NORME INTERNATIONALE

INTERNATIONAL STANDARD

CEI
IEC
60559

Deuxième édition
Second edition
1989-01

Arithmétique binaire en virgule flottante pour systèmes à microprocesseurs

Binary floating-point arithmetic for microprocessor systems

Numéro de référence
Reference number
CEI/IEC 60559: 1989

Numéros des publications

Depuis le 1er janvier 1997, les publications de la CEI sont numérotées à partir de 60000.

Publications consolidées

Les versions consolidées de certaines publications de la CEI incorporant les amendements sont disponibles. Par exemple, les numéros d'édition 1.0, 1.1 et 1.2 indiquent respectivement la publication de base, la publication de base incorporant l'amendement 1, et la publication de base incorporant les amendements 1 et 2.

Validité de la présente publication

Le contenu technique des publications de la CEI est constamment revu par la CEI afin qu'il reflète l'état actuel de la technique.

Des renseignements relatifs à la date de reconfirmation de la publication sont disponibles dans le Catalogue de la CEI.

Les renseignements relatifs à des questions à l'étude et des travaux en cours entrepris par le comité technique qui a établi cette publication, ainsi que la liste des publications établies, se trouvent dans les documents ci-dessous:

- **«Site web» de la CEI***
- **Catalogue des publications de la CEI**
  Publié annuellement et mis à jour régulièrement
  (Catalogue en ligne)*
- **Bulletin de la CEI**
  Disponible à la fois au «site web» de la CEI* et comme périodique imprimé

Terminologie, symboles graphiques et littéraux

En ce qui concerne la terminologie générale, le lecteur se reportera à la CEI 60050: *Vocabulaire Electrotechnique International* (VEI).

Pour les symboles graphiques, les symboles littéraux et les signes d'usage général approuvés par la CEI, le lecteur consultera la CEI 60027: *Symboles littéraux à utiliser en électrotechnique,* la CEI 60417: S*ymboles graphiques utilisables sur le matériel. Index, relevé et compilation des feuilles individuelles,* et la CEI 60617: *Symboles graphiques pour schémas.*

* Voir adresse «site web» sur la page de titre.

Numbering

As from 1 January 1997 all IEC publications are issued with a designation in the 60000 series.

Consolidated publications

Consolidated versions of some IEC publications including amendments are available. For example, edition numbers 1.0, 1.1 and 1.2 refer, respectively, to the base publication, the base publication incorporating amendment 1 and the base publication incorporating amendments 1 and 2.

Validity of this publication

The technical content of IEC publications is kept under constant review by the IEC, thus ensuring that the content reflects current technology.

Information relating to the date of the reconfirmation of the publication is available in the IEC catalogue.

Information on the subjects under consideration and work in progress undertaken by the technical committee which has prepared this publication, as well as the list of publications issued, is to be found at the following IEC sources:

- **IEC web site***
- **Catalogue of IEC publications**
  Published yearly with regular updates
  (On-line catalogue)*
- **IEC Bulletin**
  Available both at the IEC web site* and as a printed periodical

Terminology, graphical and letter symbols

For general terminology, readers are referred to IEC 60050: *International Electrotechnical Vocabulary* (IEV).

For graphical symbols, and letter symbols and signs approved by the IEC for general use, readers are referred to publications IEC 60027: *Letter symbols to be used in electrical technology*, IEC 60417: *Graphical symbols for use on equipment. Index, survey and compilation of the single sheets* and IEC 60617: *Graphical symbols for diagrams.*

* See web site address on title page.

NORME INTERNATIONALE

INTERNATIONAL STANDARD

CEI
IEC
60559

Deuxième édition
Second edition
1989-01

Arithmétique binaire en virgule flottante pour systèmes à microprocesseurs

Binary floating-point arithmetic for microprocessor systems

International Electrotechnical Commission 3, rue de Varembé Geneva, Switzerland
Telefax: +41 22 919 0300 e-mail: inmail@iec.ch IEC web site http: //www.iec.ch

IEC Commission Electrotechnique Internationale
International Electrotechnical Commission
Международная Электротехническая Комиссия

CODE PRIX
PRICE CODE S

Pour prix, voir catalogue en vigueur
For price, see current catalogue

SOMMAIRE

CONTENTS

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

ARITHMETIQUE BINAIRE EN VIRGULE FLOTTANTE POUR SYSTEMES A MICROPROCESSEURS

PREAMBULE

1) Les décisions ou accords officiels de la CEI en ce qui concerne les questions techniques, préparés par des Comités d'Etudes où sont représentés tous les Comités nationaux s'intéressant à ces questions, expriment dans la plus grande mesure possible un accord international sur les sujets examinés.

2) Ces décisions constituent des recommandations internationales et sont agréées comme telles par les Comités nationaux.

3) Dans le but d'encourager l'unification internationale, la CEI exprime le voeu que tous les Comités nationaux adoptent dans leurs règles nationales le texte de la recommandation de la CEI, dans la mesure où les conditions nationales le permettent. Toute divergence entre la recommandation de la CEI et la règle nationale correspondante doit, dans la mesure du possible, être indiquée en termes clairs dans cette dernière.

