Capítulo V
Sistemas Numéricos

1 Introdução

Em capítulos anteriores estudamos diversas funções lógicas. No próximo capítulo veremos que operações aritméticas como soma e subtração de números binários podem ser implementadas através da combinação de funções lógicas. Estas funções lógicas aritméticas, quando reunidas em um único CI, constituem uma Unidade Lógica e Arimética (ULA) ou, em inglês, ALU (*Arithmetic and Logic Unit*). Uma ULA é um componente lógico fundamental de um microprocessador, e todas as operações aritméticas por ela realizadas são efetuadas com números binários.

Neste contexto, o presente capítulo estuda a aritmética com números binários e a conversão entre números em base binária e números em outras bases, como as bases octal, decimal e hexadecimal.

2 Números Decimais

● Números em base decimal constituem os números com os quais naturalmente estamos habituados a trabalhar. O termo "naturalmente" surge do fato de possuirmos dez dedos nas mãos, o que levou os povos antigos que deram origem a nossa civilização a adotarem um sistema de contagem em base dez.

● Um número em uma base numérica **qualquer** pode ser decomposto em uma **soma de potências da base, ponderadas por um dos dígitos do conjunto de dígitos que definem a base**.

●Por exemplo, consideremos o número decimal 271.8281. Na base decimal o conjunto de dígitos é $\{0,1,2,3,4,5,6,7,8,9\}$, e estes dígitos constituem os **Fatores de Ponderação** de cada **Potência da Base**. Daí, este número pode ser decomposto na forma:

Potência da Base	10^2	10^1	10^0		10^{-1}	10^{-2}	10^{-3}	10^{-4}
Fator de Ponderação	2	7	1	.	8	2	8	1

ou, em termos analíticos:

$$271.8281 = 2\times10^2 + 7\times10^1 + 1\times10^0 + 8\times10^{-1} + 2\times10^{-2} + 8\times10^{-3} + 1\times10^{-4} \quad (1)$$

● Com um número decimal formado por N casas ou dígitos decimais podemos efetuar a contagem numérica de até 10^N objetos. Por exemplo, para $N=2$ podemos enumerar objetos de 0 a 99, totalizando $10^N=10^2=100$ objetos.

3 Números Binários

● Na base binária o conjunto de dígitos é $\{0,1\}$, e estes dígitos constituem os **Fatores de Ponderação** no somatório de **Potências da Base** (2^n) que representa analiticamente o número.

● Cada dígito binário do conjunto $\{0,1\}$ é denominado **bit** (***binary unit***).

●Por exemplo, o número binário 101.1101 pode ser decomposto na forma:

Potência da Base	4	2	1		$1/2$	$1/4$	$1/8$	$1/16$
Fator (bit) de Ponderação	1	0	1	.	1	1	0	1

ou, em termos analíticos:

$$101.1101=1\times2^2+1\times2^0+1\times2^{-1}+1\times2^{-2}+1\times2^{-4} \tag{2}$$

●Para converter um número binário em decimal basta efetuar o somatório de potências 2^n cujo bit de ponderação seja "1":

$$101.1101_2=4+1+1/2+1/4+1/16=5.8125_{10} \tag{3}$$

● Associado a qualquer número binário existem os conceitos de **Bit Mais Significativo (MSB – *Most Significant Bit*)** e de **Bit Menos Significativo (LSB – *Least Significant Bit*)**. Por exemplo, para o número 101.1101, os MSB e LSB das partes inteiras e fracionais são aqueles associados às maiores potências 2^n de cada parte:

	Parte Inteira				Parte Fracional			
Potência da Base	4	2	1		$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$
Número Binário	1	0	1	.	1	1	0	1
	↑ MSB		↑ LSB		↑ MSB			↑ LSB

● Com um número binário formado por N bits podemos efetuar a contagem numérica de até 2^N objetos. Por exemplo, para $N = 4$ podemos enumerar objetos de 0 a 15, totalizando $2^N = 2^4 = 16$ objetos:

Número Binário de 4 bits				Número Decimal
b_3 (MSB)	b_2	b_1	b_0 (LSB)	
0	0	0	0	0
0	0	0	1	1
0	0	**1**	0	2
0	0	1	1	3
0	**1**	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

Tabela 1: Contagem binária de 0 a 15. Os bits em vermelho mostram os instantes da contagem em que é necessário lançar mão do recurso do bit “vai-um” (*carry*) em conseqüência de ter sido esgotado a capacidade de contagem dos bits menos significativos utilizados até o instante em consideração.

