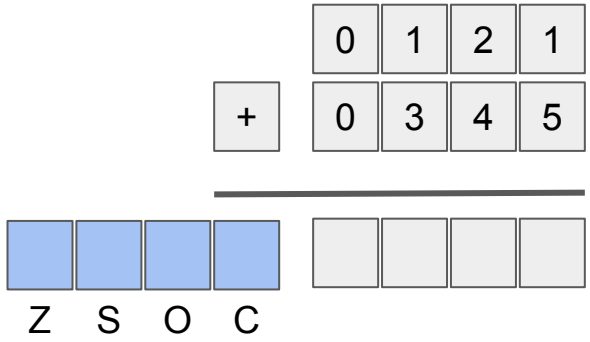
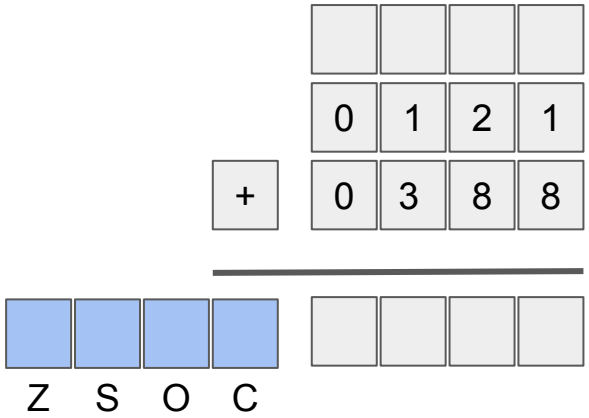
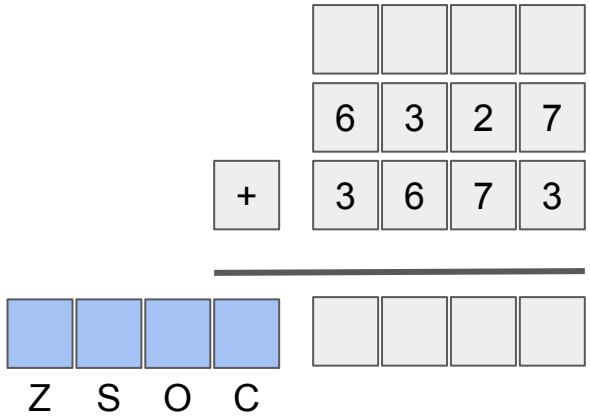
# Review of Mathematical Operations:

- Before
- First, introduce some status values:
  - Zero, Sign, Overflow, Carry
- Assume a register of size 4:

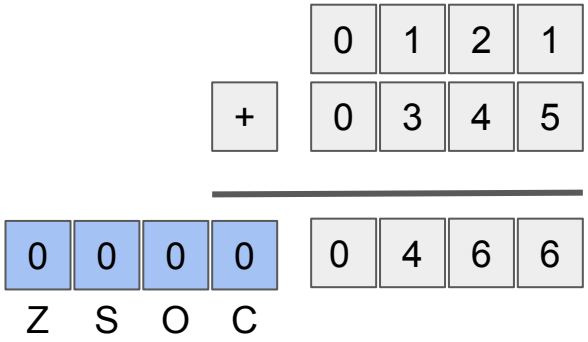


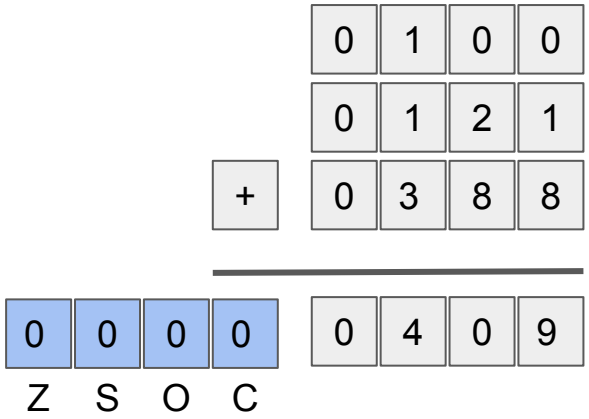



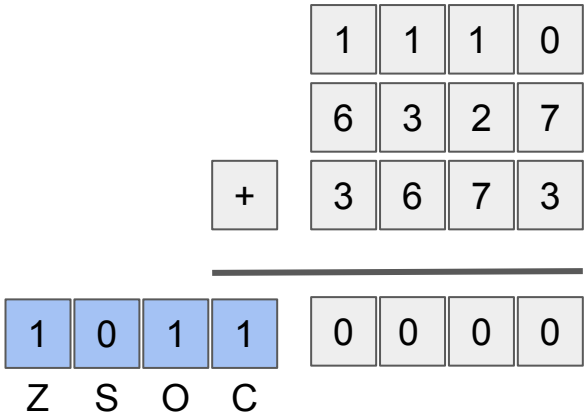



# Review of Mathematical Operations:

- After:
- First, introduce some status values:
  - Zero, Sign, Overflow, Carry
- Assume a register of size 4:



