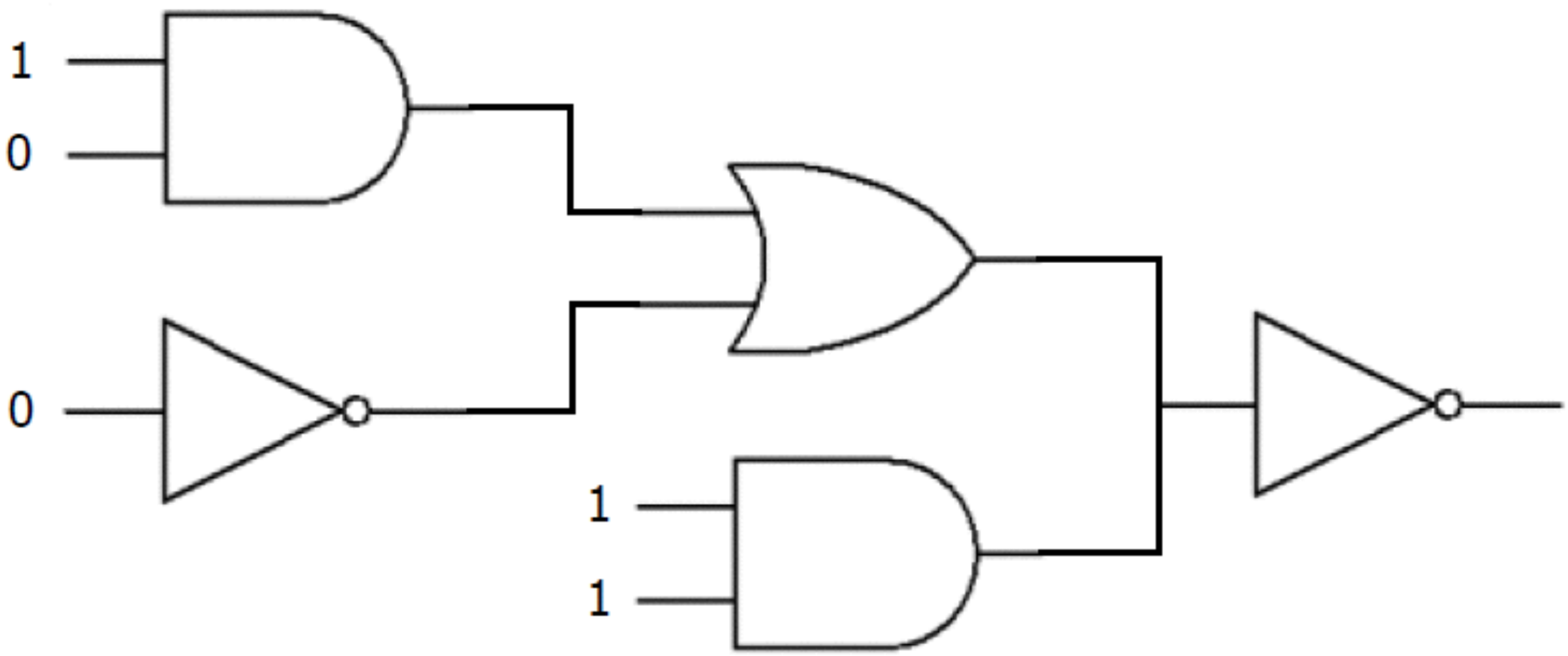
- Coleção de circuitos que produz valor de saída 0 ou 1, que permanece constante até que um pulso de outro circuito causa a mudança do valor
- A saída vai alternar entre dois valores (flip-flop) dependendo do estímulo externo
- Ideal para armazenamento de um bit no interior de um computador
- Circuito simples que é usado para construir circuitos complexos

Flip-Flop - Exemplo



Adaptado dos slides do Prof. Cleber Zanchettin (CIn-UFPE)

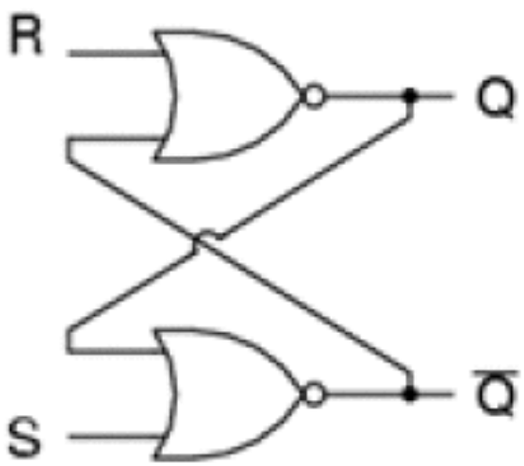
Flip-Flop - Exemplo

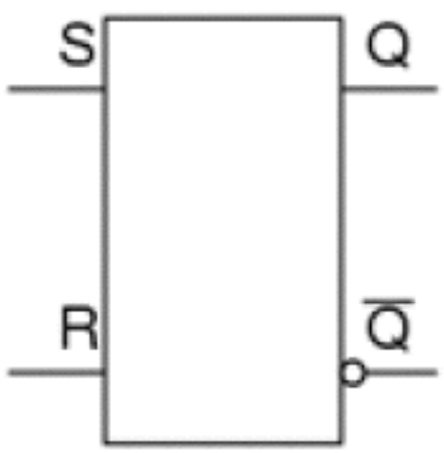


Qual a saída deste circuito, para as entradas observadas?

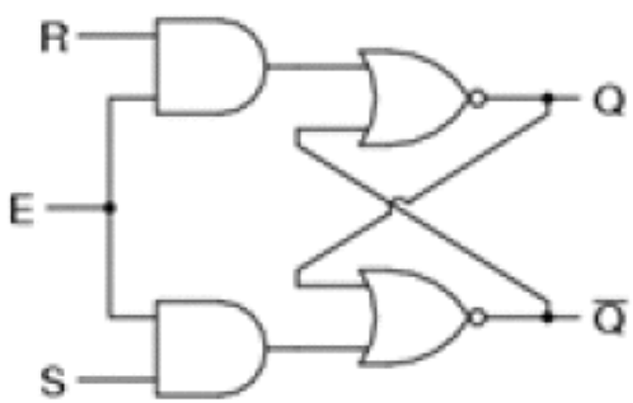
Circuito bi-estável

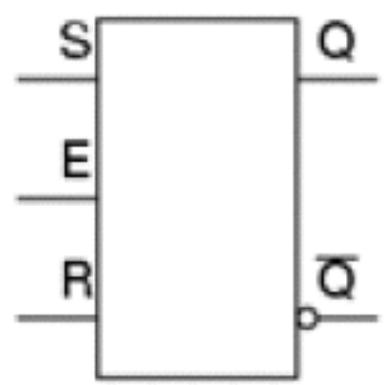
R-S flip-flop



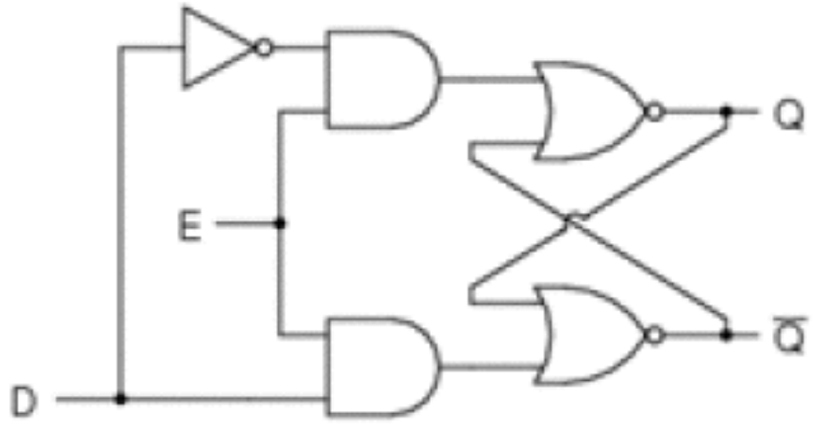


R-S enable flip-flop

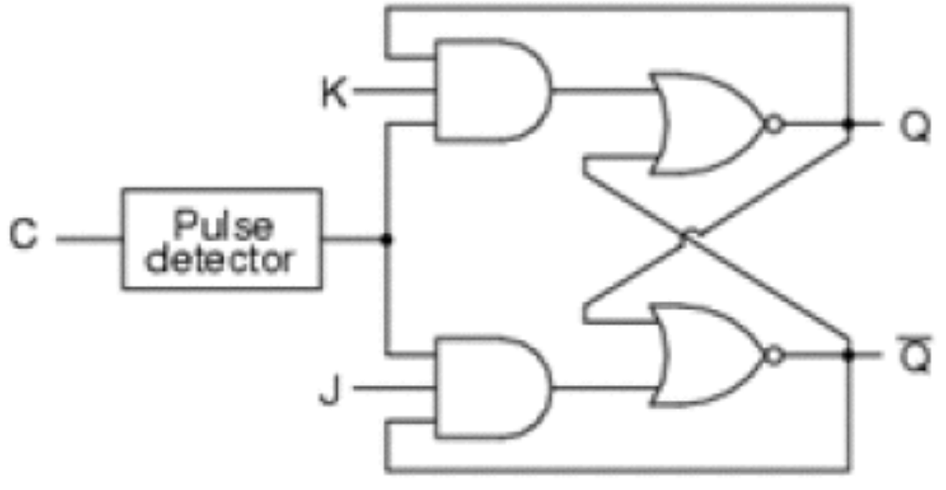




D flip-flop



J-K flip-flop



Adaptado dos slides do Prof. Cleber Zanchettin (CIn-UFPE)

Notação Hexadecimal

- Um computador lida com padrões de bits, que geralmente são longos e denominados de cadeias (*streams*) de bits
- Estas cadeias são complicadas para entendermos
- O que significa 101101010011?
- Para simplificar a representação, usamos a notação **hexadecimal**

Notação Hexadecimal

- Aproveita o fato de que padrões de bits geralmente são múltiplos de quatro
- 1011 0101 0011
  - B53
- 1010 0100 1100 1000
  - A4C8

Decimal	Binário	Hexadecimal
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

Memória Principal

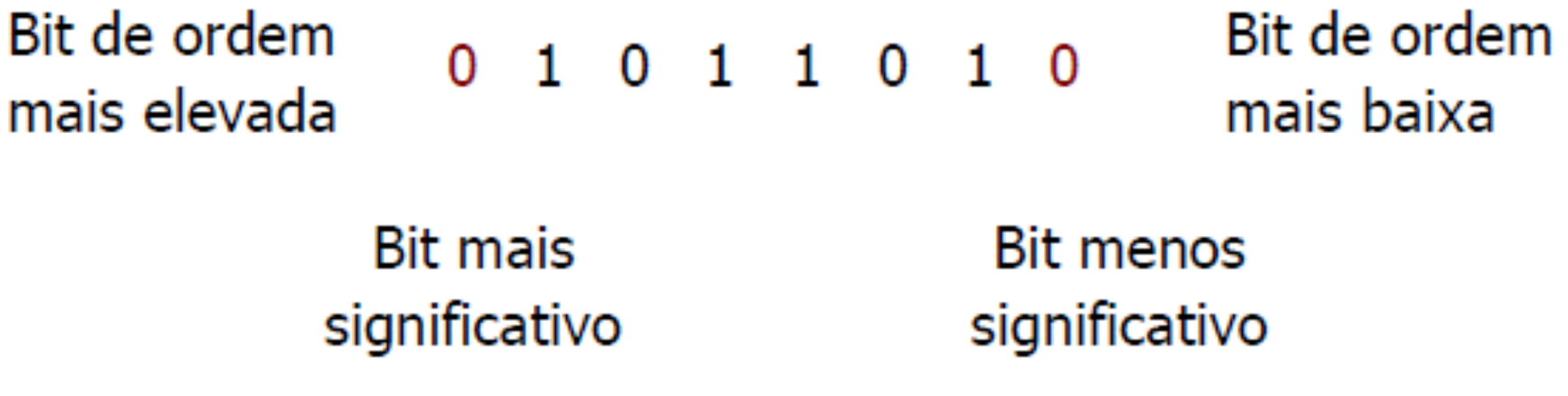
- Para armazenar dados, um computador contém uma grande coleção de circuitos (como flip-flops) capazes de armazenar bits (memória)

- A memória principal de um computador é organizada em unidades manipuláveis denominadas células

- O tamanho típico de uma célula é de 8 bits

- Uma palavra de 8 bits é chamada de *byte*

Representação

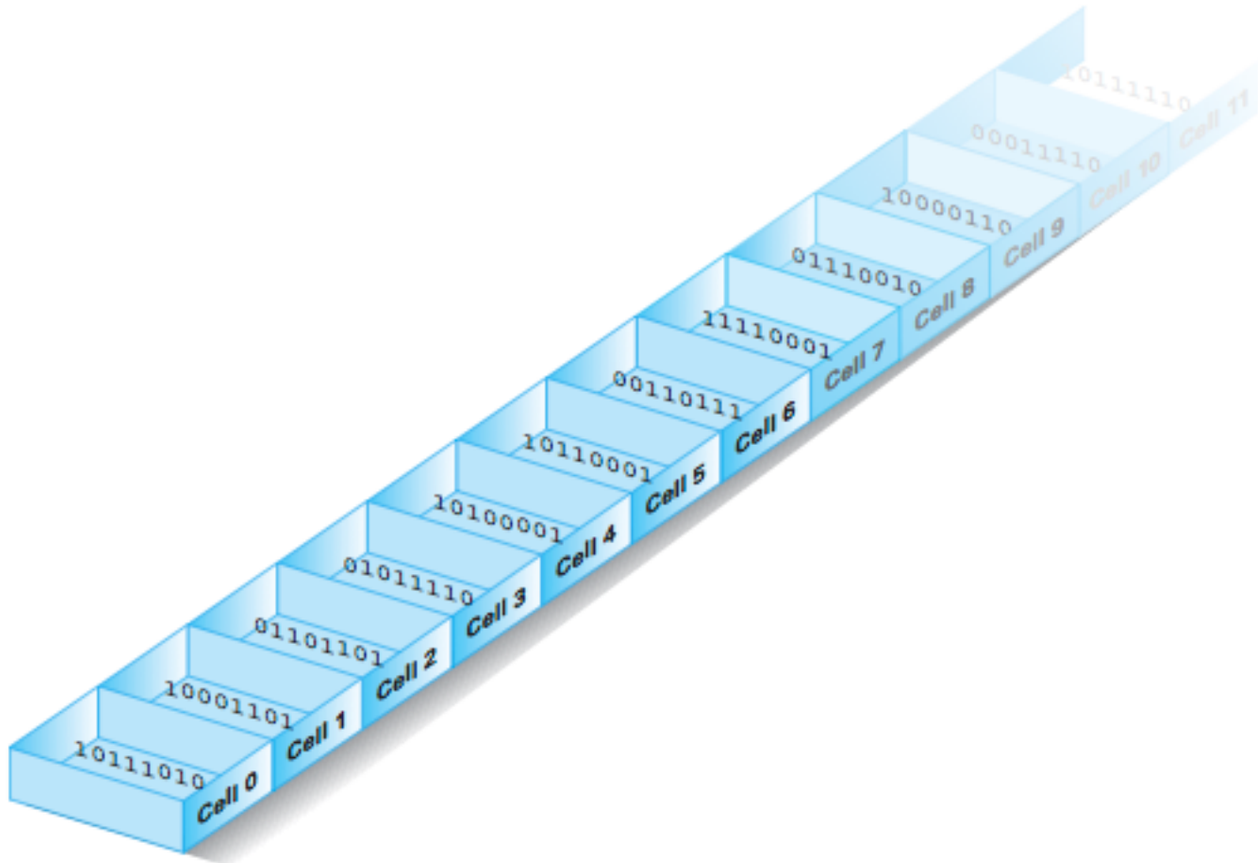


- Normalmente visualizamos os bits em uma célula de memória como organizados em uma linha

