COMP2611: Computer Organization

Arithmetic Logic Unit II

Overview

- You will learn the following in this tutorial:
  - Multiplication, Booth algorithm
  - Division
  - Floating point arithmetic

Arithmetic Logic Unit II

Review of Arithmetic logic Units

- Multiplication
- Booth Algorithm
- Division
- Floating Point Arithmetic

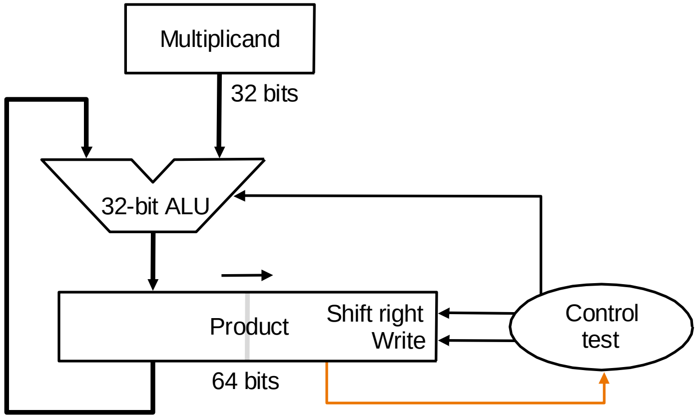
Exercises

Multiplication Hardware - Refined Version

- **32-bit ALU**

- **Two registers**:
  - **Multiplicand register: 32 bits**
  - **Product register: 64 bits**

    (right half also used for storing multiplier)



- **Operations**:
  - The right half of the product register is initialized to the multiplier, and its left half is initialized to 0
  - The two right-shifts at each step for version 2 are combined into only a single right-shift because the product and multiplier registers have been combined

- Let's consider multiplying 0010_2 and 0110_2

		Convention		Booth	
Multiplicand		0010		0010	
Multiplier	x	0110		0110	
	+	0000	+	0000	shift
	+	0010	-	0010	subtract
	+	0010	+	0000	shift
	+	0000	+	0010	add
Product	=	0001100	=	0001100	

Idea of Booth Algorithm

- Looks at two bits of multiplier at a time from right to left
- Assume that shifts are much faster than adds
- Basic idea to speed up the calculation: avoid unnecessary additions

Example

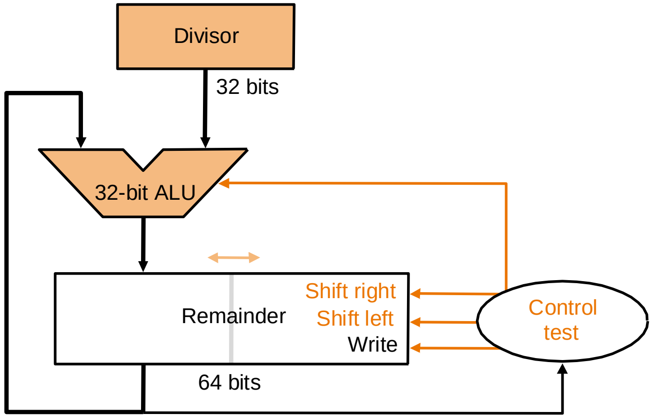
- Multiplication of two signed numbers +2 and +6 (0010 and 0110)

Multiplier

Iteration	Multiplicand (M)	Product (P)	Remark
0	0010	0000 0110 0	Initial state
1		0000 0110 0	No operation
		0000 0011 0	P = P >> 1
2		1110 0011 0	Left(P) = Left(P) - M
		1111 0001 1	P = P >> 1
3		1111 0001 1	No operation
		1111 1000 1	P = P >> 1
4		0001 1000 1	Left(P) = Left(P) + M
		0000 1100 0	P = P >> 1

Division Hardware – Improved Version

- **32-bit ALU**
- **Two registers**:
  - **Divisor register: 32 bits**
  - **Remainder register: 64 bits**
    (right half also used for storing quotient)



- **Operations**:
  - 32-bit divisor is always subtracted from the left half of remainder register
    - The result is written back to the left half of the remainder register
  - The right half of the remainder register is initialized with the dividend
    - Left shift remainder register by one before starting
  - The new order of the operations in the loop is that the remainder register will be **shifted left one time too many**
    - Thus, **final correction step:** must **right shift back only the remainder** in the left half of the remainder register

Floating-Point Addition

- Example: $9.999_{10} \times 10^{1} + 1.610_{10} \times 10^{-1}$
- Assumptions:
  - Significand size = 4 decimal digits
  - Exponent size = 2 decimal digits

Algorithm:

1. Align the decimal point of the number that has the smaller exponent
   - e.g. $1.610_{10} \times 10^{-1}$ becomes $0.016_{10} \times 10^{1}$
2. Add the significands of the two numbers together
   - e.g. $9.999_{10} \times 10^{1} + 0.016_{10} \times 10^{1} = 10.015_{10} \times 10^{1}$
3. Normalize the sum
   - e.g. $10.015_{10} \times 10^{1}$ becomes $1.0015_{10} \times 10^{2}$
4. Round the normalized sum
   - e.g. $1.0015_{10} \times 10^{2}$ becomes $1.002_{10} \times 10^{2}$

Floating-Point Multiplication

- Example: $(1.110_{10} \times 10^{10}) \times (9.200_{10} \times 10^{-5})$
- Assumptions:
  - Significand size = 4 decimal digits
  - Exponent size = 2 decimal digits

Algorithm:

1. Add the exponents together,
   - new exponent = 10 + (-5) = 5
2. Multiply the significands together
   - new significand = $1.110_{10} \times 9.200_{10} = 10.212_{10}$
3. Normalize the product,
   - $10.212_{10} \times 10^5 \Rightarrow 1.0212_{10} \times 10^6$
4. Round the product
   - $1.0212_{10} \times 10^6 \Rightarrow 1.021_{10} \times 10^6$
5. Find the sign of the product
   - $+1.021_{10} \times 10^6$

Arithmetic Logic Unit II

Review of Arithmetic logic Units

- Multiplication
- Booth Algorithm
- Division
- Floating Point Arithmetic

Exercises

Exercise

- Question 1: According to the multiplication hardware – refined version, do multiplication of two unsigned number 5 x 7 (0101 and 0111), fill in the table below.

Iteration	Multiplicand (M)	Product (P)	Remark
0	0101		Initial state
1			
2			
3			
4			

Exercise

- Question 2: According to Booth algorithm, do multiplication of two signed number +2 and -3 (0010 and 1101), fill in the table below.

Iteration	Multiplicand (M)	Product (P)	Remark
0	0010		Initial state
1			
2			
3			
4			

Exercise

- Question 3: According to the division hardware – improved version. Divide 8 (1000) by 3 (0011), fill in the table below.

Iteration	Divisor (D)	Remainder (R)	Remark
0	0011		Initial state
1			
2			
3			
4			
extra			

Exercise

- Question 4: Add 0.25_{10} and -0.875_{10} in binary according to the algorithm (Assume for simplicity that we only keep 4 bits of precision)

Exercise

- Question 5: Multiply 0.125_{10} and -0.625_{10} in binary according to the algorithm (Assume for simplicity that we only keep 4 bits of precision)

Conclusion

- Today we have reviewed:
  - Multiplication, Booth algorithm
  - Division
  - Floating point arithmetic