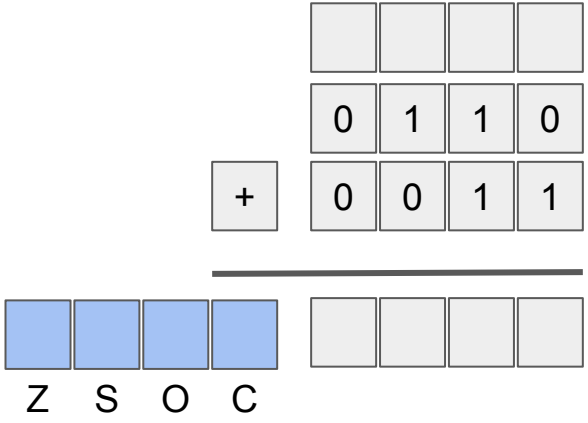
# In Binary (before)

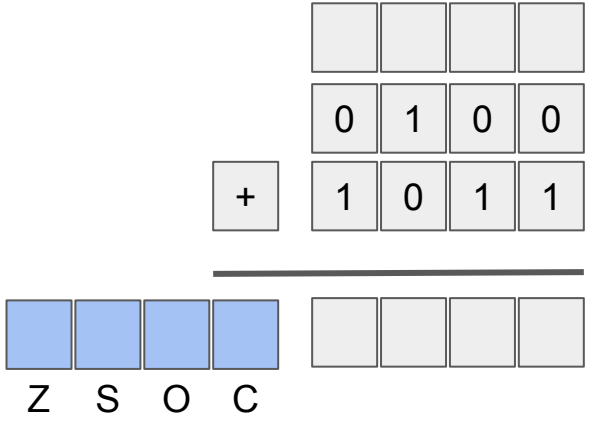
C + A + B = C, S

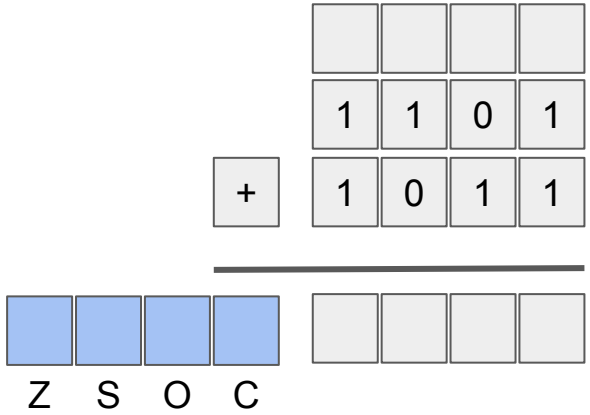
| C | A | B | C | S |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |

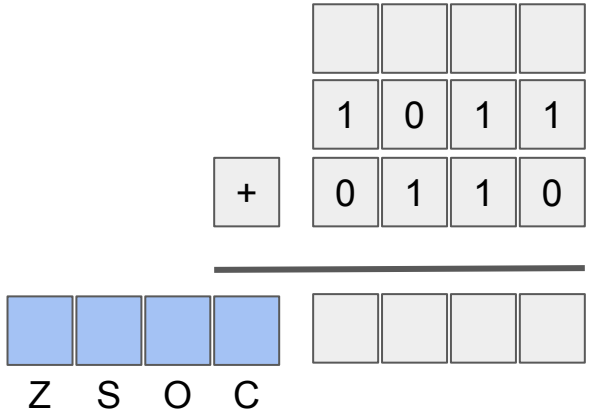
C + A + B = C, S

| C | A | B | C | S |
|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |



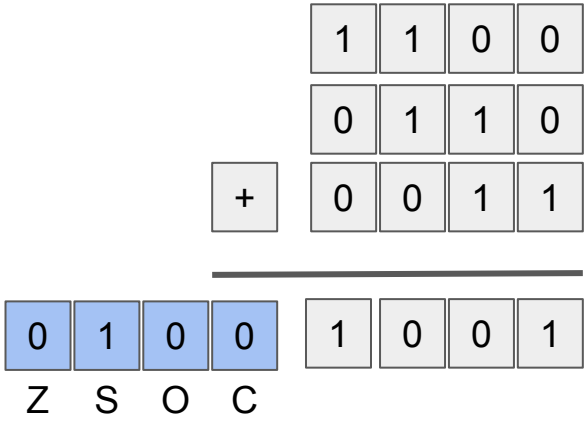
# In Binary (after)