PREFACE

La présente norme a été établie par le Sous-Comité 47B: Systèmes à microprocesseurs, du Comité d'Etudes n° 47 de la CEI: Dispositifs à semiconducteurs. (Ce Sous-Comité a été repris par l'ISO/IEC JTC 1.)

Cette deuxième édition de la Publication 559 remplace la première édition parue en 1982.

Le texte de cette norme est issu des documents suivants:

Règle des Six Mois	Rapport de vote
47B(BC)19	47B(BC)26

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à l'approbation de cette norme.

INTERNATIONAL ELECTROTECHNICAL COMMISSION

BINARY FLOATING-POINT ARITHMETIC FOR MICROPROCESSOR SYSTEMS

FOREWORD

1) The formal decisions or agreements of the IEC on technical matters, prepared by Technical Committees on which all the National Committees having a special interest therein are represented, express, as nearly as possible, an international consensus of opinion on the subjects dealt with.

2) They have the form of recommendations for international use and they are accepted by the National Committees in that sense.

3) In order to promote international unification, the IEC expresses the wish that all National Committees should adopt the text of the IEC recommendation for their national rules in so far as national conditions will permit. Any divergence between the IEC recommendation and the corresponding national rules should, as far as possible, be clearly indicated in the latter.

PREFACE

This standard has been prepared by Sub-Committee 47B: Microprocessor systems, of IEC Technical Committee No. 47: Semiconductor devices. (This Sub-Committee has been taken over by ISO/IEC JTC 1.)

This second edition of IEC Publication 559 replaces the first edition issued in 1982.

The text of this standard is based on the following documents:

Six Months' Rule	Report on Voting
47B(CO)19	47B(CO)26

Full information on the voting for the approval of this standard can be found in the Voting Report indicated in the above table.

ARITHMETIQUE BINAIRE EN VIRGULE FLOTTANTE POUR SYSTEMES A MICROPROCESSEURS

---

1. Domaine d'application

1.1 *Objectifs de réalisation*

L'objectif est qu'une réalisation d'un système à virgule flottante conforme à la présente norme puisse être effectuée entièrement par logiciel, entièrement par matériel, ou par une combinaison quelconque de logiciel et de matériel. C'est l'environnement que le programmeur ou l'utilisateur voit qui est conforme ou non conforme à cette norme. Les composants matériels qui nécessitent un support logiciel pour devenir conformes ne doivent pas être qualifiés de conformes indépendamment d'un tel logiciel.

1.2 *Inclusions*

Cette norme spécifie:

1) les formats de base et étendu des nombres en virgule flottante;
2) les opérations d'addition, de soustraction, de multiplication, de division, de calcul d'une racine carrée, du calcul d'un reste et de comparaison;
3) les conversions entre nombres entiers et nombres en virgule flottante;
4) les conversions entre différents formats en virgule flottante;
5) les conversions entre les nombres en virgule flottante en format de base et les chaînes décimales, et
6) la détection et le traitement des conditions d'exception pour les nombres en virgule flottante y compris les non-nombres ("NaN").

1.3 *Exclusions*

Cette norme ne spécifie pas:

1) les formats des chaînes décimales et des entiers;
2) l'interprétation des champs de signe et de mantisse des non-nombres ("NaN"), ou
3) les conversions de binaire à décimal et réciproquement pour les formats étendus.

2. Définitions

Exposant avec excédent

Somme de l'exposant et d'une constante (excédent ou biais) choisie de manière à rendre non négatif le domaine de l'exposant avec excédent.

BINARY FLOATING-POINT ARITHMETIC FOR MICROPROCESSOR SYSTEMS

---

1. Scope

1.1 *Implementation objectives*

It is intended that an implementation of a floating-point system conforming to this standard can be realized entirely in software, entirely in hardware, or in any combination of software and hardware. It is the environment that the programmer or user of the system sees that conforms or fails to conform to this standard. Hardware components that require software support to conform shall not be said to conform apart from such software.

1.2 *Inclusions*

This standard specifies:

1) basic and extended floating-point number formats;
2) add, subtract, multiply, divide, square root, remainder and compare operations;
3) conversions between integer and floating-point numbers;
4) conversions between different floating-point formats;
5) conversions between basic format floating-point numbers and decimal strings, and
6) floating-point exceptions and their handling, including non-numbers (NaNs).

1.3 *Exclusions*

This standard does not specify:

1) formats of decimal strings and integers;
2) interpretation of the signs and significant fields of NaNs, or
3) binary ↔ decimal conversions to and from extended formats.

2. Definitions

Biased exponent

The sum of the exponent and a constant (bias) chosen to make the biased exponent's range non-negative.

Nombre binaire en virgule flottante

Chaîne de bits caractérisée par trois éléments: un signe, un exposant signé et une mantisse. Sa valeur numérique, si elle existe, est le produit signé de sa mantisse par deux élevé à la puissance de son exposant. Dans la présente norme, une chaîne de bits n'est pas toujours distinguée du nombre qu'elle représente.

Nombre dénormalisé

Nombre en virgule flottante non nul dont l'exposant a une valeur réservée, d'habitude la valeur minimale du format et dont le bit significatif de la mantisse, explicite ou implicite, est nul.

