Architecture des ordinateurs
Cours 3
Représentation des nombres rationnels en virgule fixe ou virgule flottante

Plan du cours 3

Nombres Rationnels

Nombre Rationnel

Un nombre rationnel est un nombre qui peut s'exprimer comme le quotient de deux entiers relatifs.
On peut ainsi écrire les nombres rationnels sous forme de fractions notées :

$$\frac{a}{b}$$

où a, le *numérateur*, est un entier relatif et b, le *dénominateur*, est un entier relatif non nul.

Développement décimal dans une base B

Lé développement décimal d'un nombre est sa représentation sous forme de sommes de puissance dans la base de représentation (B > 1). La représentation est unique si chaque coefficient $0 \leq a_i < B$.

$$\sum_{i=-\infty}^{i=+\infty} a_i \times B^i$$

Il est représenté par le mot (non nécessairement borné)
$\ldots a_k a_{k-1} \ldots a_1 a_0, a_{-1} a_{-2} \ldots a_{-p} \ldots$ avec $\forall i : 0 < a_i \leq B$

Nombres Rationnels

On distingue la partie entière et la partie fractionnaire :

$$\sum_{i=0}^{i=+\infty} a_i \times B^i + \sum_{j=-\infty}^{j=-1} a_j \times B^j$$

(les 0 non significatifs sont généralement omis)

Exemples

- Tout entier naturel ou relatif : 0 ; 1 ; 378 ; -250 ; ...
- Les décimaux : 20,378 ; -10,01 ; ... (* développement décimal fini *)
- Les fractions à développement décimal infini périodique : $\frac{1}{3} = 0,333...333$ également noté $0,\underline{3}$;
- π et $\sqrt{2}$ présentent un développement décimal infini apériodique et ne sont pas rationnels, ils sont *irrationnels*.

Nombres Rationnels

2 Propriétés

- Tout nombre rationnel non nul s'exprime de manière unique comme une fraction dont le numérateur et le dénominateur sont premiers entre eux, avec un dénominateur positif.
- Le développement décimal d'un nombre rationnel est soit fini, soit infini périodique à partir d'une certaine décimale ; ceci est vrai dans n'importe quelle base B > 1.

Exemples

- $\frac{2}{4} = \frac{18}{36} = \frac{1}{2}$ et 1 et 2 sont premiers entre eux.
- $\frac{1}{2} = 0,5_{10} = 0,1_2$ les deux développement décimaux finis de $\frac{1}{2}$ dans les bases 10 et 2.
- $\frac{1}{5} = 0,2_{10} = 0,001100110011...001_2$ développement fini en base 10 et infini périodique en base 2.
- $\frac{1}{3} = 0,3333...33_{10} = 0,010101...01..._2$ développements infinis périodiques en base 10 et en base 2.

Construction du développement décimal d'un nombre rationnel dans une base B

Méthode par multiplications successives

Soit $N = \frac{a}{b}$, dont on construit le développement décimal en base $B > 1$
Partie entière de N : cf cours 1.
Partie fractionnaire de N, notée $N_f = a_{-1}.B^{-1} + a_{-2}.B^{-2} + a_{-3}.B^{-3}....$
$N_f * B = a_{-1} + N'_f$ avec $N'_f = a_{-2}.B^{-1} + a_{-3}.B^{-2} + ...) \rightarrow$ on extrait a_{-1}
$N'_f * B = a_{-2} + N''_f$ avec $N''_f = a_{-3}.B^{-1} + a_{-4}.B^{-2} + ...) \rightarrow$ on extrait a_{-2}
... et ainsi de suite (soit jusqu'à N'''_f ne contenant que des 0, soit en bornant le nombre d'itérations).

