PCS-3115 Sistemas Digitais I — Março de 2018 - v2

Sistemas de Complemento de Base

Anotações de Aula — Profs. Gomi/Marcos/Spina/Glauber

1 Objetivos deste tópico

Ao final do estudo deste tópico você saberá:

- Representar números inteiros negativos;
- Compreender o conceito de representação de Complemento de Base;
- Representar números em Complemento de 2 e Complemento de 1;
- Somar, subtrair, multiplicar e dividir em Complemento de 2.

Leitura recomendada : seções do livro do Wakerly

- 2.5 Representation of Negative Numbers;
- 2.6 Two's-Complement Addition and Subtraction;
- 2.7 One's-Complement Addition and Subtraction.

Keywords: signed-magnitude system, sign bit, signed-magnitude adder, signed-magnitude subtractor, complement number system, radix-complement system, 10's complement, computing the radix complement, two's complement, weight of MSB, extra negative number, sign extension, diminished radix complement system, 9s' complement, ones' complement, two's complement addition, overflow, overflow rules, two's complement subtraction, carry, borrow, ones' complement addition, end-around carry, onesćomplement subtraction.

2 Representação de Números Inteiros Negativos

- Como podemos representar números inteiros negativos?
- O que é o sistema de representação de complemento de base? Como funciona?
- Computadores de hoje usam o sistema Complemento de 2. Como se faz a conversão de um número positivo para seu correspondente negativo?
- Há diferença de representação entre +0 e -0?
- Quais são as vantagens do sistema Complemento de 2 sobre a representação sinal-magnitude?

3 Exercícios

1. Considere um sistema de representação em complemento de base, para a base 2, com quantidade de dígitos igual a 4 (quatro). Neste sistema pede-se que sejam realizadas as três operações determinadas nos ítens a), b) e c), com os operandos e resultados representados em complemento de base. Ainda, para estas três operações, solicita-se que seja respondida a seguinte questão: Houve transbordo na operação? Se a resposta é sim, apresentar pelo menos duas evidências que permitam detectar (explicar) o aparecimento de transbordo. Se a resposta é não justifique mostrando a evidência que comprova esta asserção.

   (a) $((-2) + (-7))_{10}$

   (b) $((-8) - (-7))_{10}$

   (c) $((+3) - (+5))_{10}$

2. Realize a operação $(+27)_{10} + (-8)_{10}$ em complemento de base 5 diminuída, com 3 dígitos, mostrando os cálculos. Houve transbordo (overflow)? Qual o valor esperado dessa operação (em complemento de base 5 diminuída e em base 10)?

3. Realize a operação $(+13)_{10} + (-9)_{10}$ em complemento de 1 com 5 dígitos. Indique a conversão de base e demonstre os cálculos realizados, indicando os bits de vai-um. Houve transbordo? Justifique sua resposta.

4. Quais são os resultados da execução do programa em Linguagem C apresentado a seguir, em um computador com microprocessador executando aritmética em Complemento de 2 de 64 bits? Explique os resultados.

```
#include <stdio.h>
#include <stdint.h>
#include <inttypes.h>

int main()
{
  int64_t a, b, c;

  a = 0x2;
  b = 0x0;
  c = b - a;

  printf("%016"PRIx64"\n",c);

  b = 0x1000000000000000;
  c = b - a;

  printf("%016"PRIx64"\n",c);
}
```

Supondo que o arquivo fonte seja barithmetic.c, o comando para compilar o programa em Linux, usando GCC será:

```
gcc -o barithmetic barithmetic.c
```

Analise o código assembly gerado pelo GCC, após remover do programa barithmetic.c as 2 instruções *printf*. O código assembly é do microprocessador Intel Core i7. Identifique e procure entender os diversos elementos que fazem parte deste código. Onde estão a operação c = b - a?

```
.file "barithmetic.c"
.text
.globl main
.type main, @function
main:
.LFB0:
.cfi_startproc
pushq %rbp
.cfi_def_cfa_offset 16
.cfi_offset 6, -16
movq %rsp, %rbp
.cfi_def_cfa_register 6
movq $2, -24(%rbp)
movq $0, -16(%rbp)
movq -16(%rbp), %rax
subq -24(%rbp), %rax
movq %rax, -8(%rbp)
movabsq $1152921504606846976, %rax
movq %rax, -16(%rbp)
movq -16(%rbp), %rax
subq -24(%rbp), %rax
movq %rax, -8(%rbp)
movl $0, %eax
popq %rbp
.cfi_def_cfa 7, 8
ret
.cfi_endproc
.LFE0:
.size main, .-main
.ident "GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.4) 5.4.0 20160609"
.section .note.GNU-stack,"",@progbits
```

5. Qual é o equivalente em hexadecimal de 12648430_{10}?

6. Encontre um número binário de 8 bits que tem o mesmo valor negativo quando interpretado como um decimal ou como um número em complemento de dois. Você consegue encontrar mais algum número com essa mesma propriedade?