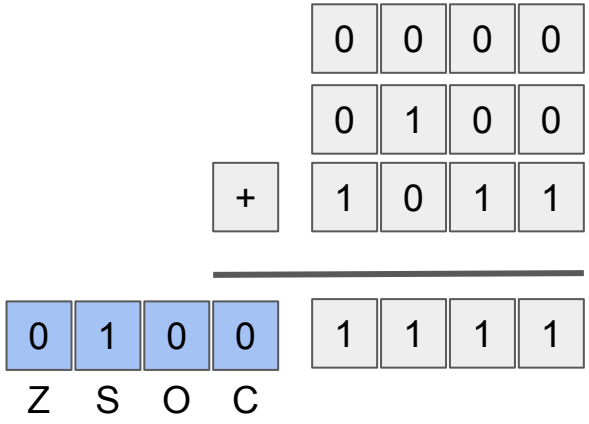
C + A + B = C, S

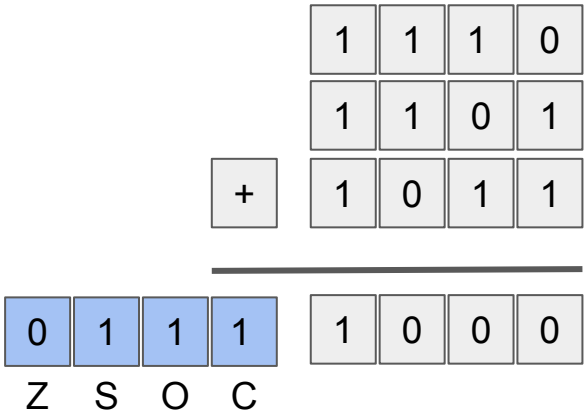
| C | A | B | C | S |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |

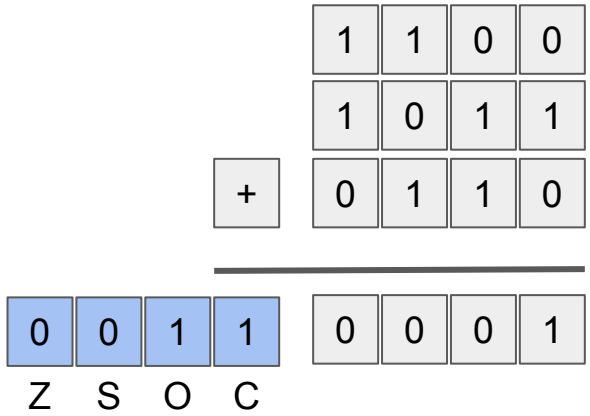
C + A + B = C, S

| C | A | B | C | S |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |



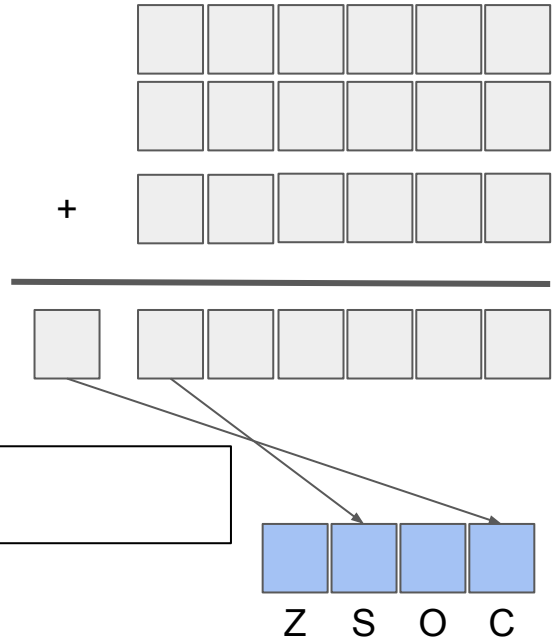



# Practice: Addition and Subtraction (template)

Steps:

1. transform the operation to addition
2. convert abs(X) to binary
3. compute 2's complement for negative numbers
4. perform binary addition
5. convert result to decimal
    - if negative result: compute 2's complement first

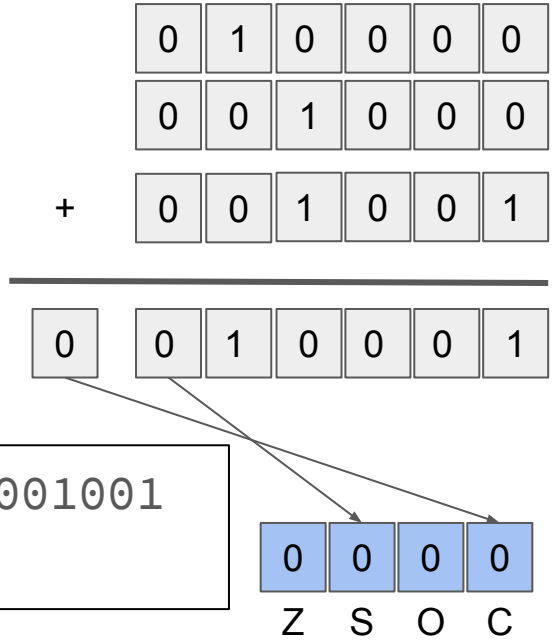



- 8 + 9:
- 7 - 4:
- -5 + 2:
- -16 - 3:

# Practice: Addition and Subtraction

Steps:

1. transform the operation to addition
2. convert abs(X) to binary
3. compute 2's complement for negative numbers
4. perform binary addition
5. convert result to decimal
    - if negative result: compute 2's complement first




➢ 8 + 9:

```
2. abs(8)= 001000, abs(9)=001001
5. 17 : 8+9= 17 (correct)
```

- 7 - 4:
- -5 + 2:
- -16 - 3:

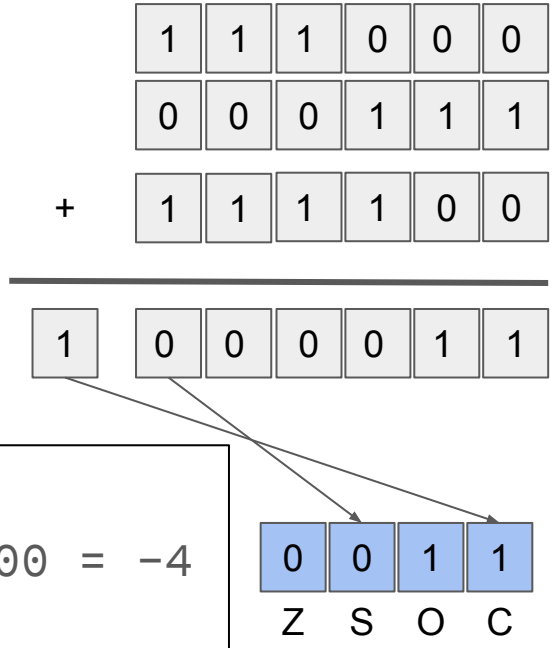
# Practice: Addition and Subtraction

Steps:

1. transform the operation to addition
2. convert abs(X) to binary
3. compute 2's complement for negative numbers
4. perform binary addition
5. convert result to decimal
    - if negative result: compute 2's complement first

- 8 + 9:
- 7 + (- 4):
- -5 + 2:
- -16 - 3:

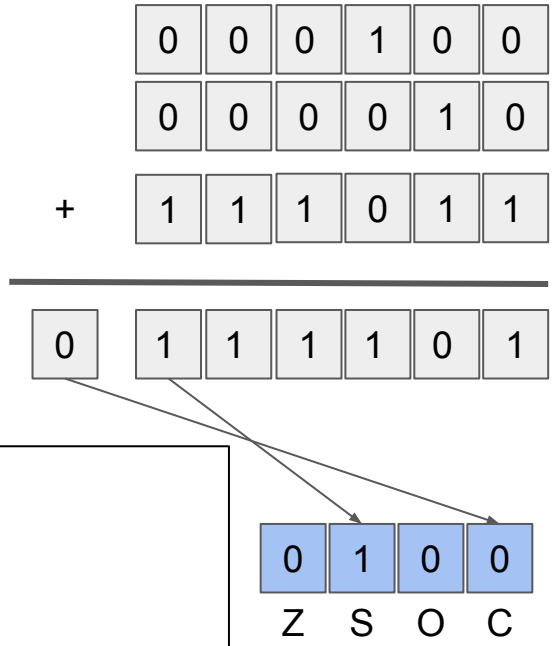
```
2. 000111, 000100
3.          111011+1 = 111100 = -4
5. 3:  7 - 4 = 3 (correct)
```



# Practice: Addition and Subtraction

Steps:

1. transform the operation to addition
2. convert abs(X) to binary
3. compute 2's complement for negative numbers
4. perform binary addition
5. convert result to decimal
    - if negative result: compute 2's complement first



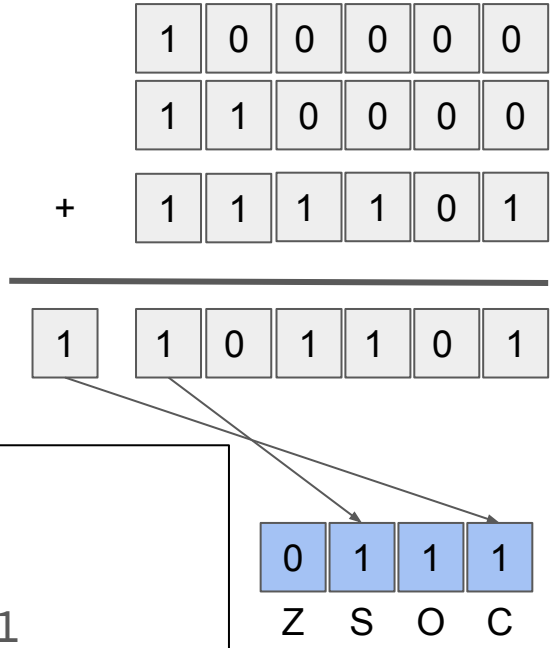

- 8 + 9:
- 7 - 4:
- ➢ (-5 )+ 2:
- -16 - 3:

```
2.  5: 000101, 2: 000010
3. -5: 111010 + 1: 111011
5. 000010 +1 = 000011 = 3
-5 + 2 = -3 (correct)
```

