Chapter 03: Computer Arithmetic

Lesson 04:

Arithmetic Operations—

Multiplication of Integer numbers

Objective

- Understand the Computer arithmetic operations in Multiplication with unsigned numbers
- Time Taken in Multiplication
- Signed Operand Multiplication of Integers

Multiplication Process

Multiplication

- Unsigned integer multiplication— handled in a similar manner to the way we multiply multidigit decimal numbers
- The first input to the multiplication

is multiplied by each bit of the second input separately, and the results added by

binary multiplication

Multiplication Process

- Simplified by the fact that the result of multiplying a number by a bit is either the original number or 0
- Hardware less complex

Multiplication Process 11 × 5

- Multiplying 11 (0b1011), multiplicand (Y) by 5 (0b0l01), multiplier (X)
- First, 0b1011 is multiplied by each bit of 0b0l01 to get the partial products shown
- Then, the partial products are added to get the final result

Example of Multiplication Process

- 0b1011 (+11)
- × 0b0101 (+5)
- 1011
- 0000
- 1011
- \+ 0000
- 0b 110 111 (+55)
-

Multiplication Process

- Note that each

  ‘Successive partial product is shifted one position to the left to account for the differing place values of the bits in the second input’

Multiplication Circuit

Example of a Multiplication Circuit

- A method for implementation of the product of two 8-bit numbers using a sequential circuit and one number 8-bit adder
- Assume that two 8-bit registers, *A* (accumulator) and *M* (multiplier) are used for addition
- *A*-*M* 16-bit combination of two registers for the partial product at each step 0 to Step 7

Example of a Multiplication Circuit

- Let Y (multiplicand) = 0b 10111011 (an unsigned number 187 decimal)
- Let X (multiplier) = 0b 01010101 (an unsigned number 85 decimal)

Step 1 of a Multiplication Process

1. The addition (denoted by step A) is done 8 times

- Shift (denoted by step B) is also done 8 times

Step 2 of a Multiplication Process

2. Shift is to the right when we use multiplier bits from msb down to lsb during steps 0 to 7

- Shift takes 17-bits into account: the carry plus A-bits and M-bits
- C will shift to msb of *A* in step B

Step 3 of a Multiplication Process

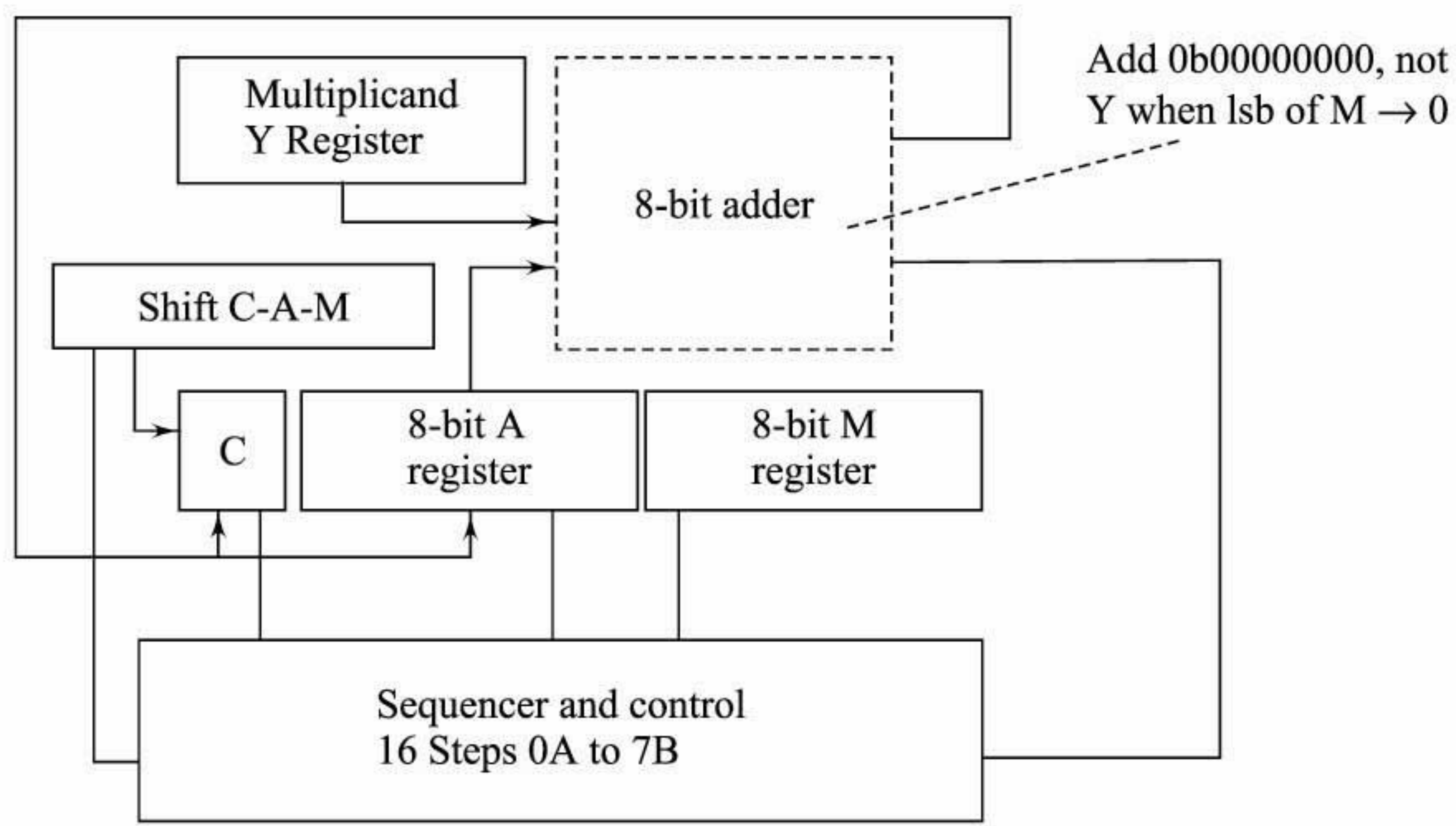
3. Each step has two parts, *A* and *B*, and a C-flag of 1-bit stores the -carry out shift-right from maximum significant bit (msb) at *A*

