Assignment 3

何泽安 (He Zean)
12011323
Mar. 30, 2022

Problem 1

We first convert the multiplicand and the multiplier into unsigned binary number: $50_{dec} = 11\ 0010_{bin}$, $9_{dec} = 1001_{bin}$. Note that for the 6-bit production, we need to use a 12-bit ALU, store the multiplier into a 6-bit register, and store the multiplicand and the product into 12-bit registers. We also need 6 iteration for this multiplication:

Iteration	Step	Multiplier	Multiplicand	Product
0	Initial values	00 100**1**	0000 0011 0010	0000 0000 0000
1	1a: 1 ⇒ Prod = Prod + Mcand	00 1001	0000 0011 0010	0000 0011 0010
	2: Shift left Multiplicand	00 1001	0000 0110 0100	0000 0011 0010
	3: Shift right Multiplier	00 010**0**	0000 0110 0100	0000 0011 0010
2	1: 0 ⇒ No operation	00 0100	0000 0110 0100	0000 0011 0010
	2: Shift left Multiplicand	00 0100	0000 1100 1000	0000 0011 0010
	3: Shift right Multiplier	00 001**0**	0000 1100 1000	0000 0011 0010
3	1: 0 ⇒ No operation	00 0010	0000 1100 1000	0000 0011 0010
	2: Shift left Multiplicand	00 0010	0001 1001 0000	0000 0011 0010
	3: Shift right Multiplier	00 000**1**	0001 1001 0000	0000 0011 0010
4	1a: 1 ⇒ Prod = Prod + Mcand	00 0001	0001 1001 0000	0001 1100 0010
	2: Shift left Multiplicand	00 0001	0011 0010 0000	0001 1100 0010
	3: Shift right Multiplier	00 000**0**	0011 0010 0000	0001 1100 0010
5	1: 0 ⇒ No operation	00 0000	0011 0010 0000	0001 1100 0010
	2: Shift left Multiplicand	00 0000	0110 0100 0000	0001 1100 0010
	3: Shift right Multiplier	00 000**0**	0110 0100 0000	0001 1100 0010
6	1: 0 ⇒ No operation	00 0000	0110 0100 0000	0001 1100 0010
	2: Shift left Multiplicand	00 0000	1100 1000 0000	0001 1100 0010
	3: Shift right Multiplier	00 0000	1100 1000 0000	0001 1100 0010

Finally, we have the product $0001\ 1100\ 0010_{bin} = 450_{dec}$.

Problem 2

We first convert the multiplicand and the multiplier into unsigned binary number: $98_{dec} = 0110\ 0010_{bin}$, $17_{dec} = 10001_{bin}$. We then have the 8-bit multiplicand stays in the register, and a 16-bit register to store the multiplier and the product (with an initialize value 0). Then do 8 iterations until all bits of multiplier are

shifted right out of the register.

Iteration	Step	Multiplicand	Product
0	Initial values	0110 0010	0000 0000 | 0001 0001
1	1a: 1 ⇒ Prod = Prod + Mcand	0110 0010	0110 0010 | 0001 0001
	2: Shift right Multiplicand	0110 0010	0 0110 0010 | 0001 000
2	1: 0 ⇒ No operation	0110 0010	0 0110 0010 | 0001 000
	2: Shift right Multiplicand	0110 0010	00 0110 0010 | 0001 00
3	1: 0 ⇒ No operation	0110 0010	00 0110 0010 | 0001 00
	2: Shift right Multiplicand	0110 0010	000 0110 0010 | 0001 0
4	1: 0 ⇒ No operation	0110 0010	000 0110 0010 | 0001 0
	2: Shift right Multiplicand	0110 0010	0000 0110 0010 | 0001
5	1a: 1 ⇒ Prod = Prod + Mcand	0110 0010	0110 1000 0010 | 0001
	2: Shift right Multiplicand	0110 0010	0 0110 1000 0010 | 000
6	1: 0 ⇒ No operation	0110 0010	0 0110 1000 0010 | 000
	2: Shift right Multiplicand	0110 0010	00 0110 1000 0010 | 00
7	1: 0 ⇒ No operation	0110 0010	00 0110 1000 0010 | 00
	2: Shift right Multiplicand	0110 0010	000 0110 1000 0010 | 0
8	1: 0 ⇒ No operation	0110 0010	000 0110 1000 0010 | 0
	2: Shift right Multiplicand	0110 0010	0000 0110 1000 0010 |

As the 8 bits all shifted away, we have the all 16 bits of the product, $0000\ 0110\ 1000\ 0010_{bin} = 1666_{dec}$.

Problem 3

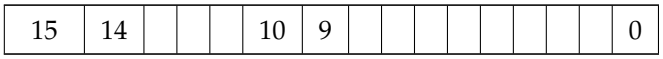
We first fill the quotient with 0, or 000000_{bin}. Then initialize the divisor with $18_{dec} = 010010_{bin}$: note that this 12-bit register should be filled with it in MSB, aka. 010010 000000.Then fill the remainder as $60_{dec} = 111100_{bin}$ (with prefix 0s).