- Para identificar células individuais, são assinalados nomes únicos (endereços)

Organização de Endereços

- Endereços são ordenados
  - próxima célula, anterior…
- Padrões de bits maiores que um byte podem ser armazenados em células consecutivas
- Combinado com os circuitos que permitem ler e escrever dados nas células



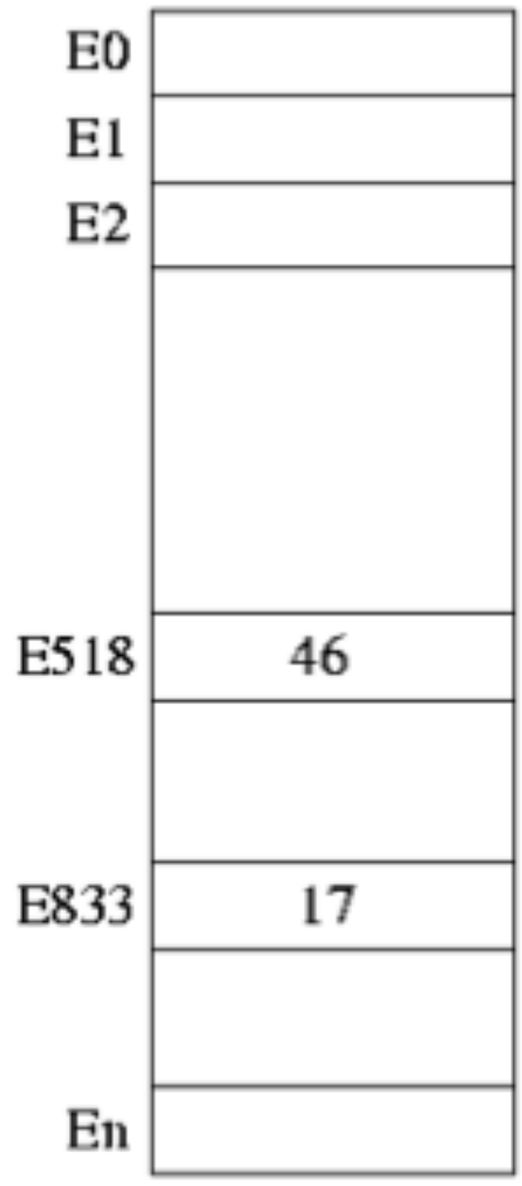
Organização de Endereços

- Por organizarmos a memória em termos de células individualmente endereçadas, podemos acessá-las independentemente, em qualquer ordem

- A memória do computador é comumente chamada de memória de acesso aleatório (*Random Access Memory* - RAM)

- Nos computadores de hoje em dia, a RAM é construída com tecnologias que garantem miniaturização e tempo de resposta rápido

  - a tecnologia usada armazena bits como pequenas cargas elétricas que dissipam rapidamente



Capacidade de Memória

- É conveniente projetar sistemas de memória onde o número total de células é uma potência de 2

- Portanto, o tamanho das memórias nos computadores iniciais era medido em 1024 (2^{10}) unidades de células

- Por ser próximo a 1000, passou-se a adotar o prefixo *kilo* em referência a esta unidade

  - 1 *kilobyte* == 1024 *bytes*

- *kilo, mega*, etc são diferentes do contexto de outras medidas

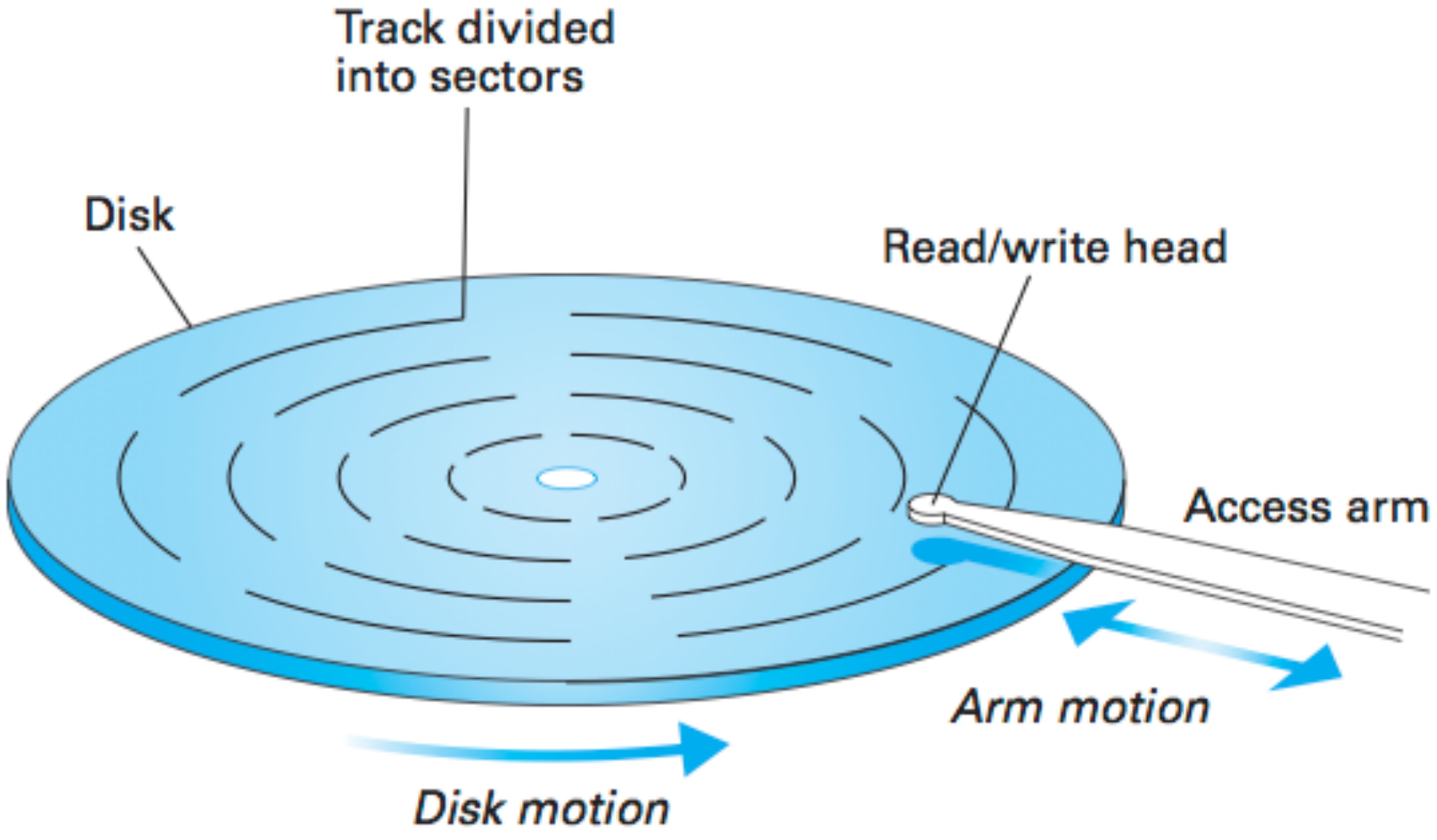
Armazenamento em Massa

- Dada a volatilidade e tamanho limitado da memória principal, muitos computadores tem dispositivos adicionais de memória

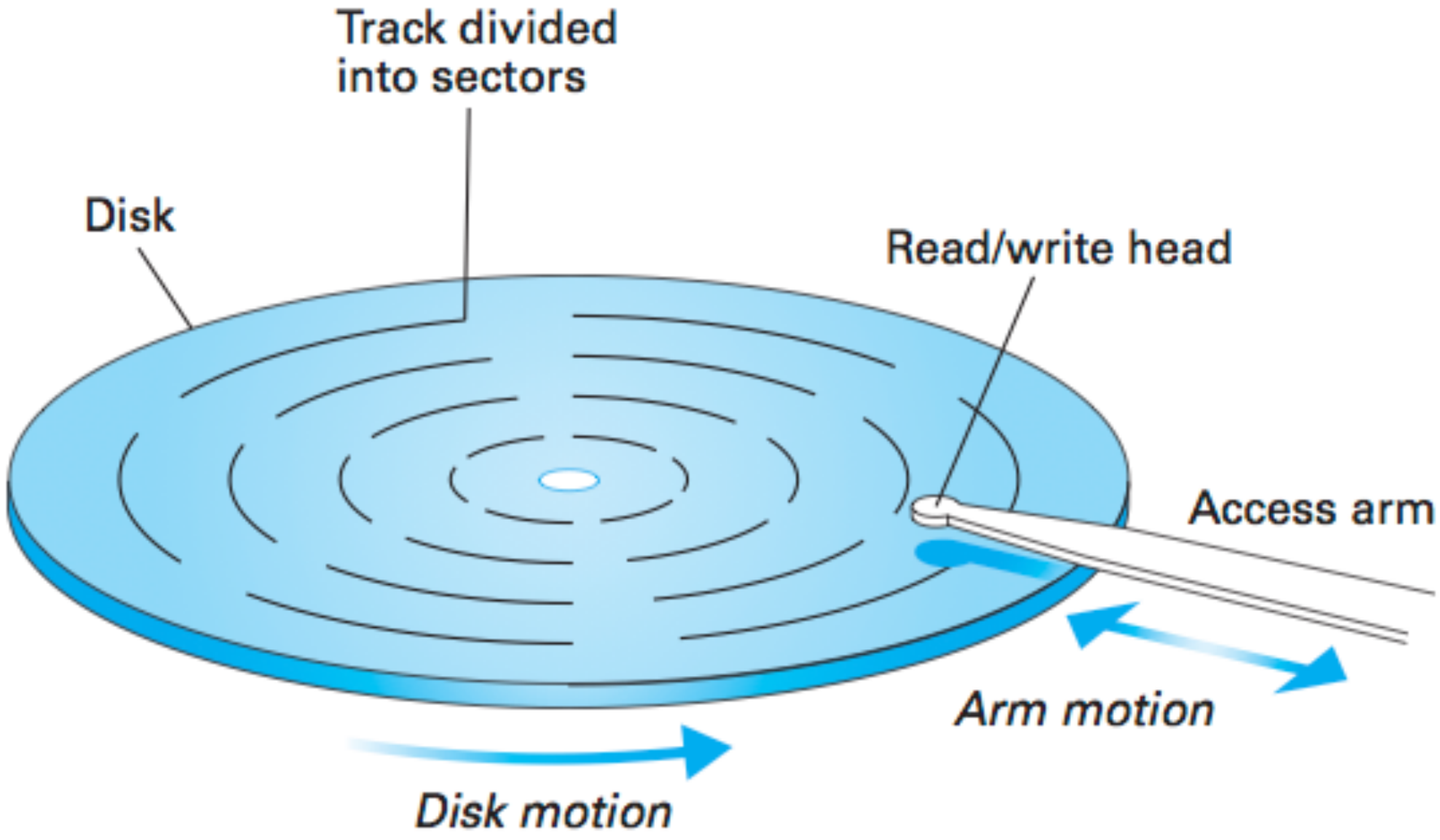
- Vantagens: Menor volatilidade, maior capacidade de armazenamento, baixo custo e ‘portabilidade’

- Desvantagem: em geral, requer algum tipo de mecânica, o que reduz significativamente o tempo para acessar e armazenar dados