Destination

Emplacement devant contenir le résultat d'une opération binaire ou unaire. La destination peut être soit désignée explicitement par l'utilisateur ou fournie de manière implicite par le système (pour les résultats intermédiaires dans les sous-expressions ou les arguments de procédures par exemple). Certains langages placent les résultats des calculs intermédiaires dans des emplacements non accessibles par l'utilisateur. Néanmoins, cette norme définit le résultat d'une opération en termes du format de cette destination aussi bien que des valeurs des opérandes.

Exposant

Elément d'un nombre binaire en virgule flottante qui représente normalement la puissance entière à laquelle deux est élevé pour déterminer la valeur du nombre représenté. Occasionnellement, l'exposant est appelé exposant signé ou exposant sans excédent.

Partie fractionnaire

Partie de la mantisse située à droite de sa virgule correspondante.

Mode

Variable qu'un utilisateur peut positionner, tester, sauvegarder et restaurer pour diriger l'exécution des opérations arithmétiques ultérieures. Le mode par défaut est le mode valable tant qu'une instruction contraire explicite n'est pas incluse dans le programme ou sa spécification.

Les modes suivants doivent être mis en place:

1) arrondi, pour commander la direction des erreurs d'arrondi, et dans certaines réalisations;

2) précision de l'arrondi, pour diminuer la précision des résultats. Le réalisateur peut fournir optionnellement les modes suivants:

3) déroutements désactivés/activés, pour gérer les conditions d'exception.

Binary floating-point number

A bit-string characterized by three components: a sign, a signed exponent, and a significand. Its numerical value, if any, is the signed product of its significand and two raised to the power of its exponent. In this standard a bit-string is not always distinguished from a number it may represent.

Denormalized number

A nonzero floating-point number, the exponent of which has a reserved value, usually the format's minimum, and the explicit or implicit leading significant bit of which is zero.

Destination

The location for the result of a binary or unary operation. The destination may be either explicitly designated by the user or implicitly supplied by the system (e.g. intermediate results in sub-expressions or arguments for procedures). Some languages place the results of intermediate calculations in destinations beyond the user's control. Nonetheless, this standard defines the result of an operation in terms of that destination's format as well as the operands' values.

Exponent

The component of a binary floating-point number that normally signifies the integer power to which two is raised in determining the value of the represented number. Occasionally the exponent is called the signed or unbiased exponent.

Fraction

The field of the significand that lies to the right of its implied binary point.

Mode

A variable that a user may set, sense, save and restore, to control the execution of subsequent arithmetic operations. The default mode is the mode that a program can assume to be in effect unless an explicitly contrary statement is included either in the program or in its specification.

The following modes shall be implemented:

1) rounding, to control the direction of rounding errors, and in certain implementations.

2) rounding precision, to shorten the precision of results. The implementor may, at his option, implement the following modes:

3) traps disabled/enabled, to handle exceptions.

NaN (non-nombre)

Non-nombre; entité symbolique codée selon le format virgule flottante. Il existe deux types de non-nombres (voir 6.2). Les non-nombres indicateurs indiquent une condition d'exception concernant une opération invalide (voir 7.1) lorsqu'ils apparaissent en tant qu'opérandes. Les non-nombres muets se propagent à travers presque toutes les opérations arithmétiques sans signaler d'exceptions.

Résultat

Chaîne de bits (représentant généralement un nombre) qui est livrée à la destination.

Mantisse

Elément d'un nombre binaire en virgule flottante constituée d'un bit significatif explicite ou implicite placé à gauche de la virgule et d'un champ fractionnaire à droite de la virgule.

Doit

Le mot "doit" recouvre les parties obligatoires de toute réalisation conforme.

Il convient - Il est recommandé - Il y a lieu

Ces termes recouvrent les parties qui sont fortement recommandées comme étant dans l'esprit de cette norme, bien que des contraintes architecturales ou autres, hors du domaine de cette norme, puissent à l'occasion rendre ces recommandations peu pratiques.

Indicateur d'état

Variable qui peut prendre deux états, actif (valeur 1) ou inactif (valeur 0). Un utilisateur peut désactiver un indicateur, le copier, ou le remettre dans un état antérieur. Lorsqu'il est actif, un indicateur peut contenir des informations supplémentaires dépendant du système et éventuellement inaccessibles à certains utilisateurs. Les opérations définies par cette norme peuvent avoir comme effet secondaire l'activation de certains des indicateurs suivants: résultat inexact, dépassement de capacité inférieur, dépassement de capacité supérieur, division par zéro et opération invalide.

Utilisateur

Toute personne, tout matériel ou logiciel non spécifié lui-même dans cette norme, ayant accès aux opérations de l'environnement de programmation spécifiées dans cette norme et qui les commande.

3. Formats

Cette norme définit quatre formats de virgule flottante en deux groupes, de base et étendu, chacun admettant deux largeurs, simple et double précision. Les niveaux de réalisation standard se distinguent par les combinaisons de formats supportés.

