Micro-contrôleur

I-Codage des Informations

JM ENJALBERT
enjalber@laas.fr

Université Paul Sabatier - Toulouse III

Septembre 2012

Plan

Introduction

- Ordinateur : Machine de traitement de données
- Fonctionne sur le principe de l'algèbre de Boole (logique binaire)
- Données à traiter de différentes natures : nombres, caractères, images, sons, etc...
- Necessité de coder les données pour pouvoir les traiter
- Programme : manipulation de données à l'aide d'instructions.
- Ecrit sous forme symbolique dans un langage algorithmique.
- Compilateur : effectue le codage pour le processeur en transformant les instructions et les données en suites de bits.
- Code : suite de bits qui représente une information
- Ne pas confondre la suite de bits (code) et l'information qu'elle représente

Principe d'un codage discret

- Soit $X = \{x_1, x_2, \dots x_p\}$ un ensemble fini de p informations à coder.
- Soit $Y = \{y_1, y_2, \dots y_q\}$ avec $q \ll p$ un ensemble de symboles.
- Un codage consiste à représenter un élément de X par une concaténation d'éléments de Y.
- A deux éléments distincts de X doivent correspondre deux mots-codes différents.
- En général on utilise un code de longueur fixe. Pour q symboles et un code de longueur l on pourra coder q^l informations.
  Par exemple pour $q = 10$ (chiffres décimaux) et $l = 4$ on peut coder $10^4 = 10000$ nombres.

Notation Hexadécimale

- Dans un ordinateur $Y = \{0, 1\}$,
  Un élément de X est codé par une suite de bits (mot-code).
- Un *octet* (suite de 8 bits) est l'objet de taille minimale accessible dans une mémoire.
- Codages des données et instructions : n multiple de 8
- Notation hexadécimale : On utilise la suite des chiffres, complétée par les premières lettres de l'alphabet : $\{0, 1, \ldots, 9, A, B, C, D, E, F\}$.
- Conversion binaire $\leftrightarrow$ hexadécimal
  Passage de la base 2 à la base 2^4 : regrouper les bits par paquets de 4 et coder chaque paquet par le symbole hexadécimal correspondant.

Relation Binaire/Hexadécimal

base 2	0000	0001	0010	0011	0100	0101	0110	0111
base 16	0	1	2	3	4	5	6	7
base 2	1000	1001	1010	1011	1100	1101	1110	1111
base 16	8	9	A	B	C	D	E	F

Exemples :

- le code binaire `1010 0101` s'écrit `A5` en hexadécimal.
- Le code hexadécimal `C2` s'écrit `1100 0010` en binaire.

On utilise différentes notations pour représenter un nombre hexadécimal :

- XXXXh (h pour hexadécimal)
- $XXXX (généralement utilisé en assembleur)
- 0xXXXX (en langage C)

Codage des caractères

- Objectif : Fournir des informations à un ordinateur : clavier composé par une centaine de touches.
- Avec 7 bits, on peut distinguer $2^7 = 128$ caractères différents.
- Normalisation du codage : code ASCII

Code ASCII

American Standard Code for Information Interchange

	bits 654							
bits 3210	000	001	010	011	100	101	110	111
0000	NUL	DLE		0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

Exemple : Le code 011 1001 est celui du caractère 9

Propriétés du code ASCII

Codage à 7 moments standard dans le monde occidental.
Codage choisi pour faciliter les opérations sur les caractères.
Soit $ord(x)$ la position du caractère x dans le code.

- $ord(B) = ord(A) + 1$
  tri par ordre alphabétique, ramené à un tri arithmétique
- $ord(A) = ord(a) - 2^5$
  facilite la transformation majuscule ↔ minuscule
- $i = ord(i) - ord(0) \quad \forall i \in [0, 9]$
  facilite le passage du code ASCII d'un chiffre a son code binaire

Les codes 00 à 1F sont utilisés pour le contrôle et la communication

- contrôle des consoles et imprimantes (ex : saut de ligne, retour en début de ligne, tabulation, ...)
- communication (ex : début de message, fin de transmission, ...)