Développement décimal en base 2

Soit $N = \frac{2}{5} = 0,4$
$N_f = 0,4$
$N_f * 2 = 0,8 < 1$ donc $a_{-1} = 0$ et $N'_f = 0,8$
$N'_f * 2 = 1,6 > 1$ donc $a_{-2} = 1$ et $N''_f = 0,6$
$N''_f * 2 = 1,2 > 1$ donc $a_{-3} = 1$ et $N_f^{(3)} = 0,2$
$N_f^{(3)} * 2 = 0,4 < 1$ donc $a_{-4} = 0$ et $N_f^{(4)} = 0,4$, on retombe sur N_f, le développement est infini périodique.
Le développement décimal est $0,01100110011...$ également noté $0,\underline{0110}$

Remarques

Les nombres rationnels sont représentés et manipulés sous la forme de mots binaires (de taille bornée), ce qui induit des limites de représentation en terme *d'amplitude* de représentation et de *précision*.

Deux représentations différentes :

- Représentation en virgule fixe : basée sur le développement décimal (en base 2), avec des champs fixes pour les parties entière et fractionnaire.
- Représentation en virgule flottante : basée sur la représentation en *ordre de grandeur* : signe/module/$2^{exposant}$, avec des champs fixes pour représenter ces 3 grandeurs.

Représentation en virgule fixe

Format virgule fixe sur <n,p> bits

- Le mot binaire support est de taille $n + p$ bits, avec n bits pour la partie entière et p pour la partie fractionnaire.
- La partie entière peut être négative, nulle ou positive ; sa représentation suit le codage des entiers relatifs sur n bits.
- La partie fractionnaire est nulle ou positive ; sa représentation suit le codage des entiers naturels sur p bits.
- Les indices des chiffres du mot binaire support varient de 0 à $n + p - 1 \rightarrow$ le chiffre pondéré par 2^0 est en position p.

$$N = (-a_{p+n-1} \times 2^{n-1} + \sum_{i=p}^{i=p+n-2} a_i \times 2^{i-p}) + (\sum_{j=0}^{j=p-1} a_j \times 2^{j-p})$$

Exemples

format	P. entière / P. fractionnaire								interprétation rationnel
<5,3>	0	0	1	1	0	1	0	1	6,625
<5,3>	1	0	1	1	0	1	0	1	- 25,375 (*)
<3,5>	0	0	1	1	0	1	0	1	1,65625

(*) = -32 + 6,625 = -25,375

Représentation en virgule fixe

Quelques représentations remarquables en virgule fixe sur <n,p> bits

- Le plus grand nombre représentable : partie entière positive et maximale, partie fractionnaire maximale : `0b0111...1111`
- Le plus petit positif représentable : partie entière nulle, partie fractionnaire la plus petite possible : `0b0000 ...0001`
- La plus négatif : partie entière négative et correction positive (entière ou fractionnaire) nulle : `0b100..000`
- Zéro : parties entière et fractionnaire nulles : `0b00...0000`

Intervalles et précisions de la représentation en virgule fixe sur bits

- Intervalle de représentation : $[-2^{n-1}, 2^{n-1} - 2^{-p}]$
- Cet intervalle est discrétisé par *pas constant* de 2^{-p}, ce qui représente la *précision* de la représentation.

Opérations arithmétiques en virgule fixe

Addition/Soustraction en virgule fixe sur <n,p> bits

Soient 2 nombres rationnels A et B à sommer, représentés en virgule fixe sur un format de <n,p> bits.

- Les virgules des opérandes sont alignées.
- L'addition est réalisée sur des mots de taille au moins égale à $n + p$ bits, la retenue issue du rang $p - 1$ (de pondération 2^{-1}) étant propagée sur le rang p (de pondération 2^0).
- L'indicateur de dépassement par débordement supérieur est *OV* (indicateur des entiers relatifs).

Exemples sur le format <5,3>

- 00100,001 + 01110,100 = 10010,101 $\rightarrow$ OV
- 00000,001 - 00000,011 = 11111,110

Autres opérations arithmétiques en virgule fixe sur <n,p> bits

- Multiplications et divisions : se ramènent au cas entiers relatifs et traitement à part de la virgule (décalage du résultat pour coïncider au format des opérandes)
- Comparaisons : se ramènent au cas entiers relatifs (les virgules sont alignées)

Représentation en virgule flottante

But

$\Rightarrow$ Représenter dans un même format des nombres très grands avec peu de précision ou des nombres très petits et très précisément.