NaN

Not a number; a symbolic entity encoded in floating-point format. There are two types of NaNs (see 6.2). Signalling NaNs signal the invalid operation exception (see 7.1) whenever they appear as operands. Quiet NaNs propagate through almost every arithmetic operation without signalling exceptions.

Result

The bit-string (usually representing a number) that is delivered to the destination.

Significant

The component of a binary floating-point number which consists of an explicit or implicit leading bit to the left of its implied binary point and a fraction field to the right.

Shall

The word "shall" signifies that which is obligatory in any conforming implementation.

Should

The word "should" signifies that which is strongly recommended as being in keeping with the intent of the standard, although architectural or other constraints beyond the scope of this standard may, on occasion, render the recommendations impractical.

Status flag

A variable that may take two states, set and clear. A user may clear a flag, copy it, or restore it to a previous state. When set, a status flag may contain additional system-dependent information, possibly inaccessible to some users. The operations of this standard may, as a side-effect, set some of the following flags: inexact result, underflow, overflow, divide by zero and invalid operation.

User

Any person, hardware, or program not itself specified by this standard, having access to and controlling those operations of the programming environment specified in this standard.

3. Formats

This standard defines four floating-point formats in two groups, basic and extended, each having two widths, single and double. The standard levels of implementation are distinguished by the combinations of formats supported.

3.1 *Ensembles de valeurs*

Ce paragraphe concerne seulement les valeurs numériques représentables dans un format, et non leur codage qui fait l'objet des paragraphes suivants. Les seules valeurs représentables dans un format donné sont celles qui sont spécifiées selon les trois paramètres entiers suivants:

P = nombre de bits significatifs (précision)

E_{max} = valeur maximale de l'exposant, et

E_{min} = valeur minimale de l'exposant

Les paramètres de chaque format sont regroupés dans le tableau 1. Pour chaque format, les seules entités qui doivent être fournies sont:

Nombres de la forme $(-1)^s 2^E (b_0 b_1 b_2 \ldots b_{p-1})$

où:

s vaut 0 ou 1;
E est un entier compris entre E_{min} et E_{max} bornes incluses, et
chaque b_i vaut 0 ou 1.

Deux valeurs infinies, $+\infty$ et $-\infty$;
au moins un non-nombre indicateur, et
au moins un non-nombre muet.

Tableau 1 - Résumé des paramètres du format

Paramètre	Format			
	Simple	Simple Etendu	Double	Double Etendu
P	24	≥32	53	≥64
E_{max}	+127	≥+1 023	+1 023	≥+16 383
E_{min}	-126	≤-1 022	-1 022	≤-16 382
Excédent de l'exposant	+127	Non spécifié	+1 023	Non spécifié
Largeur de l'exposant (bits)	8	≥11	11	≥15
Largeur du format (bits)	32	≥43	64	≥79

La description précédente énumère certaines valeurs de manière redondante, par exemple:

$$2^0(1.0) = 2^1(0.1) = 2^2(0.01) = \ldots$$

Cependant, le codage de telles valeurs non nulles peut être redondant seulement pour les formats étendus (voir 3.3). Les valeurs non nulles de la forme $\pm 2^{E_{min}} (0 \cdot b_1 b_2 \ldots b_{p-1})$ sont appelées

3.1 *Sets of values*

This sub-clause concerns only the numerical values representable within a format, not the encodings which are the subject of the following sub-clauses. The only values representable in a chosen format are those specified via the following three integer parameters:

P = number of significant bits (precision)

E_{max} = maximum exponent, and

E_{min} = minimum exponent

Each format's parameters are displayed in Table 1. Within each format just the following entities shall be provided:

Numbers of the form $(-1)^s 2^E (b_0 b_1 b_2 \ldots b_{p-1})$

where:

s is 0 or 1;
E is any integer between E_{min} and E_{max} inclusive, and each b_i is 0 or 1.

Two infinities, $+\infty$ and $-\infty$;
at least one signalling NaN, and
at least one quiet NaN.

Table 1 - Summary of format parameters

Parameter	Format			
	Single	Single Extended	Double	Double Extended
P	24	≥32	53	≥64
E_{max}	+127	≥+1 023	+1 023	≥+16 383
E_{min}	-126	≤-1 022	-1 022	≤-16 382
Exponent bias	+127	Unspecified	+1 023	Unspecified
Exponent width (bits)	8	≥11	11	≥15
Format width (bits)	32	≥43	64	≥79

The foregoing description enumerates some values redundantly, for example:

$$2^0(1.0) = 2^1(0.1) = 2^2(0.01) = \ldots$$

However, the encodings of such nonzero values may be redundant only in extended formats (see 3.3). The nonzero values of the form $\pm 2^{E_{min}} (0 \cdot b_1 b_2 \ldots b_{p-1})$ are called denormalized. Reserved exponents

"dénormalisées". Des valeurs réservées d'exposants peuvent être utilisées pour coder les non-nombres, ±∞, ±0, et les nombres dénormalisés. Pour toute variable ayant la valeur zéro, le bit de signe *s* fournit un bit supplémentaire d'information. Bien que tous les formats aient des représentations distinctes pour +0 et -0, les signes sont significatifs en certaines circonstances, comme la division par zéro, et non dans d'autres. Dans cette norme, 0 et ∞ sont écrits sans signe lorsque ce dernier n'a pas d'importance.