Step 4 of a Multiplication Process

4. Two registers shift and one addition per step = 16 shifts of 8-bit registers and 8 additions of 8-bit registers = 24 operations + 4 clear plus load M = 28

In the present example, the total number of expected operations = 24, 4 addition operations did not occur due to the nature of the multiplier

Sequential circuit and 8-bit multiplier implementation by 16 steps (8-cycles) of addition and shifts



Steps in two Dimesnsional Array to get partial product

Step	C-flag *	First Register for *A*	Second Register for *M*	Action Taken	Number of operations (instructions)
Start	0	0b00000000	0b00000000		3 for clearing *C*, *A* and *M*
	0	0b00000000	0b01010101	Load Multiplier *X* in *M*	1
Step 0A	0	10111011	01010101	Add multiplicand *Y* in *A*, result in C-*A*	1
Step 0B	0	01011101	10101010	Shift C-A-M	2
Step 1A	0	01011101	10101010	Do not add (lsb of M = 0)	0 (1)
Step 1B	0	00 101110	11010101	Shift C-A-M	2
Step 2A	0	11101001	11010101	Add *Y* , result in C-*A*	1
Step 2B	0	01110100	11101010	Shift	2

Steps in two Dimesnsional Array to get partial product

Step 3A	0	01110100	11101010	Do not add (lsb = 0)	0(1)
Step 3B	0	00111010	01110101	Shift C-A-M	2
Step 4A	0	11110101	01110101	Add *Y*, result in C-*A*	1
Step 4B	0	01111010	10111010	Shift	2
Step 5A	0	01111010	10111010	Do not add (lsb = 0)	0(1)
Step 5B	0	00111101	01011101	Shift C-A-M	2
Step 6A	0	11111000	01011101	Add *Y*, result in C-*A*	1
Step 6 B	0	01111100	00101110	Shift C-A-M	2
Step 7 A	0	01111100	00101110	Do not add (lsb = 0)	0(1)
Step 7 B	0	00111110	00010111	Shift C-A-M	2
Answer	0	0011 1110 0001 0111= 0x3e17		Decimal 15895	Total 24 (28)

* after the shift-left from the msb of *A*.

Two dimensional array of Full adders to get partial products

Multiplier X Bits	Input Y to Full Adders	Cycle Number	7	6	5	4	3	2	1	0
	0000		0	0	0	0	0	0	0	0
1 (lsb)	00001011	0					1	0	1	1
0	0000	1				0	0	0	0	←······
1	000101100	2			1	0	1	1	←······	
0 (msb)	0000	3		0	0	0	0	←······		
			FA	FA	FA	FA	FA	FA	FA	FA
			FA	FA	FA	FA	FA	FA	FA	
			FA	FA	FA	FA	FA	FA		
			FA	FA	FA	FA	FA			
			FA	FA	FA	FA				
			FA	FA	FA					
			FA	FA						
			FA							

Arrow in a column shows implementation of shift.