3.1 Conversão Decimal para Binário

● Para converter para binário as partes não-fracional e fracional de um número decimal utiliza-se, respectivamente, os métodos conhecidos como **Divisão Repetida** e a **Multiplicação Repetida**.

● Por exemplo, para converter o número decimal 45.3125 em binário, primeiramente decompomos o número na soma das partes não-fracional e fracional: $45.3125 = 45 + 0.3125$.

● Daí, aplicamos a **Divisão Repetida** à parte **não-fracional**:

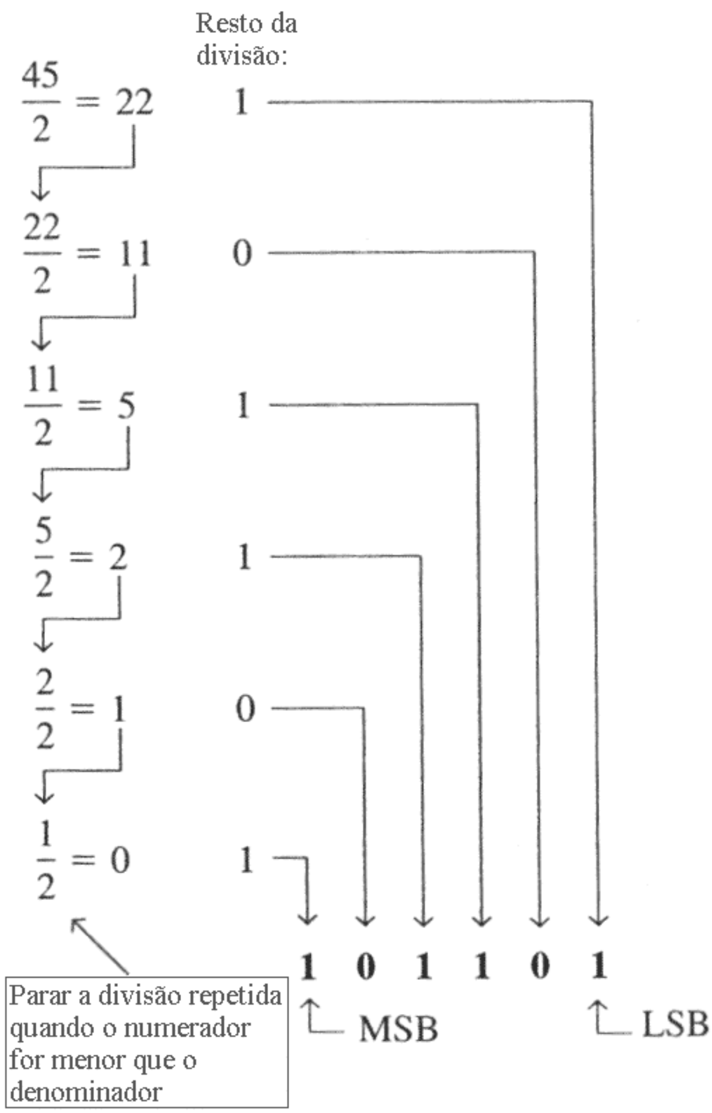

Figura 1: Conversão do número 45_{10} para seu equivalente em binário através do método Divisão por 2 Repetida.