Discos Magnéticos

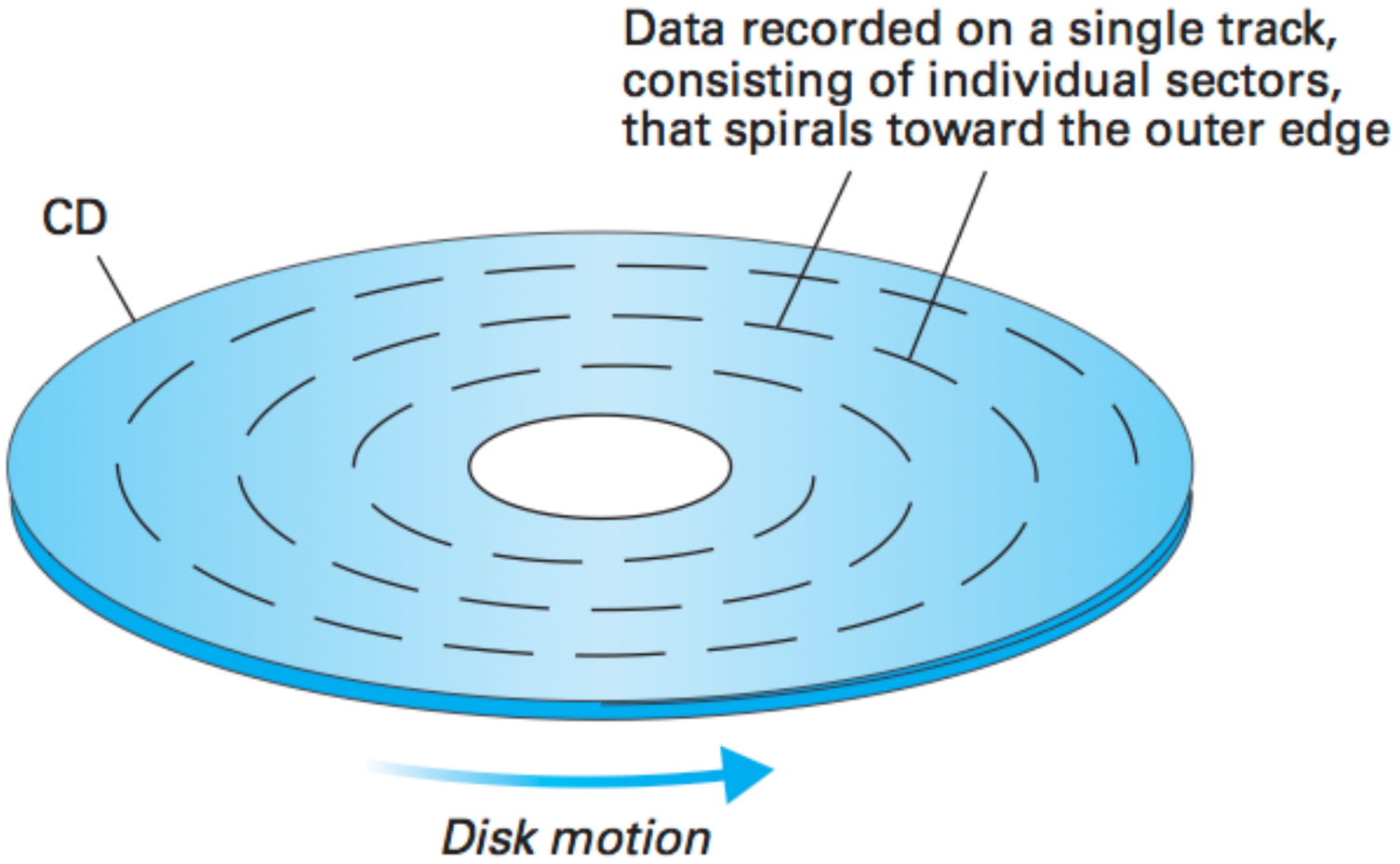


Discos Magnéticos

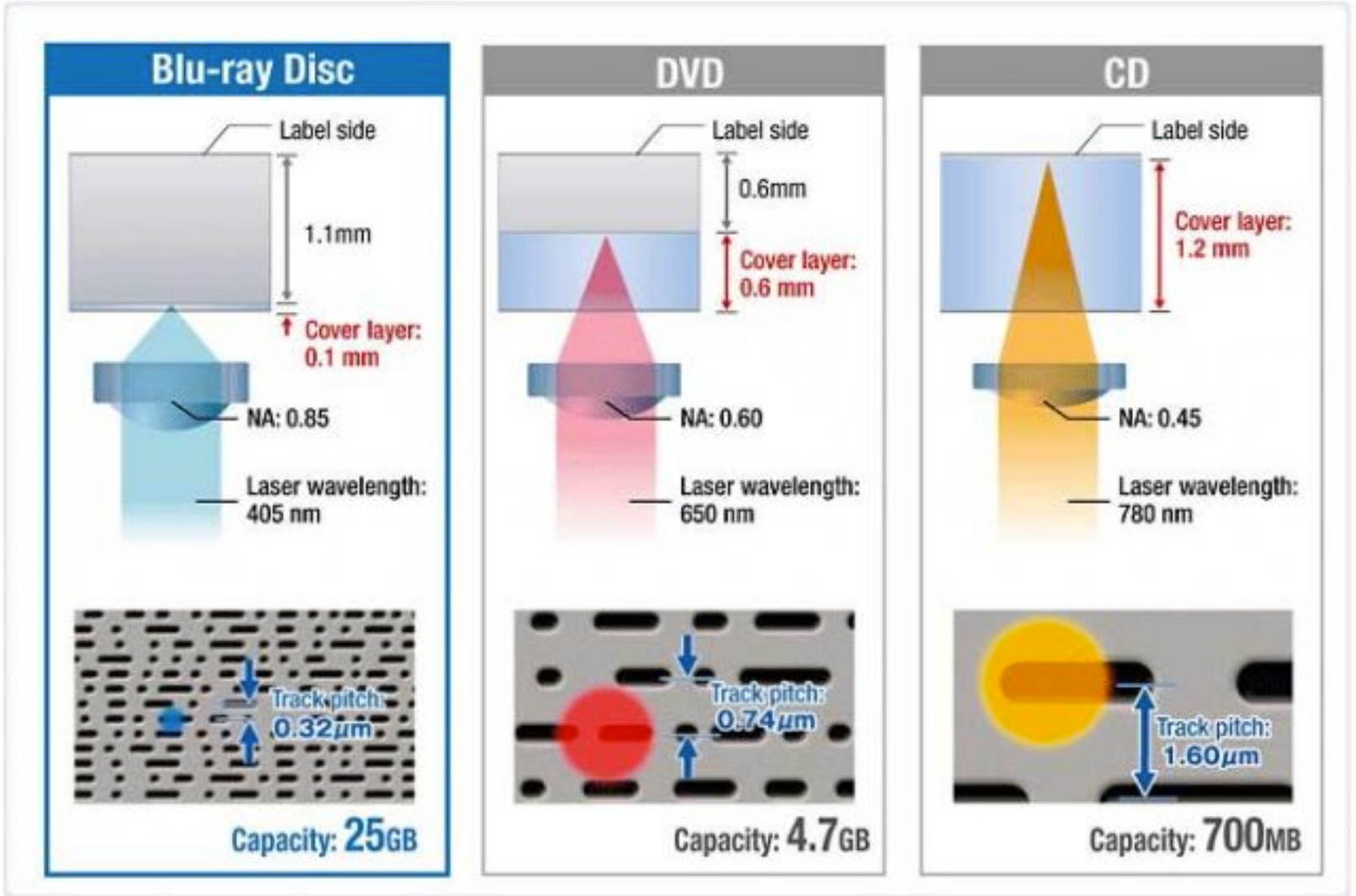


como podemos avaliar o desempenho deste sistema?

Sistemas Óticos



Sistemas Óticos



http://www.oficinadanet.com.br/imagens/post/8659/td_blu-ray-works.jpg

Memória Flash

- Reduz a desvantagem das tecnologias magnética e ótica

- Bits são armazenados ao enviar sinais eletrônicos diretamente à mídia de armazenamento, o que faz com que elétrons sejam capturados em pequenas câmaras de dióxido de silício, alterando a característica do circuito

- Estas camâras podem guardar os elétrons por muitos anos, o que torna a tecnologia adequada

Representando Informação

- Como a informação pode ser codificada com padrões de bits?

- Texto, dados numéricos, imagens, sons…

Representando Texto

- **ANSI** - American National Standard Institute
  - **ASCII** - American Standard Code for Information Interchange
    Padrão inicialmente com 7 bits, hoje usa 8 (byte)
- **Unicode** - padrão com 16 bits, 65536 padrões
- **ISO** - Padrões de 32 bits, capacidade de bilhões de símbolos

Representando Texto

- **ANSI** - American National Standard Institute

  - **ASCII** - American Standard Code for Information Interchange
    Padrão inicialmente com 7 bits, hoje usa 8 (byte)

```
01001000  01100101  01101100  01101100  01101111  00101110
   H         e         l         l         o         .
```

- **Unicode** - padrão com 16 bits, 65536 padrões

- **ISO** - Padrões de 32 bits, capacidade de bilhões de símbolos

ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	*[NULL]*	48	30	110000	60	**0**	96	60	1100000	140	`
1	1	1	1	*[START OF HEADING]*	49	31	110001	61	**1**	97	61	1100001	141	**a**
2	2	10	2	*[START OF TEXT]*	50	32	110010	62	**2**	98	62	1100010	142	**b**
3	3	11	3	*[END OF TEXT]*	51	33	110011	63	**3**	99	63	1100011	143	**c**
4	4	100	4	*[END OF TRANSMISSION]*	52	34	110100	64	**4**	100	64	1100100	144	**d**
5	5	101	5	*[ENQUIRY]*	53	35	110101	65	**5**	101	65	1100101	145	**e**
6	6	110	6	*[ACKNOWLEDGE]*	54	36	110110	66	**6**	102	66	1100110	146	**f**
7	7	111	7	*[BELL]*	55	37	110111	67	**7**	103	67	1100111	147	**g**
8	8	1000	10	*[BACKSPACE]*	56	38	111000	70	**8**	104	68	1101000	150	**h**
9	9	1001	11	*[HORIZONTAL TAB]*	57	39	111001	71	**9**	105	69	1101001	151	**i**
10	A	1010	12	*[LINE FEED]*	58	3A	111010	72	**:**	106	6A	1101010	152	**j**
11	B	1011	13	*[VERTICAL TAB]*	59	3B	111011	73	**;**	107	6B	1101011	153	**k**
12	C	1100	14	*[FORM FEED]*	60	3C	111100	74	**<**	108	6C	1101100	154	**l**
13	D	1101	15	*[CARRIAGE RETURN]*	61	3D	111101	75	**=**	109	6D	1101101	155	**m**
14	E	1110	16	*[SHIFT OUT]*	62	3E	111110	76	**>**	110	6E	1101110	156	**n**
15	F	1111	17	*[SHIFT IN]*	63	3F	111111	77	**?**	111	6F	1101111	157	**o**
16	10	10000	20	*[DATA LINK ESCAPE]*	64	40	1000000	100	**@**	112	70	1110000	160	**p**
17	11	10001	21	*[DEVICE CONTROL 1]*	65	41	1000001	101	**A**	113	71	1110001	161	**q**
18	12	10010	22	*[DEVICE CONTROL 2]*	66	42	1000010	102	**B**	114	72	1110010	162	**r**
19	13	10011	23	*[DEVICE CONTROL 3]*	67	43	1000011	103	**C**	115	73	1110011	163	**s**
20	14	10100	24	*[DEVICE CONTROL 4]*	68	44	1000100	104	**D**	116	74	1110100	164	**t**
21	15	10101	25	*[NEGATIVE ACKNOWLEDGE]*	69	45	1000101	105	**E**	117	75	1110101	165	**u**
22	16	10110	26	*[SYNCHRONOUS IDLE]*	70	46	1000110	106	**F**	118	76	1110110	166	**v**
23	17	10111	27	*[ENG OF TRANS. BLOCK]*	71	47	1000111	107	**G**	119	77	1110111	167	**w**
24	18	11000	30	*[CANCEL]*	72	48	1001000	110	**H**	120	78	1111000	170	**x**
25	19	11001	31	*[END OF MEDIUM]*	73	49	1001001	111	**I**	121	79	1111001	171	**y**
26	1A	11010	32	*[SUBSTITUTE]*	74	4A	1001010	112	**J**	122	7A	1111010	172	**z**
27	1B	11011	33	*[ESCAPE]*	75	4B	1001011	113	**K**	123	7B	1111011	173	**{**
28	1C	11100	34	*[FILE SEPARATOR]*	76	4C	1001100	114	**L**	124	7C	1111100	174	**|**
29	1D	11101	35	*[GROUP SEPARATOR]*	77	4D	1001101	115	**M**	125	7D	1111101	175	**}**
30	1E	11110	36	*[RECORD SEPARATOR]*	78	4E	1001110	116	**N**	126	7E	1111110	176	**~**
31	1F	11111	37	*[UNIT SEPARATOR]*	79	4F	1001111	117	**O**	127	7F	1111111	177	*[DEL]*
32	20	100000	40	*[SPACE]*	80	50	1010000	120	**P**					
33	21	100001	41	!	81	51	1010001	121	**Q**					

Arquivos de Texto

- Sequências de símbolos codificados em ASCII ou Unicode

- Arquivos de texto simples são manipuláveis por programas editores de texto

- Processadores de texto geralmente salvam arquivos com código proprietários representando mudança de fontes, alinhamento, etc.

Valores Numéricos

- Armazenar informação na forma de caracteres codificados é ineficiente quando devemos registrar informação de forma puramente numérica

- Para armazenar o número 25 precisaríamos de 16 bits (ASCII utiliza um ***byte*** por símbolo)

- O maior número que poderíamos armazenar com 16 bits seria 99

- Usando notação binária, podemos armazenar de 0 a 65535 utilizando os mesmos 16 bits

Notação Binária

0000

000**1**

00**1**0

00**11**

0**1**00

0**1**0**1**

0**11**0

0**111**

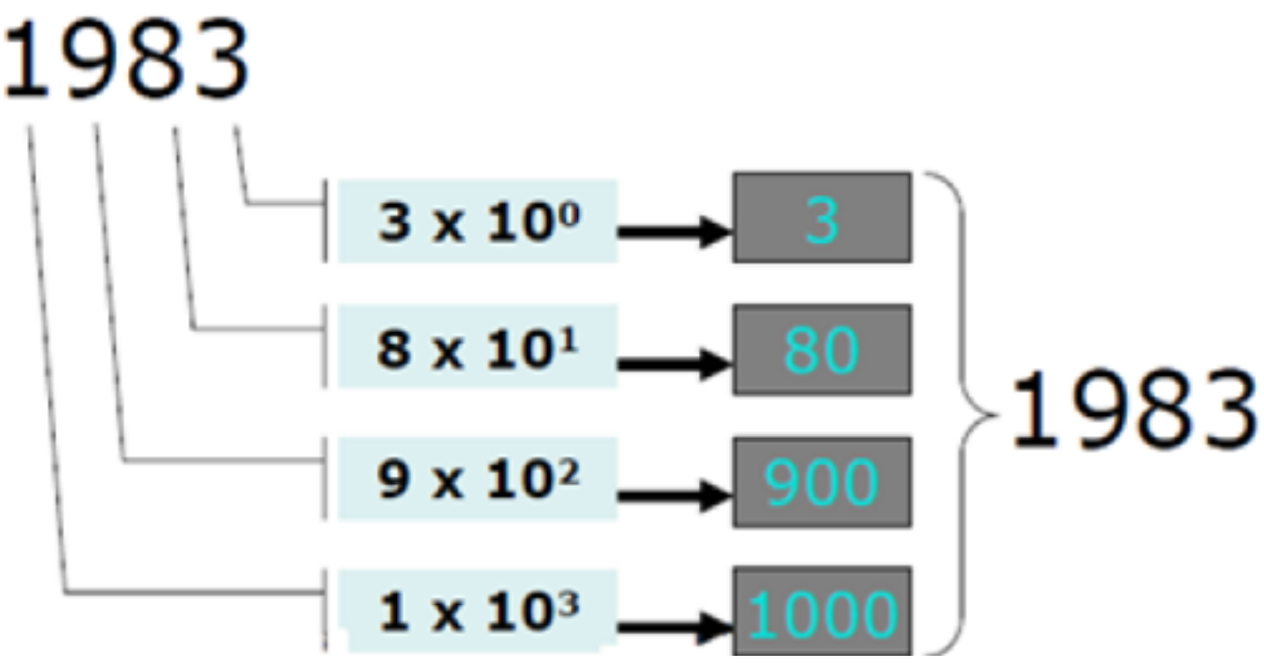
1000

Notação Binária

0000	0
000**1**	1
00**1**0	2
00**11**	3
0**1**00	4
0**1**0**1**	5
0**11**0	6
0**111**	7
1000	8

Números Decimais

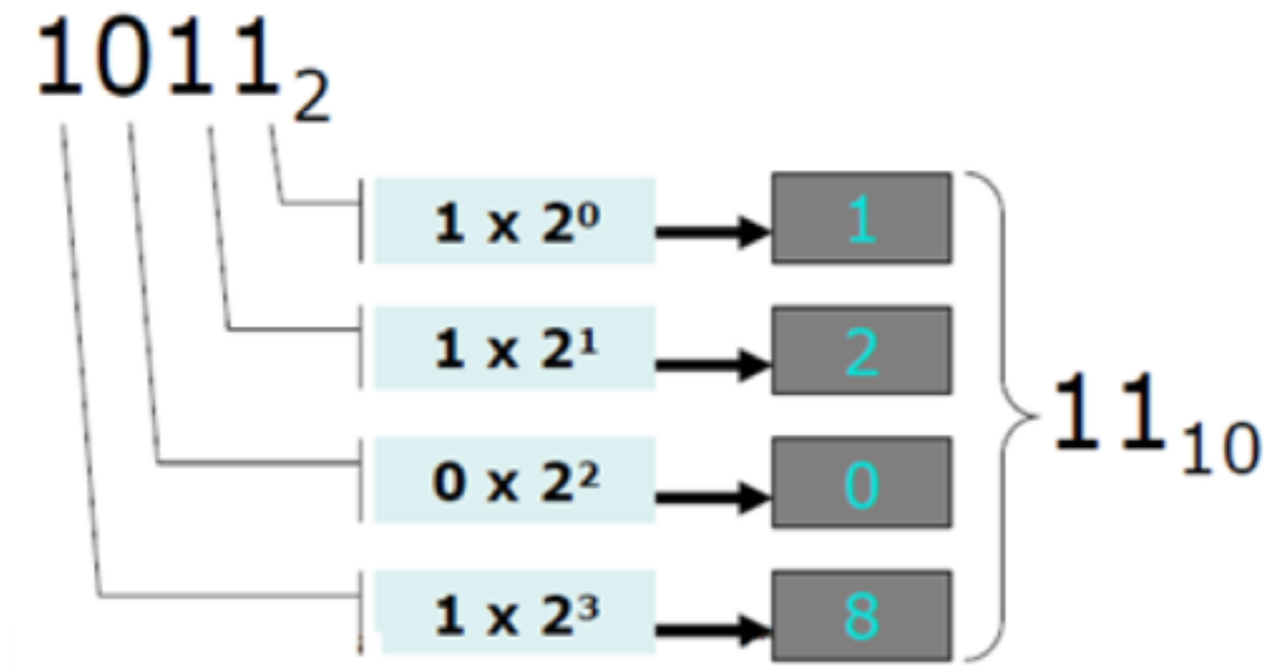
Digitos arábicos de 0 a 9



Adaptado dos slides do Prof. Cleber Zanchettin (CIn-UFPE)

