Ví dụ cho cộng, trừ và tràn số (overflow) – 5 ví dụ

❖ **Unsigned number (Số không dấu)**

Ví dụ 1:

- Cho A, B là số không dấu
- Dùng 8 bit biểu diễn

$A = CF_{(16)}$

$B = 13_{(16)}$

A + B = ?

Trả lời:

1. A = 1100 1111

   B = 10011 → B = 0001 0011 (điền đủ 8 bit)

2. 1100 1111
   \+
   0001 0011

   1110 0010 => không bị tràn

Vậy: $A + B = E2_{(16)}$

Ví dụ cho cộng, trừ và tràn số (overflow)

❖ **Unsigned number (Số không dấu)**

Ví dụ 2:

- Cho A và B là số không dấu
- Dùng 8 bit biểu diễn

$A = 200_{(10)}$

$B = 103_{(10)}$

A + B = ?

1. A = 1100 1000

   B = 110 0111 → B = 0110 0111 (điền đủ 8 bit)

2. 1100 1000
   \+
   0110 0111

   10010 1111 => phép cộng tại bit thứ 7 của A và B vẫn còn nhớ 1→ tràn

Vậy: A + B = 255 và bị tràn

(255 – giá trị lớn nhất của số 8 bits)

Như vậy, khi cộng hai số không dấu n bit, phép toàn bị tràn khi phép cộng tại bit cuối cùng (bit thứ n-1) vẫn còn nhớ

Ví dụ cho cộng, trừ và tràn số (overflow)

❖ Signed number (Số có dấu)

Khi thực hiện phép cộng/trừ hai số có dấu, phép toán bị tràn nếu xảy ra một trong 4 tình huống như bảng sau:

- **Hàng 1: Nếu cộng hai số dương, mà kết quả âm => phép toán bị tràn**
- **Hàng 2: Nếu cộng hai số âm, mà kết quả dương => phép toán bị tràn**
- **Hàng 3: Nếu trừ một số dương cho một số âm, mà kết quả âm => phép toán bị tràn (Tình huống này giống như hàng 1, trừ một số dương cho một số âm, tức là cộng một số dương với một số dương mà kết quả là âm thì bị tràn)**
- **Hàng 4: Nếu trừ một số âm cho một số dương, mà kết quả dương => phép toán bị tràn**
- **(Tình huống giống như hàng 2, trừ một số âm cho một số dương, tức là cộng một số âm với một số âm mà kết quả là dương thì bị tràn)**

Operation	Operand A	Operand B	Result indicating overflow
$A + B$	≥ 0	≥ 0	< 0
$A + B$	< 0	< 0	≥ 0
$A - B$	≥ 0	< 0	< 0
$A - B$	< 0	≥ 0	≥ 0

Có thể tóm tắt như sau:

- **Nếu là phép trừ quy về phép cộng dùng bù 2**
- **Quy tắc tràn số tính như sau:**
  - ✓ **Nếu cộng hai số trái dấu, kết quả không bao giờ bị tràn**
  - ✓ **Nếu cộng hai số cùng dấu mà ra kết quả khác dấu => tràn**

Ví dụ cho cộng, trừ và tràn số (overflow)

❖ **Signed number (Số có dấu)**

Ví dụ 3:

- Cho A và B là số có dấu
- Dùng 8 bit biểu diễn

$A = 200_{(10)}$

$B = 103_{(10)}$

A + B = ?

