

Conversões:

- Base qualquer \rightarrow Decimal: Forma Polinomial
 - $507_{10} = 5 \cdot 10^2 + 0 \cdot 10^1 + 7 \cdot 10^0$
 - $1100_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$
- Decimal \rightarrow Base Qualquer: Divisões sucessivas

Ex: $23_{10} \rightarrow 10111_2$

Δ nesta direção

- Binário \leftrightarrow Octal: Separar em grupos de 3 bits

Ex: $1111010110_2 \rightarrow \frac{1}{1} \frac{111}{7} \frac{010}{4} \frac{110}{6} \rightarrow 1746_8$
- Binário \leftrightarrow Hexadecimal: separar em grupos de 4 bits

Ex: $1111010110_2 \rightarrow \frac{11}{3} \frac{1101}{D} \frac{0110}{6} \rightarrow 3D6_{16}$

Adição e Produto em Binário

Ex: $01110001_2 + 11110001_2 = 101101001_2$

Δ se multiplicarmos dois números com m e n dígitos teremos no max. m.n dígitos.

carry que gera overflow

Conversão Decimal \rightarrow Binário (Float)

K=1
F=r₁₀
Do { F = 2.F;
d_K = ParteInteira F;
F = F - d_K;
K++;
} while (F > 0 && K < max_d);

Ex: $3,375_{10}$

F	Res
0,375	0,1
0,75	0,0
1,5	0,01
1	0,011

$\therefore 11,011_2 = 3,375_{10}$

Ponto Flutuante:

- Normalizar o número;
- Se o primeiro bit for 1 é negativo, 0 é positivo.
- Float: 1 bit 8 bits 23 bits

S	Expoente	Mantissa
---	----------	----------

- Double 1 bit 11 bits 52 bits

S	Expoente	Mantissa
---	----------	----------

- Expoente: Float Double

01111111 + expoente₂ (bias 127₁₀) | 01111111 + expoente₂ (bias 1023₁₀)

Valores Especiais:

Float		Double		Objeto Representado
Expoente	Mantissa	Expoente	Mantissa	
0	0	0	0	0
0	não zero	0	não zero	$\pm n^\circ$ desnormalizado
1-254	qualquer coisa	1-254	qualquer coisa	$\pm n^\circ$ pto. flutuante
255	0	255	0	$\pm \infty$
255	não zero	255	não zero	not a number

Byte: 8 bits Nibble: 4 bits

- Kilobyte = 1000 bytes (10^3)
- Kibibyte = 1024 bytes (2^{10})
- MegaByte = 1000 Kilobytes (10^6)
- Mebibyte = 1024 Kibibytes (2^{20})
- GigaByte = 1000 Megabytes (10^9)
- Gibibyte = 1024 Mebibytes (2^{30})

Complemento 1: $(2^d - 1) - N$ representa -N

- bit mais significativo
- 0 se é positivo.
- 1 se é negativo.

como não começa com 1 basta converter; caso comece com 1 é necessário inverter os bits e o valor convertido será negativo.

Ex: $23 - 9$

00010111
+ 11110110

0001101
+1

0001110 (14)

Complemento 2: $2^d - N$ representa -N

Ex: $9_{10} - 23_{10}$

0001001 (9)
11101001 (-23)

11111010

inverte os bits e soma 1
como começa com 1 basta subtrair 1 bit e inverter os bits. Se for positivo basta só converter.

• Overflow: Quando dois valores do mesmo sinal geram um diferente.

ASCII \rightarrow código 7 bits,

$\rightarrow 2^7 = 128$ códigos

char ocupa 1 byte por isso os valores começam com 0

Nº	ASCII	Minúscula	Maíscula
0	011 0000	a 0110 0000	A 0100 0000
1	011 0001	b 0110 0001	B 0100 0001
2	011 0010	c 0110 0010	C 0100 0010
3	011 0011	d 0110 0011	D 0100 0011
4	011 0100	e 0110 0100	E 0100 0100
5	011 0101	f 0110 0101	F 0100 0101
6	011 0110	g 0110 0110	G 0100 0110
7	011 0111	:	:
8	011 1000	y 0111 1000	Y 0100 1000
9	011 1001	z 0111 1001	Z 0100 1001

se zerar viva int

única diferença

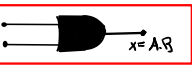
```
int charToInt(char c){
    assert(c >= '0' && c <= '9');
    return (int)c - '0';
}


int intToChar(int c){
    return c % 10 + '0';
}


void toLower(char *str, char *lowerStr){
    int i = 0;
    while(str[i] != '\0'){
        lowerStr[i] = str[i] & 0b11111111;
        i++;
        // toUpper upStr[i] = str[i] & 0b11111111;
    }
    str[i] = '\0';
}
```