Caractères de contrôle

NUL	nul	VT	tabulation verticale
SYN	synchronisation	SOH	début d'en-tête
FF	présentation de formule	ETB	fin de bloc de transmission
STX	début de texte	CR	retour début de ligne
CAN	annulation	ETX	fin de texte
SO	hors code	EM	fin de support
EOT	fin de communication	SI	en code
SUB	substitution	ENQ	demande
DLE	échappement transmission	ESC	échappement
ACK	accusé de réception	DC1	commande auxiliaire 1
FS	séparateur de fichier	BEL	sonnerie
DC2	commande auxiliaire 2	GS	séparateur de groupe
BS	retour en arrière	DC3	commande auxiliaire 3
RS	séparateur d'article	HT	tabulation horizontale
DC4	commande auxiliaire 4	US	séparateur d'unité
LF	interligne	NAK	acquittement négatif

8ième bit

Objet de taille minimale accessible en mémoire : un octet
code ASCII défini sur 7 bits,
trois solutions pour le bit de poids fort :

- le plus simple : le fixer (à 0 ou à 1)
- les claviers de PC définissent deux jeux de caractères
  - les codes 00 à 7F (le bit 7 vaut 0) constituent le code ASCII standard
  - les codes 80 à FF (le bit 7 vaut 1) sont des caractères étendus, définis par IBM.
- le bit de poids fort peut être utilisé pour faire du contrôle de parité, ce qui permet la détection de certaines erreurs de transmission.

Extensions du code ASCII

- latin-1 (ISO 8859-1) : codage sur 8 bits : les codes ASCII plus caractères accentués et particuliers (191 caractères)
- Unicode : affecte à chaque caractère un nom et un numéro unique (plus de 200 000 caractères décrits)
- UTF-8 : codage sur 1 à 4 octets d'un caractère Unicode. Compatible avec les codes ASCII

Représentation binaire UTF-8	Signification
0xxxxxxx	1 octet pour 1 à 7 bits
110xxxxx 10xxxxxx	2 octets pour 8 à 11 bits
1110xxxx 10xxxxxx 10xxxxxx	3 octets pour 12 à 16 bits
11110xxx 10xxxxxx 10xxxxxx 10xxxxxx	4 octets pour 17 à 21 bits

- UTF-16, UTF-32, ...

Codage des booléens

Booléen : variable qui peut prendre 2 valeurs : *vrai* ou *faux*.
Il suffirait d'un seul bit pour le coder, mais on utilise un octet pour son codage.

- *faux* est généralement codé par 00
- *vrai* est codé par une autre valeur (`FF`, `01`, ...)

Le choix du codage est effectué par le compilateur.

Notation positionnelle

- Nombre : notion abstraite, valeur représentée par une technique de codage.
- Un nombre est représenté par une concaténation de symboles (mot-code)
- Code choisi pour que les opérations sur les nombres soient faciles
- La valeur (poids) de chaque symbole dépend de sa position dans le mot-code
- En décimal (base 10) : 10 symboles différents (chiffres 0 à 9). Poids égal à la valeur du symbole multipliée par une puissance de 10 dépendant de sa position (Poids croissants de la droite vers la gauche)
- En base B : B symboles et les poids sont des puissances de B

Binaire naturel

Notons $\mathcal{N}(A)$ l'entier naturel de code A. Le mot-code $A = a_{n-1}a_{n-2}\dots a_0$, composé d'une séquence de n bits, représente l'entier naturel :

$$\mathcal{N}(A) = \sum_{i=0}^{n-1} a_i\, 2^i$$

Exemples :

- Le code binaire sur 8 bits A=0010 0000 représente l'entier naturel $\mathcal{N}(A) = 32$
- Le code de l'entier naturel $\mathcal{N}(A) = 1$ sur 8 bits s'écrit : A=0000 0001

Codage d'un entier en binaire naturel

- Pour déterminer les bits a_i du code, il faut décomposer en puissance de 2.
- On peut faire cela par divisions successives par 2
- Exemple :

13/2	=	6	reste 1	a_0=1
6/2	=	3	reste 0	a_1=0
3/2	=	1	reste 1	a_2=1
1/2	=	0	reste 1	a_3=1

  D'ou le code de 13 sur 4 bits : 1101 ou sur 8 bits : 0000 1101

Gamme représentable

Avec n bits, la gamme des nombres représentables est $[0, 2^n - 1]$.
Les entiers sont codés sur 1,2,4 ou 8 octets selon les besoins.