Trả lời:

1. A = 1100 1000

   B = 0110 0111

2. 1100 1000
   \+
   0110 0111

   10010 1111 => kết quả phép cộng là 0010 1111 **(không quan tâm bit 1 dư ra như số không dấu)**

3. Kiểm tra tràn:

Xét bảng bên dưới với: phép cộng, A < 0, B > 0, và kết quả dương

=> không có hàng nào tương ứng => không bị tràn

(Cộng 2 số trái dấu chắc chắn không bị tràn)

Vậy: $A + B = 00101111_{(2)} = 47_{(10)}$

Operation	Operand A	Operand B	Result indicating overflow
$A + B$	≥ 0	≥ 0	< 0
$A + B$	< 0	< 0	≥ 0
$A - B$	≥ 0	< 0	< 0
$A - B$	< 0	≥ 0	≥ 0

Ví dụ cho cộng, trừ và tràn số (overflow)

❖ **Signed number (Số có dấu)**

Ví dụ 4:

- Cho A và B là số có dấu
- Dùng 8 bit biểu diễn

$A = 247_{(10)}$

$B = 237_{(10)}$

A + B = ?

1. A = 1111 0111

   B = 1110 1101

2. 1111 0111

   \+ 1110 1101

   11110 0100 => kết quả phép cộng 111 00100 (không quan tâm bit 1 dư ra)

3. Kiểm tra tràn:

Xét bảng bên dưới với phép cộng, A < 0 và B < 0, và tổng âm → không có hàng nào tương ứng → không tràn

(Cộng 2 số âm, kết quả âm => không tràn)

Vậy: $A + B = 11100100_{(2)} = -28_{(10)}$

Operation	Operand A	Operand B	Result indicating overflow
$A + B$	≥ 0	≥ 0	< 0
$A + B$	< 0	< 0	≥ 0
$A - B$	≥ 0	< 0	< 0
$A - B$	< 0	≥ 0	≥ 0

Ví dụ cho cộng, trừ và tràn số (overflow)

❖ Signed number (Số có dấu)

Ví dụ 5:

- Cho A và B là số có dấu
- Dùng 8 bit biểu diễn

$A = 200_{(10)}$

$B = 103_{(10)}$

A - B = ?

1. A = 1100 1000

   B = 110 0111 → B = 0110 0111 (điền đủ 8 bit)

   bù 2 của B = 10011001

2. 1100 1000
   \+
   1001 1001

   10110 0001 => kết quả phép cộng 0110 0001 (không quan tâm bit 1 dư ra)

3. Kiểm tra tràn:

Xét bảng bên dưới với phép trừ, A < 0 và B > 0, kết quả dương → tương ứng với hàng thứ 3 => bị tràn

(Cộng 2 số âm mà kết quả dương => bị tràn)

Vậy: $A - B = -128_{(10)}$ và bị tràn

(-128 là giá trị nhỏ nhất của số có dấu 8 bit)

Operation	Operand A	Operand B	Result indicating overflow
$A + B$	≥ 0	≥ 0	< 0
$A + B$	< 0	< 0	≥ 0
$A - B$	≥ 0	< 0	< 0
$A - B$	< 0	≥ 0	≥ 0

Như vậy, với các phép toán bị tràn:

- Nếu là số không dấu, kết quả là giá trị lớn nhất trong giới hạn có thể biểu diễn
- Nếu là số có dấu:
  - ✓ Nếu cộng hai số âm, kết quả là giá trị nhỏ nhất trong giới hạn có thể biểu diễn
  - ✓ Nếu cộng hai số dương, kết quả là giá trị lớn nhất trong giới hạn có thể biểu diễn

Giới hạn có thể biểu diễn của một số:

- Số không dấu n-bit: từ 0 tới (2^n-1)
- Số có dấu n-bit: từ -2^{n-1} tới $(2^{n-1}-1)$

Ví dụ:

- Biểu diễn số không dấu dùng 8 bit: số sẽ từ 0 tới 255
- Biểu diễn số có dấu dùng 3 bit: số sẽ từ

