```
addu $t0, $t1, $t2 # $t0 = sum, but don't trap
xor  $t3, $t1, $t2 # Check if signs differ
slt  $t3, $t3, $zero # $t3 = 1 if signs differ
bne  $t3, $zero, No_overflow # $t1, $t2 signs ≠,
                             # so no overflow
xor $t3, $t0, $t1 # signs =; sign of sum match too?
                  # $t3 negative if sum sign different
slt $t3, $t3, $zero # $t3 = 1 if sum sign different
bne $t3, $zero, Overflow # All 3 signs ≠; goto overflow
```

Chapter3: 3.9 3.10 3.12 3.13

11 7 2020

张佳晨

3.9 [10] <§3.2> 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.

$(151)_{10} = (10010111)_2$, for signed 8-bit, $\Rightarrow (-105)_{10}$

$(214)_{10} = (11010110)_2$, for signed 8-bit, $\Rightarrow (-42)_{10}$

For signed 8-bit, the maximum value is $(01111111)_2 = (127)_{10}$

the minimum value is $(10000000)_2 = (-128)_{10}$

As $(-105)_{10} + (-42)_{10} = (-147)_{10} < (-128)_{10}$

The result is smaller than the minimum value the 8-bit could represent.

As we use saturating arithmetic method,

The answer would be -128, the maximum negative number.

3.10 [10] <§3.2> 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.

$(151)_{10} = (10010111)_2$, for signed 8-bit, $\Rightarrow (-105)_{10}$

$(214)_{10} = (11010110)_2$, for signed 8-bit, $\Rightarrow (-42)_{10}$

For signed 8-bit, the maximum value is $(01111111)_2 = (127)_{10}$

the minimum value is $(10000000)_2 = (-128)_{10}$

As $(-105)_{10} - (-42)_{10} = (-63)_{10}$,

which is between the minimum value and the maximum value

As we use saturating arithmetic, the result would be $(-63)_{10}$

3.12 [20] <§3.3> Using a table similar to that shown in Figure 3.6, calculate the product of the octal unsigned 6-bit integers 62 and 12 using the hardware described in Figure 3.3. You should show the contents of each register on each step.

$(62)_8 = (110010)_2$

$(12)_8 = (001010)_2$

$$\begin{array}{r} 110010 \\ \times\ 001010 \\ \hline 110010 \\ 110010 \\ \hline \end{array}$$

$(111110100)_2 = (500)_{10} = (764)_8$

Iteration	Step	Multiplier	Multiplicand	Product
0	Initial Values	001010	000000 110010	000000 000000
1	1: 0 ⇒ NO Operation			000000 000000
	2: Shift left Multiplicand		000001 100100	
	3. Shift Right Multiplier	000101		
2	1: 1 ⇒ Prod += Mcand			000001 100100
	2: Shift left Multiplicand		000011 001000	
	3. Shift Right Multiplier	000010		
3	1: 0 ⇒ NO Operation			000001 100100
	2: Shift left Multiplicand		000110 010000	
	3. Shift Right Multiplier	000001		
4	1: 1 ⇒ Prod += Mcand			000111 110100
	2: Shift left Multiplicand		001100 100000	
	3. Shift Right Multiplier	000000		
5	1: 0 ⇒ NO Operation			000111 110100
	2: Shift left Multiplicand		011001 000000	
	3. Shift Right Multiplier	000000		
6	1: 0 ⇒ NO Operation			000111 110100
	2: Shift left Multiplicand		110010 000000	
	3. Shift Right Multiplier	000000		

3.13 [20] <§3.3> 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.

Iteration	Step	Multiplicand	Product
0	Initial Values	01100010	0000 0000 0001 0010
1	1. 0 ⇒ NO Operation 2. Shift Right Product	0110 0010	0000 0000 0001 0010 0000 0000 0000 1001
2	1. 1 ⇒ Prod += Multiplicand 2. Shift Right Product	01100010	0110 0010 0000 1001 0011 0001 0000 0100
3	1. 0 ⇒ NO Operation 2. Shift Right Product	0110 0010	0011 0001 0000 0100 0001 1000 1000 0010
4	1. 0 ⇒ NO Operation 2. Shift Right Product	0110 0010	0001 1000 1000 0010 0000 1100 0100 0001
5	1. 1 ⇒ Prod += Multiplicand 2. Shift Right Product	0110 0010	0110 1110 0100 0001 0011 0111 0010 0000
6	1. 0 ⇒ NO Operation 2. Shift Right Product	01100010	0011 0111 0010 0000 0001 1011 1001 0000
7	1. 0 ⇒ NO Operation 2. Shift Right Product	0110 0010	0001 1011 1001 0000 0000 1101 1100 1000
8	1. 0 ⇒ NO Operation 2. Shift Right Product	0110 0010	0000 1101 1100 1000 0000 0110 1110 0100

$(62)_{16} = (0110\ 0010)_2$

$(12)_{16} = (0001\ 0010)_2$

$$\begin{array}{r} 0110\ 0010 \\ \times\ 0001\ 0010 \\ \hline 01100010 \\ 01100010 \\ \hline \end{array}$$

$(0110111100100)_2 = (1764)_{10} = (6E4)_{16}$