Software Básico

Representação de Dados: Inteiros com Sinal

INSTITUTO DE INFORMÁTICA - UFG

Reconhecimento

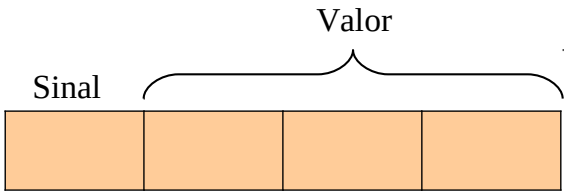
- Material produzido por:
  - Noemi Rodriguez – PUC-Rio
  - Ana Lúcia de Moura – PUC-Rio
- Adaptação
  - Bruno Silvestre – UFG

Representação de Inteiros

- Com n bits podemos representar 2^n valores
  - Para inteiros não negativos (*unsigned*) o intervalo de valores é [0, 2^n-1]
  - Para inteiros com sinal, teremos 0, valores negativos e valores positivos
- Como representar esses valores?
  - Há diferentes formas de representação

Sinal e Magnitude

- A ideia é usar o bit mais significativo como “sinal”
  - “0” → valor positivo
  - “1” → valor negativo
- Exemplo: n = 4 bits



Sinal e Magnitude

- Com 4 bits temos o sinal e 8 valores possíveis
  - Temos duas representações para zero

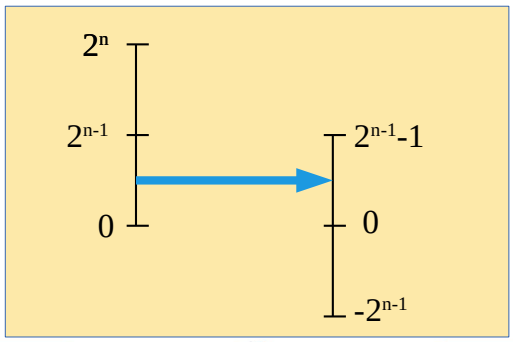
BINÁRIO	VALOR
0000	+ zero
0001 a 0111	0 a 7 decimal
1000	- zero
1001 a 1111	-1 a -7 decimal

Complemento para 2

- Representação mais usual para inteiros com sinal
- Temos um padrão de bits para 0 (zero), alguns padrões de bits representam valores positivos e alguns representam valores negativos
  - Uma única representação para 0

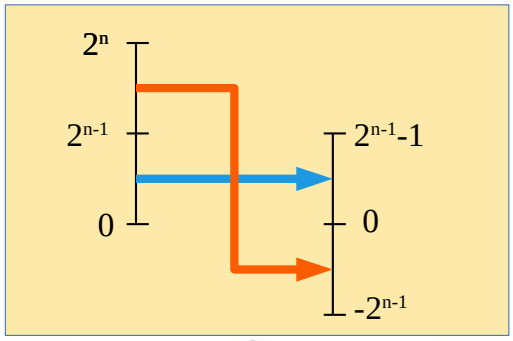
Complemento para 2

- Com n bits, temos 0 a 2^n números (padrões de bits)
- Queremos usar um padrão para 0 (zero), metade como positivos e metade como negativos



Complemento para 2

- Com n bits, temos 0 a 2^n números (padrões de bits)
- Queremos usar um padrão para 0 (zero), metade como positivos e metade como negativos



Complemento para 2

- Exemplo para n = 4, temos 2^4 padrões de bits

$2^4 - 1$ 1111
1110
1101
1100
1011
1010
1001
1000
0111
0110
0101
0011
0010
0001
0 0000