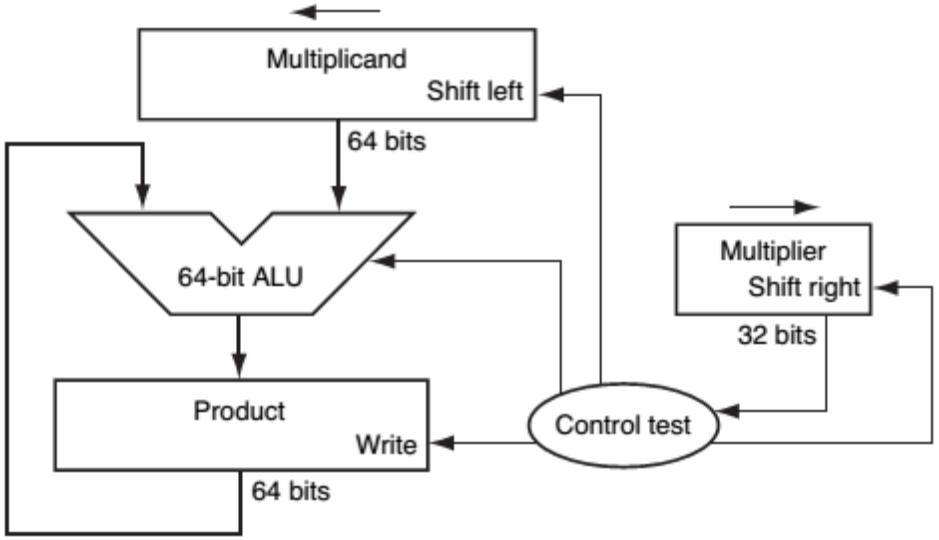
  -2^2 tới 2^2-1 hay bao gồm: -4, -3, -2, -1, 0, 1, 2, 3
- Biểu diễn số có dấu dùng 8 bit: số sẽ từ

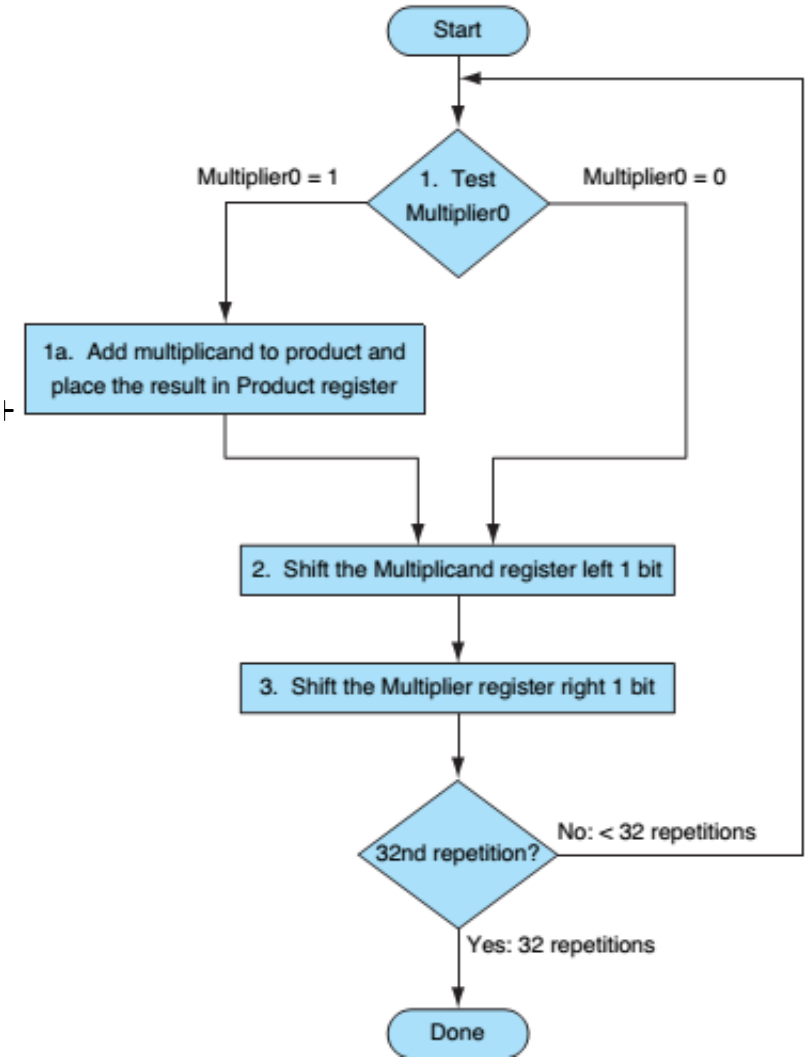
  -2^7 tới 2^7-1 hay bao gồm: -128, -127, …, 0, …, 127

