Complemento a Um e Complemento a Dois

Cristina Boeres (baseado no material de Fernanda Passos)

Instituto de Computação (UFF)

Fundamentos de Arquiteturas de Computadores

Conceito de Complemento

- Já vimos o conceito de complemento – em aritmética não decimal
- O complemento $\overline{a}$ de um número a em uma base b é um número tal que:

$$a + \overline{a} = \alpha\alpha\ldots\alpha_{(b)}$$

  - Onde α é o maior algarismo na base b.
  - E o número de algarismos da soma é igual ao número de algarismos de a.
- Em uma base b qualquer, **todo algarismo α tem um algarismo complementar $\overline{\alpha}$**.

$$\overline{\alpha} = b - 1 - \alpha$$

- De maneira menos formal, o complemento de um número pode ser calculado **substituindo** cada um de seus **algarismos pelo** respectivo **complemento**.

Conceito de Complemento

- Uma das utilidades do conceito de complemento é a simplificação do processo de subtração:
  - Calculamos o complemento (ums subtração *fácil*)
  - Somamos com o complemento
  - Incrementamos de 1 (outra soma)
  - No final, ignoramos o algarismo mais significativo

Conceito de Complemento

- Exemplo: calcular $65_{(10)} - 37_{(10)}$
  - Complemento a 9 (*i.e.*, na base 10) de $37_{(10)}$ é $62_{(10)}$
  $$99_{(10)} - 37_{(10)} = 62_{(10)}$$
  - Somando minuendo e complemento do subtraendo:
  $$65_{(10)} + 62_{(10)} = 127_{(10)}$$
  - Incrementando de um e ignorando o algarismo mais significativo, chegamos à resposta:
  $$28_{(10)}$$

Representação de Números Inteiros Negativos

- Vimos duas formas de representar números inteiros negativos:
  - Sinal e Magnitude.
  - Representação em Excesso de k.
- Desvantagem:
  - dificuldade de realizar algumas operações

Representação de Números Inteiros Negativos

- Idealmente, precisamos de uma representação que permita tratarmos **uniformemente** números positivos e negativos.
  - números em um representação uniforme não receberão **tratamento especial** para serem operados
  - principalmente **para somas**.
    - Outras operações são mais complicadas

- Por exemplo, para somar em Sinal e Magnitude, precisamos **olhar para o primeiro bit**:
  - Se for 0, efetuamos a soma.
  - Se for 1, temos que efetuar uma subtração

Complemento e os Números Negativos

- Voltando ao conceito de complemento, note que ele possui uma relação com números negativos:
- Isto é, para calcular $a_1 - a_2$ em uma base b qualquer podemos usar o método do complemento
  - Calculamos o complemento a $b - 1$ de a_2 e **transformamos a operação em uma soma**
    - Embora ainda haja os detalhes do incremento e de ignorar o algarismo mais significativo do resultado

Complemento e os Números Negativos

- Para calcular $a - a$, por exemplo, somamos a com seu próprio complemento
- Calcular $63_{(10)} - 63_{(10)}$ usando o método de complemento a 9:
  - Complemento a 9 de $63_{(10)}$ é $36_{(10)}$
  - Fazendo a soma, obtemos $63_{(10)} + 36_{(10)} = 99_{(10)}$
  - Incrementando e ignorando o algarismo mais significativo, obtemos $0_{(10)}$
- De certa forma, podemos dizer que o complemento de um número é *similar* ao seu negativo.
  - Ao menos, exerce um papel **parecido**.

Representação por Complemento a Um: Ideia Básica

- Aproveitando esta proximidade dos números negativos com o complemento da magnitude, vejamos o seguinte um esquema de representação
- Na representação por complemento a 1, números negativos são representados pelo complemento a 1 da sua magnitude
  - Números positivos são representados normalmente em base 2.
- Exemplos
  - $5_{(10)}$ é representado com 5 bits como 00101
  - $-5_{(10)}$ é representado com 5 bits como 11010
  - $37_{(10)}$ é representado com 8 bits como 00100101
  - $-37_{(10)}$ é representado com 8 bits como 11011010

Representação por Complemento a Um: Detalhes

- Note que calcular o complemento a 1 de um número binário é muito simples.
  - Basta "inverter" os bits.
    - 0 vira 1.
    - 1 vira 0.