3.2 *Formats de base*

Les nombres en format simple et double précision se composent de trois champs:

un signe *s* de 1 bit,

un exposant avec excédent e = E + excédent, et

une partie fractionnaire $f = .b_1 b_2 \ldots b_{p-1}$.

Le domaine des valeurs de l'exposant sans excédent E doit inclure tout entier placé entre les bornes E_{min} et E_{max} incluses, ainsi que deux autres valeurs réservées: $E_{min}-1$ pour coder ±0 et les nombres dénormalisés, et $E_{max}+1$ pour coder ±∞ et les non-nombres. Les paramètres ci-dessus apparaissent dans le tableau I. Chaque valeur numérique non nulle possède un codage unique. Les champs sont interprétés comme suit:

3.2.1 *Simple précision*

Un nombre *X* de 32 bits en format simple précision est constitué comme indiqué sur la figure 1. La valeur ν de *X* se déduit de ses champs constitutifs soit:

1) Si $e = 255$ et $f \neq 0$, alors ν est un non-nombre quel que soit *s*
2) Si $e = 255$ et $f = 0$, alors $\nu = (-1)^s \infty$
3) Si $0 < e < 255$, alors $\nu = (-1)^s \, 2^{e-127} \, (1,f)$
4) Si $e = 0$ et $f \neq 0$, alors $\nu = (-1)^s \, 2^{-126} \, (0,f)$ (nombres dénormalisés)
5) Si $e = 0$ et $f = 0$, alors $\nu = (-1)^s \, 0$ (zéro)

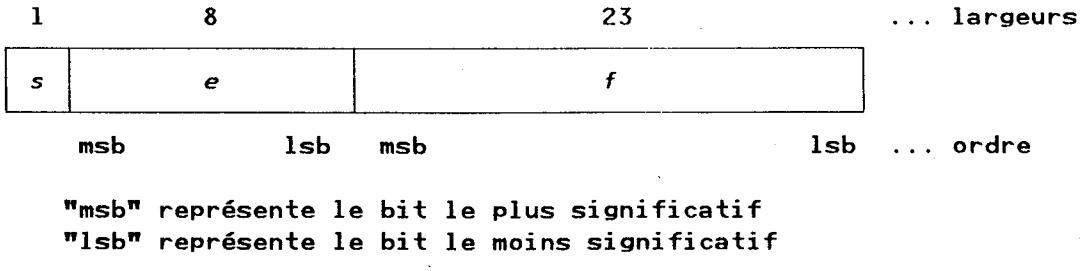

Figure 1 - Format simple précision

may be used to encode NaNs, ±∞, ±0, and denormalized numbers. For any variable that has the value zero, the sign bit *s* provides an extra bit of information. Although all formats have distinct representations for +0, and -0, the signs are significant in some circumstances, like division by zero, and not in others. In this standard 0 and ∞ are written without a sign when the sign does not matter.

3.2 *Basic formats*

Numbers in the single and double formats are composed of three fields:

a 1-bit sign *s*,

a biased exponent e = E + bias, and

a fraction $f = .b_1 b_2 \ldots b_{p-1}$.

The range of the unbiased exponent E shall include every integer between two values E_{min} and E_{max} inclusive, and also two other reserved values: $E_{min}-1$ to encode ±0 and denormalized numbers, and $E_{max}+1$ to encode ±∞ and NaNs. The foregoing parameters appear in Table 1. Each nonzero numerical value has just one encoding. The fields are interpreted as follows:

3.2.1 *Single*

A 32-bit single format number *X* is divided as shown in Figure 1. The value v of *X* is inferred from its constituent fields thus:

1) If $e = 255$ and $f \neq 0$, then v is a NaN regardless of *s*
2) If $e = 255$ and $f = 0$, then $v = (-1)^s \infty$
3) If $0 < e < 255$, then $v = (-1)^s \, 2^{e-127} \, (1.f)$
4) If $e = 0$ and $f \neq 0$, then $v = (-1)^s \, 2^{-126} \, (0.f)$ (denormalized numbers)
5) If $e = 0$ and $f = 0$, then $v = (-1)^s \, 0$ (zero)

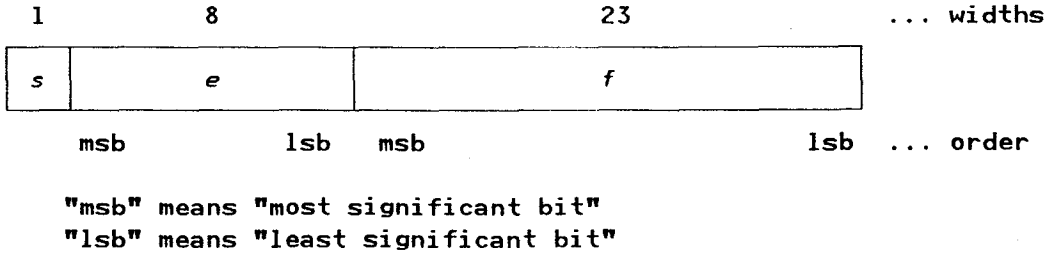

Figure 1 - Single format

3.2.2 *Double précision*

Un nombre X de 64 bits en format double précision est constitué comme indiqué sur la figure 2. La valeur v de X se déduit de ses champs constitutifs, soit:

1) Si $e = 2\,047$ et $f \neq 0$, alors v est un non-nombre quel que soit s
2) Si $e = 2\,047$ et $f = 0$, alors $v = (-1)^s \infty$
3) Si $0 < e < 2\,047$, alors $v = (-1)^s \, 2^{e-1\,023} \, (1,f)$
4) Si $e = 0$ et $f \neq 0$, alors $v = (-1)^s \, 2^{-1\,022} \, (0,f)$ (nombres dénormalisés)
5) Si $e = 0$ et $f = 0$, alors $v = (-1)^s \, 0$ (zéro)

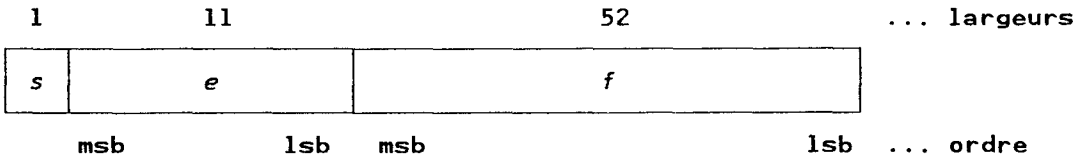

Figure 2 - Format double précision

3.3 *Formats étendus*

Les formats étendus simple précision et double précision assurent le codage de manière dépendante de la réalisation pour les ensembles de valeurs de 3.1, assujetties aux contraintes du tableau 1. Cette norme permet à une réalisation de coder certaines valeurs de manière redondante pourvu que cette redondance soit transparente à l'utilisateur selon le sens suivant: soit une réalisation doit coder chaque valeur non nulle de manière unique, soit elle ne doit faire aucune distinction entre les codages redondants des valeurs non nulles. Une réalisation peut également réserver quelques chaînes de bits pour des utilisations en dehors du domaine de cette norme; lorsqu'une telle chaîne de bits réservée est utilisée comme opérande, le résultat n'est pas spécifié par cette norme.

Une réalisation de cette norme n'est pas tenue de faire en sorte (et il y a lieu que l'utilisateur ne présume pas) que le format étendu simple précision possède un domaine plus grand que le format étendu double précision.

3.4 *Combinaisons de formats*

Toutes les réalisations conformes à cette norme doivent supporter le format simple précision. Il convient que les réalisations supportent le format étendu correspondant au format de base le plus grand qu'elles supportent et elles ne sont pas tenues de supporter d'autres formats étendus.*

---

* C'est seulement au cas ou la compatibilité ascendante et la vitesse de calcul sont importantes qu'il convient qu'un système supportant le format étendu double précision supporte aussi le format étendu simple précision.

3.2.2 *Double*

A 64-bit double format number X is divided as shown in Figure 2. The value v of X is inferred from its constituent fields thus:

1) If $e = 2\ 047$ and $f \neq 0$, then v is a NaN regardless of s
2) If $e = 2\ 047$ and $f = 0$, then $v = (-1)^s \infty$
3) If $0 < e < 2\ 047$, then $v = (-1)^s\ 2^{e-1\ 023}\ (1.f)$
4) If $e = 0$ and $f \neq 0$, then $v = (-1)^s\ 2^{-1\ 022}\ (0.f)$ (denormalized numbers)
5) If $e = 0$ and $f = 0$, then $v = (-1)^s\ 0$ (zero)

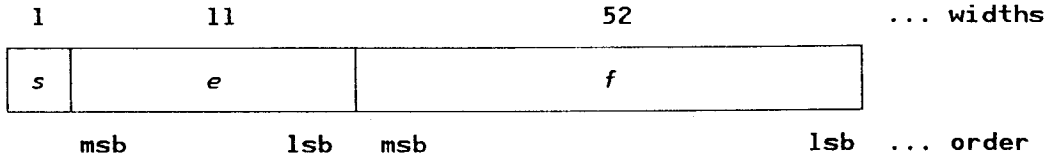

Figure 2 - Double format

3.3 *Extended formats*

The single extended and double extended formats encode in an implementation-dependent way the sets of values in 3.1 subject to the constraints of Table 1. This standard allows an implementation to encode some values redundantly, provided that redundancy is transparent to the user in the following sense: an implementation shall either encode every nonzero value uniquely or not distinguish redundant encodings of nonzero values. An implementation may also reserve some bit strings for purposes beyond the scope of this standard; when such a reserved bit string occurs as an operand the result is not specified by this standard.

An implementation of this standard is not required to provide (and the user should not assume) that single extended formats have greater range than double extended formats.

3.4 *Combinations of formats*

All implementations conforming to this standard shall support the single format. Implementations should support the extended format corresponding to the widest basic format supported, and need not support any other extended format.*

---

* Only if upward compatibility and speed are important issues should a system supporting the double extended format also support the single extended format.