● A seguir, aplicamos a **Multiplicação Repetida** à parte **fracional**:

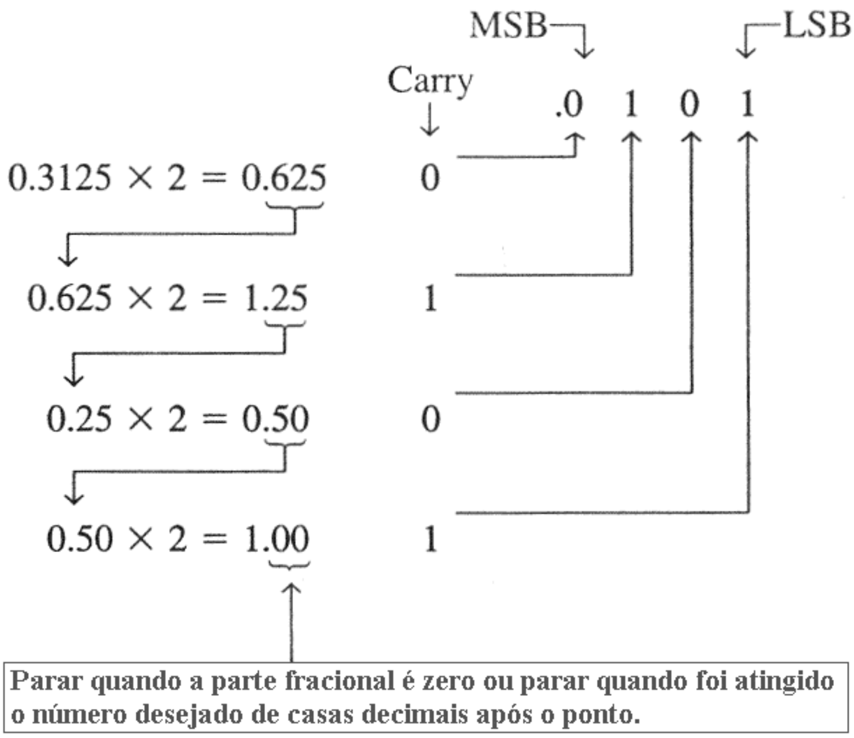

Figura 2: Conversão do número 0.3125_{10} para seu equivalente em binário através do método Multiplicação por 2 Repetida.

- Portanto, $45.3125_{10} = 101101.0101_2$.

4 Aritmética Binária Entre Números sem Sinal

4.1 Adição

- Para efetuar a adição $A+B$ entre duas palavras binárias A e B executa-se entre cada bit de A e respectivo bit de B uma das 4 possíveis operações básicas abaixo definidas, de acordo com o valor do bit de A e respectivo bit de B:

$$0+0=0$$

$$0+1=1$$

$$1+0=1$$

$$1+1=10 \rightarrow$$

O "1" no resultado é o "vai-um" (*carry*) gerado por ter sido esgotado a capacidade de contagem. O *carry* deve ser acrescentado à soma dos bits imediatamente mais significativos à esquerda daqueles que deram origem ao *carry* .

Exemplo 1: Calcule as somas binárias (a) $11+11$ (b) $100+10$ (c) $111+11$ (d) $110+100$ fazendo simultaneamente a soma dos números decimais equivalentes.

Solução:

(a) $\begin{array}{r} 11 \\ +\ 11 \\ \hline \mathbf{110} \end{array}$ $\quad\begin{array}{r} 3 \\ +\ 3 \\ \hline 6 \end{array}$ **(b)** $\begin{array}{r} 100 \\ +10 \\ \hline \mathbf{110} \end{array}$ $\quad\begin{array}{r} 4 \\ +\ 2 \\ \hline 6 \end{array}$ **(c)** $\begin{array}{r} 111 \\ +\ 11 \\ \hline \mathbf{1010} \end{array}$ $\quad\begin{array}{r} 7 \\ +\ 3 \\ \hline 10 \end{array}$ **(d)** $\begin{array}{r} 110 \\ +\ 100 \\ \hline \mathbf{1010} \end{array}$ $\quad\begin{array}{r} 6 \\ +\ 4 \\ \hline 10 \end{array}$

4.2 Subtração

- A maneira mais eficiente para efetuar a subtração $A-B$ entre duas palavras binárias A e B é executar a operação $A+\mathrm{C}^{\mathrm{II}}\{B\}$ onde $\mathrm{C}^{\mathrm{II}}\{\cdot\}$ é o operador denominado **Complemento de 2**. **A operação $\mathrm{C}^{\mathrm{II}}\{\cdot\}$ é equivalente a acrescentar o sinal “ – ” ao número binário**.