Time taken in Multiplication

Example: Compute the time taken for an 8-bit multiplication using the circuit

- Assume that an 8-bit adder adds in 0.008 μs, and a shift of carry bit + 8-bit accumulator and multiplier registers' shift by one bit after each addition takes 0.002 μs.
- Assume that the time for other operations is 0.001μs

Solution

- The multiplication circuit design has *8*-bit addition and shift lefts, both 8-times
- There will be 8-bit additions in 8-bit multiplier and 8-times shifts

Solution

- Time in 2 registers and carry clear= 0.003 μs
- Time in register loads = 0.001 μs
- Time taken for 8-bit addition before the shift = 0.008 μs.
- Time taken for 1-bit shift = 0.002 μs
- Time taken for 8-add and 8-shifts = 0.003 μs + 0.001 μs + 8 × (0.008 + 0.002) μs = 8 × 10 ns + 4= 84 ns
- Multiplication Time = 84 ns

Signed operand multiplication of integers

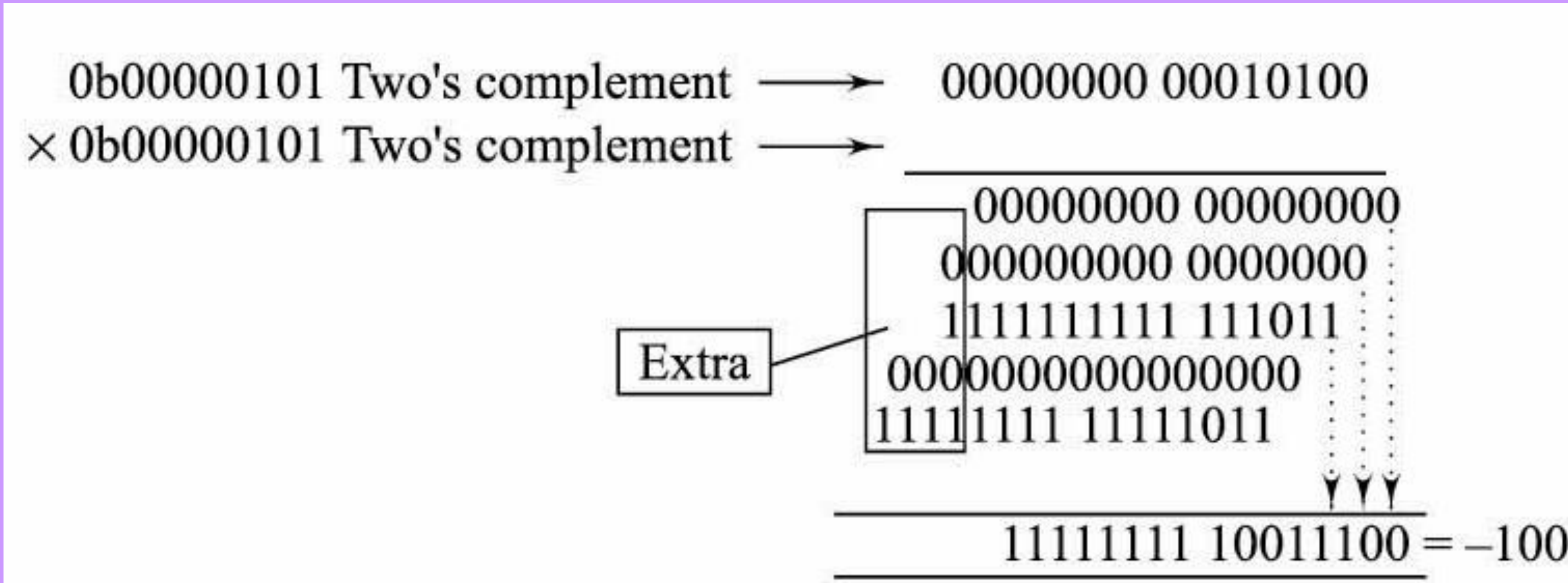
Signed Integer Multiplication

- Signed integer multiplication handled in a manner similar to the way unsigned integers
- Multiply multidigit decimal numbers and accumulate the partial products

Multiplication Process

- However, for an n-bit signed multiplier, there should be a sign extension up to 2^n bits and we must find the two's complement of both

Multiplication of signed numbers



Summary

We learnt

- Multiplication process is a process in which the first input to the multiplication is multiplied by each bit of the second input separately
- The results added by binary multiplication
- Successive partial product shifted one position to the left to account for the differing place values of the bits in the second input

End of Lesson 4 on

Arithmetic Operations─ Multiplication of Integer numbers