# Practice: Addition and Subtraction

Steps:

1. transform the operation to addition
2. convert abs(X) to binary
3. compute 2's complement for negative numbers
4. perform binary addition
5. convert result to decimal
    - if negative result: compute 2's complement first




- 8 + 9:
- 7 - 4:
- -5 + 2:
- ➢ (-16) + (-3):

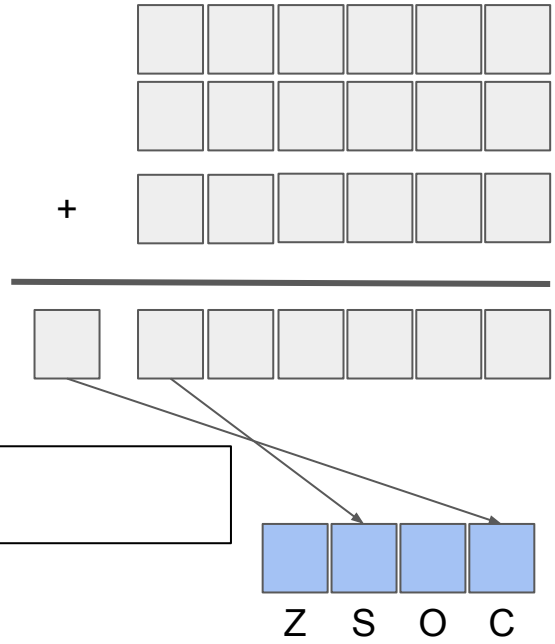
```
2.  16: 010000   3: 000011
3. -16: 110000  -3: 111101
5. -19: abs(010011), 101101
```

# More Practice

# Practice: Addition and Subtraction (template)

Steps:

1. transform the operation to addition
2. convert abs(X) to binary
3. compute 2's complement for negative numbers
4. perform binary addition
5. convert result to decimal
    - if negative result: compute 2's complement first

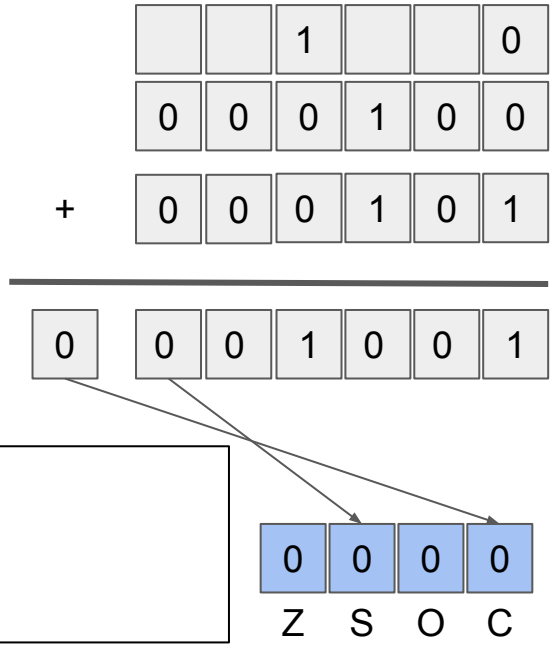



- 4 + 5:
- 7 - 6:
- -3 + 5:
- -18 - 4:

# Practice: Addition and Subtraction

Steps:

1. transform the operation to addition
2. convert abs(X) to binary
3. compute 2's complement for negative numbers
4. perform binary addition
5. convert result to decimal
    - if negative result: compute 2's complement first



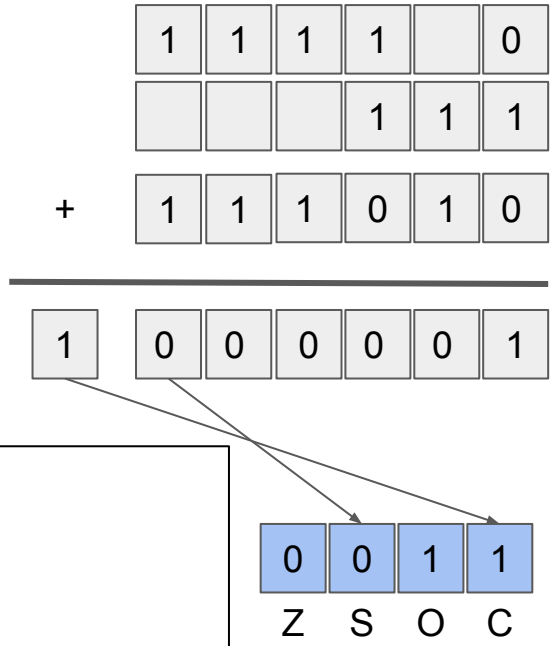

➢ 4 + 5:
- 7 - 6:
- -3 + 5:
- -18 - 4:

```
5. 1001 == 9

4 + 5 = 9 (correct)
```