INSTITUTO DE INFORMÁTICA - UFG

Complemento para 2

- Exemplo para n = 4, temos 2^4 padrões de bits

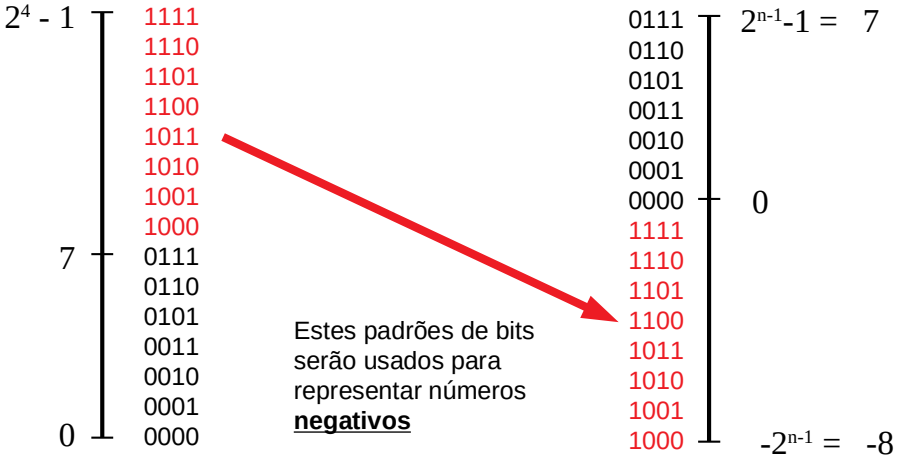
$2^4 - 1$ 1111
1110
1101
1100
1011
1010
1001
1000
7 0111
0110
0101
0011
0010
0001
0 0000

Estes padrões de bits serão usados para representar números **não-negativos**

0111 $2^{n-1}-1 = 7$
0110
0101
0011
0010
0001
0000 0
1111
1110
1101
1100
1011
1010
1001
1000 $-2^{n-1} = -8$

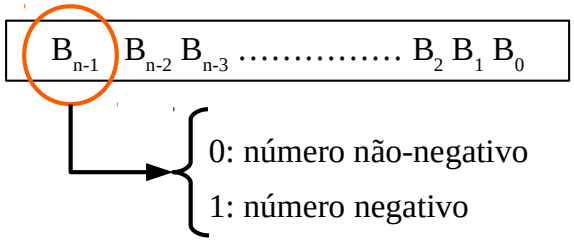
Complemento para 2

- Exemplo para n = 4, temos 2^4 padrões de bits



Complemento para 2

- O bit mais significativo também distingue valores negativos e não-negativos



Equivalência *modulo* 2^n

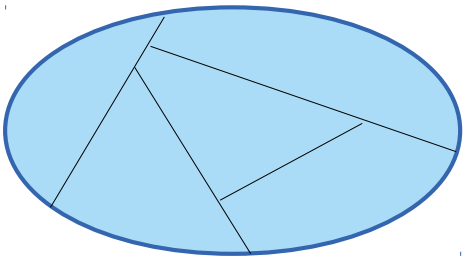
- Relação que define uma partição dos inteiros em classes de equivalência

  x Ξ y (mod m) se |x-y| = k*m, para algum k

Equivalência *modulo* 2^n

- Relação que define uma partição dos inteiros em classes de equivalência

  x Ξ y (mod m) se |x-y| = k*m, para algum k



Esta relação particiona o conjunto dos números inteiros

Equivalência *modulo* 2^n

$x \equiv y \pmod m$ se $|x-y| = k*m$, para algum k

- Exemplo para $m = 2^4$ → ou seja, 4 bits

{…, -16, 0, 16, …}
{…, -15, 1, 17, …}
{…, -14, 2, 18, …}
{…, -13, 3, 19, …}
{…, -12, 4, 20, …}
{…, -11, 5, 21, …}
{…, -10, 6, 22, …}
{…, -9, 7, 23, …}

$|1 - (-15)| = 16 = 1 * 2^4$

$|0 - 32| = 32 = 2 * 2^4$

Equivalência *modulo* 2^n

{…, -16, 0, 16, …} → 0000
{…, -15, 1, 17, …} → 0001
{…, -14, 2, 18, …} → 0010
{…, -13, 3, 19, …} → 0011
{…, -12, 4, 20, …} → 0100
{…, -11, 5, 21, …} → 0101
{…, -10, 6, 22, …} → 0110
{…, -9, 7, 23, …} → 0111