Números Binários

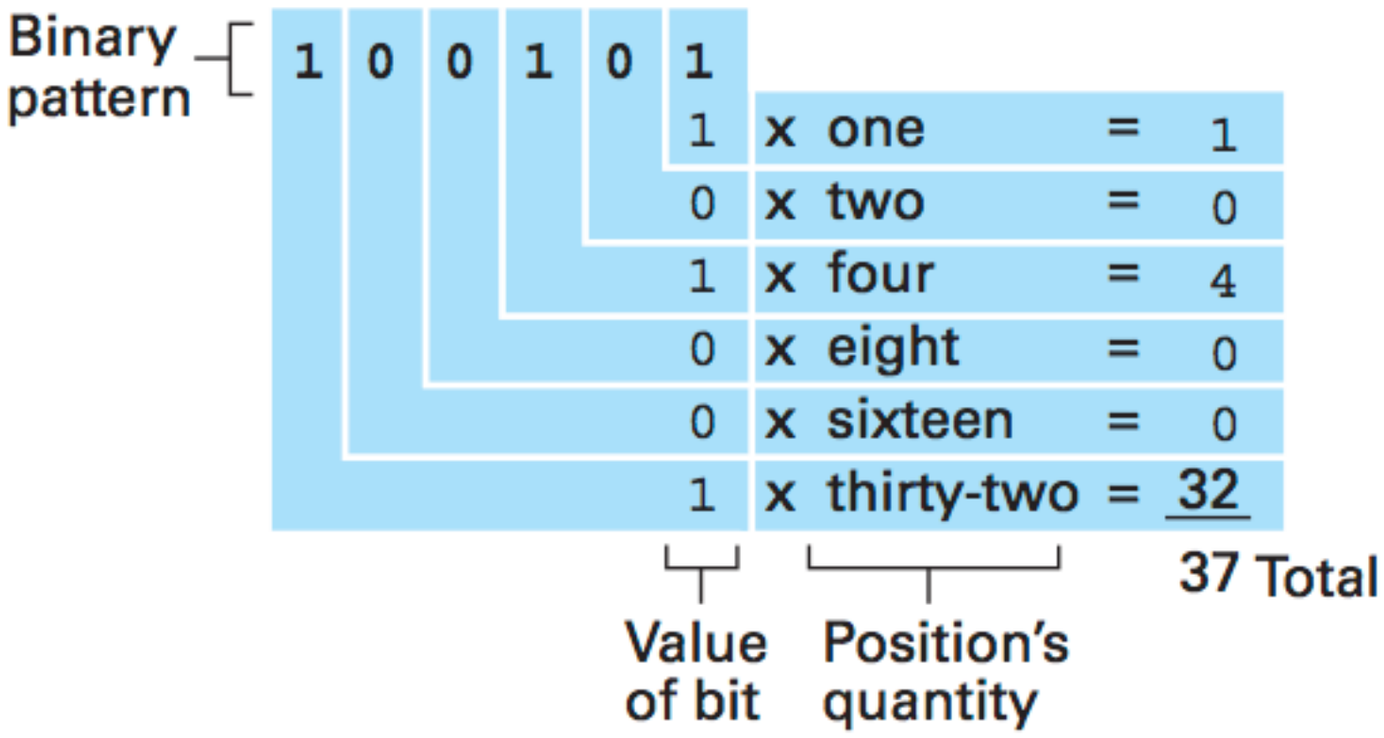
Digitos arábicos de 0 a1



Adaptado dos slides do Prof. Cleber Zanchettin (CIn-UFPE)

Números Binários

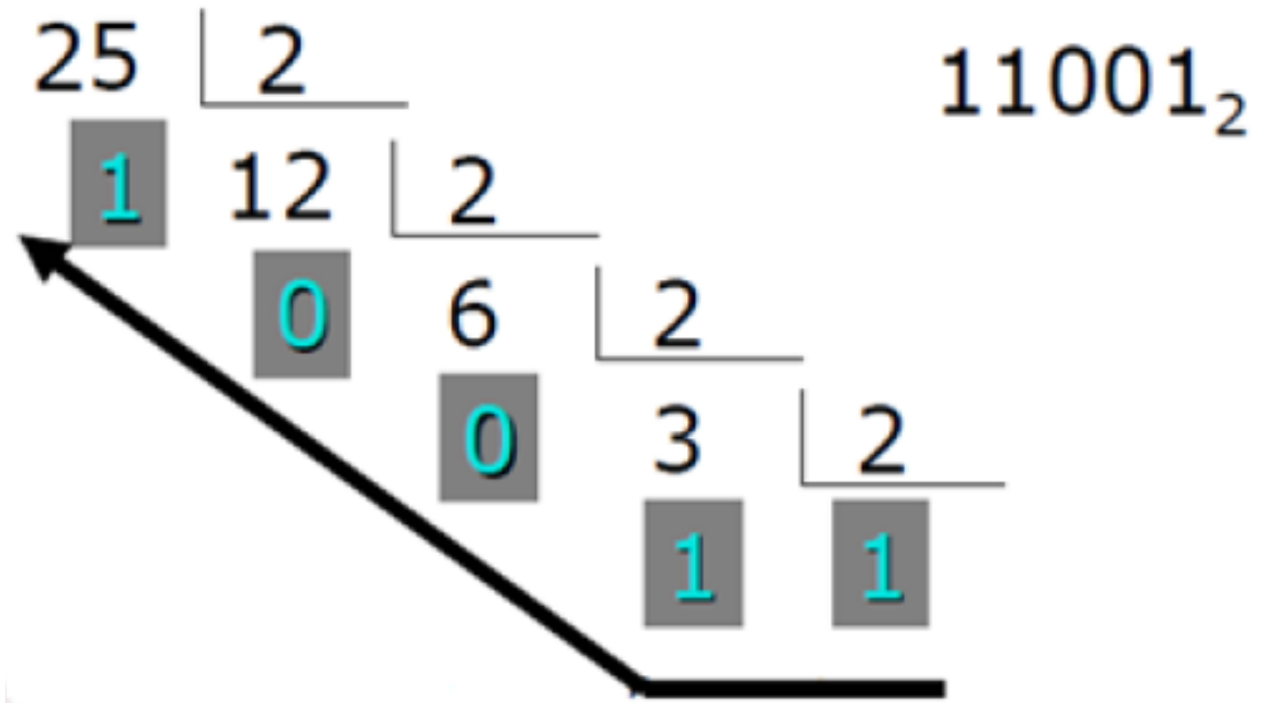
Digitos arábicos de 0 a1



Adaptado dos slides do Prof. Cleber Zanchettin (CIn-UFPE)

Conversão de base: 10 para 2

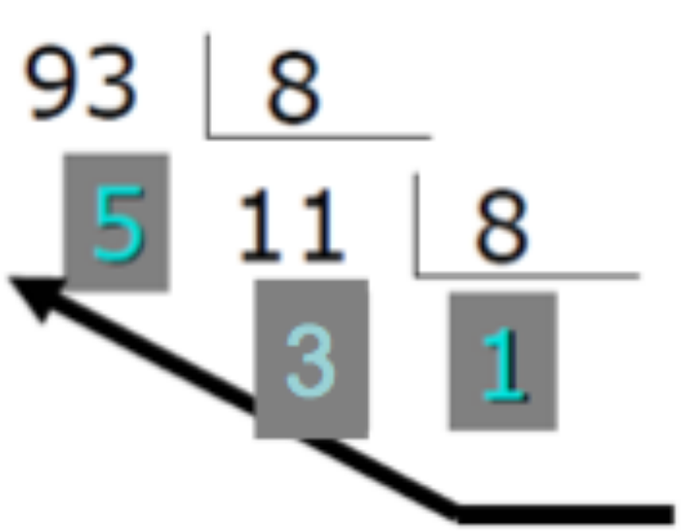
Divisões sucessivas pela base



Adaptado dos slides do Prof. Cleber Zanchettin (CIn-UFPE)

Conversão de base: 10 para 8

Divisões sucessivas pela base

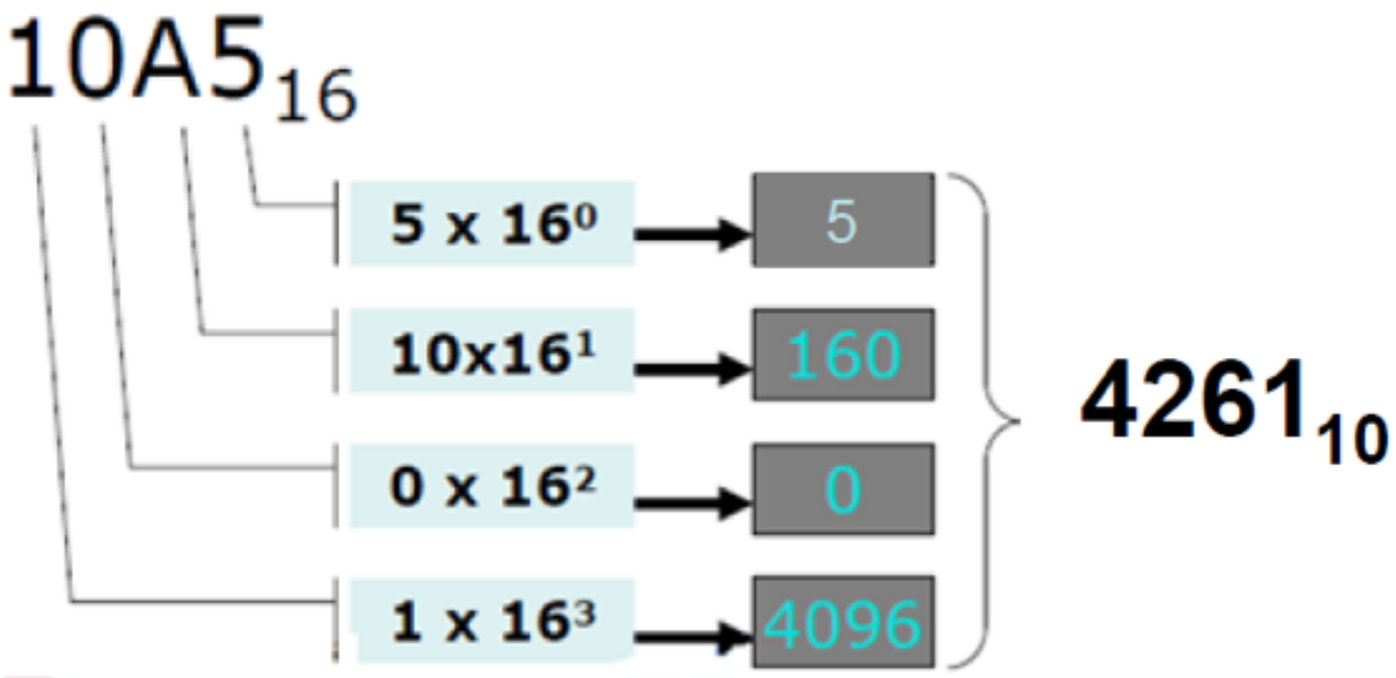


135_8

Adaptado dos slides do Prof. Cleber Zanchettin (CIn-UFPE)

Números Hexadecimais

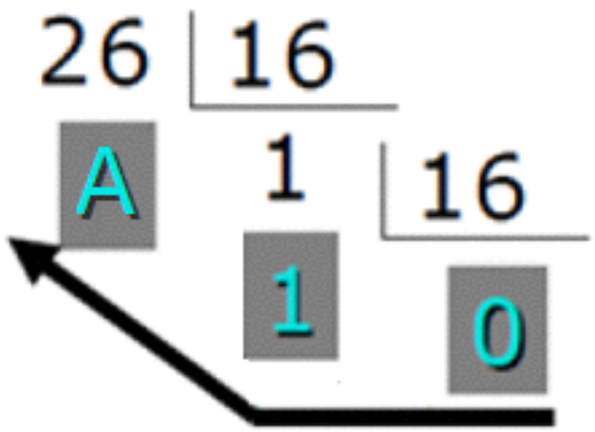
Digitos arábicos de 0 a 9 e A a F



Adaptado dos slides do Prof. Cleber Zanchettin (CIn-UFPE)

Conversão de base

Divisões sucessivas pela base



$01A_{16}$

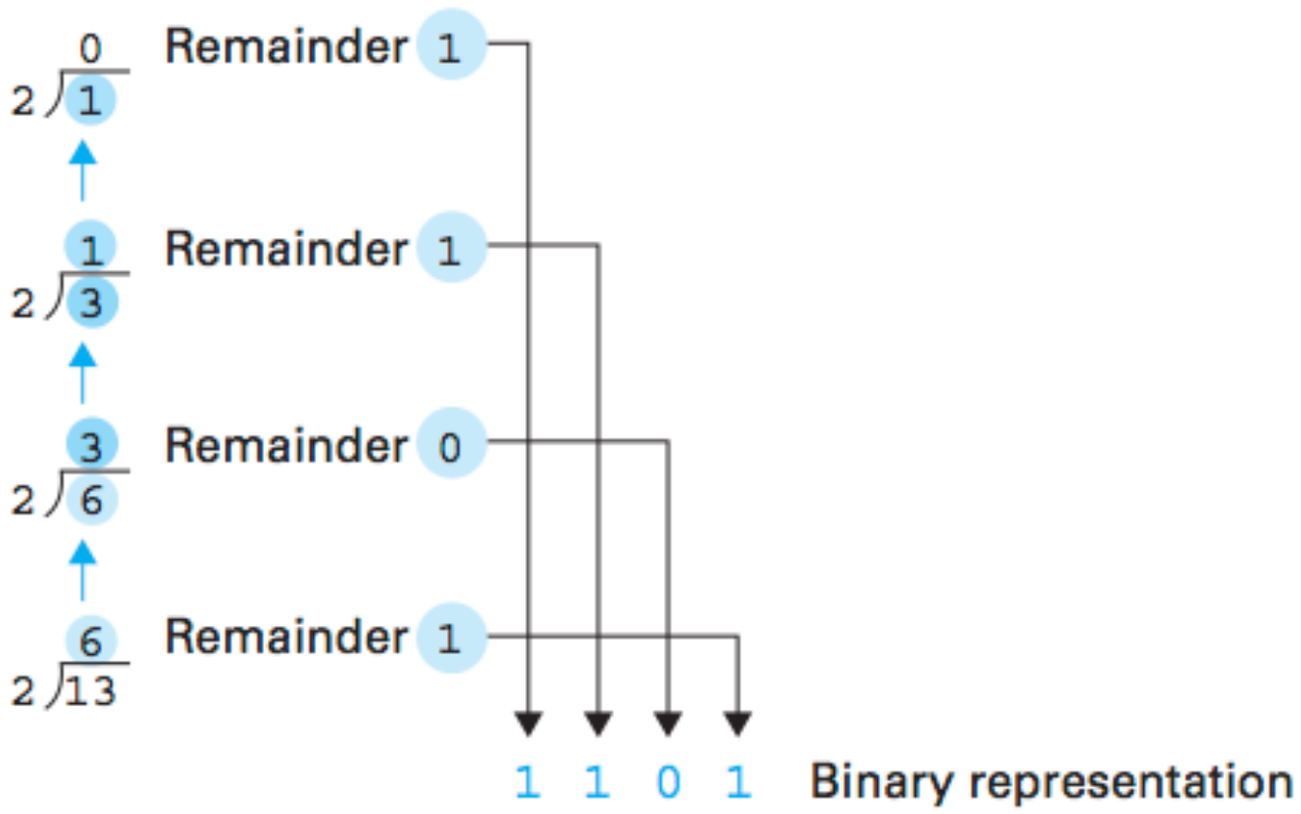
Adaptado dos slides do Prof. Cleber Zanchettin (CIn-UFPE)

Conversão de base

Step 1. Divide the value by two and record the remainder.

