

Tutorial 1

TD AFAE

Ex 1

$$x = \sum_{i=0}^{n-1} x_i \beta^i, \quad x_i \in \{0, \dots, \beta-1\}.$$

1) x est entier décimal à 9 chiffres.

Avec 9 chiffres, on peut représenter les nombres de 0 à $10^9 - 1 = 999999999$

En binaire, sur 32 bits, on peut représenter les nombres de 0 à $2^{32} - 1 = 4294967295$

Donc $10^9 - 1 < 2^{32} - 1$

$$x = \sum_{i=0}^8 10^i x_i = ((10x_8 + x_7) \times 10 + x_6) \times 10 + \dots + x_0$$

↳ schéma de Horner

uint32_t convertFromDecimal(uint32_t x[9]) {

 uint32_t res = 0;

 int i;

 for (i = 8; i >= 0; i--) {

 res = res * 10 + x[i];

}

 return res

}

2) Comme $2^{32}-1 < 10^{10}-1$, un entier de 32 bits est représentable en décimal sur 10 chiffres.

```
void convertToDecimal(uint32_t res[10],  
                      uint32_t x) {  
    int i;  
    uint32_t t;  
    t = x;  
    for (i = 0; i < 10; i++) {  
        res[i] = t % 10;  
        t = t / 10;  
    }  
}
```

La division coûte plus chère que la multiplication (environ 10 cycles d'horloge par une division contre 2-4 cycles par une multiplication).

3)

void convertVolumes (unit32_t &gallons,
unit32_t &quarts,
unit32_t &pints,
unit32_t &floz,
unit32_t &rest_ml,
unit32_t &r) {

unit32_t t;

t = r;

*rest_ml = t % 3785411;

t = t / 3785411

*floz = t % 16;

t = t / 16

*pints = t % 2;

t = t / 2;

*quarts = t % 4

t = t / 4

*gallons = t;

}

4) $x = \sum_{i=0}^{n-1} c_i \beta^i$, $c_i \in \{-\alpha, -\alpha+1, \dots, \alpha-1, \alpha\}$.
 avec $\text{card}(\{-\alpha, -\alpha+1, \dots, \alpha-1, \alpha\}) > \beta$:

`uint32_t convertFromRedundantBasis(uint32_t x[],
 uint32_t base,
 int n);`

```

    uint32_t res = 0;
    int i;
    for (i = n-1; i >= 0; i--) {
        res = res * base + x[i];
    }
    return res
}
    
```

Ex 2

1) Par induction sur la taille du nombre
 * cas de base :

snt $a, b \in \{0, \dots, \beta-1\}$ alors

$$atb = BC_{out} + s$$

comme $0 \leq a \leq \beta-1$
 $0 \leq b \leq \beta-1 \Rightarrow 0 \leq atb \leq 2\beta-2$

Par définition $Cout = \lfloor \frac{a+b}{\beta} \rfloor$

Or $0 \leq \frac{a+b}{\beta} \leq 2 - \frac{2}{\beta} < 2$ donc

$$Cout = \lfloor \frac{a+b}{\beta} \rfloor < 2 \Rightarrow 0 \leq Cout \leq 1$$

Il faut aussi montrer que $s \in \{0, -, \beta-1\}$.

$$\frac{a+b}{\beta} - 1 < Cout \leq \frac{a+b}{\beta}$$

$$\Rightarrow \underbrace{a+b - \beta}_{s} < \underbrace{\beta Cout}_{\text{Cout}} \leq \underbrace{a+b}_{s}$$

$$\underbrace{a+b - \beta Cout}_{s} < \beta \quad 0 \leq \underbrace{a+b - \beta Cout}_{s}$$

Donc on en déduit que $0 \leq s < \beta$

d'où $0 \leq s \leq \beta-1$.

* cas général

$$a+b + cin = Cout \beta + s$$

par hypothèse $cin \in \{0, 1\}$

$$0 \leq a+b + cin \leq \beta-1 + \beta-1 + 1 \leq 2\beta-1$$

$$0 \leq \frac{a+b + cin}{\beta} \leq \underbrace{2 - \frac{1}{\beta}}_{< 2} \Rightarrow Cout = \lfloor \frac{a+b + cin}{\beta} \rfloor$$

\downarrow
 $Cout < 2$

$$0 \leq cout \leq 1$$

Pour montrer que $s \in \{0, \dots, \beta-1\}$, même démonstration que par le cas de base.