Ví dụ cho phép nhân (3 ví dụ)

Ví dụ 1:

Thực hiện phép nhân $2_{(10)}$ x $3_{(10)}$ (sử dụng số 4 bit không dấu) theo cấu trúc phần cứng như hình





Lưu đồ giải thuật đi kèm cho cấu trúc phần cứng

Ví dụ 1:

$2_{(10)}$ x $3_{(10)}$ = ?

$2_{(10)}$ = 0010 (multiplicand)

$3_{(10)}$ = 0011 (multiplier)

Cấu trúc phần cứng như hình vẽ là nhân 2 số 32 bits, kết quả là số 64 bits,

Có: thanh ghi multiplicand 64 bits

thanh ghi multiplier là 32 bits

thanh ghi product là 64 bits

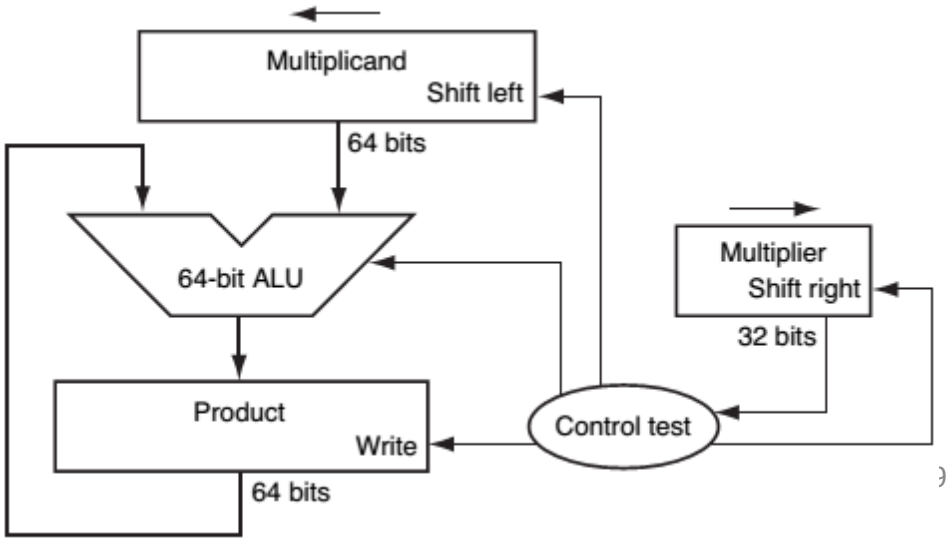
Ví dụ 1 yêu cầu nhân 2 số 4 bits không dấu, sử dụng cấu trúc phần cứng tương tự như hình, vậy kết quả phải là số 8 bits

=> thanh ghi multiplicand 8 bits (giá trị khởi tao 0000 0010)

thanh ghi multiplier là 4 bits (giá trị khởi tạo 0011)

thanh ghi product là 8 bits (giá trị khởi tạo 0000 0000)

Iteration	Step	Multiplier	Multiplicand	Product
0	Khởi tạo	0011	0000 0010	0000 0000



- Sau khi khởi tạo xong. Mỗi vòng lặp (interation) sẽ gồm 3 bước:
  - B1. Kiểm tra bit 0 của multiplier xem có bằng 1 hay không; nếu bằng 1 thì product = product + multiplicand; nếu bằng 0, không làm gì cả
  - B2. Dịch trái Multiplicand 1 bit
  - B3. Dịch phải Multiplier 1 bit
- Số vòng lặp cho giải thuật này đúng bằng số bit dùng biểu diễn (ví dụ 1 yêu cầu dùng số 4 bit, thì có 4 vòng lặp)
- Sau khi kết thúc số vòng lặp, giá trị trong thanh ghi product chính là kết quả phép nhân