Step 2. As long as the quotient obtained is not zero, continue to divide the newest quotient by two and record the remainder.

Step 3. Now that a quotient of zero has been obtained, the binary representation of the original value consists of the remainders listed from right to left in the order they were recorded.



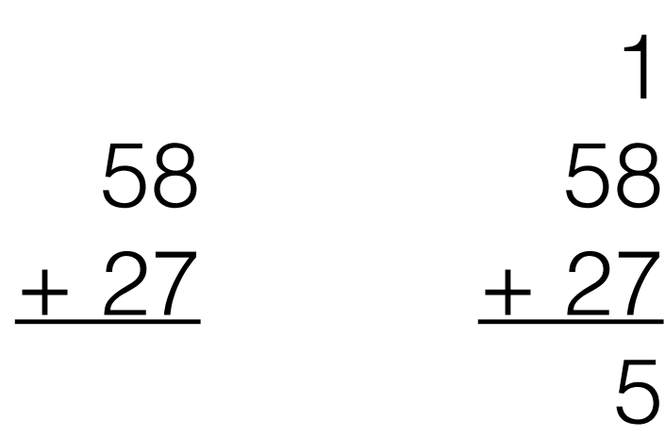
Adição Binária

- Vamos lembrar o processo de adição na base 10:

$$\begin{array}{r} 58 \\ +\ 27 \\ \hline \end{array}$$

Adição Binária

- Vamos lembrar o processo de adição na base 10:



Adição Binária

- Vamos lembrar o processo de adição na base 10:

	1	
58	58	58
+ 27	+ 27	+ 27
	5	85

Tabuada de Adição Binária

0	**1**	**0**	**1**
+ 0	**+ 0**	**+ 1**	**+ 1**
0	**1**	**1**	**10**

00111010
+ 00011011
?

Tabuada de Adição Binária

0	**1**	**0**	**1**
+ 0	**+ 0**	**+ 1**	**+ 1**
0	**1**	**1**	**10**

```
  00111010
+ 00011011
  01010101
```

Frações em Binário

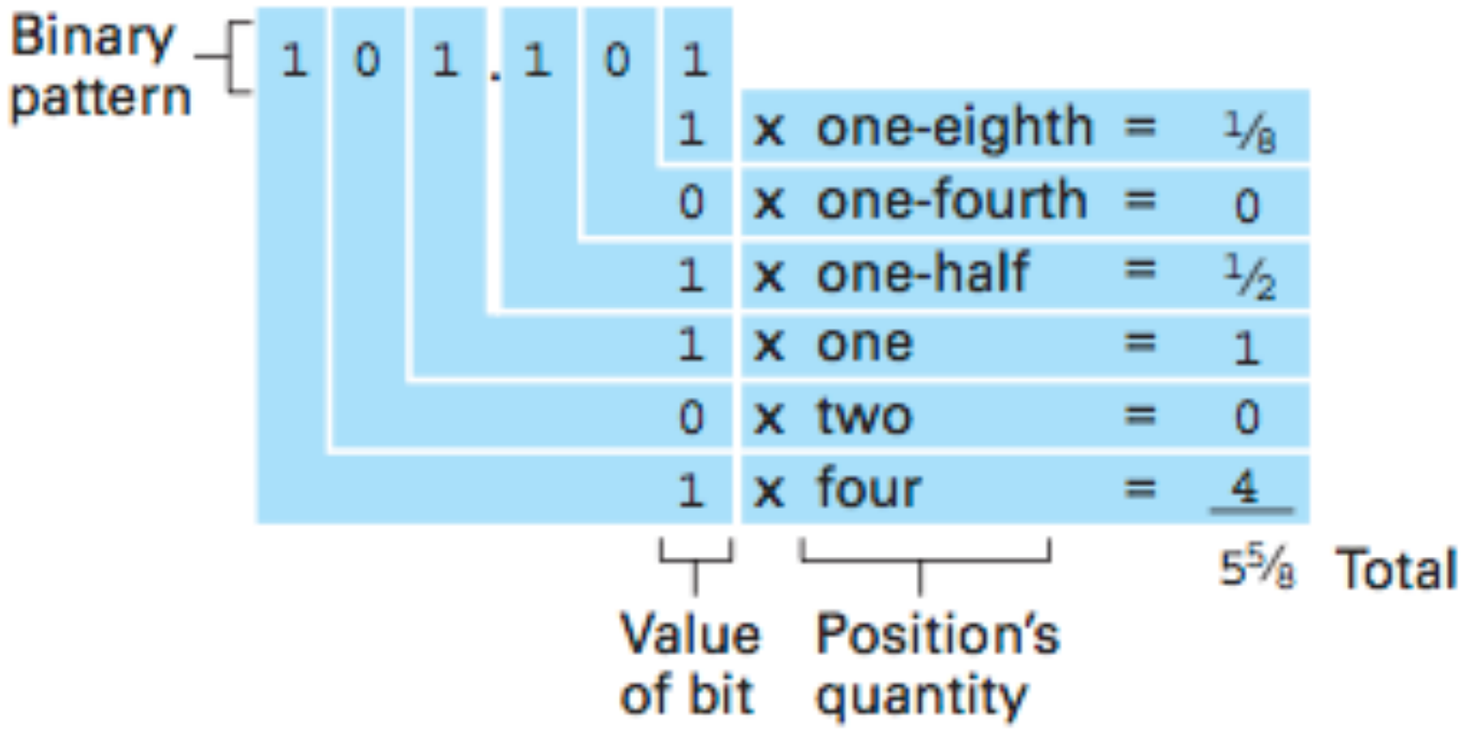
- Utilizamos uma vírgula (*radix point*) que funciona da mesma forma que o ponto/vírgula da notação decimal

- Dígitos à esquerda do ponto representam a parte inteira do valor e são interpretados conforme já discutido

- Dígitos à direita representam a parte fracionária do valor, sendo interpretados de forma semelhante aos outros bits, exceto que suas posições representam quantidades fracionárias

  - A primeira posição à direita da vírgula representa 1/2 (que é 2^{-1}); a segunda posição representa 1/4 (que é 2^{-2}); e assim por diante…

Decodificando frações em binário



Adição de frações em binário

```
  10.011
+ 100.110
```

Adição de frações em binário

```
  10.011
+ 100.110
---------
 111.001
```

Representando Números Inteiros

- Podemos utilizar sistemas notacionais baseados no sistema binário, mas com propriedades adicionais que os tornam mais compatíveis com o projeto de computadores

- Toda escolha tem consequências, vamos estudar as propriedades de cada sistema

Complemento de Dois

- Sistema mais popular, usa um número fixo de bits para representar cada valor numérico

- Nos equipamentos atuais, é comum usar um sistema de complemento de dois em que cada valor é representado por um padrão de 32 bits

- Para estudar as propriedades destes sistemas, utilizaremos sistemas menores

Complemento de Dois

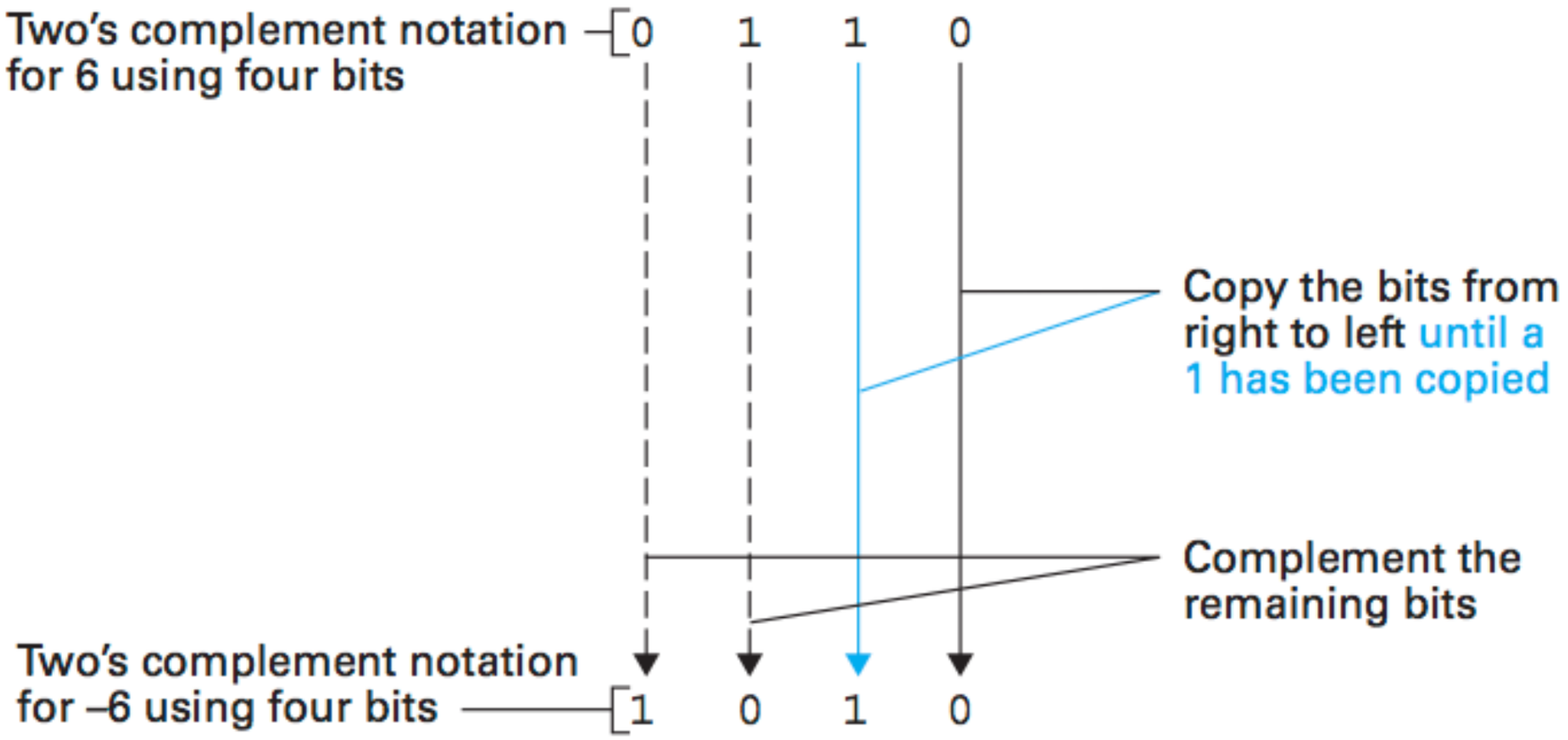
Bit pattern	Value represented
011	3
010	2
001	1
000	0
111	−1
110	−2
101	−3
100	−4

Bit pattern	Value represented
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	−1
1110	−2
1101	−3
1100	−4
1011	−5
1010	−6
1001	−7
1000	−8

Complemento de Dois

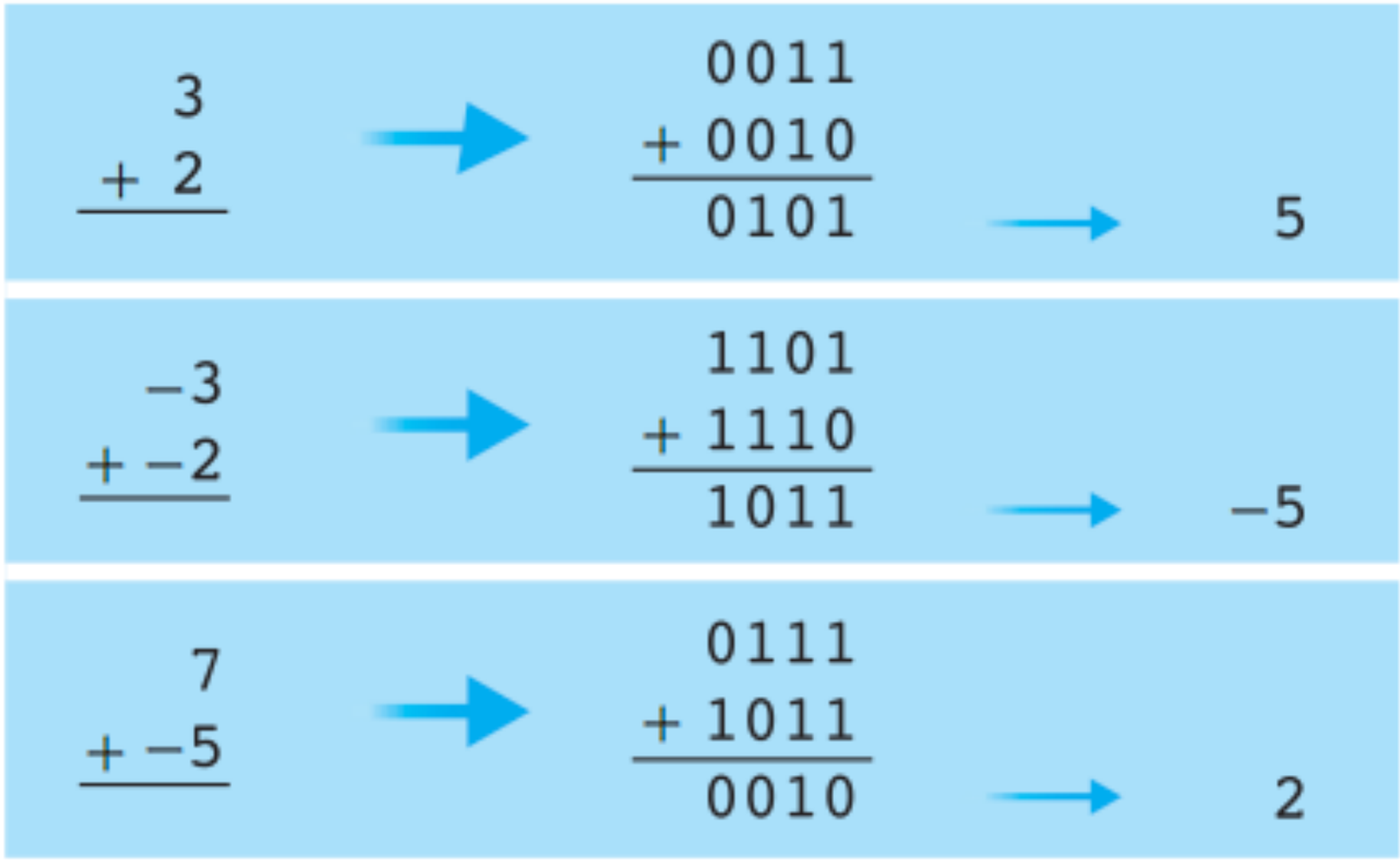
Bit pattern	Value represented
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	−1
1110	−2
1101	−3
1100	−4
1011	−5
1010	−6
1001	−7
1000	−8