n	gamme représentable
8	[0 ... 255]
16	[0 ... 65535]
32	[0 ... 4295967295] = [0 ... $\simeq 4,30\,10^9$]
64	[0 ... $\simeq 1,85\,10^{19}$]

En langage C, le choix est fait par les déclarations suivantes :

```
unsigned char               ⇒ n = 8
short unsigned int          ⇒ n = 16
long unsigned int           ⇒ n = 32
longlong unsigned int       ⇒ n = 64
```


Entiers relatifs

On doit coder le signe (+ ou -) en plus de la valeur absolue
Trois techniques principales :

- Codage en signe et valeur absolue
- Codage en complément à 2
- Codage par excédent

Codage en signe et valeur absolue

Principe :

- On code le signe dans le bit le plus à gauche du mot-code et la valeur absolue dans les $n-1$ bits restants.
- Par convention le bit de signe vaut 0 pour un nombre positif (1 pour un nombre négatif)
- La valeur absolue est codée en binaire naturel sur $n-1$ bits.
- La gamme des nombres représentables est $[-2^{n-1}+1, 2^{n-1}-1]$.

Inconvénients :

- 2 codes pour le nombre zéro
- nécessite de tester le signe avant d'additioner ou de soustraire 2 nombres

Valeur codée

Soit le mot-code $A = a_{n-1}a_{n-2}\ldots a_0$. Il représente le nombre entier relatif :

$$\mathcal{S}(A) = (-1)^{a_{n-1}} \times \sum_{i=0}^{n-2} a_i\, 2^i$$

Exemples :

- Le code 1000 0001 sur 8 bits représente l'entier relatif : -1
- L'entier relatif 2 se code sur 8 bits : 0000 0010

Complément à 2

- Soit un entier x codé sur n bits, le complément à 2 de x est l'entier $\widetilde{x}$ tel que $x + \widetilde{x} = 2^n$
- Soit $A = a_{n-1}a_{n-2}\dots a_1 a_0$
  le complément à 1 de A s'écrit : $\overline{A} = \overline{a}_{n-1}\overline{a}_{n-2}\dots\overline{a}_1\overline{a}_0$
- D'ou : $\mathcal{N}(A) + \mathcal{N}(\overline{A}) = 2^n - 1$
- Donc $\widetilde{\mathcal{N}}(A) = \overline{\mathcal{N}}(A) + 1$
- Soit encore $\widetilde{A} = \overline{A} + 1$
- exemple sur 4 bits : $A = 1010$ $\overline{A} = 0101$ $\widetilde{A} = 0110$

Codage par la technique du complément à 2

Si on code les entiers relatifs sur n bits, il faut pouvoir représenter autant de nombres négatifs que de positifs.

- Une solution consiste à accorder un poids négatif au bit $n-1$.
- Notons $\mathcal{Z}(A)$ le nombre relatif de mot-code $A = a_{n-1}a_{n-2}\dots a_0$.
- Le code A représente l'entier relatif :

$$\mathcal{Z}(A) = -a_{n-1}\,2^{n-1} + \sum_{i=0}^{n-2} a_i\,2^i$$

Exemples :
Le code 1000 0001 sur 8 bits représente l'entier relatif : -127
L'entier relatif 2 se code sur 8 bits : 0000 0010

Propriétés

$$\begin{aligned}\mathcal{Z}(A) &= -2^{n-1}a_{n-1} + \sum_{i=0}^{n-2} a_i\,2^i \\ &= -2^n a_{n-1} + \sum_{i=0}^{n-1} a_i\,2^i \\ &= -2^n a_{n-1} + \mathcal{N}(A)\end{aligned}$$

l'examen du bit a_{n-1} permet de connaître le signe du nombre :

- si $a_{n-1} = 0$, alors $\mathcal{Z}(A) = \mathcal{N}(A)$. Le mot-code A représente un nombre positif qui se décode comme un nombre binaire naturel
- si $a_{n-1} = 1$, alors $\mathcal{N}(A) - \mathcal{Z}(A) = 2^n$ soit : $\mathcal{N}(A) + |\mathcal{Z}(A)| = 2^n$
  le code A du nombre (négatif) $\mathcal{Z}(A)$ est le code binaire naturel du complément à 2^n de son module,

$$\widetilde{\mathcal{N}}(A) = \mathcal{N}(\widetilde{A}) = |\mathcal{Z}(A)|$$

Technique de codage

Pour coder un entier relatif sur n bits en complément à 2, l'opération dépend donc du signe :