Ví dụ 1:

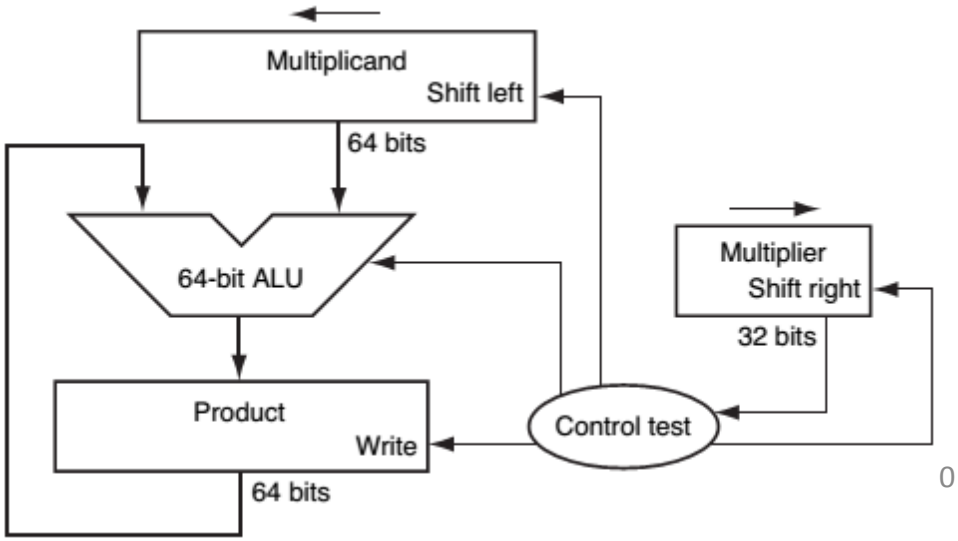
$2_{(10)}$ x $3_{(10)}$ = ?

$2_{(10)}$ = 0010 (multiplicand)

$3_{(10)}$ = 0011 (multiplier)

Iteration	Step	Multiplier	Multiplicand	Product
0	Initial values	0011	0000 0010	0000 0000
1	1a: 1 ⇒ Prod = Prod + Mcand	0011	0000 0010	0000 0010
	2: Shift left Multiplicand	0011	0000 0100	0000 0010
	3: Shift right Multiplier	0001	0000 0100	0000 0010
2	1a: 1 ⇒ Prod = Prod + Mcand	0001	0000 0100	0000 0110
	2: Shift left Multiplicand	0001	0000 1000	0000 0110
	3: Shift right Multiplier	0000	0000 1000	0000 0110
3	1: 0 ⇒ No operation	0000	0000 1000	0000 0110
	2: Shift left Multiplicand	0000	0001 0000	0000 0110
	3: Shift right Multiplier	0000	0001 0000	0000 0110
4	1: 0 ⇒ No operation	0000	0001 0000	0000 0110
	2: Shift left Multiplicand	0000	0010 0000	0000 0110
	3: Shift right Multiplier	0000	0010 0000	0000 0110