Notation scientifique normalisée des nombres rationnels dans une base B

Un nombre rationnel non nul peut être représenté de façon unique sous la forme :

$$N = (-1)^S \times MODULE \times B^{MAGNITUDE} et 0 \leq MODULE < B$$

C'est une représentation en *MODULE*, *SIGNE* et *ordre de grandeur* (ou *MAGNITUDE*).

Exemples en base 10 et en base 2

- $123,89 = 8,2389 \times 10^2$
- $11000000,1 = 1,10000001 \times 2^7$
- $0,001101 = 1,1010000 \times 2^{-3}$

La représentation en virgule flottante définit le format du mot binaire représentant le nombre rationnel : le nombre de bits et le codage de chacun des champs permettant de reconstituer *SIGNE*, *MODULE* et *MAGNITUDE*.

Norme IEEE-754 (1)

Besoin de normalisation pour faciliter les développements / usage des coprocesseurs / bibliothèques manipulant les flottants

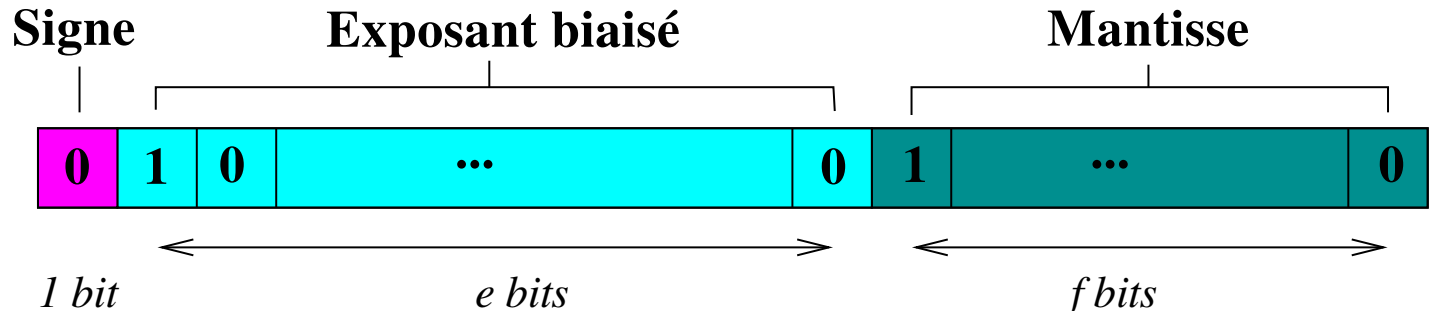
Norme IEEE-754 : dès 1985 (coprocesseurs flottants), puis 2008 (GPU), puis nouveaux formats avec l'avénement du DeepLearning.

Initialement, 2 formats :

- simple et double précisions (32 et 64b).
- Représentations normalisée et dénormalisée
- Modes d'arrondis / opérations arithmétiques et trigonométriques / exceptions

Ces formats (et leurs successeurs) sont utilisés par la très grande majorité des unités flottantes matérielles.

