REPRESENTAÇÃO DE NÚMEROS E ARITMÉTICA COMPUTACIONAL, SINAIS, VÍRGULAS E ESTOUROS

SINAIS, CONFORME SERÁ VISTO, UTILIZA-SE UM BIT A MAIS (SINAL E MAGNITUDE) OU COMPLEMENTO DE DOIS.

COM A RELAÇÃO A REPRESENTAÇÃO DE NÚMEROS FRACIONÁRIOS:

COMO FOI VISTO: A REPRESENTAÇÃO DE QUALQUER INFORMAÇÃO NO COMPUTADOR É FEITA ATRAVÉS DE '0s' E '1s'.

ENTÃO COMO REPRESETRAR NÚMEROS COM SINAL E NÚMEROS FRACIONÁRIOS?

POR EXEMPLO, ESTE:

(+ 37 , 5)10

+ 37 , 5

FÁCIL, FÁCIL:

00101011	00110011	00011011	00101100	00110101
+	3	7	,	5

ESSA REPRESENTAÇÃO, **EM ASCII**, SOMENTE É UTILIZADA QUANDO O NÚMERO É INTRODUZIDO NO COMPUTADOR COMO UM **TEXTO** LIVRE.

OBVIAMENTE, NESSA FORMA O NÚMERO INTRODUZIDO NÃO PODERÁ SER OPERADO.

NA SUA REPRESENTAÇÃO BINÁRIA PURA A VÍRGULA NÃO É REPRESENTADA, MAS SIM, **ASSUMIDADE** SUA POSIÇÃO NO NÚMERO. POR EXEMPLO:

O NÚMERO : 110111,110

SERIA REPRESENTADO INTERNAMENTE COMO 11011110 E O SISTEMA, DE ALGUMA FORMA, SABERIA QUE OS 3 ÚLTIMOS ALGARISMOS À DIREITA SERIA A PARTE FRACIONÁRIA

QUANTO AOS ESTOUROS:

NA NOSSA MATEMÁTICA NÃO EXISTEM LIMITES A NÃO SER QUANTO AO TAMANHO DO PAPEL.

QUANTOS NÚMEROS EXISTEM ENTRE 3,01 E 3,02?

3,01

3,011

3,012

3,013

3,012245

..... 3,02

ENTRETANTO, NO COMPUTADOR, DEVIDO A SUA NATUREZA FINITA DE REPRESENTAÇÃO, OS ERROS NAS OPERAÇÕES ARITMÉTICAS, PODEM OCORRER DE MUITAS E PERIGOSAS MANEIRAS.

EXEMPLO: SEJAM OS NÚMEROS DECIMAIS COM DOIS ALGARISMOS:

A = 45 , B = 63 e C = -53

CONSIDERANDO A OPERAÇÃO: A + (B + C)

TERÍAMOS : 45 + (63 – 53)=

45 + 10 = 55

ENTRETANTO, COM OS MESMOS NÚMEROS:

A = 45 , B = 63 e C = -53

CONSIDERANDO AGORA A OPERAÇÃO: (A + B) + C NÃO PODERIAMOS OBTER O RESULTADO CORRETO

$$(A + B) + C =$$

$$(45 + 63) - 53 =$$

$$108 - 53 = 55$$

OU NÃO SERIA

$$108 - 53 = -45?$$

REPRESENTAÇÃO DE NÚMEROS EM PONTO FIXO E EM PONTO FLUTUANTE

VAMOS SUPOR QUE O TAMANHO DA MEMÓRIA SEJA DE OITO ESPAÇOS, EM CADA ESPAÇO PODEMOS ARMAZENAR UM ALGARISMO DA BASE DECIMAL OU SEJA DE 0 A 9.

POR EXEMPLO, O NÚMERO ZERO SERIA REPRESENTADO POR:

CONSIDERANDO APENAS VALORES INTEIROS POSITIVOS PODERIAMOS ARMAZENAR 10**8 VALORES DISTINTOS E O MAIOR VALOR INTEIRO POSITIVO SERIA:

9	9	9	9	9	9	9	9

COMO PODERÍAMOS ARMAZENAR TAMBÉM VALORES INTEIROS NEGATIVOS?