{…, -8, 8, 24, …} → 1000
{…, -7, 9, 25, …} → 1001
{…, -6, 10, 26, …} → 1010
{…, -5, 11, 27, …} → 1011
{…, -4, 12, 28, …} → 1100
{…, -3, 13, 29, …} → 1101
{…, -2, 14, 30, …} → 1110
{…, -1, 15, 31, …} → 1111

Usar o menor inteiro positivo da classe

Representação Complemento para 2

- Se $x \geq 0 \rightarrow rep_2(x) = x$
- Se $x < 0 \rightarrow rep_2(x) = 2^n + x$

Representação Complemento para 2

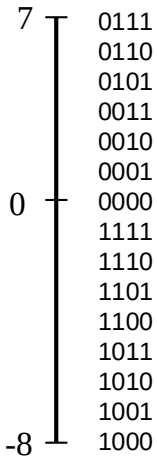
- Se $x \geq 0 \rightarrow rep_2(x) = x$
- Se $x < 0 \rightarrow rep_2(x) = 2^n + x$

Menor inteiro positivo da classe

Representação Complemento para 2

- Se $x \geq 0 \rightarrow rep_2(x) = x$
- Se $x < 0 \rightarrow rep_2(x) = 2^n + x$

- Exemplos para 4 bits → 2^4
  - $rep_2(-2) = 2^4 + (-2) = 16 - 2 = 14 \rightarrow 1110$
  - $rep_2(-8) = 2^4 + (-8) = 16 - 8 = 8 \rightarrow 1000$
  - $rep_2(-1) = 2^4 + (-1) = 16 - 1 = 15 \rightarrow 1111$



Conversão
Decimal → Binário

Encontrar Representação Binária

- Se $x \geq 0 \rightarrow x$
  - Usamos divisões sucessivas por 2 como visto em números sem sinal

Encontrar Representação Binária

- Se $x < 0 \rightarrow 2^n + x$

Encontrar Representação Binária

- Se $x < 0 \rightarrow 2^n + x$

  $2^n + x = 2^n - 1 + x + 1$

Encontrar Representação Binária

- Se $x < 0 \rightarrow 2^n + x$

  $2^n + x = 2^n - 1 + x + 1$

  $2^n + x = (2^n - 1) - (- x) + 1$

  $(2^n - 1)$ → 1111…..111

10000	10000000
- 1	- 1
1111	11111111
n = 4	n = 8

Encontrar Representação Binária

- Se $x < 0 \rightarrow 2^n + x$

  $2^n + x = 2^n - 1 + x + 1$

  $2^n + x = (2^n - 1) - (- x) + 1$

  $(2^n - 1) - (- x)$ → $\sim (- x)$

1111	1111	1111
- 0101 = 5 = -(-5)	- 0111 = 7 = - (-7)	- 0011 = 3 = - (-3)
1010	1000	1100

Encontrar Representação Binária

- Se $x < 0 \rightarrow 2^n + x$

  $2^n + x = 2^n - 1 + x + 1$

  $2^n + x = (2^n - 1) - (- x) + 1$

  $2^n + x = (\sim (- x)) + 1$

  $2^n + x = (\sim |x|) + 1$

Encontrar Representação Binária

- Se x < 0 → $2^n + x$
  - Pegar o positivo de x → |x|
  - Inverter seus bits → ~ |x|
  - Somar 1 → (~ |x|) + 1

Encontrar Representação Binária

- Se x < 0 → $2^n + x$
  - Pegar o positivo de x → |x|
  - Inverter seus bits → ~ |x|
  - Somar 1 → (~ |x|) + 1

-7 → 7 → ~ 0000 0111 → 1111 1000

\+ 1

1111 1001 = -7

Conversão
Binário → Decimal

Encontrar Representação Decimal

- Dada uma sequência de bits em complemento para 2, qual é o seu valor decimal?