- Exemplos:
  - Se a tem representação 01101100, $-a$ tem representação 10010011.
  - Se a tem representação 1010, $-a$ tem representação 0101.

Representação por Complemento a Um: Unicidade

- Uma propriedade importante de um esquema de representação é a **unicidade**.
  - *i.e.*, uma dada sequência de bits deve representar **um único valor**
  - Exemplo: em Sinal e Magnitude, a sequência 01101 representa **apenas** o número $13_{(10)}$.

- No entanto, vejamos representação por Complemento a Um
  - Vamos analisar alguns exemplos, considerando 5 bits:

    $\rightarrow +19_{(10)}$ tem representação 10011

    $\rightarrow -19_{(10)}$ tem representação 01100

    $\rightarrow +12_{(10)}$ tem representação 01100

    { Mesmo Valor

  - Conclusão: encontramos uma sequência de bits (01100) que representa **dois valores simultaneamente**.

Representação por Complemento a Um: Unicidade (II)

- Esse problema pode ser evitado:
  - Determinação dos **limites** para o uso da representação

- O problema ocorreu porque tentamos representar um número **positivo** que necessita do **bit mais significativo igual a 1**.
  - seu complemento (negativo) tenha o primeiro bit igual a 0
  - Mas esta representação já é utilizada por outro número positivo

- Então, seja a regra adicional:
  - O bit mais significativo determina o sinal do número:
    - ★ **0, para positivos**
    - ★ **1 para negativos**

  - Mesma lógica da representação por sinal e magnitude
  - problema da falta de unicidade resolvido

Representação por Complemento a Um: Limites

- Qual é o maior número que pode ser representado em Complemento a Um com n bits?
  - Dada a regra adicional apresentada no último slide: $011\dots1$.
  - *i.e.*, um zero (positivo) seguido de $n-1$ uns.
  - Na base 10, equivale a: $2^{(n-2)} + 2^{(n-3)} + \cdots + 2^0 = 2^{(n-1)} - 1$.
  - Para $n = 8$, por exemplo, maior valor é 127.
- E qual é o menor número?
  - Número mais negativo: valor negativo com a maior magnitude.
  - Como números negativos tem seus bits invertidos, a maior magnitude contém o maior número de bits zero.
  - *i.e.*, $100\dots0$, que tem magnitude $011\dots1 = 2^{(n-1)} - 1$.
  - Logo, menor número representável é $-(2^{(n-1)} - 1)$.

Representação por Complemento a Um: Realizando Contas

- A grande motivação para a exploração de outros esquemas de representação é a necessidade de simplificar operações aritméticas
- Até que ponto o Complemento a Um é bem sucedido?

Complemento a Um: Realizando Contas

- Considere o exemplo da soma:
  - Soma de dois números positivos – trivial
    - Suas representações são simplesmente as conversões para base 2
    - Exemplo com 5 bits: $01010 + 00101 = 01111$.
  - Soma de um número positivo com outro negativo – mais complicada
    - Se tentarmos a soma direta como para os positivos, teremos problemas:
    - Exemplo com 5 bits: $01010 + 11010 = 100100$.
    - O resultado esperado seria 00101.
  - Somas entre dois números negativos– também problemática

Complemento a Um: Realizando Contas

- Algoritmo simples de soma que funciona sempre:

---

1. Seja *n* o número de bits usados para a representação
2. Some os números normalmente, como se fossem positivos
3. Se a soma resultar no $n + 1$-*ésimo* bit igual a 1
   - ignore-o e incremente o resultado em uma unidade

