S34PH412 : Systèmes Microprogrammés

Architecture des Ordinateurs

Cours 2

Université de la Réunion

Année 2015/2016

Alicalapa F.

UNIVERSITÉ DE LA RÉUNION
S'ouvrir aux mondes

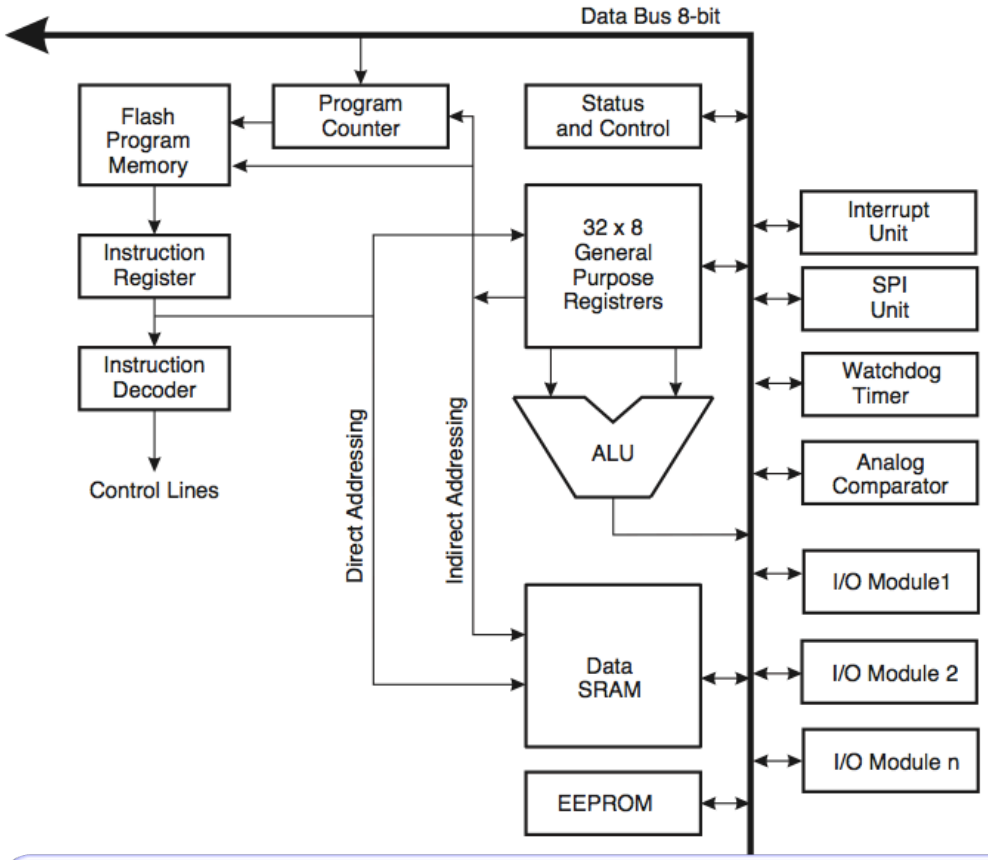
UFR Sciences et Technologies UNIVERSITÉ DE LA RÉUNION

Concepts du cours précédent :

- Dualité Ordinateur/uP en terme d'architecture
- Système embarqué
- Mode de communication série/parallèle : avantages et inconvénients
- Les 3 exigences des systèmes embarqués actuels
- Les avantages du numérique
- Complémentarité analogique/numérique (CAN/CNA)
- La robustesse du signal numérique
- La brique technologique de base / binaire / machine à calculer
- Octet et ses dérivées (Mo, Go, To)
- Binaire, octal, hexadécimal
- Nombres à virgule en binaire
- Nombres signés en binaire