Principes des représentations IEEE-754

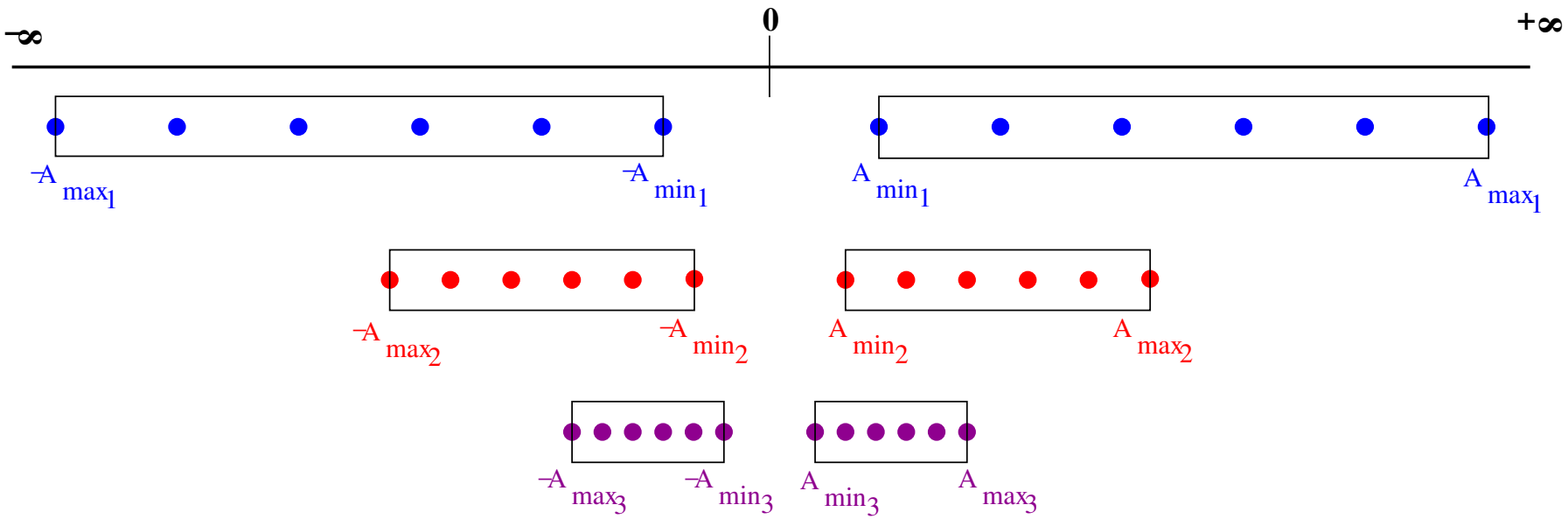


- Signe : $S = b_{f+e-1}$: $0 \rightarrow +$ et $1 \rightarrow -$
- Magnitude (ou exposant) : codé sur les bits $b_{e-1+f}...b_f$ du champ *Exposant biaisé* selon la représentation des entiers naturels biaisée avec un décalage fixe (appelé *biais* ou *bias*) égal à $-2^{e-1} - 1$.
  $E = -bias + \sum_{i=f}^{j=f+e-1} b_i.2^{i-f}$ Les cas particuliers sont identifiés par les valeurs extrêmes de l'intervalle décrit par le champ *Exposant biaisé* (nombre dénormalisé, infinités, NaN).
- Module : codé sur les bits $b_{f-1}, ..b_0$ du champ *Mantisse*, selon la représentation des entiers naturels sur f bits.
  - Cas normalisé : le module est de la forme **1,xxx**. On ne stocke que la partie fractionnaire du module, le 1 implicite doit être ajouté, alors $M = 1 + \sum_{i=0}^{i=f-1} b_i.2^{i-f}$
  - Cas dénormalisé : le module est de la forme **0,xxx**. $M = 0 + \sum_{i=0}^{i=f-1} b_i.2^{i-f}$

Interprétation du mot binaire selon le codage virgule flottante (cas normalisé/dénormalisé) :
$N = (-1)^S \times M \times 2^E$