- A operação $\mathrm{C}^{\mathrm{II}}\{B\}$ efetuada sobre uma palavra binária B é dada por $\mathrm{C}^{\mathrm{II}}\{B\}=\mathrm{C}^{\mathrm{I}}\{B\}+1$, onde $\mathrm{C}^{\mathrm{I}}\{B\}$ é a operação de inversão (NOT) do valor lógico de cada bit da palavra binária B (operação conhecida como **Complemento de 1**).

- Por exemplo, a diferença $A-B$ entre os números $A=0110_2=6_{10}$ e $B=0100_2=4_{10}$ é dada por:

$$A-B=A+\mathrm{C}^{\mathrm{II}}\{B\}=A+\mathrm{C}^{\mathrm{I}}\{B\}+1=0110+\mathrm{C}^{\mathrm{I}}\{0100\}+1=0110+1011+1 \quad (4)$$

Que resulta em:

$$\begin{array}{rcccc} & 1 & 0 & 1 & 1 \\ + & 0 & 0 & 0 & 1 \\ \hline & 1 & 1 & 0 & 0 \end{array}$$

prosseguindo:

$$\begin{array}{rccccc} & & 0 & 1 & 1 & 0 \\ & + & 1 & 1 & 0 & 0 \\ \hline \text{descartar o } \textit{carry} \rightarrow & (1) & 0 & 0 & 1 & 0 \end{array} = 2_{10}$$

●Alternativamente, podemos implementar a operação $\mathrm{C}^{\mathrm{II}}\{B\}$ através do seguinte procedimento: **Efetuamos a leitura da palavra binária B da direita para a esquerda até encontrarmos o primeiro "1" e a seguir invertemos o valor lógico de todos os bits à esquerda do primeiro "1"**.

●Por exemplo, vamos supor que queremos achar o Complemento de 2 do número binário $A = 10110_2 = 22_{10}$. Utilizando a técnica descrita no parágrafo anterior temos $\mathrm{C}^{\mathrm{II}}\{10110\} = 01010$. Se o resultado estiver correto, então $A + \mathrm{C}^{\mathrm{II}}\{A\} = 00000$, porque a operação $\mathrm{C}^{\mathrm{II}}\{A\}$ é equivalente a efetuarmos o acréscimo do sinal negativo, isto é, $-\mathrm{C}^{\mathrm{II}}\{A\}$. Senão, vejamos:

$$\begin{array}{rcccccc} & & 1 & 0 & 1 & 1 & 0 \\ & + & 0 & 1 & 0 & 1 & 0 \\ \hline \text{descartar o } \textit{carry} \rightarrow & (1) & 0 & 0 & 0 & 0 & 0 \end{array}$$

5 Aritmética Binária Entre Números com Sinal em Complemento de 2

5.1 Adição A+B

⇒ Soma-se ambos os números e descarta-se o *carry*. Por exemplo, sejam as seguintes somas de números de 8 bits:

- A e B são positivos:

$$\begin{array}{r} 00000111 \\ +\ 00000100 \\ \hline 00001011 \end{array} \qquad \begin{array}{r} 7 \\ +\ 4 \\ \hline 11 \end{array}$$

- |A| > |B| com B < 0:

$$\begin{array}{rr} & 00001111 \\ & +\ 11111010 \\ \hline \text{descartar o carry} \longrightarrow 1 & 00001001 \end{array} \qquad \begin{array}{r} 15 \\ +\ -6 \\ \hline 9 \end{array}$$

- |A| < |B| com B < 0:

$$\begin{array}{r} 00010000 \\ +\ 11101000 \\ \hline 11111000 \end{array} \qquad \begin{array}{r} 16 \\ +\ -24 \\ \hline -8 \end{array}$$