Kết quả phép nhân

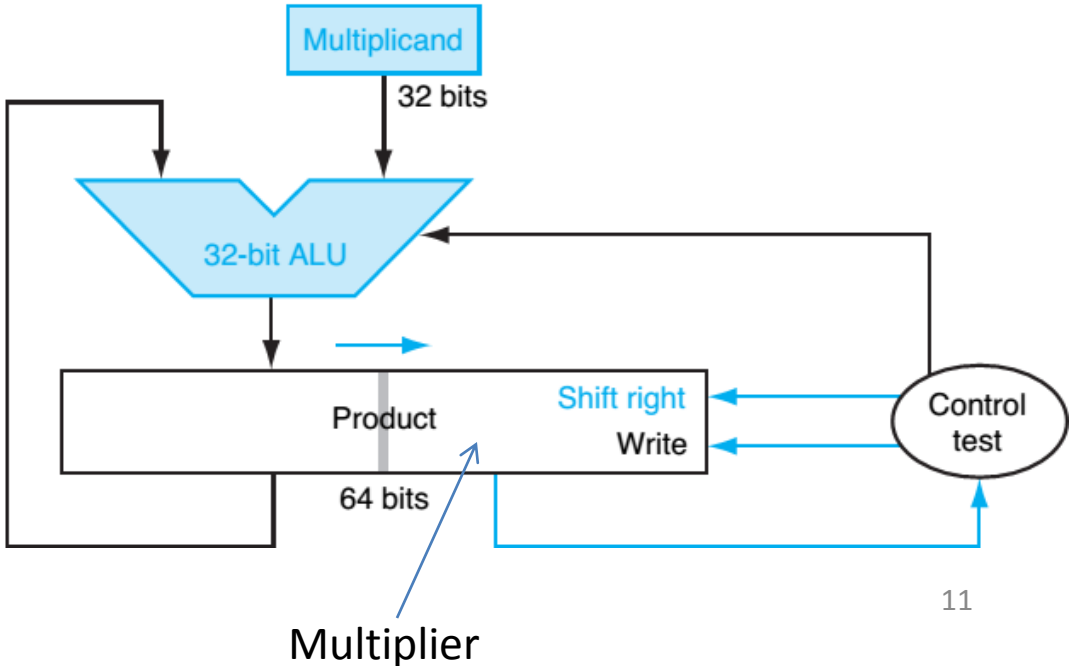


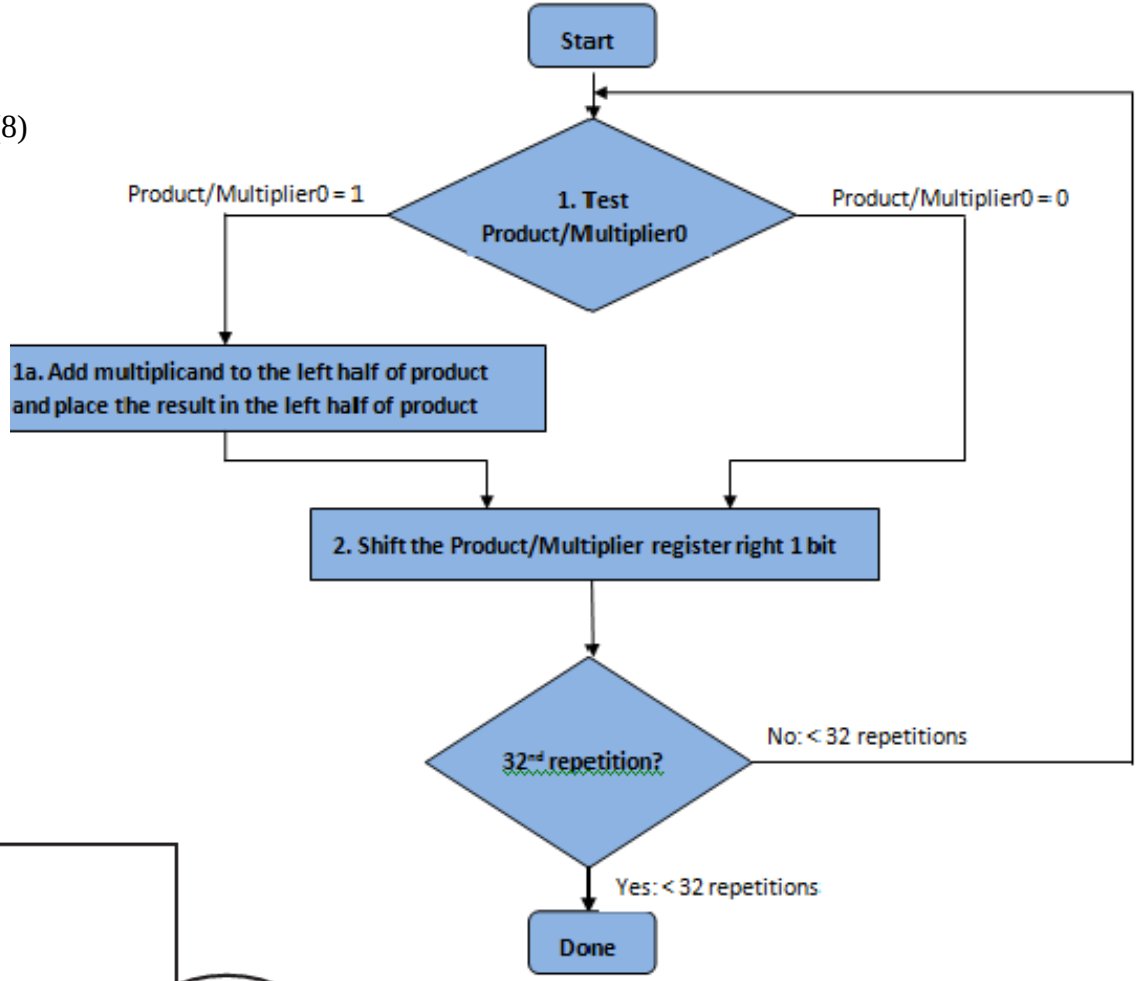
Ví dụ cho phép nhân

Ví dụ 2:

Thực hiện phép nhân $50_{(8)}$ x $23_{(8)}$ (sử dụng số 6 bit) theo cấu trúc phần cứng như hình

Thanh ghi product như trong hình là 64 bits, nhưng 32 bit đầu (nữa thấp hoặc nữa phải) được dùng để đặt multiplier vào khi khởi tao, nên dùng tên: Product/Multiplier