UMA FORMA SIMPLES, É A VELHA HISTÓRIA: RESERVAR UM ESPAÇO DE MEMÓRIA DOS 8 EXISTENTES PARA ARMAZENAR O SINAL DO NÚMERO.

SE NESSE ESPAÇO RESERVADOR O VALOR ARMAZENADO FOR 0, O NÚMERO É POSITIVO SE FOR 1 O NÚMERO É NEGATIVO:

POR EXEMPLO O NÚMERO - 8345 SERIA REPRESENTADO POR:

1	0	0	0	8	3	4	5

E PARA ARMAZENAR NÚMEROS REAIS? EXISTEM DUAS MANEIRAS BASTANTE UTILIZADAS PARA ISSO:

PRIMEIRA: RESERVAMOS UMA PARTE DAS 8 POSIÇÕES PARA ARMAZENAR A PARTE INTEIRA E UMA OUTRA PARTE PARA ARMAZENAR A PARTE FRACIONÁRIA DO NÚMERO EM QUESTÃO.

COM AMBAS AS PARTES SEGUINDO A REPRESENTAÇÃO DE NÚMEROS INTEIROS JÁ VISTO ANTERIORMENTE.

VAMOS SUPOR QUE RESERVAMOS 5 POSIÇÕES PARA ARMAZENAR A PARTE INTEIRA DO NÚMERO, INCLUINDO O SEU SINAL, E AS TRÊS POSIÇÕES RESTANTES FICAM RESERVADAS PARA A PARTE FRACIONÁRIA.

O NÚMERO – 345.9 SERIA ENTÃO REPRESENTADO POR:

1	0	3	4	5	9	0	0

ESSE TIPO DE REPRESENTAÇÃO É CONHECIDA COM EM PONTO FIXO POIS O PONTO DECIMAL ESTÁ SEMPRE IMPLICITAMENTE REPRESENTADO NA MESMA POSIÇÃO. LIMITAÇÕES: O MAIOR E O MENOR NÚMERO REAL QUE PODERIAM SER ARMAZENADOS NA MÁQUINA SERIAM:

+9999.999 e –9999.999

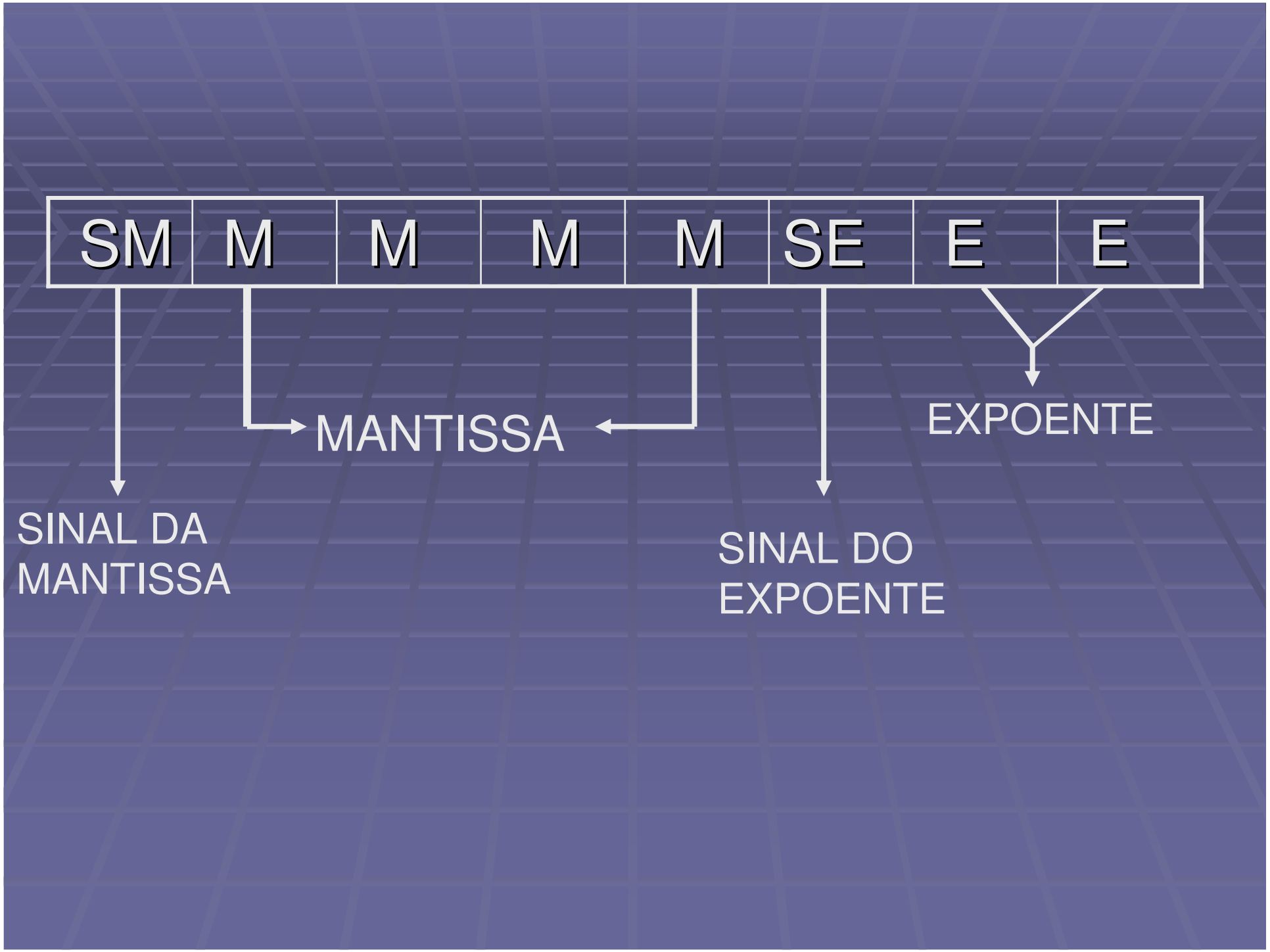
+9999.999 e –9999.999

0	9	9	9	9	9	9	9
1	9	9	9	9	9	9	9

SEGUNDA MANEIRA: ESSA MANEIRA DE REPRESENTAR NÚMEROS REAIS UTILIZA A NOTAÇÃO CIENTÍFICA A QUAL ESCREVERIA O NÚMERO – 345.9 COMO:

$- 0.3459 \times 10^3$

É Reservado uma parte das posições para representar o (3) expoente da base 10 (com sinal) e o restante para representar a mantissa (parte fracionária, com sinal) (5)


SM
M
M
M
M
SE
E
E
MANTISSA
EXPOENTE
SINAL DA MANTISSA
SINAL DO EXPOENTE

O NÚMERO: –345.9

QUE É IGUAL A -0.3459×10^{3}

REPRESENTADO DESSA FORMA, FICARIA:

1	3	4	5	9	0	0	3

ESSE TIPO DE REPRESENTAÇÃO É CONHECIDA COMO REPRESENTAÇÃO **EM PONTO FLUTUANTE** POIS A POSIÇÃO DO PONTO DECIMAL DO NÚMERO VARIA CONFORME O VALOR DO EXPOENTE.

LIMITAÇÕES: PODEMOS AGORA REPRESENTAR VALORES REAIS QUE VARIAM

DE: + ou - 0.9999×10^{99}

ATÉ: + ou - 0.9999×10^{-99}

OVERFLOW E UNDERFLOW NA NOSSA MAQUININHA:

SUPONHA QUE QUEREMOS MULTIPLICAR:

0.23×10^{50} POR -0.455×10^{60}

QUAL SERIA O RESULTADO?

$0.23 \times 10^{50} \quad X \quad -0.455 \times 10^{60}$

0	2	3	0	0	0	5	0
1	4	5	5	0	0	6	0

-0.10456×10^{110}

ESSE VALOR NÃO PODERIA SER ARMAZENADO POIS O EXPOENTE É MAIOR DO QUE 99.

ESSE TIPO DE ERRO É CONHECIDO COMO: OVERFLOW

UM ERRO SIMILAR A ESSE É QUANDO O EXPOENTE ALCANÇA UM VALOR NEGATIVO QUE TAMBÉM NÃO POSSA SER REPRESENTADO.

POR EXEMPLO, QUANTO TEMOS UM EXPONTE EM 10^{-110}.

NESSE CASO O VALOR É MUITO PRÓXIMO DE ZERO, MAS NÃO PODE SER PRECISAMENTE ARMAZENADO NA MEMÓRIA.