Encontrar Representação Decimal

- Dada uma sequência de bits em complemento para 2, qual é o seu valor decimal?

0 1 0 1

- Se o número é positivo:

$$V_{10} = 1 * 2^2 + 0 * 2^1 + 1 * 2^0$$
$$= 4 + 0 + 1$$
$$= 5$$

Encontrar Representação Decimal

- Dada uma sequência de bits em complemento para 2, qual é o seu valor decimal?

1 0 1 1

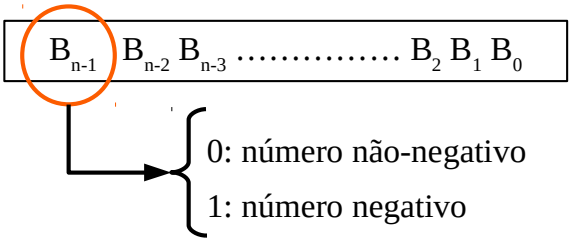
- Se o número é negativo:

$$V_{10} = (\sim x) + 1$$
$$= (\sim 1011) + 1$$
$$= 0100 + 1$$
$$= 0101$$
$$= 5 \quad \rightarrow -5 \text{ (pois o número é negativo)}$$

Intervalo de Valores em Complemento para 2

Intervalo de Valores

- Com n bits, em complemento para 2:
  - Menor valor é -2^{n-1}
  - Maior valor é $2^{n-1} - 1$
- Zero “rouba” um valor



Intervalo de Valores

- Com n bits, em complemento para 2:
  - Menor valor é -2^{n-1}
  - Maior valor é $2^{n-1} - 1$
- Zero “rouba” um valor
- Exemplo:
  - 8 bits → $[-2^{n-1}, \ldots, 0, \ldots, 2^{n-1} - 1]$
    → [-128, … , 0, … , 127]

Intervalo de Valores

- O padrão C não requer representação complemento para 2
  - Mas a maioria das máquinas o faz
  - Não é boa prática assumir a faixa de valores
- O cabeçalho <limits.h> define constantes para os tipos de dados inteiros
  - INT_MAX, INT_MIN, UINT_MAX

Soma e Subtração em Complemento para 2

Soma e Subtração com Complemento para 2

- Em complemento para 2, somas e subtrações usam adição
  - Subtração é a soma com o complemento
- Aritmética módulo 2^n garante correção mesmo com sinais diferentes (a menos de *overflow*)

Se $a \equiv b \pmod m$, $c \equiv d \pmod m$ → $(a+c) \equiv (b+d) \pmod m$

Se $a \equiv b \pmod m$, $c \equiv d \pmod m$ → $(a-c) \equiv (b-d) \pmod m$

Soma e Subtração com Complemento para 2

- Exemplo – usando 4 bits:

  $(1 - 2) \pmod{2^4} =$

Soma e Subtração com Complemento para 2

- Exemplo – usando 4 bits:

  $(1 - 2) \pmod{2^4} =$

  $(1 + (-2)) \pmod{2^4} =$

Soma e Subtração com Complemento para 2

- Exemplo – usando 4 bits:

  $(1 - 2) \pmod{2^4} =$

  $(1 + (-2)) \pmod{2^4} =$

Se $a \equiv b \pmod m$, $c \equiv d \pmod m$ $\rightarrow$ $(a+c) \equiv (b+d) \pmod m$

Soma e Subtração com Complemento para 2

- Exemplo – usando 4 bits:

  $(1 - 2) \pmod{2^4} =$

  $(1 + (-2)) \pmod{2^4} =$

Se $a \equiv b \pmod m$, $c \equiv d \pmod m$ $\rightarrow$ $(a+c) \equiv (b+d) \pmod m$

