**Bộ nhân**

**Thuật toán nhân**

Multiplicand: 1000 (10₁₀)

xMultiplier: x 1001 (9₁₀)

----------------------

1000 (1 × 1000, đặt ở hàng đơn vị)

0000 (0 × 1000, dịch trái 1)

0000 (0 × 1000, dịch trái 2)

+ 1000 (1 × 1000, dịch trái 3)

----------------------

Product: 1001000 (72₁₀)

**Quy tắc**:

* Mỗi bit của **multiplier** (nhân tử) quyết định có **thêm multiplicand** (số bị nhân) vào tích trung gian hay không.
* Mỗi tích trung gian được **dịch trái** (shift left) theo vị trí bit.
* Cộng tất cả tích trung gian → sản phẩm cuối cùng.

**Độ dài kết quả nhân**

Nếu **multiplicand** dài n bit, **multiplier** dài m bit → sản phẩm dài tối đa n+ m bit.

Ví dụ: nhân 2 số 32 bit có thể ra 64 bit → vì vậy trong CPU phải quan tâm đến **overflow** nếu chỉ giữ lại 32 bit thấp.

**Tính chất trong số nhị phân**

Trong hệ thập phân: nhân có thể là 0×multiplicand, 1×multiplicand, 2×multiplicand, …

Nhưng trong nhị phân thì đơn giản hơn:

* Nếu bit multiplier = 1 → thêm multiplicand vào.
* Nếu bit multiplier = 0 → bỏ qua (thêm 0).
* Vì vậy nhân nhị phân chỉ cần **cộng và dịch trái**.

**Ý nghĩa cho phần cứng**

Cách tiếp cận ban đầu thường gọi là **sequential shift-and-add multiplier**:

* Duyệt từng bit multiplier từ phải sang trái.
* Nếu bit = 1 → cộng multiplicand vào tích tạm.
* Dịch multiplicand sang trái mỗi bước.

**Ý tưởng tổng quát**

Thanh ghi:

* **Multiplier (32 bit)**: chứa số nhân (sẽ shift dần để xét từng bit).
* **Multiplicand (64 bit)**: chứa số bị nhân, khởi tạo ở nửa phải 32 bit, nửa trái bằng 0 → sau đó shift trái mỗi bước.
* **Product (64 bit)**: khởi tạo bằng 0, chứa kết quả tích cộng dồn.

**Các bước lặp (mỗi chu kỳ)**

Cho mỗi bit của Multiplier (bắt đầu từ LSB):

1. **Kiểm tra bit thấp nhất (Multiplier₀)**:

* Nếu = 1 → cộng Multiplicand vào Product.
* Nếu = 0 → bỏ qua.

1. **Shift Multiplicand trái 1 bit**:

* Tương tự như dịch trái khi nhân trên giấy để tính hàng chục, hàng trăm…

1. **Shift Multiplier phải 1 bit**:

* Để chuẩn bị kiểm tra bit kế tiếp ở lần lặp sau.
* Sau 32 bước (nếu multiplier là 32 bit) → thu được Product 64 bit.

**Hạn chế & Ưu điểm**

* **Ưu điểm**: phần cứng đơn giản, dễ triển khai (ALU 64 bit + vài thanh ghi + bộ điều khiển).
* **Nhược điểm**: chậm (phải lặp 32 lần).

**Thực tế**: CPU thường ít dùng nhân so với cộng/trừ → nên việc nhân tốn nhiều chu kỳ không ảnh hưởng quá nhiều. Nhưng với Amdahl’s Law, nếu nhân xuất hiện tương đối nhiều, nó có thể giới hạn hiệu năng tổng thể.

**Tối ưu hóa tiếp theo**

* **Song song hóa**: cộng và dịch trong cùng 1 bước.
* **Thu nhỏ phần cứng**: giảm độ rộng thanh ghi/adder bằng cách quan sát các vùng “không dùng”.
* Từ đây → phát triển thành các thuật toán nhân nhanh hơn: Booth’s Algorithm, Wallace Tree, Carry-Save Multiplier,…