Principes des représentations IEEE-754

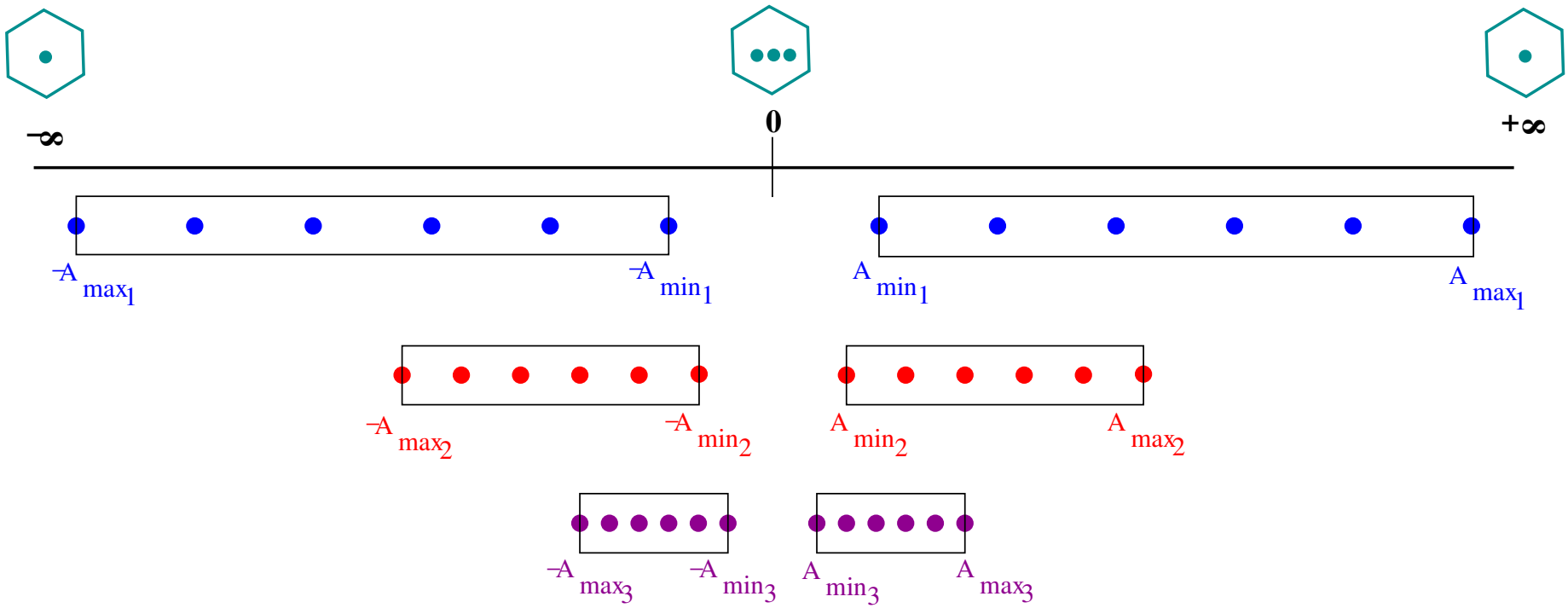
- La forme normalisée ne permet pas de représenter la valeur 0.
- L'espace des valeurs représentables par la forme normalisée est l'union de couples d'intervalles discrets centrés sur 0, d'amplitudes dépendant de *f* et de pas de discrétisation dépendant de la valeur de MAGNITUDE.



Principes des représentations IEEE-754

$\Rightarrow$ Ajouts de cas particuliers (repérés par des valeurs particulières du champs Exposant biaisé) :

	Signe	Exposant biaisé	Mantisse
Zéro	0	0..0	0..0
$+\infty$	0	1..1	0..0
$-\infty$	1	1..1	0..0
NaN	x	1..1	xx1xx
Nombre dénormalisé	x	0..0	xx1xx



Représentation IEEE-754 - Simple Précision

- Format 32 bits, $e = 8$ et $f = 23$ donc $bias = -(2^7 - 1) = -127$

Exemple de décodage, cas normalisé

- Mot à décoder : `0b 1 01000011 11101000000000000000000`
- Signe : 1 donc nombre négatif
- Exposant biaisé : `01000011` ; différent de `00000000` ou `11111111` donc le nombre est normalisé. Valeur numérique de l'exposant biaisé : 67, donc exposant $E = 67 - 127 = -60$
- Mantisse : `11101000000000000000000` = $1.2^{-1} + 1.2^{-2} + 1.2^{-3} + 0.2^{-4} + 1.2^{-5} = 0,5 + 0,25 + 0,125 + 0,03125 = 0,90625$. Comme la représentation est normalisée, le module vaut 1+Mantisse, donc $M = 1,90625$.
- $N = (-1)^S \times M \times 2^E = -1,90625 \times 2^{-60} = -1,653408313 \times 10^{-18}$