- A e B são negativos:

$$\begin{array}{rr} & 11111011 \\ + & 11110111 \\ \hline \text{descartar o carry} \longrightarrow 1 & 11110010 \end{array} \qquad \begin{array}{r} -5 \\ + \; -9 \\ \hline -14 \end{array}$$

- *Overflow* (ocorre quando o número de bits necessário para representar a soma excede o número de bits dos números sendo somados):

$$\begin{array}{rr} & 01111101 \\ + & 00111010 \\ \hline & 10110111 \end{array} \qquad \begin{array}{r} 125 \\ + \quad 58 \\ \hline 183 \end{array}$$

sinal incorreto

magnitude incorreta

Nota: No exemplo acima, o número resultante 183 requer 8 bits de magnitude para ser representado. No entanto, na aritmética em Complemento de 2 o MSB (o 8° bit no caso) é o bit representativo do sinal, sendo a magnitude representada pelos bits menos significativos restantes à direita. Portanto, neste exemplo, ocorre a adição de um *carry* ao MSB responsável pelo sinal, invalidando o resultado (*overflow*) na aritmética em Complemento de 2. Note que somente pode ocorrer *overflow* quando A e B são positivos ou A e B são negativos.

5.2 Subtração A - B

⟹ Executa-se $A + \mathrm{C}^{\mathrm{II}}\{B\}$ e descarta-se o *carry*.

Exemplo 2: Calcule as seguintes somas de números de 8 bits:

(a) 00001000 − 00000011 **(b)** 00001100 − 11110111
(c) 11100111 − 00010011 **(d)** 10001000 − 11100010

Solução:

(a) $8 - 3 = 8 + (-3) = 5$

$$\begin{array}{rrl} & 00001000 & \text{Minuend } (+8) \\ & +\ 11111101 & \text{2's complement of subtrahend } (-3) \\ \hline \text{Discard carry} \longrightarrow & \mathbf{1\ \ 00000101} & \text{Difference } (+5) \end{array}$$

(b) $12 - (-9) = 12 + 9 = 21$

00001100	Minuend (+12)
+ 00001001	2's complement of subtrahend (+9)
00010101	Difference (+21)

(c) $-25 - (+19) = -25 + (-19) = -44$

	11100111	Minuend (−25)
	+ 11101101	2's complement of subtrahend (−19)
Discard carry ⟶	**1 11010100**	Difference (−44)

(d) $-120 - (-30) = -120 + 30 = -90$

10001000	Minuend (−120)
+ 00011110	2's complement of subtrahend (+30)
10100110	Difference (−90)

6 Números Hexadecimais

● Um número em base binária apresenta o inconveniente de necessitar um grande número de dígitos para sua representação. Por exemplo, o número 9 em base decimal é representado por um único dígito 9_{10}, mas se representado em base binária são necessários 4 dígitos (bits): $1001_2 = 9_{10}$.

⟹ Note que **quanto maior for a base menor será o número de dígitos da base necessários para a representação do número**.

● Uma forma conveniente para a representação de números binários é a base 16 (base hexadecimal), porque 16 é uma potência inteira de 2 o que, conforme veremos, facilita bastante a conversão entre as duas bases.

● Com um número hexadecimal formado por N dígitos podemos efetuar a contagem numérica de até 16^N objetos. Por exemplo, para $N = 1$ podemos enumerar objetos de 0 a 15, totalizando $16^N = 16^1 = 16$ objetos:

Decimal	Binary	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

Tabela 2: Contagem hexadecimal de 0 a 15 e equivalentes decimal e binário.

Note que na base hexadecimal o conjunto de dígitos possíveis é $\{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$.

● Como em qualquer base numérica, o "vai-um" (*carry*) na base hexadecimal ocorre em conseqüência de ter sido esgotado a capacidade de contagem dos dígitos menos significativos. Por exemplo, a continuação da contagem hexadecimal da Tabela 2 seria:

10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E, 1F, 20, 21, 22, 23
24, 25, 26, 27, 28, 29, 2A, 2B, 2C, 2D, 2E, 2F, 30, 31, . . .