Complemento de Dois

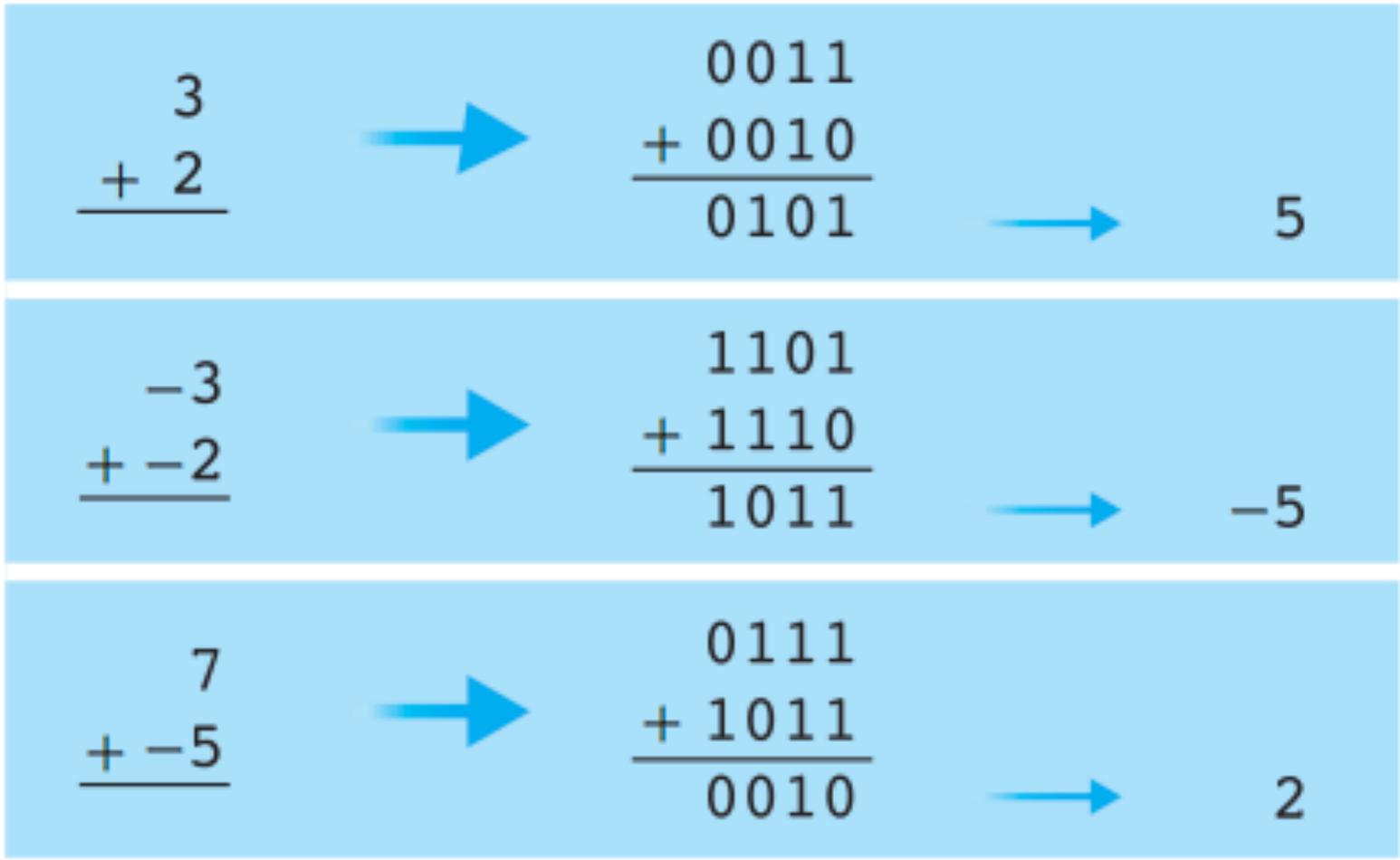


Bit pattern	Value represented
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	−1
1110	−2
1101	−3
1100	−4
1011	−5
1010	−6
1001	−7
1000	−8

Adição



Adição



Vantagem é que a adição de números positivos ou negativos segue o **mesmo algoritmo**, portanto, usa o mesmo circuito.

Que problema pode acontecer com este sistema?

Estouro (*overflow*)

- Em qualquer sistema de complemento de dois, há sempre um limite para o tamanho dos números que podem ser representados

- Ao usar o sistema com padrões de quatro bits, o maior inteiro positivo que pode ser representado é 7 e o negativo é -8.

- Em particular, o valor 9 não pode ser representado, portanto não obtemos uma resposta correta para a operação de soma 5+4 (resultado apareceria como -7)

- Com 32 bits, temos 2.147.483.647 valores antes do estouro

- Com 16 bits, temos 32.768 valores

Notação de Excesso

- Assim como no sistema de complemento de dois, há um tamanho fixo dos padrões de bits

- Para usar o sistema, estabelecemos o tamanho do padrão e escrevemos todos os padrões diferentes daquele tamanho na ordem binária

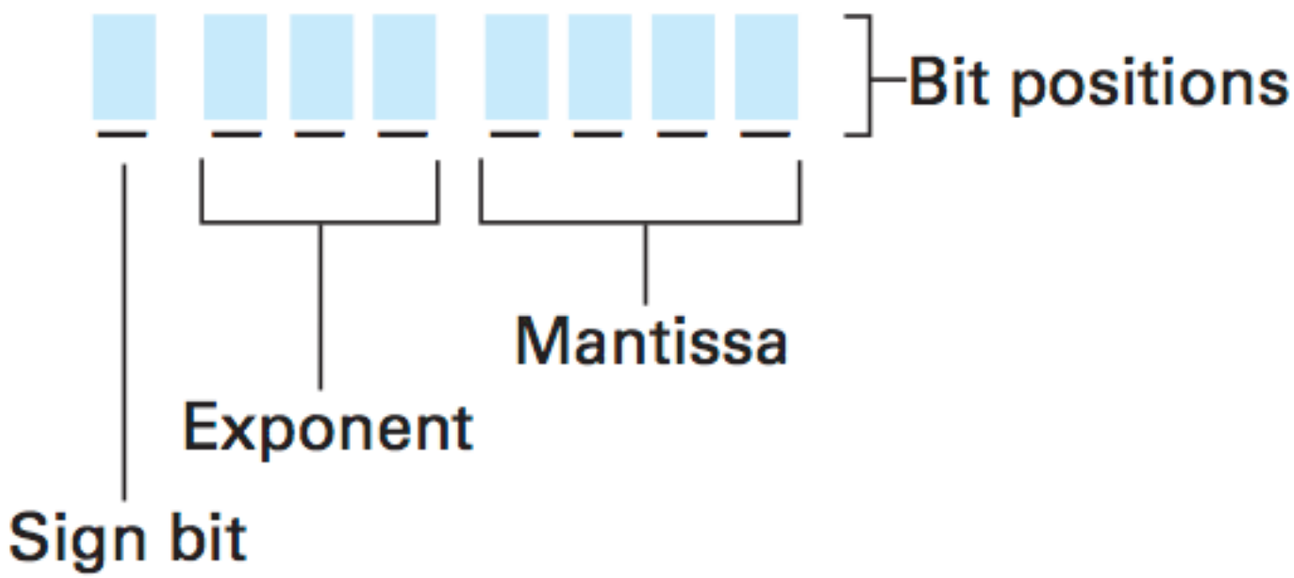
- Escolhemos o primeiro padrão começando com 1 para representar zero. Os padrões seguintes representam 1, 2, 3… e os anteriores -1, -2, -3…

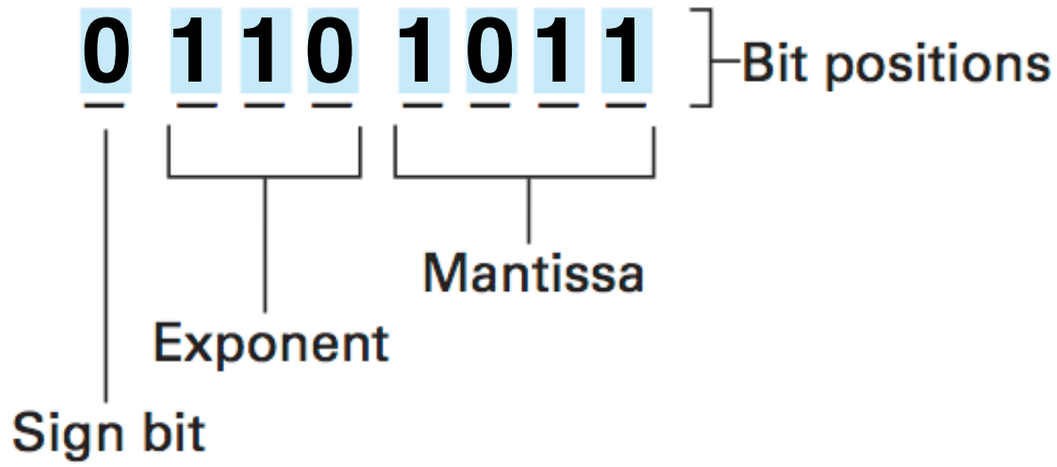
Bit pattern	Value represented
1111	7
1110	6
1101	5
1100	4
1011	3
1010	2
1001	1
1000	0
0111	-1
0110	-2
0101	-3
0100	-4
0011	-5
0010	-6
0001	-7
0000	-8

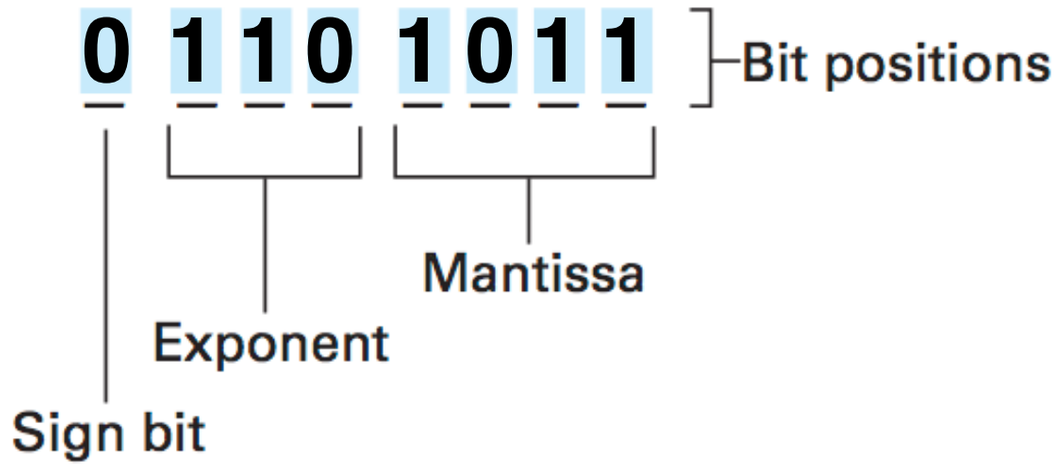
Representando Frações

- Em contraste com os números inteiros, precisamos representar também a posição do *radix point*

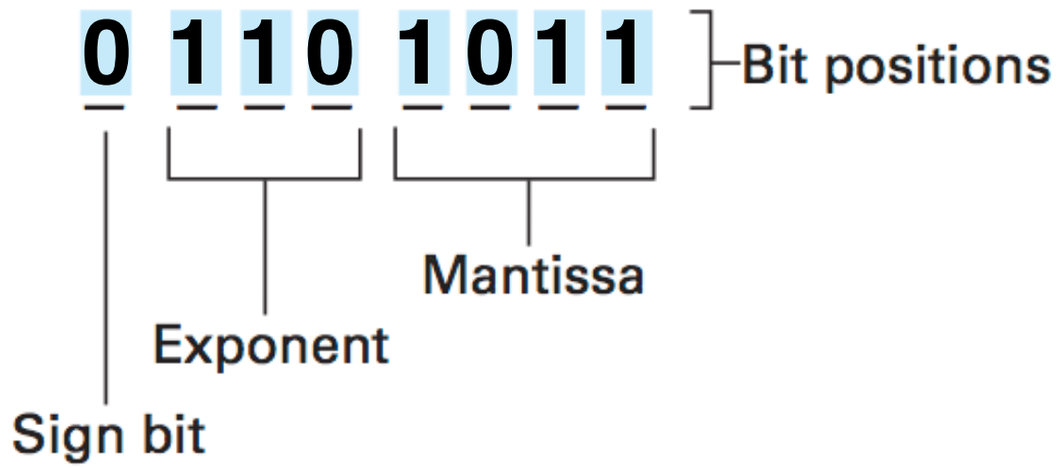
- Uma maneira popular de fazer isto é chamada de notação de ponto flutuante




0 1 1 0 1 0 1 1
Bit positions
Mantissa
Exponent
Sign bit

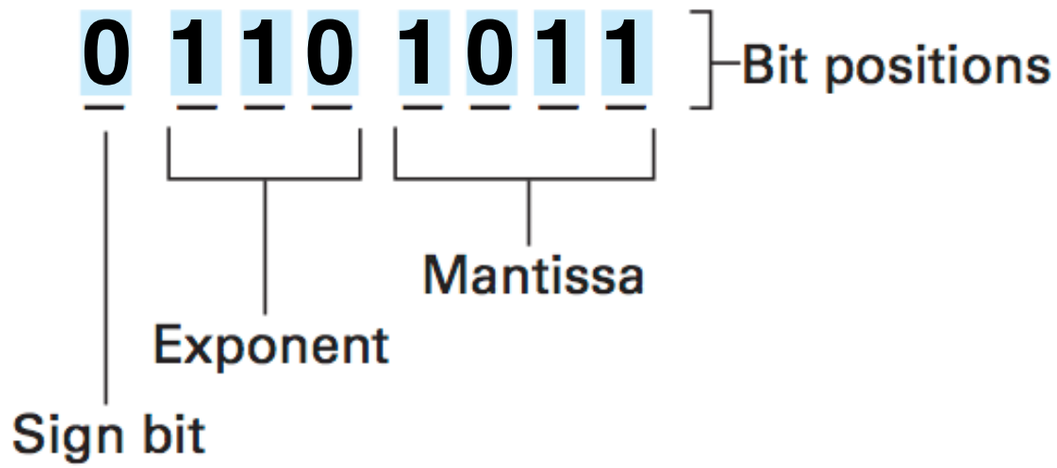


Extraímos a mantissa e colocamos a vírgula binária à sua esquerda:



Extraímos a mantissa e colocamos a vírgula binária à sua esquerda:

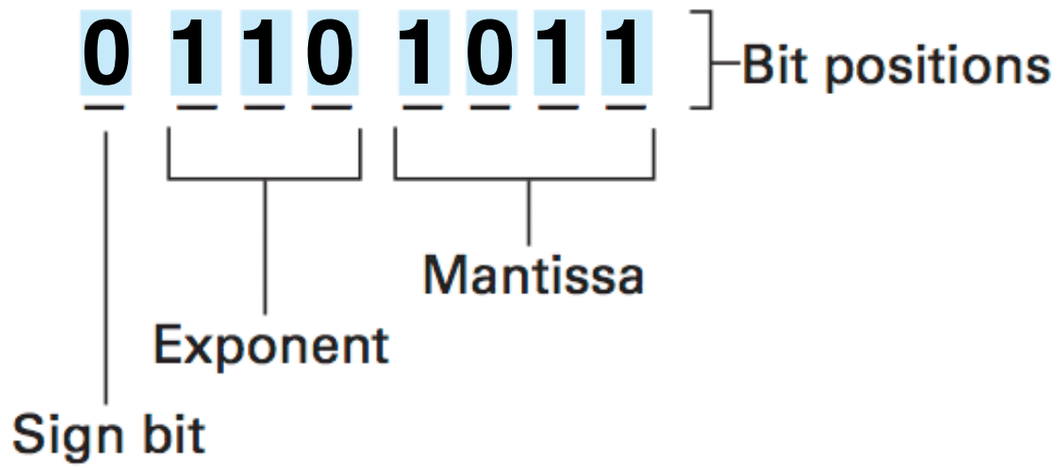
,1011



Extraímos a mantissa e colocamos a vírgula binária à sua esquerda:

,1011

Extraímos o conteúdo do campo do expoente (**110**) e interpretamos como um inteiro armazenado com notação de excesso 3-bits. Neste caso, **110** representa 2.