- Exemplos com 5 bits (números representados em Complemento a Um):

$$\begin{array}{rr} & 01010 \\ + & 00101 \\ \hline & 01111 \end{array} \qquad \begin{array}{rr} & 00101 \\ + & 10101 \\ \hline & 11010 \end{array} \qquad \begin{array}{rr} & 01010 \\ + & 11010 \\ \hline & \boxed{1}00100 \\ + & 1 \\ \hline & 00101 \end{array} \qquad \begin{array}{rr} & 10101 \\ + & 11010 \\ \hline & \boxed{1}01111 \\ + & 1 \\ \hline & 10000 \end{array}$$

Complemento a Um: Realizando Contas

- Se a soma é fácil, a subtração também é
- Basta lembrar que

$$a_1 - a_2 = a_1 + (-a_2)$$

- Trocar o sinal de um número é trivial:
  - Basta inverter todos os bits

- Algoritmo para subtração:
  1. Inverta os bits do subtraendo.
  2. Aplique o algoritmo de soma entre o resultado e o minuendo

---

Complemento a Um: Realizando Contas

- Exemplos com 5 bits (números representados em Complemento a Um):

$$
\begin{array}{rr}
 & 01010 \\
- & 00101 \\
\hline
 & 01010 \\
+ & 11010 \\
\hline
 & \boxed{1}00100 \\
+ & 1 \\
\hline
 & 00101
\end{array}
\qquad
\begin{array}{rr}
 & 01010 \\
- & 11010 \\
\hline
 & 01010 \\
+ & 00101 \\
\hline
 & 01111
\end{array}
$$

Complemento a Um: Realizando Contas

- Multiplicação e divisão continuam mais "complicadas":
  - Não podemos operar diretamente nas representações de Complemento a Um.
  - Se algum dos operandos for negativo, voltamos para a representação sem Complemento a Um e ignoramos o sinal.
  - Em seguida, realizamos a operação (como se ambos fossem).
  - Finalmente, verificamos se os sinais dos operandos eram diferentes.
    - Basta olhar para o primeiro bit de cada um.
  - Em caso afirmativo, invertemos os bits do resultado.

Complemento a Um *vs.* Outras Representações

- Em comparação com Sinal e Magnitude e Representação em Excesso de k, Complemento a Um traz vantagens
  - A inversão de sinal de um número é quase tão simples quanto em Sinal e Magnitude
  - E mais simples que na representação por excesso
  - Além disso, a soma é mais simples que em ambas as outras representações
  - Combinando estes dois fatores, concluímos que a subtração também é mais fácil
- Embora apresente estas vantagens, o Complemento a Um também tem seus problemas

Complemento a Um: Problemas

- O principal problema do Complemento a Um é o mesmo da representação por Sinal e Magnitude.
  - Existem duas representações para o 0.
- Podemos verificar isso com um exemplo:
  - Em uma representação com 5 bits, qual valor é representado pela sequência 11111?
    - O valor deve ser negativo, já que o primeiro bit é 1.
    - Para saber sua magnitude, invetemos todos os bits, obtendo 00000.
- Embora o exemplo tenha sido dado com 5 bits, o mesmo ocorre com qualquer número de bits
  - A sequência $11\ldots1$ é às vezes chamada de **zero negativo**.
  - Ocorre, por exemplo, quando somamos dois números de mesmo módulo, mas sinais opostos.
    - *e.g.*, com 5 bits, $01010 + 10101 = 11111$.

Complemento a Um: Uso

- A existência de duas representações para o 0 é inconveniente
- Além disso, reduz o intervalo de números que podem ser representados com um dado número de bits
- Por conta disso, o Complemento a Um não é muito utilizado hoje

- Mas é a base para um esquema de representação melhor
  - **Complemento a Dois**

Representação por Complemento a Dois: Ideia Básica

- No Complemento a Um, o negativo de um número é obtido invertendo-se todos os bits.
  - Complemento de 0 é 1, complemento de 1 é 0.
- Esta é a causa do zero negativo.
  - Ao somar um número com seu negativo (operação que resulta necessariamente em 0), obtemos um valor no qual todos os bits são 1.
- **Idealmente**, a soma de um número com seu negativo deveria resultar em **todos os bits 0**.
- Será que existe uma maneira de adaptar o Complemento a Um para obter este efeito?