6.1 Conversão Binário para Hexadecimal

⇒ Começando da direita para esquerda, subdivide-se o número binário em grupos de 4 bits (*nibbles*), e substitui-se pelo equivalente hexadecimal da Tabela 2. Se não for possível formar um grupo completo de 4 bits à esquerda do número binário, acrescenta-se 1,2 ou 3 zeros para tanto.

Exemplo 3:

Converta os seguintes números binários para hexadecimal:

(a) 1100101001010111 **(b)** 111111000101101001

Solução:

(a) $\underbrace{1100}\underbrace{1010}\underbrace{0101}\underbrace{0111}$ → C A 5 7 = $\mathbf{CA57_{16}}$

(b) $\underbrace{0011}\underbrace{1111}\underbrace{0001}\underbrace{0110}\underbrace{1001}$ → 3 F 1 6 9 = $\mathbf{3F169_{16}}$

Dois zeros foram adicionados em (b) p/ completar o grupo de 4 bits à esquerda.

6.2 Conversão Hexadecimal para Binário

⇒ Executa-se o processo inverso do apresentado na Seção 6.1, isto é, substitui-se cada dígito hexadecimal pelo *nibble* equivalente de acordo com a Tabela 2.

Exemplo 4:

Determine os números binários correspondentes aos seguintes números hexadecimais:

(a) $10A4_{16}$ **(b)** $CF8E_{16}$ **(c)** 9742_{16}

Solução:

(a) 1 0 A 4 → **1000010100100**

(b) C F 8 E → **1100111110001110**

(c) 9 7 4 2 → **1001011101000010**

O MSB em (a) implicitamente deve ser interpretado como tendo 3 zeros precedentes, de modo a formar um grupo de 4 bits.

6.3 Conversão Hexadecimal para Decimal

Exemplo 5:

Converta os seguintes números hexadacimais para números decimais:

(a) $1C_{16}$ **(b)** $A85_{16}$

Solução: (convertendo 1° para binário e depois para decimal)

(a) 1 C → 00011100 $= 2^4 + 2^3 + 2^2 = 16 + 8 + 4 = \mathbf{28_{10}}$

(b) A 8 5 → 101010000101 $= 2^{11} + 2^9 + 2^7 + 2^2 + 2^0 = 2048 + 512 + 128 + 4 + 1 = \mathbf{2693_{10}}$

Exemplo 6:

Converta os seguintes números hexadacimais para números decimais:
(a) $E5_{16}$ **(b)** $B2F8_{16}$

Solução:

(a) $E5_{16} = (E \times 16) + (5 \times 1) = (14 \times 16) + (5 \times 1) = 224 + 5 = \mathbf{229}_{10}$
(b) $B2F8_{16} = (B \times 4096) + (2 \times 256) + (F \times 16) + (8 \times 1)$
$= (11 \times 4096) + (2 \times 256) + (15 \times 16) + (8 \times 1)$
$= 45{,}056 + 512 + 240 + 8 = \mathbf{45{,}816}_{10}$

6.4 Conversão Decimal para Hexadecimal

Exemplo 7:

Converta o número decimal 650 para hexadecimal, utilizando a divisão repetida.

Solução:

resto:

$\frac{650}{16} = 40.625 \rightarrow 0.625 \times 16 = 10 = A$

$\frac{40}{16} = 2.5 \longrightarrow 0.5 \times 16 = 8 = 8$

$\frac{2}{16} = 0.125 \longrightarrow 0.125 \times 16 = 2 = 2$

Parar quando o numerador for menor que o denominador.

2 8 A **número hexadecimal**

MSD LSD

7 Números Octais

- Uma outra forma conveniente para a representação de números binários é a base 8 (base octal), porque 8 é uma potência inteira de 2 o que, conforme vimos para a base hexadecimal, facilita a conversão entre as duas bases.
- Na base octal o conjunto de dígitos possíveis é $\{0,1,2,3,4,5,6,7\}$.
- A enumeração de objetos na base octal é como segue:

0, 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17, 20, 21 ...

7.1 Conversão Octal para Decimal

Exemplo 8:

Converter para decimal o número 2374_8 .

