Representação em Ponto Fixo

Tipos primitivos de dados:
- caracter
- lógico
- numérico

Linguagens de alto nível: tipos mais complexos (array, registro, etc)

Mas, no processo de compilação, os dados são convertidos nas formas primitivas, para o hardware entendê-los e executa-los.

Tipo caracter
Os dados são codificados em ASCII e armazenados na memória. Cada caracter ocupa um byte da memória.

Tipo lógico ou booleano
As variáveis possuem somente 2 valores possíveis: FALSO ou VERDADEIRO

Tipo numérico

Quando trabalhamos com valores numéricos, temos que levar em consideração três fatos que podem acarretar inconvenientes no projeto e na utilização da máquina e que, na nossa vida cotidiana (aritmética com papel e lápis), não causam nenhum problema:
- a representação do sinal do número;
- a representação da vírgula (ou ponto) que separa a parte inteira da parte fracionária de um número não inteiro;
- a quantidade limite de algarismos possível de ser processada pela UAL (Unidade Aritmética e Lógica) de um processador.

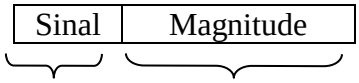
As representações numéricas vistas anteriormente não levavam em consideração o sinal do número. Tem-se duas maneiras de representar números em ponto fixo com sinal: por Sinal e Magnitude ou por Complemento de 2.

Representação em ponto fixo

É a representação utilizada para expressar números inteiros (supõe-se que a vírgula ou ponto esteja na extremidade direita do número).
Ex.: 157 = 157,0
183594 = 183494,0
A vírgula não é fisicamente representada na memória, mas o sistema assume que ela está sempre na mesma posição.
Números fracionários → representação em ponto flutuante
Números negativos e positivos → reserva-se um bit por sinal. Podem ser representados por:
- sinal e magnitude
- complemento 2.

Representação em ponto fixo por Sinal e Magnitude
O bit mais à esquerda é reservado para o sinal e os demais para a magnitude do número.

Formato:



Ex.: em 6 bits:

+16 = 0 10000

Sinal Magnitude
\+ 16

-16 = 1 10000

Sinal Magnitude
\- 16

em 8 bits:
+ 16 = 0001 0000
-16 = 1001 0000

em 16 bits:
+16 = 0000 0000 0001 0000
-16 = 1000 0000 0001 0000

Faixa de representação:

Para n bits:
$-(2^{n-1} - 1)$ a $+(2^{n-1} - 1)$

Ex.: para 8 bits:
$-(2^{8-1} - 1)$ a $+(2^{8-1} - 1)$
$-(2^{7} - 1)$ a $+(2^{7} - 1)$
$-(128 - 1)$ a $+(128 - 1)$
-127 a +127

Essa representação possui algumas características que a tornam menos vantajosa que as outras, razão por que não é utilizada atualmente nos processadores.
Desvantagens:
a) existem duas representações para o zero, o que é matematicamente incorreto e ocasiona a necessidade de um circuito lógico exclusivo para evitar uma má interpretação do valor;
b) para que as operações matemáticas possam ser realizadas na ULA (Unidade Aritmética e Lógica) precisa-se de dois circuitos diferentes, um para realizar a adição e outro para realizar a subtração, tornando o algoritmo de soma/subtração mais complexo e mais lento.

Representação em Ponto Fixo por Complemento de 2

Como na representação em Sinal Magnitude, também se reserva um bit para representação do sinal.

Representação de números positivos

Bit de sinal = 0
Magnitude: binário correspondente ao número positivo

Representação de números negativos

Bit de sinal = 1
Magnitude: complemento de 2 do binário positivo

Para que se possa encontrar o complemento de 2 de um número binário, primeiro é necessário calcular seu complemento de 1. Isso é obtido, de maneira prática, simplesmente invertendo cada um dos bits, ou seja, troca-se cada um dos dígitos por seu complemento.
Exemplo do cálculo de complemento de 1 do número binário 10110101:

Número binário	10110101
Complemento de 1	01001010

Agora, para encontrar o complemento de 2 do mesmo número, somamos 1 ao resultado encontrado:

Número binário	10110101
Complemento de 1	01001010 +1
Complemento de 2	01001011

Faixa de representação:

Para n bits:

-2^{n-1} a $+(2^{n-1} - 1)$.

Assim, para 8 bits, tem-se:

-2^{8-1} a $+(2^{8-1} - 1) = -2^7$ a $+(2^7 - 1) = -128$ a $+(128 - 1)$.

Ou seja, com 8 bits, pode-se representar valores entre – 128 e + 127.

Exemplos de representação numérica em complemento de 2:

a) Utilizando 5 bits:

+8 = 01000

-8 = ?

8 = 01000

Complemento de 2 de 8:

Número binário	01000
Complemento de 1	10111 +1
Complemento de 2	11000

b) Utilizando 8 bits:

+8 = 00001000

8 = 00001000

Complemento de 2 de 8:

Número binário	00001000
Complemento de 1	11110111 +1
Complemento de 2	11111000

A quase totalidade dos computadores modernos utilizam aritmética de complemento a 2 (quando se trata de representação em ponto fixo), devido às duas grandes vantagens daquele método sobre sinal e magnitude, e mesmo sobre complemento a 1...:

- possuir uma única representação para o zero;
- necessitar de apenas um circuito somador para realizar, não só operações de soma, mas também operações de subtração (mais barato).