ESSE TIPO DE ERRO É CONHECIDO COMO: **UNDERFLOW**

EXISTE AINDA UM OUTRO TIPO DE ERRO MAIS SUTIL E MUITO COMUM QUE OCORRE QUANDO O RESULTADO DE UMA UMA OPERAÇÃO É UM NÚMERO COM MAIS ALGARISMOS SIGNIFICATIVOS DO QUE SE PODE ARMAZENAR NA MEMÓRIA (ultrapassa o espaço reservado para a mantissa).

POR EXEMPLO O RESULTADO DA DIVISÃO DE 23.5 POR 8 QUE É 2.9375 SERIA REPRESENTADO NO NOSSO COMPUTADOR COMO:

0	2	9	3	7	0	0	1

QUE EQUIVALERIA A 2.937

ESSE ERRO É CONHECIDO COMO **TRUNCAMENTO** E QUANDO TRABALHAMOS COM CÁLCULOS NUMÉRICOS SOFISTICADOS DEVEMOS ESTAR CIENTES DA PERDA DE PRECISÃO NAS OPERAÇÕES REALIZADAS.

SINAL E MAGNITUDE

LIMITES DE REPRESENTAÇÃO

SEJA TAMANHO DA MEMÓRIA IGUAL A N ALGARISMOS, FAIXA LIMITE DE NÚMEROS INTEIROS:

$-(2^{**(N-1)} - 1) >>>>$

$+(2^{**(N-1)} - 1)$

Por que?

DO VALOR DE N SUBTRAI-SE 1 PARA O SINAL E DO VALOR DE $2^{(N-1)}$ TAMBÉM SUBTRAI-SE 1 PARA O VALOR ZERO (+0 e -0).

EXEMPLOS

MEMÓRIA DE 6 BITS:

$-(2^5 - 1) >>> + (2^5 - 1)$

-31, -30, -29, ... , -1, -0

>>> +0, +1, +2, ... , +31

EXEMPLOS

MEMÓRIA DE 16 BITS:

$-(2^{15} - 1) >>> + (2^{15} - 1)$

-32767, -32766, ... , -1, -0

>>> +0, +1, +2, ... , +32767

EXEMPLOS

MEMÓRIA DE 16 BITS, porém representando um número fracionário (10 bits para a mantissa, 5 para a parte fracionária e 1 bit para o sinal)

??????

ARITMÉTICA EM SINAL E MAGNITUDE

SEJA UMA MEMÓRIA COM 4 BITS:

QUAIS OS RESULTADOS DAS SOMAS:

(+4) + (-2) e (+2) + (-2)

0100 (+4)	0010 (+2)
+ 1010 (-2)	+ 1010 (-2)
----------	----------
1110 (-6)	1100 (-4)

UÉ? O QUE ESTÁ ERRADO???

O BIT DE SINAL, EM REPRESENTAÇÃO **SINAL MAGNITUDE**, NÃO PODE SER CONSIDERADO NA OPERAÇÃO. ESSE BIT SERVE APENAS PARA IDENTIFICAR O TIPO DA OPERAÇÃO E O SINAL DO RESULTADO.

COM LÁPIS E PAPEL COMO SE FAZ ESSA OPERAÇÃO:

$$13 - 17 = \ ?$$

NA REALIDADE NOS EXECUTAMOS O SEGUINTE ALGORITMO:

ALGORITMO DA SOMA:

(+13) + (–17) = (-4)

1) VERIFICA OS SINAIS DOS OPERANDOS;

2) SE TIVEREM O MESMO SINAL, SOMA-SE AS MAGNITUDES, O SINAL DO RESULTADO É O MESMO DOS OPERANDOS;

E SE TIVEREM SINAIS DIFERENTES?

3) SE TIVEREM SINAIS DIFERENTES:

3.1) IDENTIFICA A MAIOR MAGNITUDE E REGISTRA SEU SINAL;

3.2) SUBTRAI A MENOR MAGNITUDE DA MAIOR;

3.3) O RESULTADO TERÁ O SINAL DA MAIOR MAGNITUDE.