# Practice: Addition and Subtraction

Steps:

1. transform the operation to addition
2. convert abs(X) to binary
3. compute 2's complement for negative numbers
4. perform binary addition
5. convert result to decimal
    - if negative result: compute 2's complement first




- 4 + 5:
- 7 + (- 6):
- -3 + 5:
- -18 - 4:

```
2. 000110  (6)
3. 111001 + 1 == 111010
5. 1
7 + -6 = 1 == 1 (correct)
```

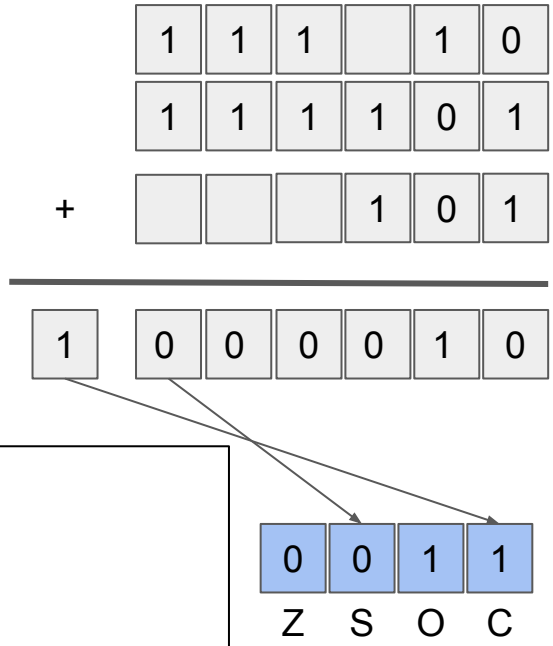
# Practice: Addition and Subtraction

Steps:

1. transform the operation to addition
2. convert abs(X) to binary
3. compute 2's complement for negative numbers
4. perform binary addition
5. convert result to decimal
    - if negative result: compute 2's complement first

- 4 + 5:
- 7 + - 6:
- ➢ (-3) + 5:
- -18 - 4:

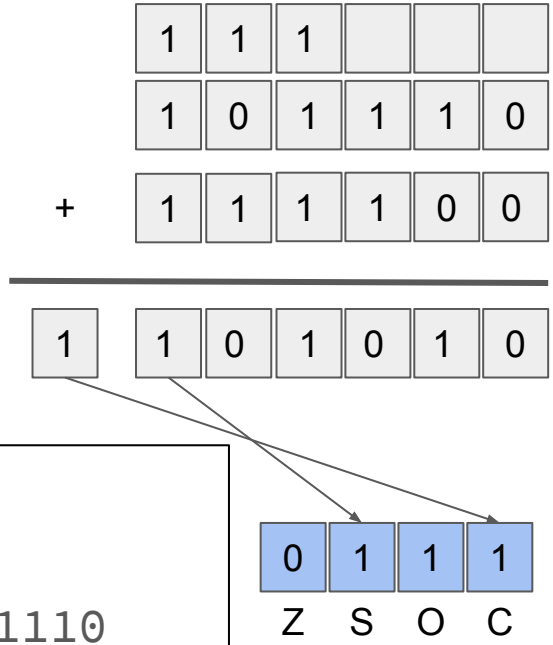
```
2. 000011 abs(3)
3. 111100 + 1 = 111101
5. 2
-3 + 5 == 2 (correct)
```



# Practice: Addition and Subtraction

Steps:

1. transform the operation to addition
2. convert abs(X) to binary
3. compute 2's complement for negative numbers
4. perform binary addition
5. convert result to decimal
    - if negative result: compute 2's complement first

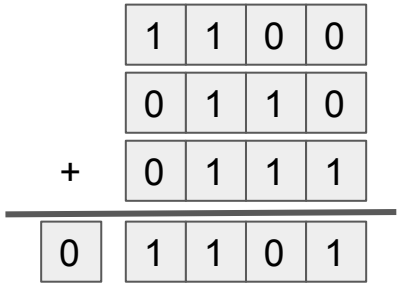



- 4 + 5:
- 7 + - 6:
- -3 + 5:
- ➢ (-18) + (- 4):

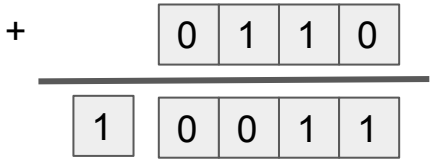
```
2: abs(-18) : 010010
   abs(-4)  : 000100
3. -18 :  101101 + 1 == 101110
    -4 :  111011 + 1 == 111100

5. 101010 -> 010101 + 1 -> 010110
```