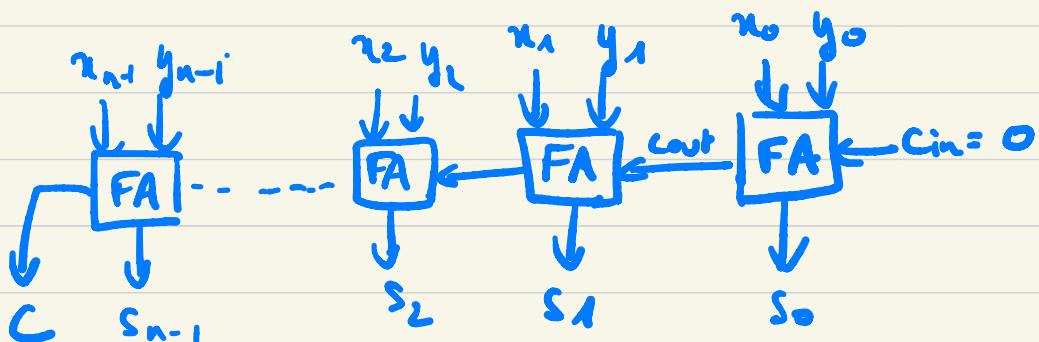
2) void fulladder (uint32_t *s, uint32_t &cout,
uint32_t a, uint32_t b,
uint32_t c_in) {

```

*cout = 0;
*s = a + b;
if (*s < a) (*cout)++;
*s += c_in;
if (*s < c_in) (*cout)++;
}

```

3)



void addition (uint32_t s[], uint32_t *c,
 uint32_t x[], uint32_t y[],
 int n) {

 uint32_t c_in, c_out;

 int i;

 c_in = 0;

 for (i = 0; i < n; i++) {

 fulladder (s[i], x[i], y[i],
 c_in);

 c_in = c_out;

}

*c = c_out;

}

Ex5

1)

1. 0x3FF00000000000000

3 F

exposent biaise Q M M M M M M

exposant non binaire = 0

$$2^{10} - 1$$

$$\text{markissx} = 0$$

Le nombre représent est 1.

2. Le variazioni rappresentative + O

$$3. \quad 8 : 1000 \Rightarrow \text{le nombre repeat est } 0.$$

$$2) 3,125 = 1,1001 \times 2^5$$

$$\text{exponent: } 1 + 1023 = 1024.$$

$$\begin{array}{r} 0 | 1000000 | 0000 | 1001 | 0 + 0 \\ \hline 4 \quad 0 \quad 0 \quad 9 \quad 0 \quad 0 \end{array}$$

xbd : 0x4009000000000000

Tutorial 2

Ex 2

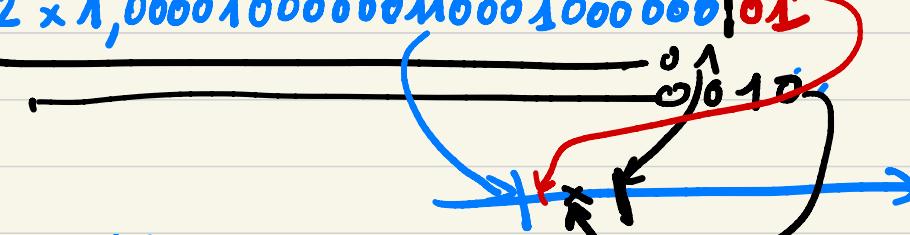
$$2) \quad a = 2^{12} + 1$$

$$b = 2^{13} + 2^3 + 1$$

$$ab = 2^{25} + 2^{20} + 2^{13} + 2^{12} + 2^3 + 1$$

$$= 2^{25} (1,00001000000110001000)$$

$$\quad \quad \quad 0001$$

$$ab = 2^{25} \times 1,00001000000100001000000|01$$


$$a \oplus b = 2^{25} \times 1,00001000000110001000000$$

signe : 0

$$\text{exposant binaire} : 127 + 25 = 152$$

$$0|1001000|00001000000110001000000$$

$$0x4C040C40$$