Se $1 \equiv 1 \pmod{2^4}$, $-2 \equiv 14 \pmod{2^4}$ $\rightarrow$ $(1+(-2)) \equiv (1+14) \pmod{2^4}$

Soma e Subtração com Complemento para 2

- Exemplo – usando 4 bits:

  $(1 - 2) \pmod{2^4} =$

  $(1 + (-2)) \pmod{2^4} =$

  $(1 + 14) \pmod{2^4} =$

  $15 \pmod{2^4} \rightarrow 15 \rightarrow 1111 \rightarrow rep_2(-1)$

Soma e Subtração com Complemento para 2

- Exemplos:

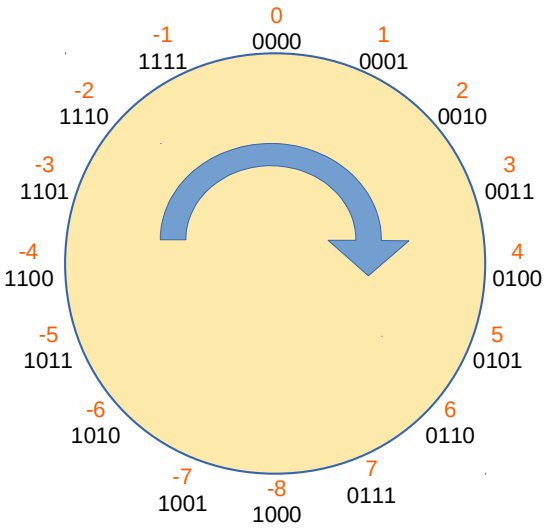
  $2 + 3 \rightarrow 0010 + 0011 \rightarrow 0101 \rightarrow 5$

  $7 - 1 \rightarrow 7 + (-1) \rightarrow 0111 + 1111 \rightarrow 0110 \rightarrow 6$

  $(-3) + 6 \rightarrow 1101 + 0110 \rightarrow 0011 \rightarrow 3$

  $(-1) + (-1) \rightarrow 1111 + 1111 \rightarrow 1110 \rightarrow (-2)$

Usando 4 bits → mod 4



Conversões entre Tipos Inteiros

Signed VS Unsigned

- Na conversão entre tipos do mesmo tamanho, o padrão de bits não muda
  - Ou seja, apenas é interpretado de forma diferente
- Exemplo:

```
int16_t   x = -1;              // x = 1111 1111 1111 1111 = -1
uint16_t  y = (uint6_t) x;     // y = 1111 1111 1111 1111 = 65535
```

Operadores Relacionais

- Operações de comparação (<, <=, etc) devem tratar operandos com e sem sinal
  - Existem instruções de máquina para cada caso
  - O compilador C gera o código adequado ao tipo de operandos

```
int a, b;
unsigned int c, d;

...
if (a < b)    /* operandos com sinal */
  ...
if (c < d)    /* operandos sem sinal */
  ...
```

Comportamento peculiar...

- Em expressões com operandos de tipo signed int com unsigned int, todos os valores são tratados como unsigned int

```
int a = -1;
int b =  0;
unsigned int z = 0;

if (a < b)   // true

if (a < z)   // false !!!

if (a < 0)   // true

if (a < 0U)  // false !!!
```