Lưu đồ giải thuật đi kèm cho cấu trúc phần cứng

Ví dụ 2:

$50_{(8)}$ x $23_{(8)}$ = ?

$50_{(8)}$ = 101000 (multiplicand)

$23_{(8)}$ = 010011 (multiplier)

Cấu trúc phần cứng như hình vẽ là nhân 2 số 32 bits, kết quả là số 64 bits,

Có: thanh ghi multiplicand 32 bits

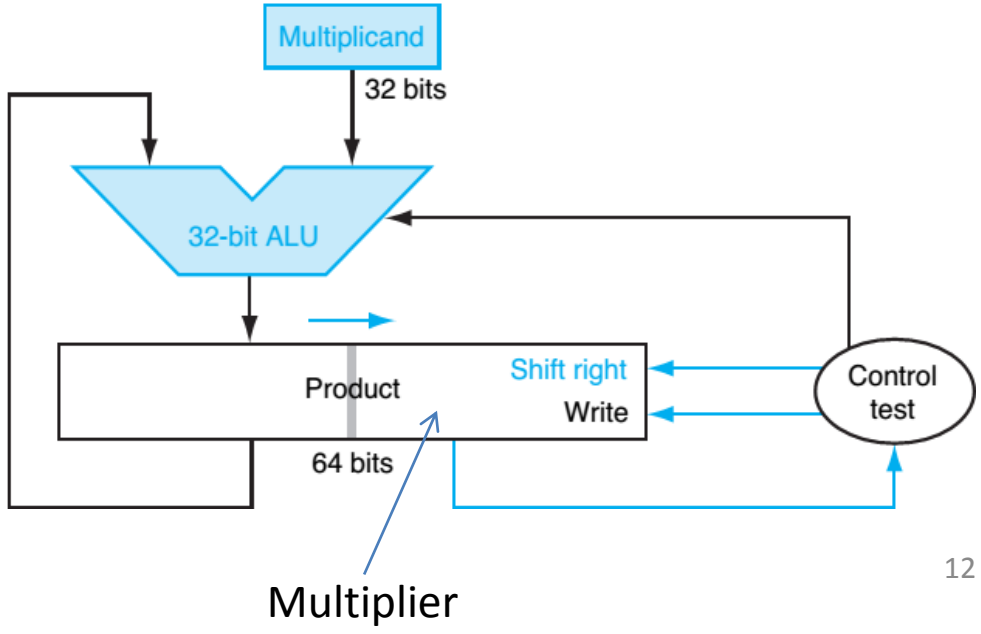
thanh ghi product 64 bits (khi khởi tạo, đưa multiplier vào 32bits thấp của product, còn nữa cao khởi tạo 0)