4. Arrondi

L'arrondi considère un nombre comme infiniment précis et, si nécessaire, le modifie pour l'adapter au format de la destination tout en signalant l'exception d'inexactitude (voir 7.5). Sauf pour la conversion binaire-décimale (dont les conditions plus faibles sont spécifiées en 5.6), chaque opération spécifiée dans l'article 5 doit être effectuée comme si elle produisait tout d'abord un résultat intermédiaire correct avec une précision infinie et avec un domaine illimité, puis exécutait une opération d'arrondi sur ce résultat selon un des modes de cet article.

Les modes d'arrondi affectent toutes les opérations arithmétiques, sauf la comparaison et le calcul du reste. Les modes d'arrondi peuvent affecter les signes des sommes de valeur nulle (voir 6.3) et affectent les seuils au-delà desquels un dépassement (voir 7.3) et un dépassement inférieur (voir 7.4) peuvent être signalés.

4.1 *Arrondi au plus près*

Une réalisation de cette norme doit fournir un arrondi optimal (au plus près) comme mode d'arrondi par défaut. Dans ce mode, elle doit fournir la valeur représentable la plus proche du résultat théorique infiniment précis; si les deux valeurs représentables les plus proches sont à égale distance du résultat théorique, la valeur fournie sera celle dont le bit de poids faible est égal à zéro. Cependant, un résultat infiniment précis avec une valeur au moins de $2^{E_{max}}(2 - 2^{-P})$ doit être arrondi à ∞ sans changement de signe; dans ce cas, E_{max} et P sont déterminés par le format de destination (article 3), à moins que ce dernier ne soit annulé par un mode de précision d'arrondi (voir 4.3).

4.2 *Arrondis orientés*

Une réalisation doit aussi fournir trois modes d'arrondi orientés au choix de l'utilisateur: arrondi vers $+\infty$, arrondi vers $-\infty$ et arrondi vers 0.

Lors de l'arrondi vers $+\infty$, le résultat doit être la valeur du format (éventuellement $+\infty$) la plus proche du résultat théorique de précision infinie et non inférieure à ce résultat théorique. Lors de l'arrondi vers $-\infty$, le résultat doit être la valeur du format (éventuellement $-\infty$) la plus proche du résultat théorique de précision infinie et non supérieure à ce résultat théorique.

Lors de l'arrondi vers 0, le résultat doit être la valeur du format la plus proche du résultat théorique de précision infinie, et non supérieure en valeur absolue à ce résultat théorique.

4.3 *Précision d'arrondi*

En général, un résultat est arrondi avec la précision correspondant au format de sa destination. Cependant, certains systèmes donnent des résultats en format double précision ou en format étendu. Sur un tel système, l'utilisateur, qui peut être un compilateur de langage évolué, doit pouvoir spécifier qu'un résultat doit être arrondi en simple précision, même lorsqu'il est mémorisé en format double précision ou

4. Rounding

Rounding takes a number regarded as infinitely precise and, if necessary, modifies it to fit in the destination's format while signalling the inexact exception (see 7.5). Except for binary ↔ decimal conversion (the weaker conditions of which are specified in 5.6), every operation specified in clause 5 shall be performed as if it first produced an intermediate result correct to infinite precision and with unbounded range, and then rounded that result according to one of the modes in this clause.

The rounding modes affect all arithmetic operations except comparison and remainder. The rounding modes may affect the signs of zero sums (see 6.3), and do affect the threshold beyond which overflow (see 7.3) and underflow (see 7.4) may be signalled.

4.1 *Round to nearest*

An implementation of this standard shall provide round to nearest as the default rounding mode. In this mode, the representable value nearest to the infinitely precise result shall be delivered; if the two nearest representable values are equally near, the one with its least significant bit equal to zero shall be delivered. However, an infinitely precise result with magnitude at least $2^{E_{max}}(2 - 2^{-P})$ shall round to ∞ with no change in sign; here E_{max} and P are determined by the destination format (clause 3) unless overridden by a rounding precision mode (see 4.3).

4.2 *Directed roundings*

An implementation shall also provide three user-selectable directed rounding modes: round toward $+\infty$, round toward $-\infty$, and round toward 0.

When rounding toward $+\infty$, the result shall be the format's value (possibly $+\infty$) closest to and no less than the infinitely precise result. When rounding toward $-\infty$, the result shall be the format's value (possibly $-\infty$) closest to and no greater than the infinitely precise result.

When rounding toward 0, the result shall be the format's value closest to and no greater in magnitude than the infinitely precise result.

4.3 *Rounding precision*

Normally a result is rounded to the precision of its destination. However, some systems deliver results only to double or extended destinations. On such a system the user, which may be a high-level language compiler, shall be able to specify that a result be rounded instead to single precision, though it may be stored in the double or

étendu avec un exposant ayant un domaine plus étendu*. De la même manière, un système qui fournit des résultats seulement sous le format étendu double précision doit permettre à l'utilisateur de spécifier l'arrondi en simple ou double précision. Il faut remarquer que pour satisfaire aux spécifications de 4.1, le résultat ne peut pas être entaché de plus d'une erreur d'arrondi.

