

Arithmétique signée et non signée dans un processeur

Module Info 2

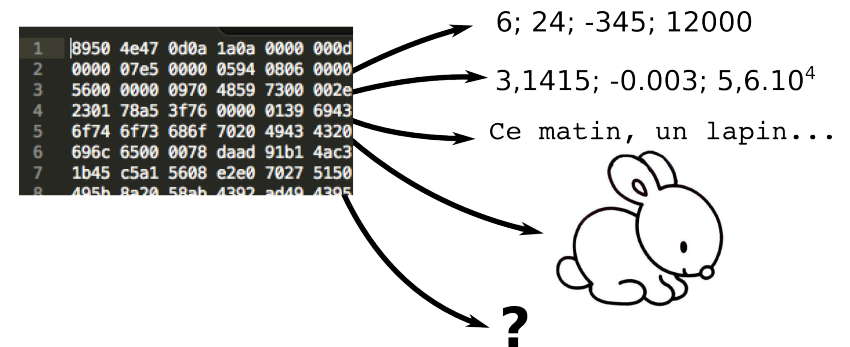
Sebastien.Kramm@univ-rouen.fr

IUT GEII Rouen

2013-2014

Codage de l'information dans un ordinateur

- ▶ Les octets en mémoire peuvent représenter :
 - ▶ un programme ;
 - ▶ ou des données, qui sont **codées** d'une certaine façon. (entiers, flottants, texte, image, son, video, ...)



Codage des entiers

- ▶ Soit n bits représentant une valeur entière
- ▶ En la considérant **non signée**, on a 2^n valeurs, numérotées de 0 à $2^n - 1$
(exemple : sur 8 bits, 0 à 255)
- ▶ Si on veut représenter des valeurs signées, on réserve un bit pour le signe, on a donc 2^{n-1} valeurs possibles.
(exemple : sur 7 bits, 128 valeurs possibles)

Par convention

Le bit de signe est **toujours** le bit de poids fort.

- ▶ 0 : nombre positif
- ▶ 1 : nombre négatif

Remarque : la valeur 0 est considérée comme positive.

Représentation nombres signés : complément à 2

- ▶ Si positif, codage binaire "classique"
Exemple : $0x27 = 0010\ 0111 = 2 \times 16 + 7 = 39$
- ▶ Si négatif, codage "complément à 2"
 1. Complémentation de tous les bits
 2. Addition de 1 à la valeur obtenue

Exemples

- ▶ Codage de la valeur -14 sur 8 bits : $14_{10} = 8 + 4 + 2 = (0000.1110)_2$
 1. Complément à 1 : $0000.1110 = 1111.0001$
 2. Ajout de 1 : $1111.0001 + 1 = 1111.0010 = 0xF2$
- ▶ Codage de la valeur -14 sur 16 bits : $14_{10} = (0000.0000.0000.1110)_2$
 1. Complément à 1 : $1111.1111.1111.0001$
 2. Ajout de 1 : $1111.1111.1111.0010 = 0xFFFF$

Pourquoi le complément à 2 ?

- ▶ Permet d'utiliser la même ALU pour le calcul en signé et en non-signé :

$$\text{\$B6} + \text{\$03} = \text{\$B9}$$

$$\text{Signé : } -74 + (+3) = -71$$

$$\text{Non signé: } 182 + 3 = 185$$

- ▶ Transforme les soustractions en additions

$$5 - 2 = 5 + (-2) = \text{\$05} + \text{\$FE} = \text{\$03}$$

Remarque : la retenue obtenue est ici **ignorée**.

Exemple de représentation

- ▶ L'octet 0x41 peut représenter :

- ▶ La valeur non signée 65 ($4 \times 16 + 1$)
- ▶ La valeur signée +65
- ▶ Le code ASCII de 'A'

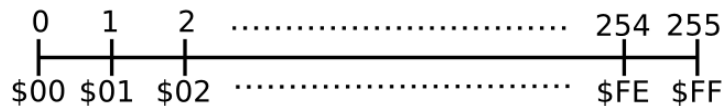
- ▶ L'octet 0xE9 peut représenter :