- si le nombre est positif ou nul, prendre son code binaire naturel sur n bits
- si le nombre est négatif, trouver le code binaire naturel de son module sur n bits puis prendre le complément à 2 de ce code.
  Exemple. Coder -3 sur 4 bits. Le code binaire de 3 est 0011. Le code binaire de -3 est donc 1101.
- Autre possibilité : chercher le code A de l'entier naturel $\mathcal{N}(A) = 2^n + \mathcal{Z}(A)$
  Exemple. Coder -3 sur 4 bits. On cherche le code binaire de 16-3=13. Le code binaire de -3 est donc 1101.

Gamme représentable

La gamme des nombres représentables est $[-2^{n-1}, 2^{n-1}-1]$.

n	gamme représentable
8	[-128 ... +127]
16	[-32768 ... +32767]
32	$[\simeq -2,15\,10^9 \ldots \simeq +2,15\,10^9]$
64	$[\simeq -9,22\,10^{18} \ldots \simeq +9,22\,10^{18}]$

Codage par la technique de l'excédent

Notons $\mathcal{X}_e(A)$ le nombre relatif de mot-code $A = a_{n-1}a_{n-2}\ldots a_0$.
Le code A représente l'entier relatif :

$$\mathcal{X}_e(A) = \sum_{i=0}^{n-1} a_i\,2^i - e = \mathcal{N}(A) - e$$

- Ce système est appelé codage *excédent e* ou encore codage *binaire décalé de e* ou encore *binaire biaisé de e*.
- Il consiste à décaler les codes des nombres de telle sorte qu'il y ait autant de négatifs que de positifs.
- Comme la gamme des nombres représentables est $[-e, 2^n - 1 - e]$, les valeurs de e peuvent donc être $e = 2^{n-1}$ ou bien $e = 2^{n-1} - 1$

Codage d'un nombre par excédent

L'opération de codage d'un entier relatif s'effectue de la façon suivante :

1. ajouter l'excédent au nombre à coder
2. prendre le code binaire naturel du résultat

Exemple. On travaille avec $n = 4$ et $e = 8$.
Soit à coder $\mathcal{X}_8(A)$=-3.
On ajoute l'excédent $(-3 + 8 = 5)$, puis on prend le code binaire de $5 \Rightarrow A = 0101$.

Résumé

En résumé, les différences entre les quatre techniques de codage des nombres entiers étudiées sont visualisées dans le tableau pour des codes de 4 bits.

code bin A	binaire naturel $\mathcal{N}(A)$	signe et val.abs. $\mathcal{S}(A)$	compl. à 2 $\mathcal{Z}(A)$	excéd. 8 $\mathcal{X}_8(A)$	code bin A	binaire naturel $\mathcal{N}(A)$	signe et val.abs. $\mathcal{S}(A)$	compl. à 2 $\mathcal{Z}(A)$	excéd. 8 $\mathcal{X}_8(A)$
0000	0	+0	0	-8	1000	8	-0	-8	0
0001	1	+1	+1	-7	1001	9	-1	-7	+1
0010	2	+2	+2	-6	1010	10	-2	-6	+2
0011	3	+3	+3	-5	1011	11	-3	-5	+3
0100	4	+4	+4	-4	1100	12	-4	-4	+4
0101	5	+5	+5	-3	1101	13	-5	-3	+5
0110	6	+6	+6	-2	1110	14	-6	-2	+6
0111	7	+7	+7	-1	1111	15	-7	-1	+7

Connaissant le mot-code, il faut connaitre la convention de codage pour savoir ce qu'il représente.

Codage des réels

Les calculateurs ne manipulent pas que des nombres entiers mais doivent pouvoir aussi manipuler des nombres réels.
Deux techniques de codage pour représenter des réels :

1. la virgule fixe à employer lorsque les réels à manipuler sont à peu près du même ordre de grandeur
2. la virgule flottante dans le cas général

A noter que l'ensemble des réels est infini alors que l'ensemble des codes pour n donné est fini et égal à 2^n. le codage d'un réel introduit donc la plupart du temps une erreur.