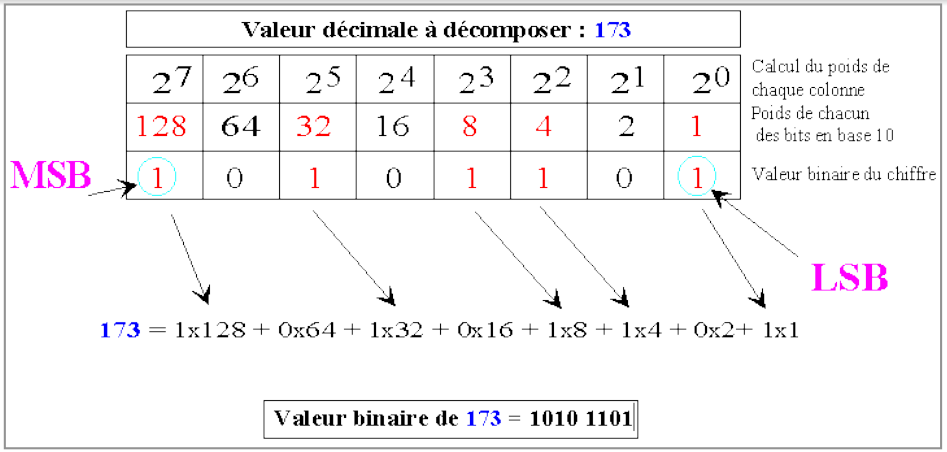
- **Dualité** uC/ordinateur et **uP/uC : video « video_What is a Microcontroller_ » …. Youtube**

- **Dualité** uC/ordinateur et **uP/uC : video « video_What is a Microcontroller_ »**



Qu'est ce qu'un périphérique pour un uC ?

- Binaire naturel



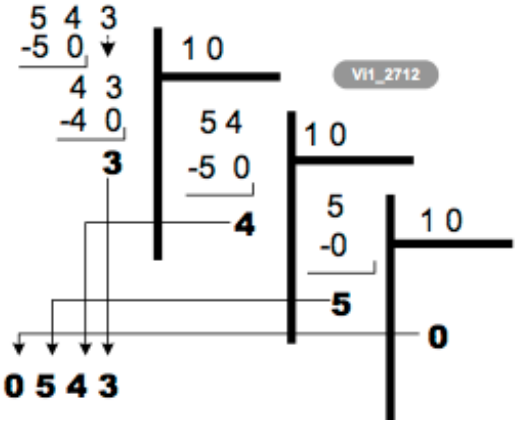
- Exercice 1 : traduire $(10)_{10}$ en base 2 ?
- Exercice 2 : Donner la valeur décimale du nombre 10101, dans le cas où il est codé en base 2
- Exercice 3 : Réaliser la somme en base 2 des nombres décimaux suivants : 25 et 14 : $(25)_{10}=(11001)_2$ et $(14)_{10}=(1110)_2$

- **Fiche TD1 … fiche sur MOODLE**

VII. Systèmes de numération

Pour revoir la base 2 et l'algorithme de conversion associé :

1. Ecrire 173_{10} en binaire (en présentant votre conversion sous forme de divisions successives par 2 « méthode des divisions successives » - voir l'exemple proposé en base 10 ci-dessous).
2. Puis passer du binaire à l'octal et à l'hexadécimal.
3. Rappeler brièvement le principe de passage de la base binaire à octal et à hexadécimal.



- Exercice 4 du cours :
  - Pour revoir la base 2 et l'algorithme de conversion associé :
  1. Écrire 173_{10} en binaire (en présentant votre conversion sous forme de divisions successives par 2 « méthode des divisions successives »).
  2. Puis passer du binaire à l'octal et à l'hexadécimal.
  3. Rappeler brièvement le principe de passage de la base binaire à octal et à hexadécimal.
- Exercice 5 du cours :
  - Donner la valeur décimale du nombre 10101, dans le cas où il est codé en base 2, 8 et 16.