# BCD: Addition

- Addition performed on the nibble level: 6+7



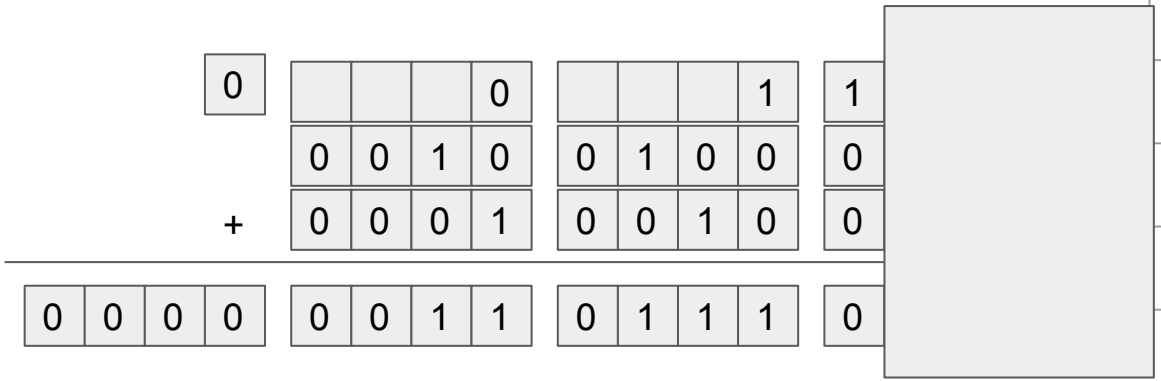

if (overflow or invalid code ) then




| N | Code | N | Code |
|---|---|---|---|
| 0 | 0000 | 8 | 1000 |
| 1 | 0001 | 9 | 1001 |
| 2 | 0010 | | 1010 |
| 3 | 0011 | | 1011 |
| 4 | 0100 | | 1100 |
| 5 | 0101 | x | 1101 |
| 6 | 0110 | | 1110 |
| 7 | 0111 | | 1111 |

# BCD: Binary Coded Decimal

- Another encoding for numbers,
  where precision is required
- Four bits are used to encode each digit
- Perform addition per nibble
- Example: 246 + 127




| N | Code | N | Code |
|---|---|---|---|
| 0 | 0000 | 8 | 1000 |
| 1 | 0001 | 9 | 1001 |
| 2 | 0010 | | 1010 |
| 3 | 0011 | | 1011 |
| 4 | 0100 | | 1100 |
| 5 | 0101 | | 1101 |
| 6 | 0110 | | 1110 |
| 7 | 0111 | | 1111 |

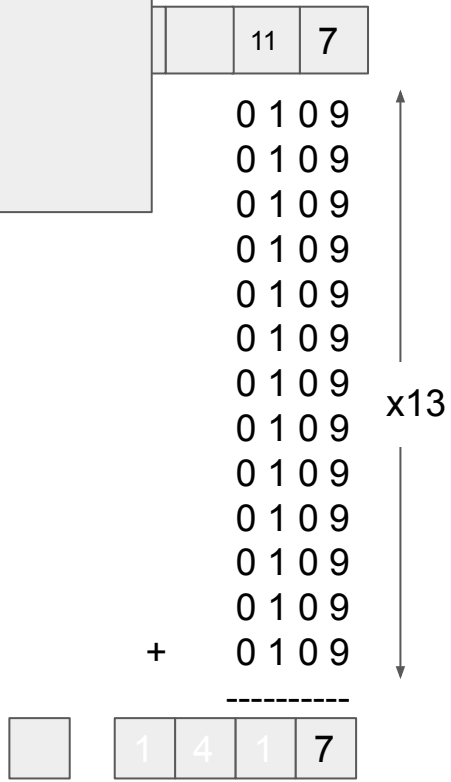
# Multiplication

- Problem: 109 x 13 = 1417
- Approach: Successive Additions
  - Consider: 9 + 9 + 9 .. + 9 (13 times) = ?
  - What is carry value for the 10's column?
- Approach: Long Multiplication

```
    013   (A)
*   109   (B)
-------
    117   (A*9)*1
   0000   (A*0)*10
+ 01300   (A*1)*100
-------
   1417
```