Representação por Complemento a Dois: Ideia Básica (II)

- Considere dois números quaisquer a e b, tal que $b = -a$ e $a \neq 0$.
- Suponha que neste esquema ideal de representação, com n bits, eles sejam escritos como:
  - $a = \alpha_{n-1}\alpha_{n-2}\ldots\alpha_0$.
  - $b = \beta_{n-1}\beta_{n-2}\ldots\beta_0$.
  - Onde α_i e β_i são algarismos binários (0 ou 1), para $0 \leq i < n$.
- Queremos que, quando somadas, estas representações resultem em todos os bits 0.
- Algo como:

$$
\begin{array}{ccccc}
 & \alpha_{n-1} & \alpha_{n-2} & \ldots & \alpha_0 \\
+ & \beta_{n-1} & \beta_{n-2} & \ldots & \beta_0 \\
\hline
 & 0 & 0 & \ldots & 0
\end{array}
$$

- Note, no entanto, que isso só é possível se todos os α_i e β_i forem também 0.

Representação por Complemento a Dois: Ideia Básica (III)

- Não podemos ter todos os α_i e β_i iguais a zero.
  - Teríamos uma quebra da unicidade da representação.
- Precisamos de outra alternativa.
- Proposta:
  - Relaxamos a restrição de $a + b$ resultar em todos os bits 0.
  - Ao contrário, vamos exigir isso apenas para os n primeiros bits.
    - ★ Em outras palavras, vamos permitir que a soma resulte em um bit adicional à esquerda igual a 1.
    - ★ Como só podemos usar n bits, o $(n+1)$-*ésimo* será descartado de qualquer forma.
- Algo como:

$$
\begin{array}{cccccc}
 & \alpha_{n-1} & \alpha_{n-2} & \ldots & \alpha_0 \\
+ & \beta_{n-1} & \beta_{n-2} & \ldots & \beta_0 \\
\hline
1 & 0 & 0 & \ldots & 0
\end{array}
$$

Representação por Complemento a Dois: Ideia Básica (IV)

- Esta abordagem parece mais promissora.
- Dada a representação de a, podemos descobrir a representação de b calculando $100\ldots0 - a_{(2)}$.
- Por exemplo:
  - Suponha que $n = 5$ bits e a tenha a representação 01101.
  - Para descobrir b, calculamos $100000_{(2)} - 01101_{(2)} = 10011_{(2)}$
- O valor b calculado desta forma é chamado de Complemento a Dois de a.
  - Esta ideia é o cerne do esquema de representação por Complemento a Dois.

Representação por Complemento a Dois: Formalização

- A representação por Complemento a Dois de um dado número inteiro *a* com *n* bits é definida da seguinte forma:
  - Se *a* é positivo, sua representação é idêntica à representação de *a* em base 2.
    - Completa-se com zeros à esquerda para garantir que a representação utilize todos os *n* bits.
  - Se *a* é negativo, sua representação é o Complemento a Dois de $|a|$.
    - *i.e.*, o resultado de $100\ldots0_{(2)} - |a|$.