- Exercice 6 du cours : certains de ces nombres sont équivalents : lesquels ? 14792_{10}, 1432_{2}, $39C8_{16}$, 34710_{8},

- Exercice 7 du cours :
  - Réaliser la soustraction :

$$\begin{array}{cccccc} & A & 3 & B & 5 \\ - & 1 & F & 1 & C \end{array}$$

I. Notion de numération binaire (le monde du numérique et de l'analogique)
   a. Vocabulaire de base du numérique : bit, byte, Ko, octet
   b. Domaines numérique et analogique
   c. Numération : utilité ?
   d. Le codage binaire, Binaire naturel
   e. D'autres bases existent – octal et hexadécimal
   f. Binaire réfléchi, code de Gray
   g. Autres codes
   h. Représentation des nombres signés en binaire
   i. nombres flottants ou à virgule en base 2

II. Rapide Historique de l'évolution des ordinateurs

III. Le modèle Von Neuman et représentation hiérarchique

IV. Les différents composants du système et leurs caractéristiques

- Représentation des nombres flottants ou à virgule en base 2

  - 1ère méthode : par **conversion de base**
  - Inspiration = Cas **décimal** :

0.	3	1	2	5
poids	10^{-1}	10^{-2}	10^{-3}	10^{-5}

  - 0.3125*10=3.125 ⇨ 1er digit décimal après la virgule = 3 (remontée des digits)
  - (**enlève le « 3 »**) 0.125*10=1.25 ⇨ 2ème digit décimal après la virgule = 1 etc…….

- Représentation des nombres flottants ou à virgule en base 2

  - 1ère méthode : par **conversion de base**

  - Multiplication successive par 2
  - Exemple :

$(0.3125)_{10}$= binaire ?

0.3125***2**=**0**.625 (premier digit « 0 ») ⇨ $(0.3125)_{10}$= (0,**0**…) – enlève le « 0 »

0.625***2**=**1**.25 (2ème digit « 1 ») ⇨ $(0.3125)_{10}$= (0,0**1**…) – enlève le « 1 »

puis 0.25***2**=**0**.5 puis 0.5*2=**1** Fin

$(0.3125)_{10}$=$(0.0101)_2$ à retenir pour après.

- Représentation des nombres flottants ou à virgule en base 2

  - 1ère méthode : par **conversion de base**
  - Quand s'arrête-t-on ?

0,347 . 2 = 0,694 < 1 je pose 0 : 0,347 = 0,0...
0,694 . 2 = 1,388 > 1 je pose 1 : 0,347 = 0,01...
0,388 . 2 = 0,766 < 1 je pose 0 : 0,347 = 0,010...
0,766 . 2 = 1,552 > 1 je pose 1 : 0,347 = 0,0101...
0,552 . 2 = 1,104 > 1 je pose 1 : 0,347 = 0,01011...
0,104 . 2 = 0,208 < 1 je pose 0 : 0,347 = 0,010110...
0,208 . 2 = 0,416 < 1 je pose 0 : 0,347 = 0,0101100...
0,416 . 2 = 0,832 < 1 je pose 0 : 0,347 = 0,01011000...
0,832 . 2 = 1,664 > 1 je pose 1 : 0,347 = 0,010110001...
0,664 . 2 = 1,328 > 1 je pose 1 : 0,347 = 0,0101100011...

  - S'arrêter quand il y a équivalence de représentation entre bases (voir exercice TD : $2^N=10^Y$)

- Représentation des nombres flottants ou à virgule en base 2

  - 2ème méthode : représentation en **virgule fixe** (qqfois rencontrée)
  - chaque nombre est séparé en deux parties
  - Position virgule est fixe

Ex : sur 16 bits en

15	,	3125_{10}
0000 1111	,	$0101\ 0000_2$

  - Remarque 1 : pour des « petites » partie entière ou décimale, beaucoup de digits qui sont à « 0 » ne sont pas utilisés

