Chapter-3
Arithmetic for Computers

Integer Addition: 7+6

```
      1  1  0
   0  0  0  1  1  1
+  0  0  0  1  1  0
-------------------
      1  1  0  1
```

Overflow if result out of range

Add (+ve, -ve) ⟶ No overflow

Add (+ve, +ve) ⟶ overflow if result sign 1

Add (−ve, −ve) ⟶ overflow if result sign 0

$-2^{(n-1)}$ to $+2^{(n-1)}-1$

$\Rightarrow -16$ to 15 } 5 bit

(+11) + (-1) = 10

1 ⟶ 00001
11110
11111

```
+11 ⟶ 01011
 -1 ⟶ 11111
-------------
   ≡ 01010
```

(-11)+(-15) = -26 (Overflow)

```
-11 ⟶ 10101
-15 ⟶ 10001
-------------
  1  00110
  ↑
```

(+11)+(+15) = 26

```
+11 ⟶ 01011
+15 ⟶ 01111
-------------
   ≡ 11010
     4 3 2 1 0
```

$-2^4+2^3+2^1 = -6$

$(+8)-(-4) = +12$

$(-8)-(+4) = -12$

$3-(-15) = 18 \longrightarrow$ overflow

$3+15$

$$\begin{array}{r} 00011 \\ 01111 \\ \hline 10010 \end{array}$$

$-8-(+10) = -18 \longrightarrow$ overflow

$-8-10$

$$\begin{array}{r} 11000 \\ 10110 \\ \hline 01110 \end{array}$$

Overflow

+ve

Dealing with Overflow

addu, addui, subu (C program)

Step 1: If the current instruction has overflow, then the program counter holds the address of that instruction.

Step 2: Then the program counter saves the address of that instruction in exception pc register.

Step 3: Then the program counter jumps to the function, which handles overflow.

This function contains the set of instructions to overcome overflow.

Step 4: Lastly, the program counter retrives EPC address using mfc0 instruction and resume instruction.

Multiplication

Length of product is the sum of operand lengths.

```
    1000   → multiplicand
  X 1001   → multiplier
  ------
    1000
   0000
  0000
 1000
 ---------
 1001000  → Product
```

→ Long Multiplication Approach

→ 4-bit architecture

→ Multiplicand = 8 → 0000 1000

→ Multiplier = 9 → 1001

Iteration	Multiplicand 00001000	Multiplier 1001	Product 00000000
1	00001000 00010000 00010000	1001 1001 0100	00001000 00001000 00001000
2	00100000 00100000	0100 0010	00001000 00001000
3	01000000 01000000	0010 0001	00001000 00001000
4	01000000 10000000 10000000	0001 0001 0000	01001000 01001000 01001000

8 * 9 = 72