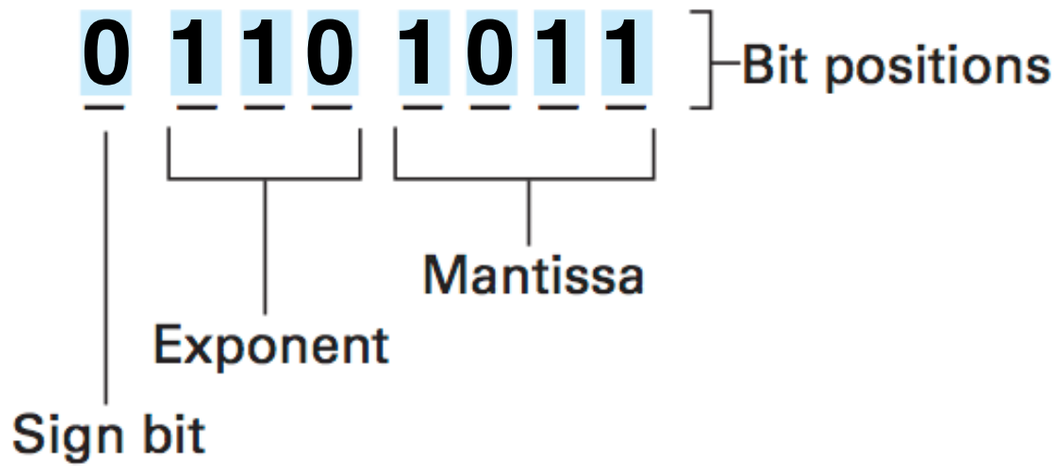
Extraímos a mantissa e colocamos a vírgula binária à sua esquerda:

,1011

Extraímos o conteúdo do campo do expoente (**110**) e interpretamos como um inteiro armazenado com notação de excesso 3-bits.

Neste caso, **110** representa 2.

Desta forma, deslocamos a vírgula dois dígitos para a direita



Extraímos a mantissa e colocamos a vírgula binária à sua esquerda:

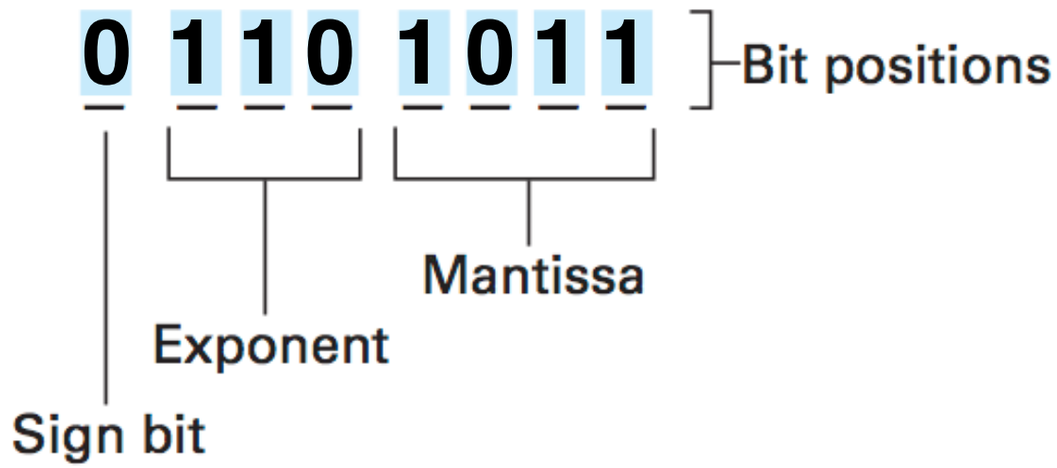
,1011

Extraímos o conteúdo do campo do expoente (**110**) e interpretamos como um inteiro armazenado com notação de excesso 3-bits.

Neste caso, **110** representa 2.

Desta forma, deslocamos a vírgula dois dígitos para a direita

10,11 = 2(3/4)



Extraímos a mantissa e colocamos a vírgula binária à sua esquerda:

,1011

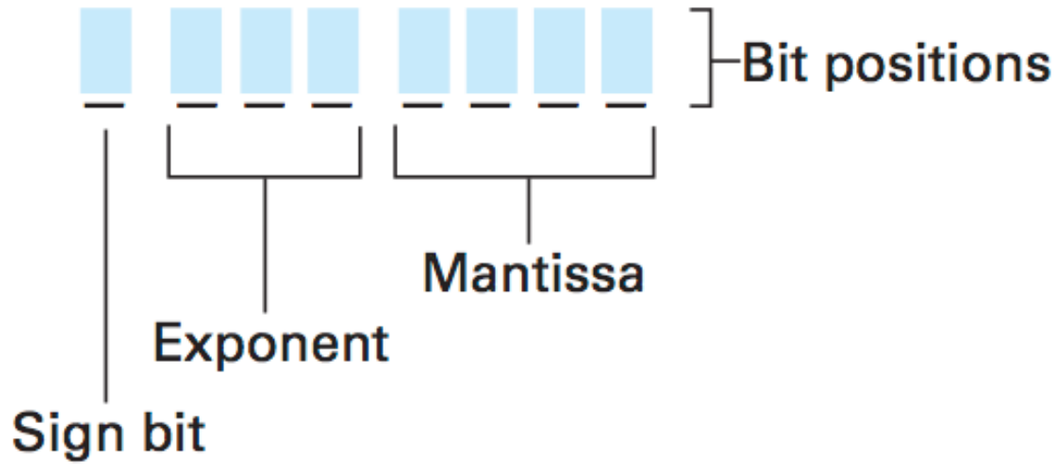
Extraímos o conteúdo do campo do expoente (**110**) e interpretamos como um inteiro armazenado com notação de excesso 3-bits.

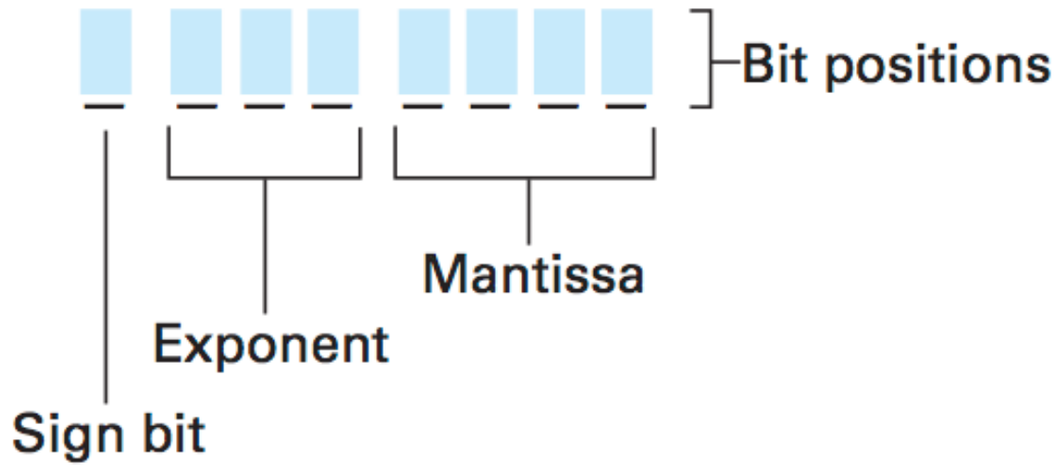
Neste caso, **110** representa 2.

Desta forma, deslocamos a vírgula dois dígitos para a direita

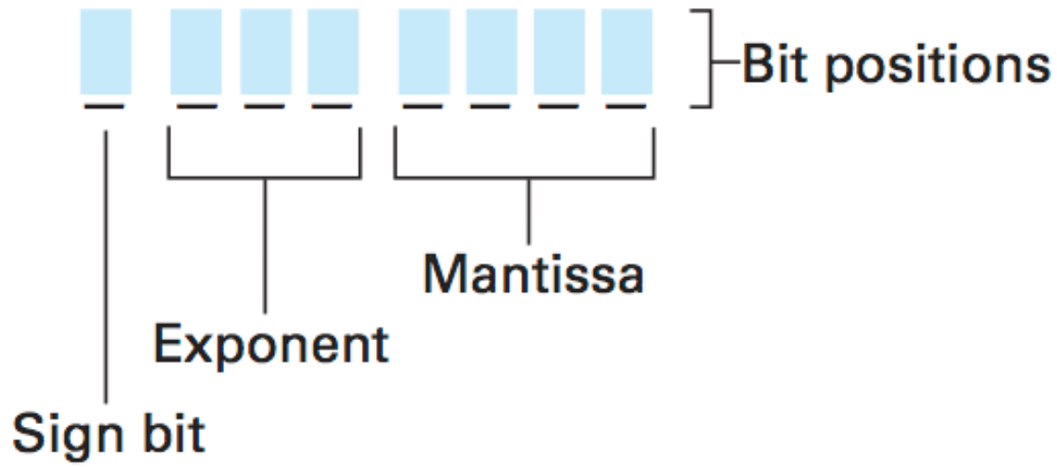
10,11 = 2(3/4)

O bit de sinal é 0, então o valor é não-negativo.


Bit positions
Mantissa
Exponent
Sign bit

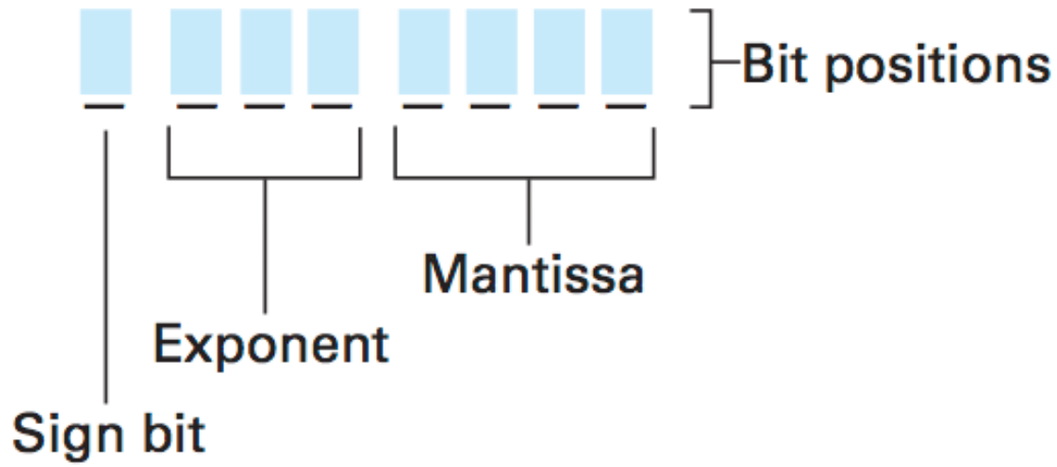


Para codificar 1 1/8 em notação de ponto flutuante, primeiro expressamos o número em binário e obtemos 1.001. Em seguida colocamos o padrão na mantissa



Para codificar 1 1/8 em notação de ponto flutuante, primeiro expressamos o número em binário e obtemos 1.001. Em seguida colocamos o padrão na mantissa

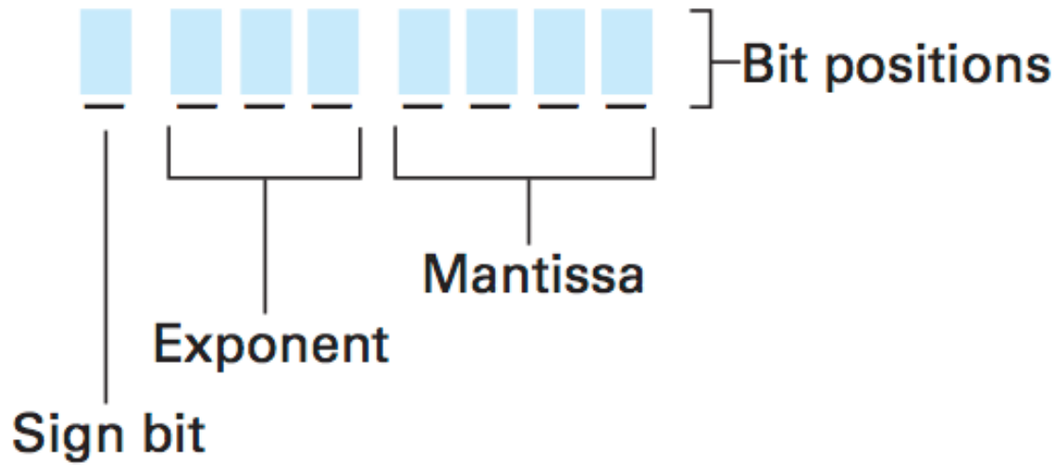
_ _ _ _ **1 0 0 1**



Para codificar 1 1/8 em notação de ponto flutuante, primeiro expressamos o número em binário e obtemos 1.001. Em seguida colocamos o padrão na mantissa

_ _ _ _ **1 0 0 1**

Agora precisamos preencher o campo do expoente. Devemos mover a vírgula um bit à direita, portanto o expoente deve ser o número 1 positivo representado em notação de excesso (*101*)

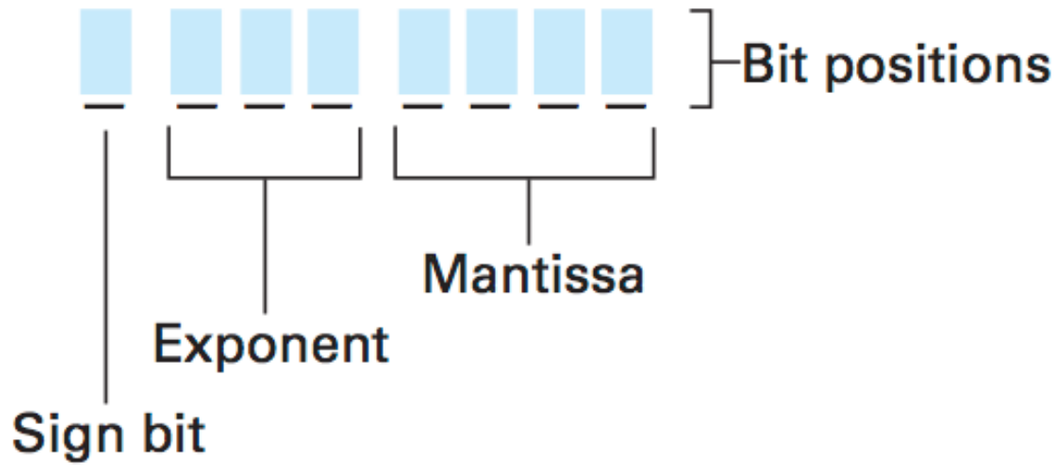


Para codificar 1 1/8 em notação de ponto flutuante, primeiro expressamos o número em binário e obtemos 1.001. Em seguida colocamos o padrão na mantissa

_ _ _ _ **1 0 0 1**

Agora precisamos preencher o campo do expoente. Devemos mover a vírgula um bit à direita, portanto o expoente deve ser o número 1 positivo representado em notação de excesso (*101*)