Solução:

$$\begin{array}{cccc} 8^3 & 8^2 & 8^1 & 8^0 \\ 2 & 3 & 7 & 4 \end{array}$$

$$\begin{aligned} 2374_8 &= (2 \times 8^3) + (3 \times 8^2) + (7 \times 8^1) + (4 \times 8^0) \\ &= (2 \times 512) + (3 \times 64) + (7 \times 8) + (4 \times 1) \\ &= 1024 + 192 + 56 + 4 = 1276_{10} \end{aligned}$$

7.2 Conversão Decimal para Octal

Exemplo 9:

Converter para octal o número 359_{10} .

Solução:

resto:

$$\frac{359}{8} = 44.875 \rightarrow 0.875 \times 8 = 7$$

$$\frac{44}{8} = 5.5 \longrightarrow 0.5 \times 8 = 4$$

$$\frac{5}{8} = 0.625 \longrightarrow 0.625 \times 8 = 5$$

Parar quando o numerador for menor que o denominador

5 4 7 **número octal**

MSD LSD

7.3 Conversão Octal para Binário

⇒ Substitui-se cada dígito hexadecimal pela palavra binária de 3 bits equivalente, de acordo com a Tabela 3 a seguir:

Octal/binary conversion.

Octal Digit	0	1	2	3	4	5	6	7
Binary	000	001	010	011	100	101	110	111

Tabela 3: Equivalência entre dígito octal – palavra binária correspondente.

Exemplo 10:

Converta para binário:

(a) 13_8 **(b)** 25_8 **(c)** 140_8 **(d)** 7526_8

Solução:

(a) $\begin{matrix} 1 & 3 \\ \downarrow & \downarrow \\ \overbrace{\mathbf{001}} & \overbrace{\mathbf{011}} \end{matrix}$ **(b)** $\begin{matrix} 2 & 5 \\ \downarrow & \downarrow \\ \overbrace{\mathbf{010}} & \overbrace{\mathbf{101}} \end{matrix}$ **(c)** $\begin{matrix} 1 & 4 & 0 \\ \downarrow & \downarrow & \downarrow \\ \overbrace{\mathbf{001}} & \overbrace{\mathbf{100}} & \overbrace{\mathbf{000}} \end{matrix}$ **(d)** $\begin{matrix} 7 & 5 & 2 & 6 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ \overbrace{\mathbf{111}} & \overbrace{\mathbf{101}} & \overbrace{\mathbf{010}} & \overbrace{\mathbf{110}} \end{matrix}$

7.4 Conversão Binário para Octal

⇒ Começando da direita para esquerda, subdivide-se o número binário em grupos de 3 bits, e substitui-se pelo equivalente dígito octal da Tabela 3. Se não for possível formar um grupo completo de 3 bits à esquerda do número binário, acrescenta-se 1 ou 2 zeros para tanto.

Exemplo 11:

Converta para octal:

(a) 110101 **(b)** 101111001 **(c)** 100110011010 **(d)** 11010000100

Solução:

(a) $\begin{matrix} \underbrace{110} & \underbrace{101} & \\ \downarrow & \downarrow & \\ 6 & 5 & = \mathbf{65_8} \end{matrix}$ **(b)** $\begin{matrix} \underbrace{101} & \underbrace{111} & \underbrace{001} & \\ \downarrow & \downarrow & \downarrow & \\ 5 & 7 & 1 & = \mathbf{571_8} \end{matrix}$

(c) $\begin{matrix} \underbrace{100} & \underbrace{110} & \underbrace{011} & \underbrace{010} & \\ \downarrow & \downarrow & \downarrow & \downarrow & \\ 4 & 6 & 3 & 2 & = \mathbf{4632_8} \end{matrix}$ **(d)** $\begin{matrix} \underbrace{011} & \underbrace{010} & \underbrace{000} & \underbrace{100} & \\ \downarrow & \downarrow & \downarrow & \downarrow & \\ 3 & 2 & 0 & 4 & = \mathbf{3204_8} \end{matrix}$