Representação por Complemento a Dois: Formalização

- Note que, assim como fizemos no Complemento a Um, teremos uma restrição em relação ao bit mais à esquerda.
  - Em números positivos, ele é necessariamente 0
  - Em números negativos, ele é necessariamente 1
- Isso garante a propriedade da unicidade de uma representação
  - Embora limite os valores que podem ser representados
  - Por exemplo, $200_{(10)}$ não poderá ser representado com 8 bits
    - ★ $200_{(10)} = 11001000$.
    - ★ Bit mais à esquerda é 1, reservado para valores negativos

Representação por Complemento a Dois: Exemplos

- Representar $47_{(10)}$ com 7 bits:
  - $47_{(10)} = 101111_{(2)}$
  - Completando com zeros à esquerda obtemos a representação final: 0101111
- Representar $-47_{(10)}$ com 7 bits:
  - Partindo do resultado anterior ($|-47| = 47$), basta calcular o complemento:
    - $10000000_{(2)} - 101111_{(2)} = 1010001$
  - Para simples confência, seja o cálculo do complemento na base 10:
    - $2^7_{(10)} - 47_{(10)} = 81_{(10)} = 1010001_{(2)}$.

Atalho para o Cálculo do Complemento a 2

- Calcular o complemento a 2 envolve realizar uma subtração.
  - Pode ser feita em qualquer base
  - Mas para números grandes, pode ser trabalhosa

- Um método bem mais fácil (possível, devido a base binária):
  - Inverter bits
  - Incrementar em uma unidade

- Exemplos (com 5 bits):
  - Complemento a dois de 01011 é $10100 + 1 = 10101$.
  - Complemento a dois de 11001 é $00110 + 1 = 00111$.
  - Complemento a dois de 10010 é $01101 + 1 = 01110$.
  - Complemento a dois de 10101 é $01010 + 1 = 01011$.

Complemento a Dois: Limites

- Qual é o maior número que se pode representar em Complemento a Dois com n bits?
  - Números positivos sempre começam com 0.
  - Após o zero inicial, o maior número possível terá todos os outros bits iguais a 1.
    - ★ $011\ldots1$.
  - Em decimal, equivale ao valor $2^{(n-2)} + 2^{(n-3)} + \cdots + 2^0 = 2^{(n-1)} - 1$.
- E qual o menor número?
  - Números negativos sempre começam por 1.
  - O valor absoluto é dado por 2^n subtraído da representação.
  - Para maximizar o valor absoluto, devemos minimizar os algarismos da representação:
    - ★ $100\ldots0$.
  - Em decimal, equivale a $-(2^n - 2^{(n-1)}) = -2^{(n-1)}$.

Complemento a Dois: Realizando Contas

- As operações de soma e subtração em Complemento a Dois funcionam de forma similar àquelas em Complemento a Um.
  - São até mais simples
- No caso da soma, dados dois números representados em Complemento a Dois, fazemos a soma normal em base 2
  - Se o resultado ocupar mais de n bits, pegamos apenas os n bits menos significativos
- Exemplos com 5 bits (números representados em Complemento a Dois):

$$\begin{array}{rr} & 01010 \\ + & 00101 \\ \hline & 01111 \end{array} \qquad \begin{array}{rr} & 00101 \\ + & 10101 \\ \hline & 11010 \end{array} \qquad \begin{array}{rr} & 01010 \\ + & 11010 \\ \hline & \cancel{1}00100 \end{array} \qquad \begin{array}{rr} & 10111 \\ + & 11010 \\ \hline & \cancel{1}10001 \end{array}$$

Representação por Complemento a Dois: Realizando Contas (II)

- Assim como no Complemento a Um, subtrações podem ser transformadas em somas.
  - Altera-se o sinal do subtraendo e soma-se com o minuendo.
- A alteração de sinal é feita calculando o complemento a dois do número.
  - Invertem-se os bits, e soma-se um
- Exemplos com 5 bits:

$$
\begin{array}{rr}
 & 01010 \\
- & 00101 \\
\hline
 & 01010 \\
+ & 11011 \\
\hline
 & \cancel{1}00101
\end{array}
\qquad \text{Complemento a Dois}
\qquad\qquad
\begin{array}{rr}
 & 01010 \\
- & 11011 \\
\hline
 & 01010 \\
+ & 00101 \\
\hline
 & 01111
\end{array}
\qquad \text{Complemento a Dois}
$$

Complemento a Dois: Realizando Contas (III)

- Multiplicações e divisões seguem o mesmo processo do Complemento a Um.
  - Não podem ser realizados diretamente na representação em Complemento a Dois.
  - Operandos negativos têm seus sinais trocados.
    - Invertem-se os bits, e soma-se um.
  - Operação é realizada.
  - Se os sinais (bits mais significativos) dos operandos originais eram diferentes, resultado é negativo (bit mais significativo é 1).
- Note que há algoritmos para multiplicação direta de valores em Complemento a Dois.
  - Como o Algoritmo de Booth.