Requires (at worst) 10^N additions

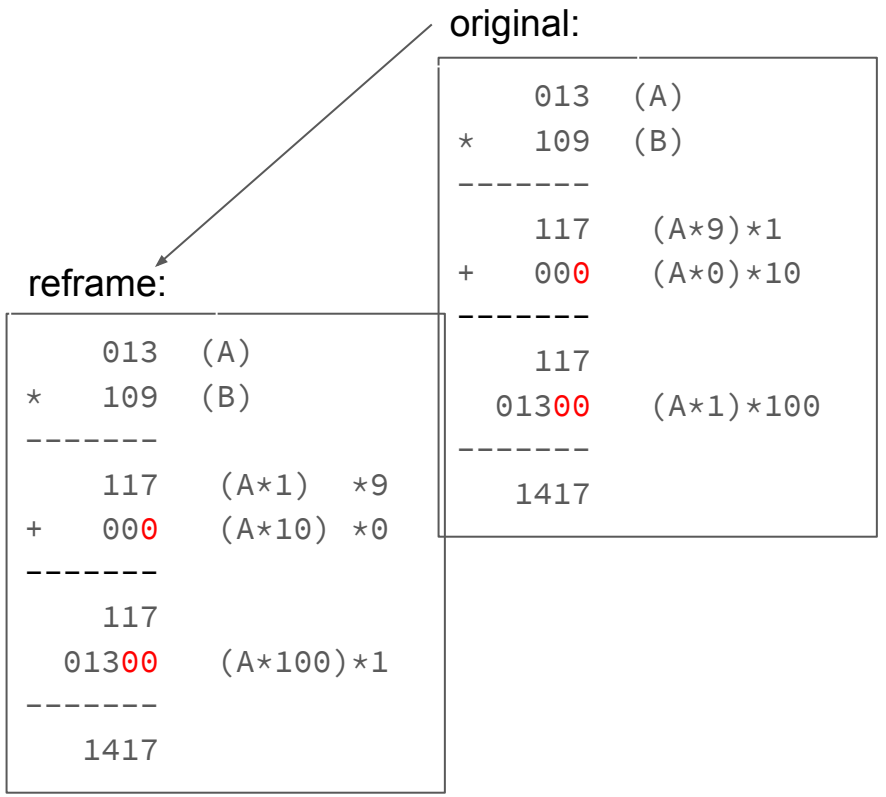
```
    013   (A)
*   109   (B)
-------
    117    (A*9)*1
+  0000    (A*0)*10
-------
    117
  01300    (A*1)*100
-------
   1417
```



# Algorithm for Multiplication

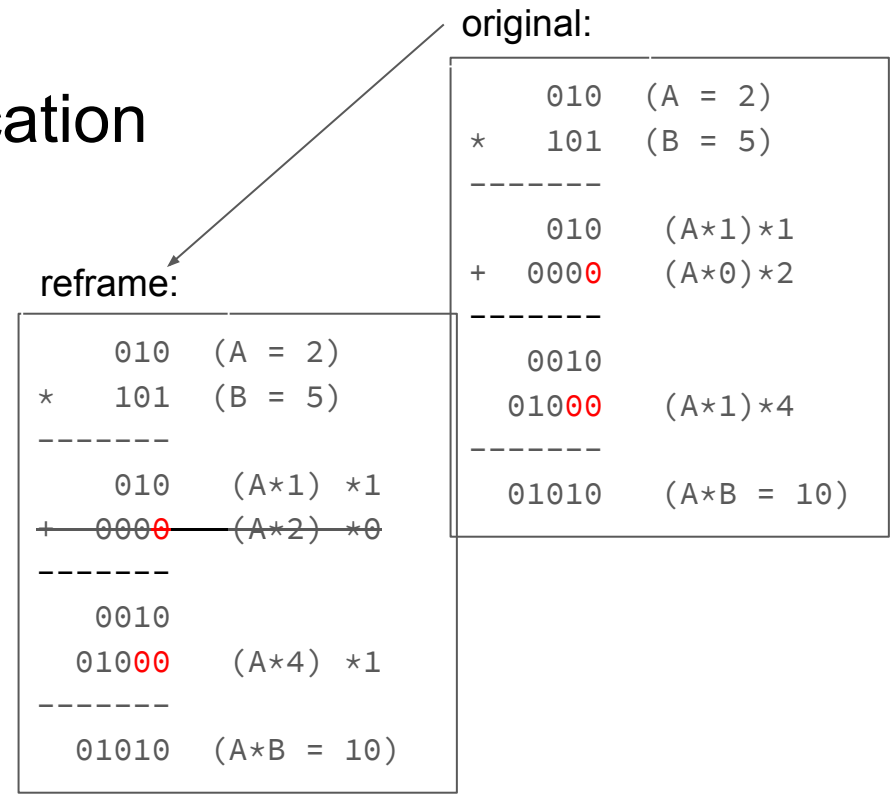
```
sum = 0;
for (d = 0 ; d < 4 ; d ++ ) {
      sum += A * B[d];
     A = A * 10 ;  // Shift to the left
}
```

```
// B[0] = 9
// B[1] = 0
// B[2] = 1
```



# Algorithm for Binary Multiplication

```
sum = 0;
for (d = 0 ; d < 4 ; d ++ ) {
    if (B[d]  == 1) {
        sum += A * B[d];
    }
    A = A * 2 ;  // Shift to the left
    A << 1 ;
}
```




Requires *word_size* additions