CS202

Homework3

11911827 张皓淇

3.9

> Assume 151 and 214 are signed 8-bit decimal integers stored in two's complement format. Calculate 151 + 214 using saturating arithmetic. The result should be written in decimal. Show your work.

Since 151 and 214 are stored in two's complement format, so we need to find the original number.

$(151)_{10} = (10010111)_2$

$(214)_{10} = (11010110)_2$

Then subtract 1:

$(10010111)_2 - 1 = (10010110)_2$

$(11010110)_2 - 1 = (11010101)_2$

Then get the original number:

$(01101001)_2 = (105)_{10}$

The first number is -105

$(00101010)_2 = (42)_{10}$

The second number is -42

(-105)+(-42)=-147

Since -147<-128, the final result would be -128 by using saturation arithmetic.

3.10

> Assume 151 and 214 are signed 8-bit decimal integers stored in two's complement format. Calculate 151 - 214 using saturating arithmetic. The result should be written in decimal. Show your work.

According to 3.9, we can get

(-105)-(-42)=-63

Since -63>-128, the final result would be -63.

3.11

> Assume 151 and 214 are unsigned 8-bit integers. Calculate 151 + 214 using saturating arithmetic. The result should be written in decimal. Show your work.

151+214=365

Since 365>255, the final result would be 255 by using saturation arithmetic.

3.13

> Using a table similar to that shown in Figure 3.6, calculate the product of the hexadecimal unsigned 8-bit integers 62 and 12 using the hardware described in Figure 3.5. You should show the contents of each register on each step.

$(62)_{hex} * (12)_{hex}$

Iteration	Step	Multiplicand	Product
0	Initial values	0000 0110 0010	0000 0000 0001 0010
1	1: 0 => No operation	0000 0110 0010	0000 0000 0001 0010
	2: Shift right Product	0000 0110 0010	0000 0000 0000 1001
2	1: 1 => high 32 bits of Prod+ = Mcand	0000 0110 0010	0110 0010 0000 1001
	2: Shift right Product	0000 0110 0010	0011 0001 0000 0100
3	1: 0 => No operation	0000 0110 0010	0011 0001 0000 0100
	2: Shift right Product	0000 0110 0010	0001 1000 1000 0010
4	1: 0 => No operation	0000 0110 0010	0001 1000 1000 0010
	2: Shift right Product	0000 0110 0010	0000 1100 0100 0001
5	1: 1 => high 32 bits of Prod+ = Mcand	0000 0110 0010	0110 1110 0100 0001
	2: Shift right Product	0000 0110 0010	0011 0111 0010 0000
6	1: 0 => No operation	0000 0110 0010	0011 0111 0010 0000
	2: Shift right Product	0000 0110 0010	0001 1011 1001 0000
7	1: 0 => No operation	0000 0110 0010	0001 1011 1001 0000
	2: Shift right Product	0000 0110 0010	0000 1101 1100 1000
8	1: 0 => No operation	0000 0110 0010	0000 1101 1100 1000
	2: Shift right Product	0000 0110 0010	0000 0110 1110 0100

$(0000011011100100)_2 = (06E4)_{hex}$

$(62)_{hex} * (12)_{hex} = (6E4)_{hex}$

3.16

> Calculate the time necessary to perform a multiply using the approach given in Figure 3.7 if an integer is 8 bits wide and an adder takes 4 time units.

There are 4 adders in the first layer, 2 adders in the second layer, 1 adder in the first layer.

3 layers in total.

Calculate time: 3*4=12 time units

3.18

> Using a table similar to that shown in Figure 3.10, calculate 74 divided by 21 using the hardware described in Figure 3.8. You should show the contents of each register on each step. Assume both inputs are unsigned 6-bit integers.

Iteration	Step	Quotient	Divisor	Remainder
0	Initial values	0000	0001 0101 0000 0000	0000 0000 0100 1010
1	1: Rem = Rem – Div	0000	0001 0101 0000 0000	1110 1011 0100 1010
	2b: Rem < 0 ⇒ +Div, sll Q, Q0 = 0	0000	0001 0101 0000 0000	0000 0000 0100 1010
	3: Shift Div right	0000	0000 1010 1000 0000	0000 0000 0100 1010
2	1: Rem = Rem – Div	0000	0000 1010 1000 0000	1111 0101 1100 1010
	2b: Rem < 0 ⇒ +Div, sll Q, Q0 = 0	0000	0000 1010 1000 0000	0000 0000 0100 1010
	3: Shift Div right	0000	0000 0101 0100 0000	0000 0000 0100 1010
3	1: Rem = Rem – Div	0000	0000 0101 0100 0000	1111 1011 0000 1010
	2b: Rem < 0 ⇒ +Div, sll Q, Q0 = 0	0000	0000 0101 0100 0000	0000 0000 0100 1010
	3: Shift Div right	0000	0000 0010 1010 0000	0000 0000 0100 1010
4	1: Rem = Rem – Div	0000	0000 0010 1010 0000	1111 1101 1010 1010
	2b: Rem < 0 ⇒ +Div, sll Q, Q0 = 0	0000	0000 0010 1010 0000	0000 0000 0100 1010
	3: Shift Div right	0000	0000 0001 0101 0000	0000 0000 0100 1010
5	1: Rem = Rem – Div	0000	0000 0001 0101 0000	1111 1110 1111 1010
	2b: Rem < 0 ⇒ +Div, sll Q, Q0 = 0	0000	0000 0001 0101 0000	0000 0000 0100 1010
	3: Shift Div right	0000	0000 0000 1010 1000	0000 0000 0100 1010
6	1: Rem = Rem – Div	0000	0000 0000 1010 1000	1111 1111 1010 0010
	2b: Rem < 0 ⇒ +Div, sll Q, Q0 = 0	0000	0000 0000 1010 1000	0000 0000 0100 1010
	3: Shift Div right	0000	0000 0000 0101 0100	0000 0000 0100 1010

Iteration	Step	Quotient	Divisor	Remainder
7	1: Rem = Rem – Div	0000	0000 0000 0101 0100	1111 1111 1111 0110
	2b: Rem < 0 ⇒ +Div, sll Q, Q0 = 0	0000	0000 0000 0101 0100	0000 0000 0100 1010
	3: Shift Div right	0000	0000 0000 0010 1010	0000 0000 0100 1010
8	1: Rem = Rem – Div	0000	0000 0000 0010 1010	0000 0000 0010 0000
	2a: Rem ≥ 0 ⇒ sll Q, Q0 = 1	0001	0000 0000 0010 1010	0000 0000 0010 0000
	3: Shift Div right	0001	0000 0000 0001 0101	0000 0000 0010 0000
9	1: Rem = Rem – Div	0001	0000 0000 0001 0101	0000 0000 0000 1011
	2a: Rem ≥ 0 ⇒ sll Q, Q0 = 1	0011	0000 0000 0001 0101	0000 0000 0000 1011
Done		0011	0000 0000 0001 0101	0000 0000 0000 1011

$Quotient : (0011)_2 = (3)_{10}$

$Remainder : (1011)_2 = (11)_{10}$