7. Cada uma das seguintes operações é correta em pelo menos um sistema numérico. Determine radicais possíveis para os números em cada operação:

   a) $1234 + 5432 = 6666$

   b) $41/3 = 13$

   c) $33/3 = 11$

   d) $23 + 44 + 14 + 32 = 223$

   e) $302/20 = 12.1$

   f) $\sqrt{41} = 5$

8. A primeira expedição para Marte encontrou apenas as ruínas de uma civilização. A partir dos artefatos e figuras, os exploradores deduziram que as criaturas que haviam produzido esta civilização eram seres de quatro patas com um tentáculo que possuía vários "dedos" na ponta. Depois de muito estudo, os exploradores conseguiram traduzir a matemática marciana. Eles encontraram a seguinte equação:

$$5x^2 - 50x + 125 = 0$$

   com as soluções indicadas $x = 5$ e $x = 8$. O valor $x = 5$ parecia correto, mas $x = 8$ precisou ser explicado. Neste momento, os exploradores refletiram na maneira como o sistema numérico da Terra se desenvolveu e encontraram evidência de que o sistema marciano teve uma história parecida. Quantos dedos você diria que os marcianos possuíam? (de *The Bent of Tau Beta Pi*, fevereiro de 1956.)

9. Suponha que um número de $4n$ bits, denominado B, é representado por um número hexadecimal H de n dígitos. Prove que o complemento de dois de B é representado pelo complemento de 16 de H. Providencie uma afirmação similar a essa para números em representação octal.

10. Repita o exercício anterior usando o complemento de um de B e o complemento de 15 de H.

11. Repita o exercício anterior usando expressões e regras apropriadas para a adição em complemento de um.

12. Determine uma regra para o *overflow* na adição de números em complemento de dois em termos de operações de contagem na representação modular da Figura 2-3.

13. Mostre que um número em complemento de dois pode ser convertido para uma representação com mais bits através de uma *extensão de bits*. Ou seja, dado um número X em complemento de dois com n bits, mostre que a representação de X em complemento de dois com m bits, onde $m > n$, pode ser obtida colocando-se $m - n$ cópias do bit de sinal de X a esquerda da representação de n bits de X.

14. Mostre que um número em complemento de dois pode ser convertido para uma representação com menos bits através da remoção de bits de ordem superior. Ou seja, dado um número X em complemento de dois com n bits, mostre que o número Y em complemento de dois com m bits obtido descartando-se os d bits a esquerda de X representa o

mesmo número que X se e somente se todos os bits descartados forem iguais ao bit de sinal de Y.

15. Por que a pontuação de "complemento de dois" e "complemento de um" são inconsistentes? (Olhe as duas primeiras citações nas referências.)

16. Um somador binário de n bits pode ser usado para realizar uma subtração sem sinal de n bits, $X - Y$, através da operação $X + \overline{Y} + 1$, onde X e Y são números de n bits sem sinal e $\overline{Y}$ representa o complemento bit-a-bit de Y. Demonstre este fato da seguinte maneira. Primeiro, prove que $(X - Y) = (X + \overline{Y} + 1) - 2^n$. Segundo, prove que o *carry* do somador de n bits é o oposto do *borrow* da subtração de n bits. Ou seja, mostre que a operação $X - Y$ produz um *borrow* a partir da posição do MSB se e somente se $X + \overline{Y} + 1$ não produz um *carry* a partir da posição do MSB.

17. Na maioria dos casos, o produto de dois números de n bits em complemento de dois precisa de menos de $2n$ para ser representado. Na verdade, há apenas um caso em que $2n$ bits são necessários. Qual é este caso?

18. Prove que o complemento de dois de um número pode ser multiplicado por 2 fazendo o *shift* de uma posição para a esquerda, com um *carry* de 0 na posição menos significativa e ignorando qualquer *carry* da posição mais significativa, assumindo que não ocorre *overflow*. Descreva a regra para detectar quando ocorre *overflow*.

19. Descreva uma técnica similar àquela descrita no item anterior para multiplicar números em complemento de um por 2. Demonstre que essa técnica está correta.

20. Mostre como subtrair números BCD (*Binary Coded Decimal*) descrevendo regras para gerar *borrows* e aplicar um fator de correção. Mostre como as suas regras se aplicam a cada uma das seguintes subtrações: $8 - 3$, $4 - 8$, $5 - 9$, $2 - 7$.