- Représentation des nombres flottants ou à virgule en base 2
  - 2ème méthode : représentation en **virgule fixe**
  - Remarque 2 : adapté pour l'addition mais pas pour la multiplication
  - Ex sur 2 digits après la virgule :

$$
\begin{array}{cccccc}
 & & 1 & 1\bullet & 0 & 1_2 \\
+ & & & 0\bullet & 1 & 0_2 \\
\hline
0 & 0 & 1 & 1\bullet & 1 & 1
\end{array}
$$

OK

$$
\begin{array}{cccccc}
 & & & 1 & 1\bullet & 0 \quad 1_2 \\
* & & & & 1\bullet & 1 \quad 1_2 \\
\hline
 & 1 & 1 & 1 & 0 & 1 \\
 & & 1 & 1 & 0 & 1 \quad . \\
 & 1 & 1 & 0 & 1 & . \quad . \\
\hline
10 & 1 & \bullet 1 & 0 & \not{1} & \not{1}
\end{array}
$$

  - Pour multiplication : 2 visions : erreur arrondi (si suppression 2 derniers digits) ou déplacement de la virgule « supposée fixe » si garde 4 digits !

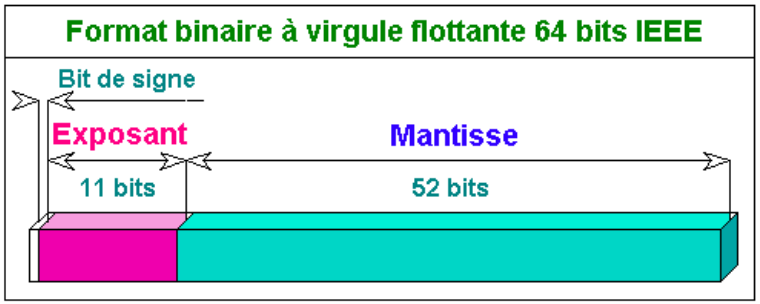
- **3ème méthode** : Représentation en **virgule flottante**

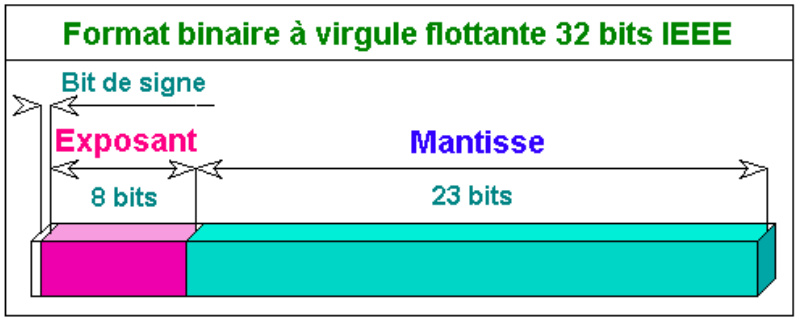
  - Depuis 1960, la représentation en virgule flottante, plus souple, s'est imposée, IEEE 754
  - plus économique en taille mémoire occupée,
  - permet de coder plus grande plage de valeurs
  - les nombres s'écrivent comme en notation scientifique :

    $$N = M*B^E$$

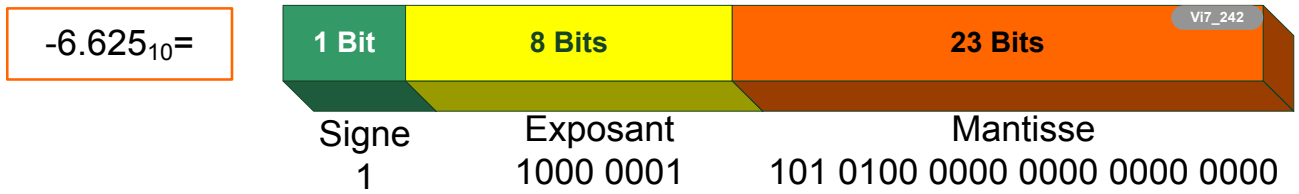
    M : mantisse, E : exposant, B : base

  - Exemple : π(base 10) **0,31415926 * 10^1**
  - Il **existe** flottant **simple précision** et **double précision** (voir différence après) :