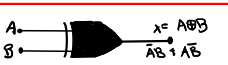
Álgebra de Boole:


• NOT:  $x = \bar{A}$

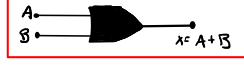
• AND:  $x = A \cdot B$

• NAND:  $x = \overline{A \cdot B}$

• NOR:  $x = \overline{A + B}$

• XOR:  $x = A \oplus B$

• XNOR:  $x = \overline{A \oplus B}$

• OR:  $x = A + B$

Precedência:

- Not Δ caso não haja parêntesis
- AND
- OR

A	B	$\overline{A \cdot B}$	$\overline{A + B}$	$A \oplus B$	$\overline{A \oplus B}$
0	0	1	1	0	1
0	1	1	0	1	0
1	0	1	0	1	0
1	1	0	0	0	1

Soma dos Produtos:

- Pegar as linhas que a função gera 1.
- Realizar o produto (AND) das variáveis envolvidas na linha.
- Negamos as variáveis que aparecem com 0.
- Fazemos a soma (OR) de todos os mintermos, (linhas que geram 1)

Produto das Somas

- Pegar as linhas que a função gera 0.
- Realizar a soma (OR) das variáveis envolvidas na linha.
- Negamos as variáveis que aparecem com 1.
- Fazemos o produto (AND) de todos os mintermos, (linhas que geram 0)

Associatividade:

- $\hookrightarrow A + B + C = (A + B) + C = A + (B + C)$
- $\hookrightarrow A \cdot B \cdot C = (A \cdot B) \cdot C = A \cdot (B \cdot C)$

Comutatividade

- $\hookrightarrow A + B = B + A$
- $\hookrightarrow AB = BA$

Distributividade

- $\hookrightarrow A \cdot (B + C) = AB + AC$
- $\hookrightarrow A + (B \cdot C) = (A + B) \cdot (A + C)$
- $\hookrightarrow (A + B) \cdot (C + D) = AC + AD + BC + BD$
- $\hookrightarrow (A \cdot B) + (C \cdot D) = (A + C) \cdot (B + D)$

Adição Lógica

- $\hookrightarrow A + 0 = A$
- $\hookrightarrow A + 1 = 1$
- $\hookrightarrow A + A = A$
- $\hookrightarrow A + \bar{A} = 1$

Multiplicação Lógica

- $\hookrightarrow A \cdot 0 = 0$
- $\hookrightarrow A \cdot 1 = A$
- $\hookrightarrow A \cdot A = A$
- $\hookrightarrow A \cdot \bar{A} = 0$

Multivariados

- $\hookrightarrow A + AB = A$
- $\hookrightarrow A + \bar{A}B = A + B$
- $\hookrightarrow \bar{A} + AB = \bar{A} + B$

De Morgan

- $\hookrightarrow \overline{AB} = \bar{A} + \bar{B}$
- $\hookrightarrow \overline{A + B} = \bar{A} \cdot \bar{B}$

Mapas de Karnaugh

1. Montar Mapa

		C			
		0	0	1	1
A	B	0	1	1	0
	0	1	2	4	3
0	1	5	6	8	7
	1	13	14	16	15
1	0	9	10	12	11

2. Criar Grupos

- Criar grupos de 1, 2, 4, 8, 16 elementos 2^n
- Grupos são retângulos (1x1, 1x2, 1x4, 2x2, 2x4, 4x4)
- MAX elementos por grupos
- MIN números de grupos
- Cada 1 em pelo menos um grupo.

3. Criar FND

- Cada grupo é uma cláusula (conjunção)
- Fazer disjunção das cláusulas
- Quais variáveis são comuns para todos os elementos do grupo
 - \rightarrow Se for 1 entra normal na conjunção $\rightarrow A$
 - \rightarrow Se for 0 entra invertido na conjunção $\rightarrow \bar{A}$

Don't Care: Caso tenha basta escolher o que trouxer a melhor simplificação.