Multiply 16-bit Integers using an 8-bit Multiplier

Nguyễn Quang Phú
Ngày 5 tháng 1 năm 2025

Tóm tắt nội dung

Bài toán "Multiply 16-bit Integers using an 8-bit Multiplier" là một ví dụ thực tế trong lập trình hệ thống và tối ưu hóa hiệu năng. Với giới hạn về phần cứng chỉ hỗ trợ phép nhân 8-bit, chúng ta cần nhân hai số nguyên 16-bit thông qua chia nhỏ các phần 8-bit. Bài toán không chỉ minh họa khả năng sử dụng hiệu quả tài nguyên phần cứng mà còn nhấn mạnh vào kỹ thuật kết hợp dữ liệu và xử lý nhị phân.

---

1. Giới thiệu

1.1 Mô tả bài toán

- Cho hai số nguyên 16-bit m và n.
- Cần thực hiện phép nhân hai số trên nhưng chỉ sử dụng phép nhân 8-bit.
- Kết quả đầu ra là một số nguyên 32-bit.

1.2 Lý do và ứng dụng

Phép nhân giới hạn 8-bit thường xuất hiện trong các vi xử lý cũ hoặc hệ thống nhúng có tài nguyên hạn chế. Việc nhân số lớn hơn bằng cách chia nhỏ thành các phần phù hợp với phần cứng là một kỹ thuật quan trọng trong lập trình tối ưu.

---

2. Ý tưởng thuật toán

- Phân tích số 16-bit:
    - Chia mỗi số thành hai phần:
        - mLow và mHigh là 8 bit thấp và cao của số m.
        - nLow và nHigh là 8 bit thấp và cao của số n.
- Thực hiện 4 phép nhân nhỏ:
    - Nhân các phần đã chia của m và n:
        - mLow × nLow
        - mHigh × nLow
        - mLow × nHigh
        - mHigh × nHigh
- Kết hợp kết quả:
    - Dịch các giá trị tương ứng để ghép thành số 32-bit:

        - Giữ nguyên $mLow \times nLow$.
        - Dịch trái 8 bit với $mHigh \times nLow$ và $mLow \times nHigh$.
        - Dịch trái 16 bit với $mHigh \times nHigh$.
- Tính toán tổng:
    - Cộng tất cả các kết quả lại để ra kết quả cuối cùng.

---

3. Phân tích thuật toán

- Độ phức tạp thời gian: O(1) (thực hiện 4 phép nhân 8-bit và một số phép cộng, dịch bit).
- Độ phức tạp không gian: O(1) (chỉ sử dụng một số biến tạm thời).

---

4. Mô tả giải pháp (Mã giả)

```
function multiply16bit (m, n):
  // Chia m và n thành các phần 8-bit
  mLow ← m & 0x00FF
  mHigh ← (m & 0xFF00) >> 8
  nLow ← n & 0x00FF
  nHigh ← (n & 0xFF00) >> 8

  // Thực hiện các phép nhân 8-bit
  mLow_nLow ← multiply8bit(mLow, nLow)
  mHigh_nLow ← multiply8bit(mHigh, nLow)
  mLow_nHigh ← multiply8bit(mLow, nHigh)
  mHigh_nHigh ← multiply8bit(mHigh, nHigh)

  // Kết hợp kết quả
  result ← mLow_nLow + ((mHigh_nLow + mLow_nHigh) << 8) + (mHigh_nHigh << 16)

  return result
```

---

5. Ví dụ minh họa

Giả sử hai số nguyên 16-bit là:

- $m = 23472$ (nhị phân: 0101101111010000)
- $n = 2600$ (nhị phân: 0000101000111000).

Phép nhân thông thường

- Kết quả trực tiếp từ phép nhân thông thường là:
    - $23472 \times 2600 = 61027200$.

Tính toán theo thuật toán

1. Phân tích số 16-bit:
    - mLow = m & 0x00FF = 208 (nhị phân: 11010000)
    - mHigh = (m & 0xFF00) >> 8 = 91 (nhị phân: 01011011)
    - nLow = n & 0x00FF = 40 (nhị phân: 00101000)
    - nHigh = (n & 0xFF00) >> 8 = 10 (nhị phân: 00001010).
2. Thực hiện 4 phép nhân 8-bit:
    - mLow × nLow = 208 × 40 = 8320.
    - mHigh × nLow = 91 × 40= 3640.
    - mLow × nHigh = 208 × 10 = 2080.
    - mHigh × nHigh = 91 × 10 = 910.
3. Kết hợp kết quả từ các phép nhân:
    - Giữ nguyên mLow × nLow = 8320.
    - Dịch trái 8 bit mHigh × nLow : 3640 << 8 = 932352.
    - Dịch trái 8 bit mLow × nHigh : 2080 << 8 = 532480.
    - Dịch trái 16 bit mHigh × nHigh : 910 << 16 = 59637760.
4. Tổng hợp tất cả các kết quả:
    - Tổng = 8320 + 932352 + 532480 + 59637760 = 61027200.

So sánh kết quả

- Phép nhân thông thường: 23472 × 2600 = 61027200.
- Kết quả từ thuật toán: 61027200.

Hai kết quả là **tương đương**, cho thấy thuật toán hoạt động chính xác.

---

6. Kết luận

Phương pháp chia nhỏ số 16-bit thành các phần 8-bit để thực hiện phép nhân không chỉ giúp tối ưu hóa trên phần cứng giới hạn, mà còn minh họa rõ ràng khả năng sử dụng thuật toán để mở rộng giới hạn của hệ thống. Đây là một ví dụ quan trọng trong tối ưu hiệu năng trên các hệ thống nhúng.

---

7. Tài liệu tham khảo

- **Mark A. Weiss**, *Data Structures and Algorithm Analysis in C++*, 4th edition, 2013.
- **GeeksforGeeks**, *"Multiply two integers without using multiplication operator".*