Exercícios

1) Converta os seguintes valores decimais em valores binários equivalentes:

a) 329
b) 284
c) 473
d) 69
e) 135
f) 215
g) 581
h) 197

2) Converta os seguintes valores binários em valores decimais equivalentes:

a) 11011101010
b) 11001101101
c) 10000001111
d) 11101100010
e) 1110011010001
f) 111111000011
g) 101100011000
h) 100000000110

3) Converta os seguintes valores decimais em valores octais equivalentes:

a) 177
b) 254
c) 112
d) 719
e) 343
f) 27
g) 821
h) 197

4) Converta os seguintes valores octais em valores decimais equivalentes:

a) 2136
b) 1741
c) 613
d) 546
e) 120
f) 317
g) 720
h) 665

5) Converta os seguintes valores decimais em valores hexadecimais equivalentes:

a) 447
b) 544
c) 223
d) 71
e) 622
f) 97
g) 121
h) 297

6) Converta os seguintes valores hexadecimais em valores decimais equivalentes:

a) 3A2
b) 33B
c) 621
d) 99
e) 1ED4
f) 7EF
g) 22C
h) 110A

7) Efetue as seguintes conversões de base:

a) $(11011100011)_2 = (\ \ )_{16}$
b) $(5331)_8 = (\ \ )_2$
c) $(100011011)_2 = (\ \ )_8$
d) $(413)_8 = (\ \ )_2$
e) $(2317)_8 = (\ \ )_2$
f) $(3651)_{16} = (\ \ )_2$
g) $(11001011011011)_2 = (\ \ )_8$
h) $(100100001001)_2 = (\ \ )_8$
i) $(11111111)_2 = (\ \ )_8$
j) $(257)_8 = (\ \ )_2$
l) $(10110)_2 = (\ \ )_{16}$
m) $(10111111)_2 = (\ \ )_{16}$
n) $(92)_{16} = (\ \ )_2$
o) $(1A6)_{16} = (\ \ )_2$
p) $(37FD)_{16} = (\ \ )_2$

8) Realize as conversões de base a seguir:

a) $(0,7265625)_{10} = (\ \ )_2, (\ \ )_8, (\ \ )_{16}$
b) $(0,78125)_{10} = (\ \ )_2, (\ \ )_8, (\ \ )_{16}$
c) $(35,4)_{10} = (\ \ )_2, (\ \ )_8, (\ \ )_{16}$
d) $(10000100,101)_2 = (\ \ )_{10}$
e) $(11,001011)_2 = (\ \ )_{10}$
f) $(0,000101)_2 = (\ \ )_{10}$
g) $(10100,0001)_2 = (\ \ )_{10}$
h) $(0,1)_2 = (\ \ )_{10}$
i) $(E5D7,A3)_{16} = (\ \ )_{10}$
j) $(0,22D0E)_{16} = (\ \ )_{10}$
l) $(72,3)_8 = (\ \ )_{10}$
m) $(21,374)_8 = (\ \ )_{10}$

9) Usando a representação numérica em sinal e magnitude converta o valor decimal -8 para binário, utilizando 5 e 8 bits.

10) Utilizando as notações em sinal e magnitude e complemento de 2, represente o número decimal -114, com 8 bits.

Respostas dos exercícios:

1)
a) 101001001
b) 100011100
c) 111011001
d) 1000101
e) 10000111
f) 11010111
g) 1001000101
h) 11000101

2)
a) 1770
b) 1645
c) 1039
d) 1890
e) 7377
f) 4035
g) 2840
h) 2054

3)
a) 261
b) 376
c) 160
d) 1317
e) 527
f) 33
g) 1465
h) 305

4)
a) 1118
b) 993
c) 395
d) 358
e) 80
f) 207
g) 464
h) 437

5)
a) 1BF
b) 220
c) DF
d) 47
e) 26E
f) 61
g) 79
h) 129

6)
a) 930
b) 827
c) 1569
d) 153
e) 7892
f) 2031
g) 556
h) 4362

7)

a) $(6E3)_{16}$
b) $(101011011001)_2$
c) $(433)_8$
d) $(100001011)_2$
e) $(010011001111)_2$
f) $(0011011001010001)_2$
g) $(31333)_8$
h) $(4411)_8$
i) $(377)_8$
j) $(010101111)_2$
l) $(16)_{16}$
m) $(BF)_{16}$
n) $(10010010)_2$
o) $(000110100110)_2$
p) $(0011011111111101)_2$

8)

a) $(0,1011101)_2$, $(0,564)_8$, $(0,BA)_{16}$
b) $(0,11001)_2$, $(0,62)_8$, $(0,C8)_{16}$
c) $(100011,011...)_2$, $(43,3146...)_8$, $(23,66...)_{16}$
d) $(132,625)_{10}$
e) $(3,171875)_{10}$
f) $(0,078125)_{10}$
g) $(20,0625)_{10}$
h) $(0,5)_{10}$
i) $(58839, 63671875)_{10}$
j) $(0,136)_{10}$
l) $(58,375)_{10}$
m) $(17,486)_{10}$

9) 5 bits: 11000 e 8 bits: 10001000

10) 10001110 e 1110010