EXEMPLOS:

UTILIZANDO O ALGORITMO DA SOMA EM SINAL E MAGNITUDE, EFETUAR AS SEGUINTES SOMAS CONSIDERANDO A PALAVRA DE DADOS (TAMANHO DOS REGISTRADORES DA UAL)

COM 6 BITS

(+13) + (+12) = ? (6BITS)

+13 001101

+12 001100

+25 011001

(-17) + (-9) = ? (6BITS)

-17 110001

-9 101001

-26 111010

(+18) + (-11) = ? (6BITS)

+18 010010

-11 101011

+7 000111

OBSERVAR QUE É UMA SUBTRAÇÃO:

+18 0 10010

-11 1 01011

+7 0 00111

0 – 1 >>> 2 – 1 = 1 VAI 1

1 – 1 >>> 3 – 2 = 1 VAI 1

0 – 0 >>> 2 – 1 = 1 VAI 1

0 – 1 >>> 2 – 2 = 0 VAI 1

1 – 0 >>> 1 – 1 = 0

(-21) + (+10) = ? (6BITS)

-21 110101

+10 001010

-11 101011

DA MESMA FORMA:

- 21 >>> 1 (10101) menos

+10 >>> 0 (01010) = 1(01011) = (-11)

1 – 0 >>> 1 – 0 = 1

0 – 1 >>> 2 – 1 = 1 VAI 1

1 – 0 >>> 1 – 1 = 0

0 – 1 >>> 2 – 1 = 1 VAI 1

1 – 0 >>> 1 – 1 = 0

EXEMPLOS:

UTILIZANDO O ALGORITMO DA SOMA EM SINAL E MAGNITUDE, EFETUAR AS SEGUINTES SOMAS CONSIDERANDO A PALAVRA DE DADOS (TAMANHO DOS REGISTRADORES DA UAL)

COM 16 BITS

(-21) + (+10) = ? (16BITS)

-21 1000000000010101

+10 0000000000001010

-11 1000000000001011

(-17) + (-19) = ? (6BITS)

-17 110001

-19 110011

-36 1100100

Ok ???????

NOT OK !!!!!!!!

1 011

-17 110001

-19 110011

-36 100100

"VAI UM" PARA O BIT DE SINAL

1 011

-17 110001

-19 110011

-36 100100

NA REPRESENTAÇÃO EM SINAL E MAGNITUDE, UM "VAI UM" PARA O BIT DO SINAL, SEMPRE INDICA QUE HOUVE ESTOURO (OVERFLOW)

6 BITS >> FAIXA DE REPRESENTAÇÃO -31 a +31

PARA ARMAZENAR -36, NO MÍNIMO, PRECISARIA DE 7 BITS

CUJA FAIXA PASSA A SER :

-63 a +63

SUBTRAÇÃO NA REPRESENTAÇÃO SINAL E MAGNITUDE.

RESULTADO = MINUENDO - SUBTRAENDO

ALGORITMO DA SUBTRAÇÃO IGUAL AO DA SOMA COM UM PASSO A MAIS:

- TROCA-SE O SINAL DO SUBTRAENDO
- EXECUTA O ALGORITMO DA SOMA.

EXEMPLO, EFETUAR A SUBTRAÇÃO, COM 6 BITS:

(-18) – (+12)

(-18) – (+12)

+12=0 01100 >>> -12 = 1 01100

SINAIS IGUAIS SOMA OS OPERANDOS:

-18 1 10010

-12 1 01100

-30 1 11110

OUTRO: (-27) – (-14) (6 BITS)

-14= 1 01110 >> +14 = 0 01110

SINAIS DIFERENTES SUBTRAI OS OPERANDOS:

- 27 1 11011

+14 0 01110

- 13 1 01101

OUTRO: (+27) – (+31) (6 BITS)

+31= 0 11111 >> -31 = 1 11111

SINAIS DIFERENTES SUBTRAI OS OPERANDOS:

- 31 1 11111

+ 27 0 11011

- 4 1 00100

OUTRO: (+19) – (-25) (6 BITS)

-25= 1 11001 >> +25 = 0 11001

SINAIS IGUAIS SOMA OS OPERANDOS:

\+ 19 0 10011

\+ 25 0 11001

1 11

\+ 19 0 10011

\+ 25 0 11001

\+ 44 0 101100

ZEBRA: VAI UM PARA O BIT DO SINAL