

Multiplicação de inteiros

Fundamentos de Arquitetura de Computadores

Prof. John Lenon C. Gardenghi

Engenharia de Software
Faculdade do Gama
Universidade de Brasília

Multiplicação de inteiros sem sinal

The diagram illustrates the binary multiplication of two integers without sign. It shows the multiplicand (M) as 1000, the multiplier (Q) as 1001, and the resulting product (P) as 1001000. The multiplier is shifted to the left relative to the multiplicand, and the partial products are summed to produce the final result.

$$\begin{array}{r} \text{multiplicando (M)} \quad 1000 \\ \text{multiplicador (Q)} \quad 1001 \\ \hline 1000 \\ 0000 \\ 0000 \\ 1000 \\ \hline \text{produto (P)} \quad 1001000 \end{array}$$

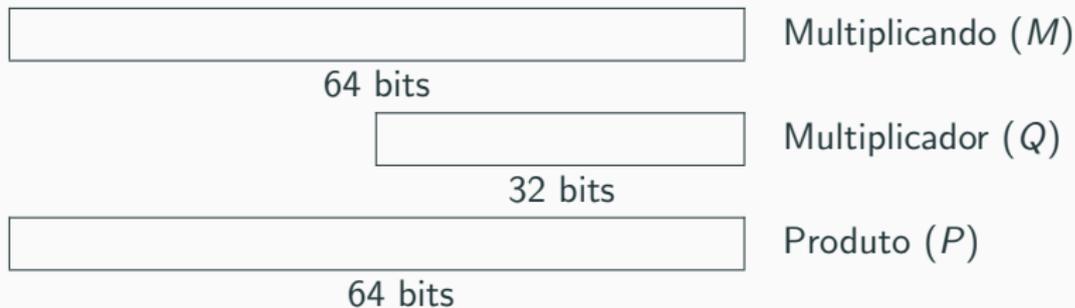
Multiplicação de inteiros sem sinal

The diagram illustrates the binary multiplication of two 4-bit numbers. The multiplicand (M) is 1000 and the multiplier (Q) is 1001. The product (P) is 1001000. The multiplier's bits are used to generate partial products: the least significant bit (1) produces 1000, the second bit (0) produces 0000, the third bit (0) produces 0000, and the most significant bit (1) produces 1000. These are then summed to produce the final product 1001000.

multiplicando (M)	1000
multiplicador (Q)	x 1001

	1000
	0000
	0000
	1000

produto (P)	1001000



Algoritmo básico de multiplicação

1. Inicialize $P = 0$ e contador = 1.
2. Faça $P = P + q_0 \times M$.
3. Faça o deslocamento lógico de um bit à esquerda em M .
4. Faça o deslocamento lógico de um bit à direita em Q .
5. Se contador = 32, pare. Senão, contador = contador +1 e volte ao Passo 2.

Otimizando a multiplicação

- É possível otimizar o algoritmo básico:
 1. E se, ao invés de deslocar o multiplicando à esquerda, deslocarmos o produto para a direita?

Otimizando a multiplicação

- É possível otimizar o algoritmo básico:
 1. E se, ao invés de deslocar o multiplicando à esquerda, deslocarmos o produto para a direita?
 2. E se salvarmos o multiplicador na porção menos significativa do produto?

Otimizando a multiplicação

- É possível otimizar o algoritmo básico:
 1. E se, ao invés de deslocar o multiplicando à esquerda, deslocarmos o produto para a direita?
 2. E se salvarmos o multiplicador na porção menos significativa do produto?
- Com isso,
 - O registrador para o multiplicador não é mais necessário
 - O registrador para o multiplicando não precisa mais ter 64 bits, mas apenas 32

Algoritmo otimizado para multiplicação

1. $P[63..32] = 0$.
2. $P[31..0] = Q$.
3. Se $P[0] = 1$, $P[63..32] = P[63..32] + M$.
4. Faça um deslocamento de 1 bit à direita em P .
5. Se não for a 32ª repetição, volte ao Passo 3.

Exemplo: $8_{\text{dec}} \times 9_{\text{dec}} = 1000_{\text{bin}} \times 1001_{\text{bin}}$.

Multiplicação de inteiros com sinal

- Há algumas alternativas:
 1. Converter M e Q para positivo sem sinal, e negar P se o sinal de M for diferente de Q .

Multiplicação de inteiros com sinal

- Há algumas alternativas:
 1. Converter M e Q para positivo sem sinal, e negar P se o sinal de M for diferente de Q .
 2. Usar o Algoritmo de Booth.

- Dois registradores de 32 bits para o produto:
 - HI: 32 bits mais significativos
 - LO: 32 bits menos significativos

Instruções no assembly MIPS

- Dois registradores de 32 bits para o produto:
 - HI: 32 bits mais significativos
 - LO: 32 bits menos significativos

- Instruções
 - `mult rs, rt / multu rs, rt`
 - produto de 64 bits nos registradores HI/LO

Instruções no assembly MIPS

- Dois registradores de 32 bits para o produto:
 - HI: 32 bits mais significativos
 - LO: 32 bits menos significativos

- Instruções
 - `mult rs, rt / multu rs, rt`
 - produto de 64 bits nos registradores HI/LO
 - `mfhi rd / mflo rd`
 - move dos registradores HI e LO para o rd

Instruções no assembly MIPS

- Dois registradores de 32 bits para o produto:
 - HI: 32 bits mais significativos
 - LO: 32 bits menos significativos
- Instruções
 - `mult rs, rt / multu rs, rt`
 - produto de 64 bits nos registradores HI/LO
 - `mfhi rd / mflo rd`
 - move dos registradores HI e LO para o rd
 - `mul rd, rs, rt`
 - os 32 bits menos significativos do produto em rd