Codage en virgule fixe

Soit des nombres codés sur n bits avec m bits pour la partie fractionnaire.
La virgule se situe entre le bit de rang m et celui de rang $m-1$.
Le mot-code $A = a_{n-1} \dots a_m, a_{m-1} \dots a_0$ représente le nombre réel :

$$\Phi_m(A) = \sum_{i=0}^{n-1} a_i \, 2^{i-m}$$

ou encore :

$$\Phi_m(A) = (\sum_{i=0}^{n-1} a_i \, 2^i) \times 2^{-m} = \mathcal{N}(A) \times 2^{-m}$$

Le nombre de bits après la virgule (m) est appellé *facteur de cadrage*.
Tous les réels codés selon cette technique doivent avoir le même facteur de cadrage

Propriétés

- Si on s'intéresse uniquement aux nombres positifs, cette technique permet de représenter 2^n réels à partir de 0, espacés de 2^{-m}.
- Pour coder un réel non représentable de façon exacte, une erreur est introduite par le codage. Elle est au maximum de 2^{-m} si on procède par troncature et de $2^{-(m+1)}$ dans le cas d'un arrondi. La virgule fixe garantit donc une précision *absolue* lors de l'opération du codage
- L'intérêt de cette technique vient du fait que les opérations d'addition et de soustraction s'effectuent par des opérations entières puisque indépendantes de la position de la virgule

Virgule fixe et complément à 2

Pour pouvoir représenter des réels positifs et négatifs, le codage se fait aussi en utilisant la technique du complément à 2
Le mot-code $A = a_{n-1} \, a_{n-2} \dots a_m, a_{m-1} \dots a_0$ représente le nombre réel :

$$\Phi_m(A) = -a_{n-1} \, 2^{n-1-m} + \sum_{i=0}^{n-2} a_i \, 2^{i-m}$$

soit :

$$\Phi_m(A) = (-a_{n-1} \, 2^{n-1} + \sum_{i=0}^{n-2} a_i \, 2^i) \times 2^{-m} = \mathcal{Z}(A) \times 2^{-m}$$

qui montre que le codage des réels positifs et négatifs en virgule fixe se ramène à des manipulations d'entiers relatifs.

Exemples

Soit un codage avec $n = 8$ et $m = 4$.

- Le code `A=1011 0110` représente le nombre : $-2^3 + 2^1 + 2^0 + 2^{-2} + 2^{-3} = -8 + 2 + 1 + 0,25 + 0,125 = -4,625$
- Pour coder $\Phi_4(A) = 4,5$ on peut le multiplier par $2^4 = 16$ et chercher le code de l'entier relatif en complément à 2 : $\mathcal{Z}(A) = 72$ soit $A = 01001000$
- on peut aussi chercher séparément le code de la partie entière : 100, celui de la partie décimale : 1 puis concaténer les deux codes et compléter avec des 0 à gauche et à droite : $A = 01001000$

Résumé

- Avantage : Ramène les calculs sur des réels à des calculs sur des entiers, lesquels sont très rapides.
- Inconvénient : Utilisable seulement pour traiter des réels qui ont le même ordre de grandeur
- Usage : s'emploie en gestion (euros, centimes) et en traitement du signal (les convertisseurs analogiques-numériques codent au maximum sur 16 bits)

Virgule flottante - Introduction

- En calcul scientifique, les nombres ont des ordres de grandeur très différents.
  Exemples :
  Vitesse de la lumière : $c = 299792458 m/s$
  Charge élémentaire : $e = 1,602176565 \times 10^{-19}$
- On les représente donc en notation "scientifique" dans laquelle seuls les *chiffres significatifs* et l'ordre de grandeur sont codés.

Normalisation

Le module d'un nombre réel peut être écrit sous la forme $m \times B^e$ avec :

m	la mantisse (réel en virgule fixe)
B	la base utilisée pour le codage
e	l'exposant (entier relatif)

Pour que la mantisse ne comporte que des chiffres significatifs, elle doit être normalisée, c'est-à-dire que sa partie entière est composée d'un nombre prédéfini de chiffres significatifs. Elle est choisie telle que :

$$m \in [B^{p-1}, B^p[$$

où p est un entier naturel qui résulte d'une convention (il indique le nombre de chiffres constituant la partie entière de la mantisse).