Complemento a Dois: Única Representação para Zero

- Neste caso, existe uma única representação para o Zero
  - Diferentemente do Complemento a Um

- Vamos analisar o caso de soma que apresenta o "zero negativo" em Complemento a Um, com 5 bits:
  - Eram somas entre a e $-a$.
  - Exemplo: $9_{(10)} + (-9_{(10)})$.
    - Representação em Complemento a Um: $01001 + 10110 = 11111$.
    - Representação em Complemento a Dois $01001 + 10111 = 100000$ e o 1 mais significativo é ignorado
- Neste caso, em Complemento a Dois, não há este problema.
- O zero sempre será representado apenas por $0 \ldots 0$.
  - Podendo ter o 1 na posição $n + 1$ (que será ignorado)

Complemento a Dois: Resumo

- Representação de números positivos:
  - Idêntica ao número escrito em base 2.
  - Zeros a esquerda são adicionados para que número fique com n bits.
  - **Bit mais significativo tem que ser igual a 0**
- Representação de números negativos:
  - Escreve-se a representação em Complemento a Dois do **valor absoluto**.
  - Troca-se o sinal
    - Invertem-se os bits, e soma-se um
  - **Bit mais significativo tem que ser igual a 1**

Representação por Complemento a Dois: Resumo (II)

- Somas:
  - Realizadas diretamente
  - Da mesma forma como se somam números inteiros positivos escritos na base 2
  - **Único detalhe: resultado fica restrito aos *n* bits menos significativos**
- Subtração:
  - Transformada em soma
  - Inverte-se o sinal do subtraendo
    - Invertem-se os bits, e soma-se um
- Determinar valor de número em Complemento a Dois:
  - Se o número é positivo (bit mais significativo é 0), basta convertê-lo para decimal
  - Se o número é negativo (bit mais significativo é 1), valor absoluto é determinado invertendo o sinal.
    - Invertem-se os bits, e soma-se um

Complemento a Dois: Propriedades e Dicas

- Dado um número representado em Complemento a Dois com *n* bits, podemos estendê-lo para mais bits:
  - Se o número positivo, basta adicionar zeros à esquerda.
  - Se o número é negativo, basta adicionar uns.
  - De forma geral, repete-se o bit mais significativo tantas vezes quanto necessário.
- Exemplos:
  - 01101 com cinco bits tem o mesmo valor de 00001101 com 8 bits.
  - 10001 com cinco bits tem o mesmo valor de 11110001 com 8 bits.
- Cuidado ao comparar valores escritos em Complemento a Dois:
  - Valores positivos são simples
  - Mas valores negativos podem enganar
  - Exemplo: 10001 é *mais negativo* que 11111

Representação por Complemento a Dois *vs.* Outras Representações

- Com exceção de algumas aplicações bastante especializadas, o Complemento a Dois é geralmente a melhor escolha.
  - Um único valor para o 0.
  - Operações realizadas de maneira uniforme sobre números positivos ou negativos.
  - Algoritmo trivial para a soma.
  - Inversão do sinal também bastante simples.

Representação por Complemento a Dois: Uso

- O Complemento a Dois é o esquema de representação mais popular nos computadores modernos.
  - Para números inteiros.
- Por permitir um tratamento uniforme de números com e sem sinal, *hardware* só precisa implementar um tipo de somador.
  - Simplifica e reduz custos.