Ví dụ 2 yêu cầu nhân 2 số 6 bits, sử dụng cấu trúc phần cứng tương tự như hình, vậy kết quả phải là số 12 bits

⇒ thanh ghi multiplicand 6 bits (giá trị khởi tao 101000)

thanh ghi product là 12 bits (6 bit thấp là multiplier, 6 bit cao là 0 → 000000 010011)

Iteration	Step/Action	Multiplicand	Product/Multiplier
0	Khởi tạo	101000	000000 010011



- Sau khi khởi tạo xong. Mỗi vòng lặp (interation) sẽ gồm 2 bước:
  - B1. Kiểm tra bit 0 của Product/multiplier xem có bằng 1 hay không; nếu bằng 1 thì nữa cao của product/multiplier = nữa cao của product/multiplier + multiplicand; nếu bằng 0, không làm gì cả
  - B2. Dịch phải Product/Multiplier 1 bit
- Số vòng lặp cho giải thuật này đúng bằng số bit dùng biểu diễn (ví dụ 2 yêu cầu dùng số 6 bit, thì có 6 vòng lặp)
- Sau khi kết thúc số vòng lặp, giá trị trong thanh ghi product chính là kết quả phép nhân

Iteration	Step/Action	Multiplicand	Product/Multiplier
0	Khởi tạo	101000	000000 010011
1	1a: 1 => Nữa cao của Product/Multiplier = Nữa cao Product/Multiplier + Multiplicand	101000	101000 010011
	2: shift right Product/Multiplicand	101000	010100 001001
2	1a: 1 => Nữa cao của Product/Multiplier = Nữa cao Product/Multiplier + Multiplicand	101000	111100 001001
	2: shift right Product/Multiplicand	101000	011110 000100
3	1: 0 => không làm gì	101000	011110 000100
	2: shift right Product/Multiplicand	101000	001111 000010
4	1: 0 => không làm gì	101000	001111 000010
	2: shift right Product/Multiplicand	101000	000111 100001
5	1a: 1 => Nữa cao của Product/Multiplier = Nữa cao Product/Multiplier + Multiplicand	101000	101111 100001
	2: shift right Product/Multiplicand	101000	0101111 10000
6	1: 0 => không làm gì	101000	0101111 10000
	2: shift right Product/Multiplicand	101000	00101111 1000

Kết quả phép nhân

Hoặc có thể trình bày ngắn gọn như bảng sau:

Step	Action	Multiplicand	Product/Multiplier
0	Initial Vals	101 000	000 000 010 011
1	Prod = Prod + Mcand	101 000	101 000 010 011
	Rshift Product	101 000	010 100 001 001
2	Prod = Prod + Mcand	101 000	111 100 001 001
	Rshift Mplier	101 000	011 110 000 100
3	lsb = 0, no op	101 000	011 110 000 100
	Rshift Mplier	101 000	001 111 000 010
4	lsb = 0, no op	101 000	001 111 000 010
	Rshift Mplier	101 000	000 111 100 001
5	Prod = Prod + Mcand	101 000	101 111 100 001
	Rshift Mplier	101 000	010 111 110 000
6	lsb = 0, no op	101 000	010 111 110 000
	Rshift Mplier	101 000	001 011 111 000