Number Representation

A B C ··· K

① → 10 → ②

把 ① 的進制 轉成 10 進制
再用 10 進制 轉成 ② 的進制
算用於 進制間轉換

① **A → 10** (用係數權重表示)

$365_{(10)} = 3\cdot10^2 + 6\cdot10^1 + 5\cdot10^0$

$423_{(8)} = 4\cdot8^2 + 2\cdot8^1 + 3\cdot8^0$

$= 256 + 16 + 3 = 275_{(10)}$

② **10 → K 進位**

$\frac{M_{(10)}}{k} = a_n a_{n-1} \cdots a_2 a_1 | a_0 \ (k)$

$M_{(10)} \equiv a_0 \mod k$

e.g. 10 → 8

$275_{(10)}$ ÷ 8 = 34 ··· 3 ← a_0

34 ÷ 8 = 4 ··· 2 ← a_1

4 ÷ 8 = 0 ··· 4 ← a_2

Number in computer

- floating point number
- integer
  - signed
    - S & M
    - 1's
    - 2's
  - unsigned

① unsigned integer: 8 bits, $2^0+2^1+\cdots+2^7$, $0 \sim 2^8-1$, 若為 n bits : $0 \sim 2^n-1$

② S & M: 8 bits, $-(2^7-1) \sim 2^7-1$ ⇒ 會造成兩個 0 : e.g. 0000 和 1000

最左 bit 來表示 sign，1 為負，0 為正

③ 1's complement : 所有正有多數和無多數相同 ⇒ 負有多數就 1 變 0，0 變 1

④ 2's complement : 所有正有多數和無多數相同 ⇒ 把 1's complement +1

e.g. 0100 : +4 → 1011 + 1 = 1100 : -4

解決兩個 0 : 0000 → 1111 + 1 = 10000 ⇒ 0 為 0000

e.g. 10 10 : -6 ← 01 10 : +6

0101 : +5

1011 : -5

★ 設 4 bit，可表示之數值範圍：

$-2^3 \sim 2^3-1$

8 { 7 ⋮ 0 }

8 { -1 ⋮ -8 }

e.g. addi $s1, $s2, 15 ~ range : $-2^{15} \sim 2^{15}-1$

⇒ $-2^n \sim 2^n-1$

S&M	1's	2's
000 = +0	000 = +0	000 = +0
001 = +1	001 = +1	001 = +1
010 = +2	010 = +2	010 = +2
011 = +3	011 = +3	011 = +3
100 = -0	100 = -3	100 = -4
101 = -1	101 = -2	101 = -3
110 = -2	110 = -1	110 = -2
111 = -3	111 = -0	111 = -1

★ 設為 8 bit 的數 : $(-2^7 \sim 2^7-1)$

給定兩數為 signed decimal integer

e.g. 151, 214

實際上，∵ 151 > 127 且 214 > 127

∴ 皆為負數

⇒ 151 = 151 - 256 = -105

214 = 214 - 256 = -42

(∵ = 補數和 unsign number 差在 MSB 的 sign)

⚠ 十進位 & 二進位之間轉換

10 → 2's

$-20_{(10)} \longrightarrow 11101100_{(2)}$

① 先算正 $20_{(10)} \rightarrow 00010100_{(2)}$

② 再用 2's 改成負的

2	20	
2	10	0
2	5	0
2	2	1
2	1	0
	0	1

2's → 10

$11011100_{(2)} \longrightarrow -36$

① 先取 2's = $00100100_{(2)} = 2^5+2^2 = 36$

② 再取負號：-36

<另>: $-2^7+2^6+2^4+2^3+2^2$

$= -128+64+16+8+4 = -36$ ✓

① $1111\cdots100$ -4

$1111\cdots111$ -1

② 1011 -5

$-8\ \ 3$

1100 -4

$-8\ \ 4$

<另>:

$$\begin{array}{r} 11111_2 \\ \times\ 00000000 \\ -\ 11011100 \\ \hline 00100100 \end{array}$$

$= 2^2+2^5 = 36$

再取負號 → -36

⚠ 範圍檢查路徑

減少檢查 index out of range 的代價，設為：A[x]　$0 \le x < 100$（$a1　$t2）

upper bound

```
sltu $t0 $a1 $t2
beq $t0 $0 Index out of bound
```

⇒ 用 sltu 可以減少紅色 block 的 code

若 $a1 為負 $t2 為正，∵都作無号數 $a1 < $t2 必不成立

```
slt $t0 $a1 $0
bne $t0 $0 Index out of bound
```

⑩ 溢位 (overflow)：判斷要件：兩個正數相加為負數

以 4 bits 為例 (-8 ~ 7)　　兩個負數相加為正數

$$\begin{array}{rr} 0110 & 6 \\ +\ 0101 & 5 \\ \hline 1011 & -5 \end{array} \qquad \begin{array}{rr} 1010 & -6 \\ +\ 1011 & -5 \\ \hline 10101 & 5 \end{array}$$

若使用 signed 來作運算，產生 overflow 會出現 exception

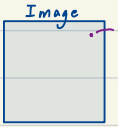
但使用 unsigned 不會，因為 unsigned 是設計用作表示記憶體位址，會用 mod 運算來處理 overflow

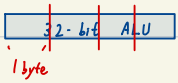
運算	運算元 A	運算元 B	產生溢位的結果
A + B	≥0 正	正 ≥0	負 <0
A + B	<0 負	負 <0	正 ≥0
A − B	≥0 正	負 <0	負 <0
A − B	<0 負	正 ≥0	正 ≥0

多媒体算数

saturating operation：當 overflow 時，設為 maxium (minium)

Image "pixel" 可表示：256×256×256 种颜色 (True color)





⇒ 可以到成4筆的 - 次可處理4個 pixel

(SIMD / Vector) (作相同運算)

└ single instruction multiple data：因為一次的又能做同一指令