Représentation IEEE-754 - Simple Précision

- Format 32 bits, $e = 8$ et $f = 23$ donc $bias = -(2^7 - 1) = -127$

Exemple de codage, cas normalisé

1. Nombre à coder $N = 12,375$
2. Normaliser le nombre en base 2 :
   - Développement décimal en base 2 : $N = 1100,011000000...000$
   - Normalisation $N = 1,100011 \times 2^3$
3. Identification des valeurs des champs :
   - S = 0
   - E = 3 donc Exposant biaisé = 3+127 = 130, soit en base 2 sur 8 bits : `0b1000 0010`
   - M = 1,100011 : la mantisse représente la partie fractionnaire uniquement, sur 23 bits : `0b1000 1100 0000 ...000`
4. N est représenté par le mot binaire `0b 0 10000010 10001100000000000000000`
5. soit `0x42980000` en hexadécimal

Représentation IEEE-754 - Simple Précision

- Format 32 bits, $e = 8$ et $f = 23$ donc $bias = -(2^7 - 1) = -127$

Intervalle englobant des valeurs représentables - nombres normalisés

- Nombre normalisé le plus grand représentable
  - `0b 0 11111110 11111111111111111111111`
  - $A_{max} = 2^{254-127} \times (1 + \sum_{i=0}^{i=22} 2^{-i}) = 2^{127} \times (2^0 - 2^{-23}) \approx 3,4 \times 10^{38}$
- Nombre normalisé le plus petit représentable (valeur absolue) :
  - `0b 0 00000001 00000000000000000000000`
  - $A_{min} = 2^{1-127} \times (1 + 0) = 2^{-126} \times 2^0 = 2^{-126} \approx 1,8 \times 10^{-38}$

Représentation IEEE-754 - Simple Précision

Nombres dénormalisés

- (Champ Eb à 0 et Mantisse contient au moins un 1) :
  - `0b 0 00000000` $b_{22}b_{21}...b_0$
  - Champ Exposant biaisé à 0 $\rightarrow$ E = -126 et N = 0,xxx (bit implicite à 0)
  - $N = 2^{-126} \times (0 + \sum_{i=0}^{i=22} b_i 2^{i-23})$
- Le plus petit dénormalisé : `0b 0 00000000 00000000000000000000001`
  $\rightarrow 2^{-126} \times 2^{-23} = 2^{-149}$
- Le plus grand dénormalisé : `0b 0 00000000 11111111111111111111111`
  $\rightarrow 2^{-126} \times (1 - 2^{-23})$
- Progression par pas constant de $2^{-126} \times 2^{-23}$

Opérations arithmétiques

Soient $A = (-1)^{S_A} \times M_A \times 2^{E_A}$ et $B = (-1)^{S_B} \times M_B \times 2^{E_B}$, deux nombres représentés en virgule flottante.

Addition ou Soustraction : $M = A + B$ ou $M = A - B$

1. Cadrage des deux opérandes : décaler le module de l'opérande de magnitude la plus petite sur celle de l'autre opérande (décalage à droite $\rightarrow$ potentielle perte de précision)
2. Sommer / Soustraire les modules
3. Renormaliser la somme (potentiel décalage à droite/gauche)

Multiplication ou division : $M = A \times B$ ou $M = \frac{A}{B}$

Les calculs s'effectuent directement sur les modules et exposants :

- $A \times B = (M_A \times M_B) \times 2^{(E_A + E_A)}$
- $\frac{A}{B} : (\frac{M_A}{M_B}) \times 2^{(E_A - E_B)}$

Ces opérations sont réalisées par des circuits dédiés (coprocesseurs flottants) ou sont émulées, en général sur une représentation étendue des champs mantisse et exposant (afin de limiter la perte de précision).