- **3ème méthode** : Représentation en virgule flottante
  - Un flottant est stocké en mémoire de la manière suivante (norme **IEEE 754**) :
  - **(SM Eb M)$_2$ soit sur 32 (simple précision) ou 64 bits (double précision)**
  - SM : **signe de la mantisse** : **1 bit**
  - Eb : **exposant biaisé (cf suite)** : sur **8 bits** en **simple précision** et **11** en **double précision**
  - M : **mantisse** : sur **23 bits** en simple précision et **52** en double précision.
  - Comment coder en virgule flottante ?



- 3ème méthode : Représentation en virgule flottante (suite)

  - Définition en **BINAIRE de la mantisse normalisée**
  - toujours de la forme **1.1**….. Une mantisse est dite normalisée si son chiffre le plus à gauche (de poids fort) est un 1 (en codage binaire).

  - Exemple : 1.101 est normalisée
    Contre-exemple : 0.001 n'est pas normalisée

  - Rem : en décimal, la notion d'écriture normalisé n'a pas de sens= 0,3141 (virgule à gauche du 1er digit non nul)…ne pas écrire 3,141 (ce premier digit peut être 5,7,9…)

  - ⇨Avantage : permet d'omettre le 1er bit puisqu'il est tout le temps à « 1 » ; permet ainsi d'augmenter la précision (pas possible décimal)

- 3ème méthode : Représentation en virgule flottante (suite)

- Exemple pour représenter un nombre réel $(-6.625)_{10}$ = $()_2$ en convention virgule flottante simple précision

1. Traduire la valeur absolue du nombre en binaire ⇨ $110{,}1010_2$
2. Créer la mantisse (il faut normaliser) : **1**,101010 x 2^2
3. Omettre le premier « 1 » et étendre à 23 bits en *simple précision* : « , 101010 0000 0000 0000 0000 0 »

$$247\,000 = 2\underbrace{4\,7\,0\,0\,0}_{5\,4\,3\,2\,1} = 2.47 \times 10^5$$

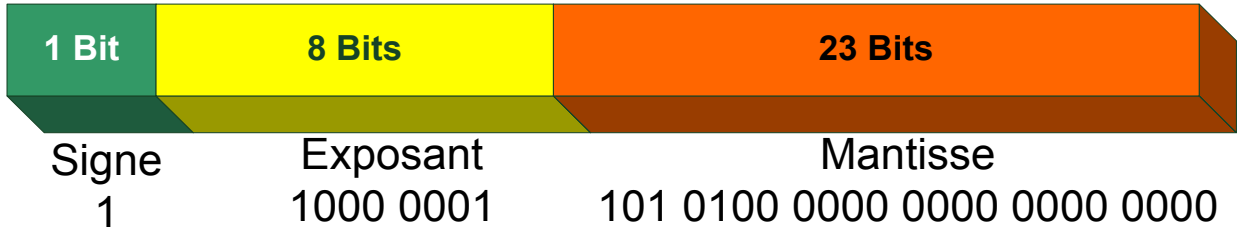
- (suite)
  4. Calculer l'exposant (ici on à 2^2), d'où exposant = 2+décalage (biais associé notation 8 bits) =$(2+\mathbf{127})_{10}=(129)_{10}$