Exemple en base 10. Soit le nombre $x = 0,00716$.

- En notation scientifique, il s'écrit $x = 0,716 \times 10^{-2}$ si on adopte la convention $p = 0$,
- il s'écrit $x = 7,16 \times 10^{-3}$ si la convention est $p = 1$.

Codage dans un calculateur

Dans un calculateur, il est facile de travailler avec une base $B = 2^k$, la valeur $k = 1$ étant la plus fréquente.
Pour le signe, un bit est suffisant.
Ainsi, un mot-code A représentant un réel en virgule flottante est composé de 3 champs S, M et E, avec :

S	code du signe (0 : positif ou nul, 1 : négatif)
M	code de la mantisse (virgule fixe)
E	code de l'exposant (entier relatif)

Débordements

- Soit α le plus petit nombre représentable en module et β le plus grand.
- Lors d'un calcul sur des réels, soit γ le module du résultat.
- si $\gamma > \beta$, le réel n'est pas représentable. On dit qu'il y a débordement par valeur supérieure (*overflow*)
- si $\gamma < \alpha$, on dit qu'il y a débordement par valeur inférieure (*underflow*)

Norme IEEE P754

Tous les processeurs actuels respectent la norme IEEE P754 qui est devenue le système standard (depuis 1985).
Elle prévoit trois formats (simple précision, double précision et précision étendue)
simple précision (32 bits)

31	30 . . . 23	22 . . . 0
S	E	F

La convention est $B = 2$ et $p = 1 \;\Rightarrow m \in [1, 2[$. La mantisse s'écrit donc sous la forme $m = 1, f$.

Champs du code

- S est le bit de signe. Par convention il vaut 1 pour les nombres négatifs
- F est le code binaire de la partie fractionnaire f de la mantisse m codée sur 23 bits comme un réel en virgule fixe.
- La partie entière n'est pas codée, elle est implicite puisque valant toujours 1.
- E est le code de l'exposant sur 8 bits représenté excédent 127 $(2^7 - 1)$. Les valeurs possibles de l'exposant sont $e \in [-126, +127]$
- Les codes extrêmes sont réservés pour coder des cas spéciaux

Valeurs de E réservées

- le code $E = 0\ldots0$ avec $F = 0$ est réservé pour coder le nombre 0, non représentable sous forme normalisée.
- le code $E = 0\ldots0$ avec $F \neq 0$ est réservé pour coder des nombres dénormalisés (résultats intermédiaires obtenus par *underflow*).
- le code $E = 1\ldots1$ avec $F = 0$ est réservé pour coder ∞.
- le code $E = 1\ldots1$ avec $F \neq 0$ sert à coder NaN (Not a Number). Ce code est utilisé pour signaler un résultat d'opération invalide ou non défini (par exemple $\frac{\infty}{\infty}$ ou $\frac{0}{0}$).

Expression de $\mathcal{R}(A)$

Dans cette norme, le mot-code $A = S|E|F$ représente le réel :

$$\mathcal{R}(A) = (-1)^S \times (1 + \Phi_{23}(F)) \times 2^{\mathcal{X}_{127}(E)}$$

La gamme des nombres représentables est $[10^{-38}, 10^{+38}]$ et la précision relative de $2^{-23} = 1,19 \times 10^{-7}$, soit de 7 chiffres décimaux significatifs.

Exemple

Coder le réel +3,0
Il peut être écrit sous forme normalisée : $+1,5 \times 2^1$ d'où :

1. S=0 (nombre positif)
2. $e = 1$, $\mathcal{N}(E) = 128$: `E=1000 0000`
3. $f = 0,5$: `F=100 0000 0000 0000 0000 0000`
4. donc : `A=0100 0000 0100 0000 0000 0000 0000 0000` soit `40400000h` en hexadécimal.

Double précision (64 bits)

Les mêmes conventions que pour le format simple précision s'appliquent. Les différences sont :
F (partie fractionnaire) codée sur 52 bits
E (exposant) codé sur 11 bits excédent 1023

63	62 ... 52	51 ... 0
S	E	F