Iteration	Step	Quotient	Divisor	Remainder
0	Initial values	00 0000	0100 1000 0000	0000 0011 1100
1	1: Rem = Rem –Div	00 0000	0100 1000 0000	1011 1011 1100
	2b: Rem < 0 ⇒ +Div, sll Q, Q0 = 0	00 0000	0100 1000 0000	0000 0011 1100
	3: Shift Div right	00 0000	0010 0100 0000	0000 0011 1100
2	1: Rem = Rem –Div	00 0000	0010 0100 0000	1101 1111 1100
	2b: Rem < 0 ⇒ +Div, sll Q, Q0 = 0	00 0000	0010 0100 0000	0000 0011 1100
	3: Shift Div right	00 0000	0001 0010 0000	0000 0011 1100
3	1: Rem = Rem –Div	00 0000	0001 0010 0000	1111 0001 1100
	2b: Rem < 0 ⇒ +Div, sll Q, Q0 = 0	00 0000	0001 0010 0000	0000 0011 1100
	3: Shift Div right	00 0000	0000 1001 0000	0000 0011 1100
4	1: Rem = Rem –Div	00 0000	0000 1001 0000	1111 1010 1100

	2b: Rem < 0 ⇒ +Div, sll Q, Q0 = 0	00 0000	0000 1001 0000	0000 0011 1100
	3: Shift Div right	00 0000	0000 0100 1000	0000 0011 1100
	1: Rem = Rem –Div	00 0000	0000 0100 1000	1111 1111 0100
5	2b: Rem < 0 ⇒ +Div, sll Q, Q0 = 0	00 0000	0000 0100 1000	0000 0011 1100
	3: Shift Div right	00 0000	0000 0010 0100	0000 0011 1100
	1: Rem = Rem –Div	00 0000	0000 0010 0100	0000 0001 1000
6	2a: Rem ≥ 0 ⇒ sll Q, Q0 = 1	00 0001	0000 0010 0100	0000 0001 1000
	3: Shift Div right	00 0001	0000 0001 0010	0000 0001 1000
	1: Rem = Rem –Div	00 0001	0000 0001 0010	0000 0000 0110
7	2a: Rem ≥ 0 ⇒ sll Q, Q0 = 1	00 0011	0000 0001 0010	0000 0000 0110
	3: Shift Div right	00 0011	0000 0000 1001	0000 0000 0110

After 7 iterations, we get the quotient as $11_{\text{bin}} = 3_{\text{dec}}$, and the remainder as $110_{\text{bin}} = 6_{\text{dec}}$.

Problem 4



$$\text{half precision} = (-1)^{b_{15}} \times 2^{(b_{14}b_{13}\cdots b_{10})_{\text{bin}}-15} \times (1.b_9b_8\cdots b_0)_{\text{bin}}$$

The sign bit of 0.9375_{dec} is 1, then we (try to approximately) represent the decimal number as binary: $0.9375 = 1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4}$ aka. $0.9375_{\text{bin}} = 0.1111_{\text{bin}} = 1.111_{\text{bin}} \cdot 2^{-1}$, which also means that we need $(b_{14}b_{13}\cdots b_{10})_{\text{bin}} - 15_{\text{dec}} = -1_{\text{dec}}$, say $(b_{14}b_{13}\cdots b_{10})_{\text{bin}} = 14_{\text{dec}} = 01110_{\text{bin}}$.

$$0.9375_{\text{dec}} = (1\ 01110\ 1110\ 0000\ 00)_{\text{half prec}}$$

By IEEE 754, the exponents 00000 and 11111 are reserved, therefore the range of fraction part is $00001_text{bin} - 15_{\text{dec}} = -14_{\text{dec}}$ to $11110_{\text{bin}} - 15_{\text{dec}} = 15_{\text{dec}}$. Then the range of fraction is from $1.\underbrace{00\cdots0}_{\text{10 bits}}$ (significand = 1.0) to $1.\underbrace{11\cdots1}_{\text{10 bits}}$ (significand ≈ 2.0). We can summarize the range of half precision as

$$(-2.0 \times 2^{15}, -1.0 \times 2^{-14}], [1.0 \times 2^{-14}, 2.0 \times 2^{15})$$

Its relative precision is

$$\begin{aligned} \Delta A/|A| &= (2^{-10} \cdot 2^y)/|1.xxx \cdot 2^y| \\ &\le (2^{-10} \cdot 2^y)/|1 \cdot 2^y| \\ &= 2^{-10} \end{aligned}$$

Problem 5

We first represent these two numbers as float point binary numbers (note that they both can be represent by 16-bit, without losing precision).

$$26.125_{\text{dec}} = 11010.001_{\text{bin}} = (1.1010\ 001 \times 2^4)_{\text{bin}}$$

$$0.2900390625_{\text{dec}} = 0.0100101001_{\text{bin}} = (1.0010\ 1001 \times 2^{-2})_{\text{bin}}$$

Then shift the smaller number right (increasing its exponent from -2 to 4, thus shift right the fraction 6 bits), as the second and third row shows (in the third row, guard, round and sticky bits are not their real value, but the shifted bits).

We can easily tell the extra bits of the shifted number: take 1 as guard, 0 as round, and since the number in the right of round bit (0100) is not all zero, set the sticky as 1.

	10 bits fraction										guard	round	sticky			
(1.)	**1**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**			
1.	*0*	*0*	*1*	*0*	*1*	*0*	*0*	*1*	*0*	*0*	⇒ shift 6 bits					
(1.)	0	0	0	0	0	*1*	*0*	*0*	*1*	*0*	*1*	*0*	*0*	*1*	*0*	*0*
(1.)	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**1**	**0**	**1**			
(1.)	**1**	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**0**	**1**	**0**	**1**			

Row 4 is the real bits and extra bits in the add process, we then add row 1 and row 4, get row 5. Now that we have guard-round-sticky = 101, which prompts us to round up. Thus the rounded result is $1.1010011011_{\text{bin}} \times 2^4 = 26.421875_{\text{dec}}$.