- ▶ La valeur non signée 233 ($15 \times 16 + 9$)
- ▶ La valeur signée -23

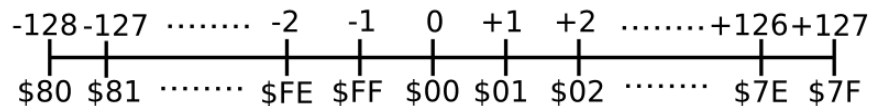
$$\text{E9} = -(\overline{1110.1001} + 1) = -(0001.0110 + 1) = -(0001.0111) = -(16 + 4 + 2 + 1)$$

Etendue numérique

- ▶ en non-signé sur 8 bits :



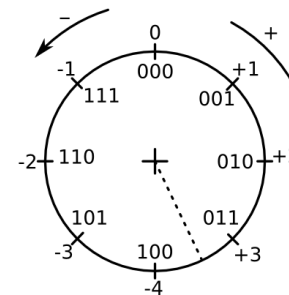
- ▶ en signé sur 8 bits :



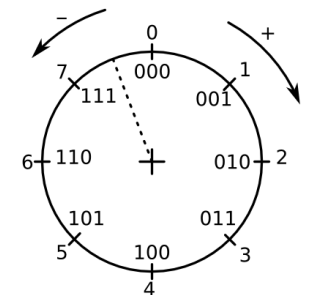
Représentation circulaire

- ▶ On peut montrer les deux représentations sous la forme d'un cercle, par exemple, pour 3 bits :

- ▶ en signé :



- ▶ en non-signé :



- ▶ On aura un débordement lorsqu'une opération va **franchir** le trait pointillé.

bits NZVC (registre CC)

- ▶ Ces bits sont positionnés :
 - ▶ par l'ALU en cas de calcul,
 - ▶ par certaines instructions de chargement de registre (exemple : `ldaa #0` \Rightarrow $Z=1$, $N=0$)
- ▶ N : Negative \Rightarrow recopie du bits de poids fort (signe)
- ▶ Z : Zero \Rightarrow =1 si valeur nulle
- ▶ V : oVerflow \Rightarrow Débordement en "Signé"
- ▶ C : Carry \Rightarrow Débordement en "Non Signé"

Exemple de calculs

- ▶ opération : $7E+1=7F$
 - ▶ En "non signé" : $126+1=127$: pas de débordement $\Rightarrow C=0$
 - ▶ En "signé" : $+126+(+1)=+127$ $\Rightarrow V=0$
- ▶ opération : $7E+2=80$
 - ▶ En "non signé" : $126+2=128$: pas de débordement $\Rightarrow C=0$
 - ▶ En "signé" : $+126+(+2)=-128$: débordement ! $\Rightarrow V=1$
- ▶ opération : $FF+1=00$
 - ▶ En "non signé" : $255+1=0$: débordement ! $\Rightarrow C=1$
 - ▶ En "signé" : $-1+(+1)=0$: pas de débordement $\Rightarrow V=0$
- ▶ opération : $80+81=01$
 - ▶ En "non signé" : $128+129=1$: débordement ! $\Rightarrow C=1$
 - ▶ En "signé" : $-128+(-127)=+1$: débordement ! $\Rightarrow V=1$

Exercices

- ▶ Réaliser les additions suivantes sur 8 bits, et en déduire l'état de NZVC après l'opération.

| Opération | Résultat | N | Z | V | C |
|---------------|----------|---|---|---|---|
| $\$40 + \30 | | | | | |
| $\$40 + \60 | | | | | |
| $\$40 + \$C0$ | | | | | |
| $\$40 + \$F0$ | | | | | |

Implémentation de tests en assembleur

- ▶ Les test sont fait via une instruction de comparaison (soustraction), suivie d'une instruction de branchement conditionnel.
- ▶ Exemple 1 : test de l'égalité :

- ▶ En C :

```

1  if( var1 != 2 )
2      MaFonction();
3  LaSuite();
    
```

- ▶ En assembleur (*Freescale*)

```

1  BOUCLE  ldaa  var1
2          cmpa  #2
3          b     etape2
4          bsr  MaFonction
5  etape2  bsr  LaSuite
    
```

- ▶ Exemple 2 : test du signe :

- ▶ En C :

```

1  if( var1 >= 0 )
2      MaFonction();
3  LaSuite();
    
```

- ▶ En assembleur (*Freescale*)

```

1  BOUCLE  ldaa  var1
2          b     etape2
3          bsr  MaFonction
4  etape2  bsr  LaSuite
    
```