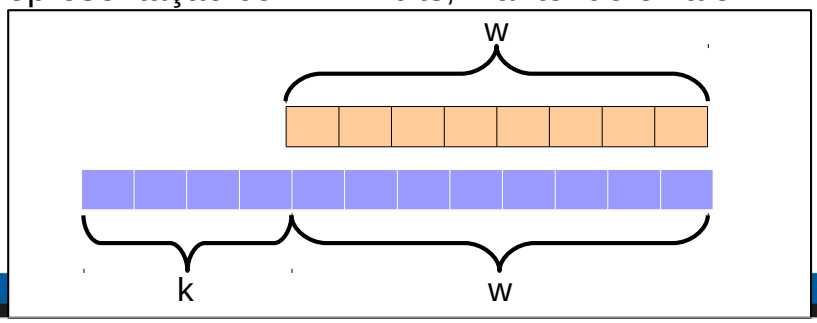

Extensão de Representação

- Conversões que aumentam o tamanho

  char → short, char → int, short → long

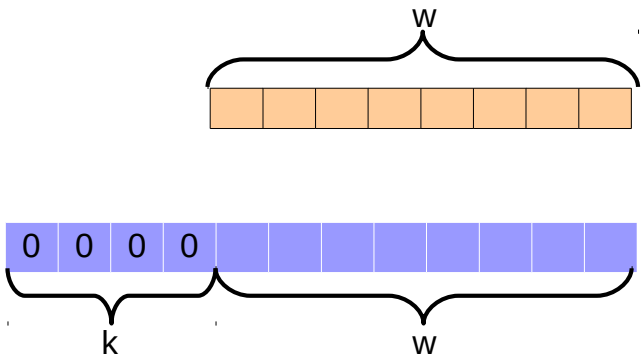
  unsingned int → unsigned long

- Converter representação com w bits para uma representação com w+k bits, mantendo o valor



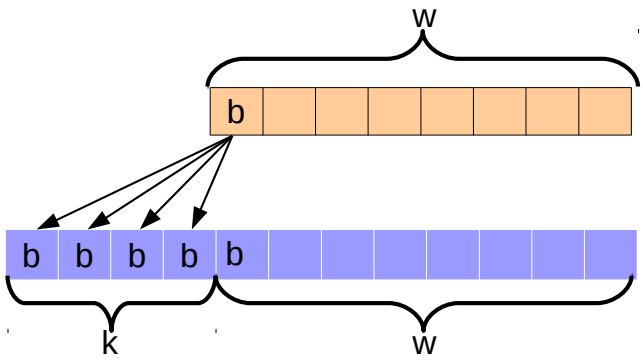
Extensão de Representação

- Se a conversão é sem sinal (*unsigned*), basta adicionar k zeros no valor final



Extensão de Representação

- Se a conversão é com sinal em complemento para 2, deve-se repetir o bit do sinal k vezes no valor final

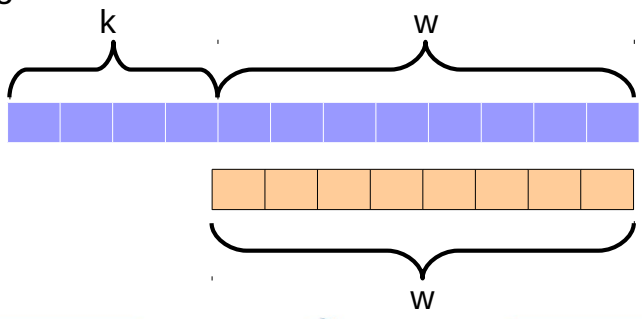


Truncamento

- Conversões que diminuem o tamanho
  - short → char, int → short, long → char
  - unsigned long → unsigned int
- Converter uma representação com w+k bits para uma representação com w bits

Truncamento

- Converter uma representação com w+k bits para uma representação com w bits
  - Truncamento → remover os k bits mais significativos



Truncamento

- Converter uma representação com w+k bits para uma representação com w bits
  - Truncamento → remover os k bits mais significativos
  - Nem sempre é possível manter o valor

unsigned

~~0000~~ 1001 → 9

1001 → 9

unsigned

~~0010~~ 1001 → 41

1001 → 9

Truncamento

- Converter uma representação com w+k bits para uma representação com w bits
  - Truncamento → remover os k bits mais significativos
  - Nem sempre é possível manter o valor

signed

~~1111~~ 1001 → -7

1001 → -7

signed

~~0010~~ 1001 → 41

1001 → -7