Approximations - Arrondis

Le résultat d'une opération arithmétique peut ne pas être représentable dans le format virgule flottante :

- Dépassement de capacité supérieur/inférieur $\rightarrow$ symboles $+\infty$ et $-\infty$
  - $1,5 \times 10^{38} \times 10 > 3,4 \times 10^{38}$
  - $2,0 \times 10^{-}38/10^{5} < 1,18 \times 10^{-38}$
- Opération invalide $\rightarrow$ symbole `NaN` (Not a Number)
  - $0.0/0.0$; ∞/∞ ; $\sqrt{-1}$; $log(-10)$; ...
- Perte de précision lors des opérations de cadrage ou renormalisation $\rightarrow$ Arrondis
  - Au plus proche
  - Dirigé (vers 0, $-\infty$ ou $+\infty$)

Approximations - Arrondis

Exemple

- Soit $N == 2^{24} = 16777216$,
- représenté par le mot : `0b 0 10010111 00000000000000000000000`
  (S=0 ; Eb = 151 donc E = 24 ; M = 1,0...0)
- Calcul de $N + 3$:
  1. Cadrage des opérandes :
     - $N = 1,000...000(23'0') \times 2^{24}$
     - $3 = 0,000...001(22'0') \times 2^{24} \rightarrow$ perte de précision du 1 de poids faible
  2. $S = 1,000...001(22'0') \times 2^{24} = 16777218$ et non pas 16777219

Autres formats flottants classiquement utilisés

Format	Paramètres				
	n	f	e	E	|A|
SP-IEEE (1985)	32	23	8	$[-126..+127]$	$[1, 18.10^{-38}..3, 4.10^{38}]$
DP-IEEE (1985)	64	52	11	$[-1022..1023]$	$[2, 23.10^{-308}..1, 79.10^{308}]$
DP-ext-IEEE (1985)	80	63	15	$[-16382..16383]$	$[3, 37.10^{-4932}..1, 18.10^{4932}]$
FP16-IEEE (2008)	16	5	10	$[-30..+31]$	$[2^{-30}..1, 9995.2^{31}]$
BP16 'Brain Float' (2019)	16	7	8	$[-126..+127]$	$[1, 18.10^{-38}..3, 4.10^{38}]$
FP8-Nvidia (2022)	8	3	4	$[-6..+7]$	$[2^{-6}..1, 875.2^{7}]$
FP8-Nvidia (2025)	8	2	5	$[-14..+15]$	$[2^{-14}..1, 75.2^{15}]$

Formats flottants sur 8b très actuels car très utilisés en deep-learning. Selon l'usage, privilégier le champ exposant (largeur de l'amplitude de |A|) ou le champ mantisse (calcul sur de nombres avec peu de variation d'ordre de grandeur).

D'après les données de :
[1] P. Darche, *Architecture des ordinateurs, Représentation des nombres et codes*, ed. G. Morin, 2000.
[2] A. Cassagne, *Numbers Representation in Computer Science, cours M2-SESI-MU4IN112*, Sorbonne Université, 2025.

Principales caractéristiques des formats virgule fixe et virgule flottante

Virgule fixe :

- Position de la virgule figée dans le format
- Domaine : 1 intervalle discrétisé par un pas fixe
- Opérations de comparaisons, additions soustractions simples (virgules alignées) : réutilisation de composants existants

Virgule flottante :

- Position de la virgule dépend de l'ordre de grandeur du nombre
- Domaine : union d'intervalles discrétisés avec des pas dépendant de l'ordre de grandeur du nombre et et de la précison de sa partie fractionnaire
- Opérations arithmétiques nécessitent des recadrages (alignement de virgule) $\rightarrow$ perte de précision possible, et résultat pas toujours représentable exactement

Très utilisés (HPC, GPU, Deeplearning), particulièrement dans des petits formats compatibles avec des traitements embarqués.