_ **1 0 1 1 0 0 1**



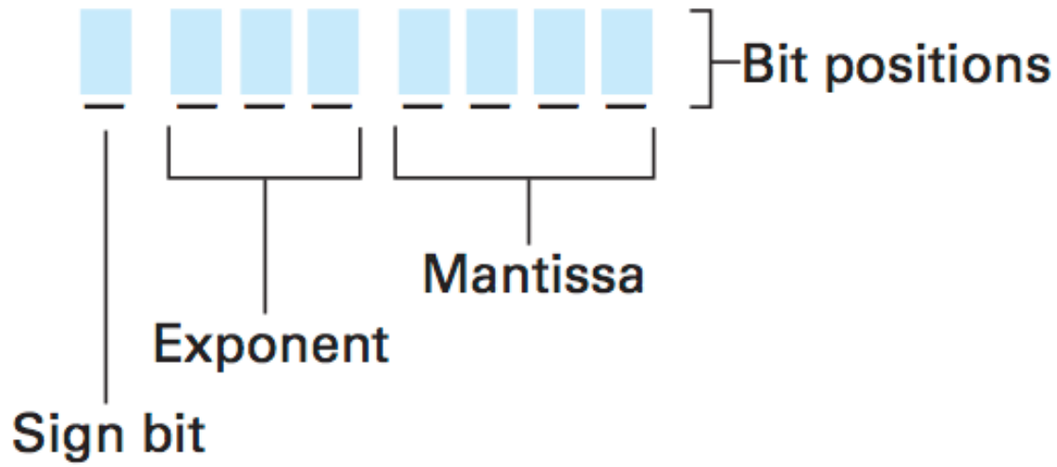
Para codificar 1 1/8 em notação de ponto flutuante, primeiro expressamos o número em binário e obtemos 1.001. Em seguida colocamos o padrão na mantissa

_ _ _ _ **1 0 0 1**

Agora precisamos preencher o campo do expoente. Devemos mover a vírgula um bit à direita, portanto o expoente deve ser o número 1 positivo representado em notação de excesso (*101*)

_ **1 0 1 1 0 0 1**

Finalmente, o número é não-negativo, então preenchemos o 0 no bit de sinal. O byte finalizado fica:



Para codificar 1 1/8 em notação de ponto flutuante, primeiro expressamos o número em binário e obtemos 1.001. Em seguida colocamos o padrão na mantissa

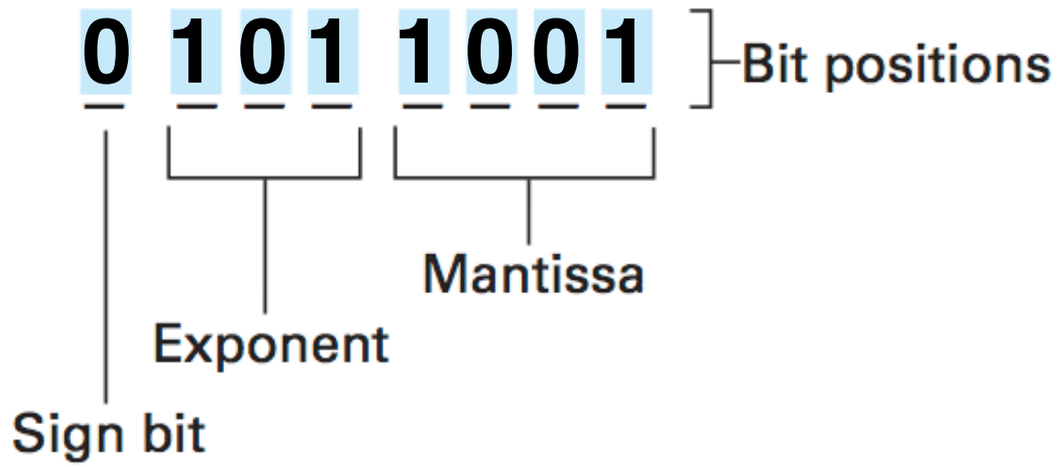
_ _ _ _ **1 0 0 1**

Agora precisamos preencher o campo do expoente. Devemos mover a vírgula um bit à direita, portanto o expoente deve ser o número 1 positivo representado em notação de excesso (*101*)

_ **1 0 1 1 0 0 1**

Finalmente, o número é não-negativo, então preenchemos o 0 no bit de sinal. O byte finalizado fica:

0 1 0 1 1 0 0 1



Para codificar 1 1/8 em notação de ponto flutuante, primeiro expressamos o número em binário e obtemos 1.001. Em seguida colocamos o padrão na mantissa

_ _ _ _ **1 0 0 1**

Agora precisamos preencher o campo do expoente. Devemos mover a vírgula um bit à direita, portanto o expoente deve ser o número 1 positivo representado em notação de excesso (*101*)

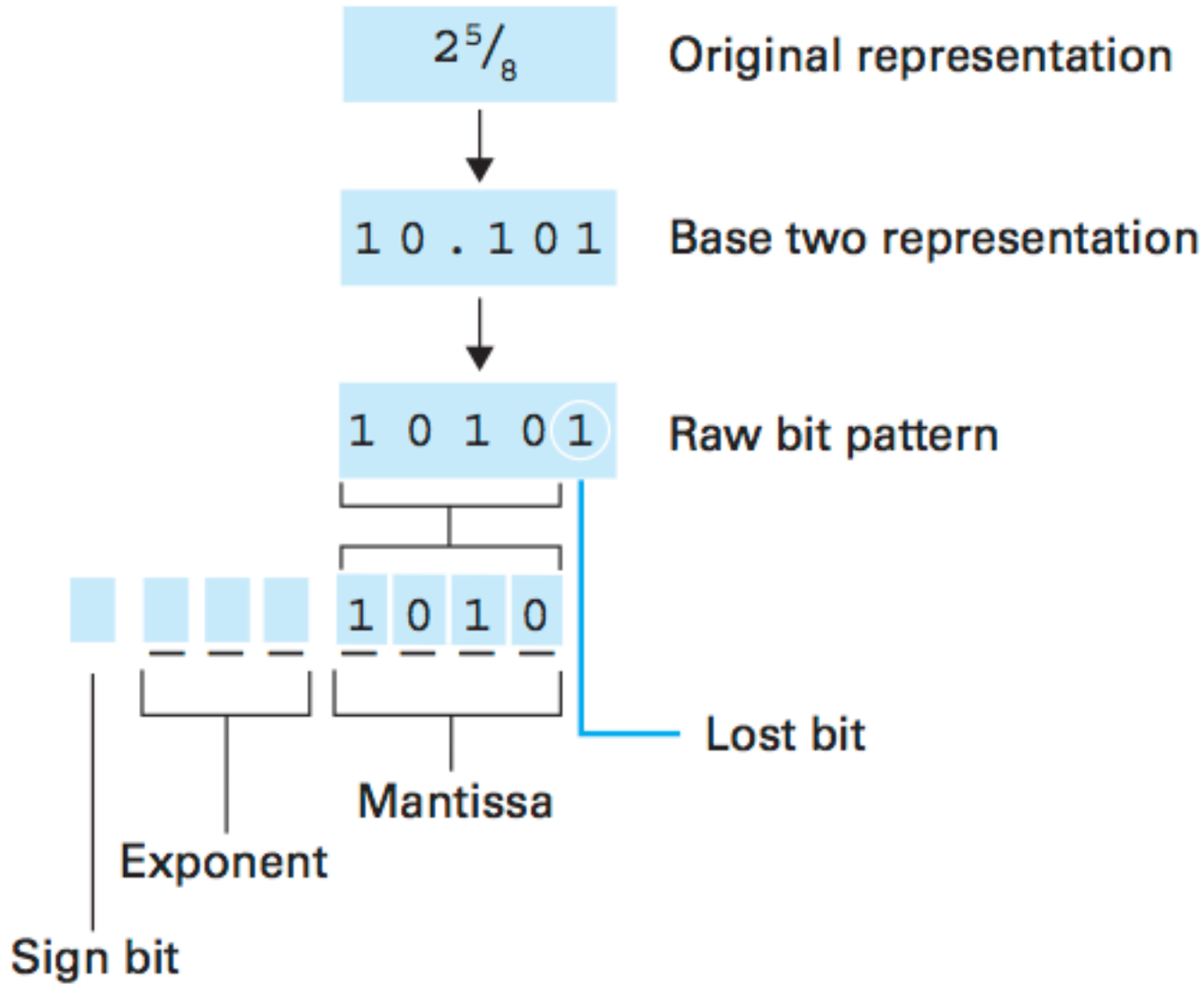
_ **1 0 1 1 0 0 1**

Finalmente, o número é não-negativo, então preenchemos o 0 no bit de sinal. O byte finalizado fica:

0 1 0 1 1 0 0 1

Que problema pode acontecer com este sistema?

Erros de truncamento



Se ignorarmos o bit, terminaremos com 2 $^1/_2$ ao invés de 2 $^5/_8$

Possível solução: aumentar a mantissa

Compressão de Dados

- Geralmente é útil (algumas vezes obrigatório) reduzir o tamanho dos dados, seja para armazená-los ou transferi-los
- Técnicas genéricas podem ser categorizadas como:
  - *Lossless*: informação não é perdida no processo de compressão
  - *Lossy*: podem levar à perda de informação, mas geralmente fornecem maior compressão de dados

Técnicas de Compressão

- Codificação de tamanho de sequência
  - *substituir sequências por um código indicativo do valor repetido e do número de vezes em que ele ocorre*
- Codificação dependente da frequência (Huffman)
  - *usar padrões de tamanho inversamente proporcional à frequência do item codificado*
- Codificação relativa
  - *registrar as diferenças entre blocos consecutivos em vez dos blocos inteiros. Isto é, cada bloco é codificado em termos de sua relação com o bloco precedente*
- Codificação (adaptável) de dicionário

Compressão de Imagens

- GIF (Graphic Interchange Format)
  - reduz o número de cores que podem ser assinaladas a um pixel para 256
  - cada pixel é representado por um byte cujo valor indica a cor
- JPEG (Joint Photographic Experts Group)
  - combina diversos métodos de compressão de imagens
  - comprime blocos de 8x8 px como uma unidade

Compressão de Áudio e Vídeo

- MPEG (*Motion Picture Experts Group*)
  - codificação relativa
  - propriedades do olho humano
- MP3 (*MPEG layer 3*)
  - propriedades do ouvido humano
    - *temporal masking*, *frequency masking*

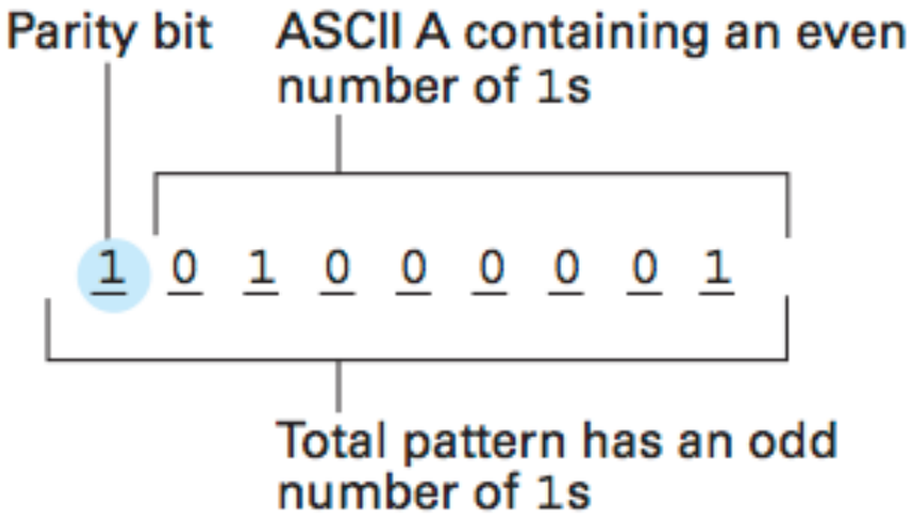
Erros de Comunicação

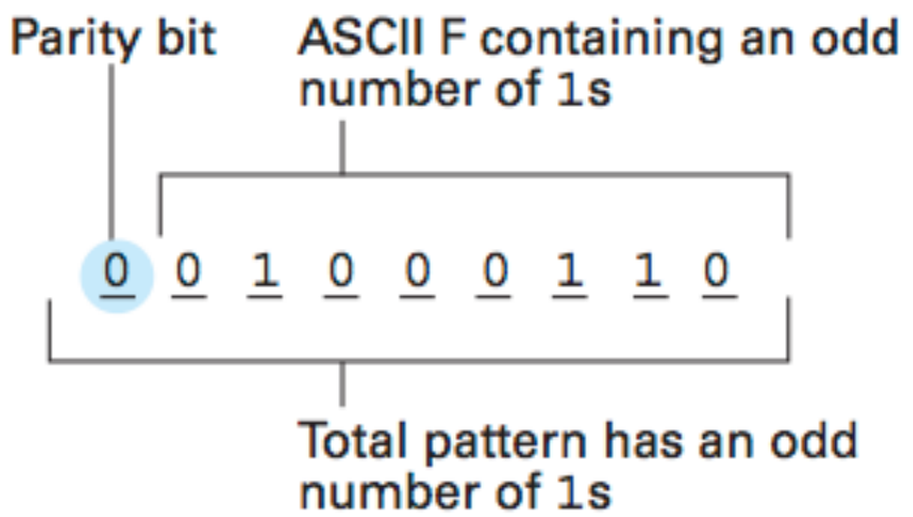
- Quando informação é transferida entre várias partes de um computador ou entre computadores, há a possibilidade de que o padrão de bits retornado não seja idêntico ao original

- Partículas de poeira, ou um circuito falho podem causar erros na gravação ou leitura dos dados

- Para resolver tais problemas, técnicas foram desenvolvidas para permitir a detecção, e até mesmo a correção de erros

Bits de paridade





Códigos de Correção de Erros

Symbol	Code
A	000000
B	001111
C	010011
D	011100
E	100110
F	101001
G	110101
H	111010

Distância de Hamming

Códigos de Correção de Erros

Character	Code	Pattern received	Distance between received pattern and code
A	0 0 0 0 0 0	0 **1** 0 **1** 0 0	2
B	0 0 1 1 1 1	0 **1** **0** 1 **0** **0**	4
C	0 1 0 0 1 1	0 1 0 **1** **0** **0**	3
D	0 1 1 1 0 0	0 1 **0** 1 0 0	**1** — Smallest distance
E	1 0 0 1 1 0	**0** **1** 0 1 **0** 0	3
F	1 0 1 0 0 1	**0** **1** **0** **1** 0 **0**	5
G	1 1 0 1 0 1	**0** 1 0 1 0 **0**	2
H	1 1 1 0 1 0	**0** 1 **0** **1** **0** 0	4

Decodificando 010100

Exercício

- Escreva um programa que converta números decimais em binários e vice-versa
  - Siga os passos do algoritmo, não vale usar funções existentes de Python
  - Tire eventuais dúvidas na lista de discussão da disciplina

- Uma possível execução do programa segue abaixo:

Por favor digite a base (B: binário, D: decimal, S: sair)
B
Por favor, digite um número
1101
O valor deste número na base decimal é 13
Por favor digite a base (B: binário, D: decimal, S: sair)
D
Por favor, digite um número
13
O valor deste número na base binária é 1101
Por favor digite a base (B: binário, D: decimal, S: sair)
S

Trabalho

- Discuta o recém-aprovado Marco Civil da Internet. **Leia o projeto de lei inteiro** (*não é longo*). Discuta pontos positivos e negativos dos artigos, implicações práticas, consequências para usuários a curto e longo prazo, e ao final, dê o seu parecer sobre o projeto.

- Ao concluir, envie por e-mail um arquivo PDF nomeado como ***MarcoCivilNomeSobrenome*.pdf até o dia 21/5.**

- Na **aula do dia 22/5** teremos um tempo para discutir o projeto. **Venham preparados com argumentos bons e embasados, a favor e contra o projeto!**