$$(signe)\times \mathbf{2}^{exposant-décalage}\times \mathbf{1},mantisse$$

$$décalage = \mathbf{2}^{n-\mathbf{1}}-\mathbf{1}$$

$$n = \text{le nombre de bits attribués à l'exposant}$$

  5. Traduire exposant en binaire $(1000\ 0001)_2$ sur 8 bits



  6. Digit de signe

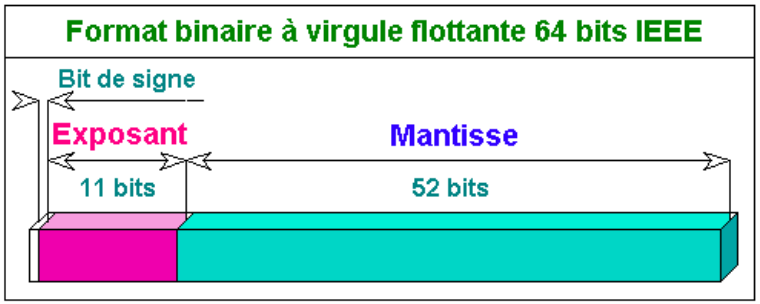
- **3ème méthode** : Représentation en **virgule flottante**

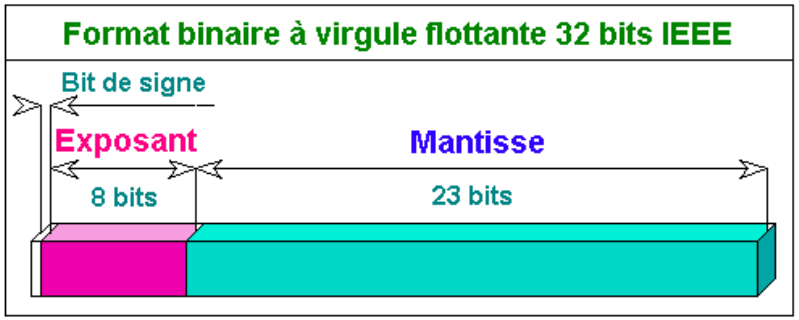
  - Depuis 1960, la représentation en virgule flottante, plus souple, s'est imposée
  - plus économique en taille mémoire occupée,
  - permet de coder plus grande plage de valeurs
  - les nombres s'écrivent comme en notation scientifique :

    $$N = M*B^E$$

    M : mantisse, E : exposant, B : base

  - Exemple : π(base 10) **0,31415926 * 10^1**
  - Il **existe** flottant **simple précision** et **double précision** (voir différence après) :





Interpretation of Exponent in the IEEE Single Format

BIASED EXPONENT	SIGN OF NUMBER	TRUE EXPONENT	SIGNIFICAND	CLASS
0000 0000	+	-	00 ... 00	positive zero
	-	-	00 ... 00	negative zero
			11 ... 11	
			to	
			00 ... 01	denormals
0000 0001	-	-126	00 ... 00	normals
to		to	to	
0111 1111		0	11 ... 11	
1000 0000	-	1	00 ... 00	normals
to		to	to	
1111 1110		127	11 ... 11	
1111 1111	+	-	00 ... 00	+ infinity
	-		00 ... 00	- infinity
	-		10 ... 00	indefinite
	-		00 ... 01	
			to	
			11 ... 11	not-a-number

Microcontroller Programming The Microchip PIC®, Julio Sanchez, Maria P. Canton, CRC Press

- Opérations sur les nombres à virgule flottante
  - Pour l'**addition** et la **soustraction**, il faut que les **exposants** aient la **même valeur, additionne mantisse (décimal)**.
  - Exemple d'addition : $0.3*10^4+0.998*10^6$ = ???
    - 1. Dénormaliser : $0.3*10^4 = 0.003*10^6$.
    - 2. Additionner les mantisses : 0.003+0.998 = 1.001
    - 3. Normaliser : $1.001*10^6 = 0.1001*10^7$.
  - Pour multiplication (ou division) :
    - addition (resp soustraction) des exposants
    - et multiplication (resp division) des mantisses (voir TDs)

Pour la multiplication (resp. la division), on additionne (resp. soustrait) les exposants et on multiplie (resp. divise) les mantisses.
Exemple : $(0.2 \times 10^{-3}) \times (0.3 \times 10^7)$

1. Addition des exposants : $-3+7=4$.
2. Multiplication des mantisses : $0.2 \times 0.3 = 0.06$.
3. Normalisation du résultat : $0.06 \times 10^4 = 0.6 \times 10^3$.