5. Opérations

Toutes les réalisations conformes à cette norme doivent fournir les opérations suivantes: addition, soustraction, multiplication, division, extraction de racine carrée, calcul du reste, arrondi vers entier en format virgule flottante, conversion entre différents formats de virgule flottante, conversion entre formats virgule flottante et entiers, conversion binaire-décimale et comparaisons. Le fait de considérer la copie sans changement de format comme une opération est une option de la réalisation. Excepté pour la conversion binaire-décimale, chacune des opérations doit être effectuée comme si elle produisait tout d'abord un résultat intermédiaire correct avec une précision infinie et un domaine de valeurs illimité, puis contraignait ce résultat intermédiaire à s'adapter au format de destination (articles 4 et 7). L'article 6 étend les spécifications suivantes de manière à couvrir ±0, ±∞ et les non-nombres; l'article 7 énumère les cas d'exception causés par des opérandes et des résultats hors gamme.

5.1 *Arithmétique*

Une réalisation doit offrir les opérations d'addition, de soustraction, de multiplication, de division et de calcul du reste pour n'importe quel couple d'opérandes de même format, et ce pour chaque format qui est supporté. Il y a lieu qu'elle offre aussi ces opérations pour des opérandes de format différents. Le format de destination (sans tenir compte de la commande de la précision d'arrondi de 4.3) doit être au moins aussi étendu que le format le plus étendu des opérandes. Tous les résultats doivent être arrondis comme spécifié dans l'article 4.

Lorsque $y \neq 0$, le reste $\gamma = x$ REM y est défini quel que soit le mode d'arrondi par la relation mathématique $\gamma = x - y \times n$, où n est l'entier le plus proche de la valeur exacte x/y; lorsque $|n - x/y| = \frac{1}{2}$, alors n est pair. Ainsi, le reste est toujours exact. Si $\gamma = 0$, son signe doit être celui de x. La commande de précision (voir 4.3) ne doit pas s'appliquer à l'opération de calcul du reste.

---

* La commande de la précision d'arrondi a pour but de permettre à des systèmes pour lesquels les résultats sont toujours en format double précision ou étendu de simuler, en l'absence de dépassement/dépassement inférieur, la précision des systèmes opérant en simple ou double précision. Il ne convient pas qu'une réalisation fournisse des opérations qui combinent des opérandes de format double précision ou étendu pour produire un résultat en simple précision, ni des opérations qui combinent des opérandes de format étendu double précision pour produire un résultat en double précision, avec seulement une opération d'arrondi.

extended format with its wider exponent range.* Similarly, a system that delivers results only to double extended destinations shall permit the user to specify rounding to single or double precision. Note that to meet the specifications in 4.1, the result cannot suffer more than one rounding error.

5. Operations

All conforming implementations of this standard shall provide operations to add, subtract, multiply, divide, extract the square root, find the remainder, round to integer in floating-point format, convert between different floating-point formats, convert between floating-point and integer formats, convert binary ↔ decimal, and compare. Whether copying without change of format is considered as an operation is an implementation option. Except for binary ↔ decimal conversion, each of the operations shall be performed as if it first produced an intermediate result correct to infinite precision and with unbounded range, and then coerced this intermediate result to fit in the destination's format (clauses 4 and 7). Clause 6 augments the following specifications to cover ±0, ±∞, and NaNs; clause 7 enumerates exceptions caused by exceptional operands and exceptional results.

5.1 *Arithmetic*

An implementation shall provide the add, subtract, multiply, divide, and remainder operations for any two operands of the same format, for each supported format; it should also provide the operations for operands of differing formats. The destination format (regardless of the rounding precision control of 4.3) shall be at least as wide as the wider operand's format. All results shall be rounded as specified in clause 4.

When $y \neq 0$, the remainder $r = x$ REM y is defined regardless of the rounding mode by the mathematical relation $r = x - y \times n$, where n is the integer nearest the exact value x/y; whenever $|n - x/y| = \frac{1}{2}$, then n is even. Thus, the remainder is always exact. If $r = 0$, its sign shall be that of x. Precision control (see 4.3) shall not apply to the remainder operation.

---

* Control of rounding precision is intended to allow systems the destinations of which are always double or extended to mimic, in the absence of over/underflow, the precisions of systems with single and double destinations. An implementation should not provide operations that combine double or extended operands to produce a single result, nor operations that combine double extended operands to produce a double result, with just one rounding.

5.2 *Racine carrée*

L'opération d'extraction de la racine carrée doit être offerte pour tous les formats supportés. Le résultat est défini et possède un signe positif pour tous les opérandes ≥0 à l'exception près que $\sqrt{-0}$ doit être égal à -0. Le format de destination doit être au moins aussi étendu que celui des opérandes. Le résultat doit être arrondi comme spécifié dans l'article 4.

5.3 *Conversions des formats virgule flottante*

Il doit être possible de convertir des nombres en virgule flottante entre tous les formats supportés. Si le résultat de la conversion a une précision moindre, le résultat doit être arrondi comme spécifié dans l'article 4. La conversion vers un format de précision supérieure, doit être exacte. Il n'y a pas d'exception.

5.4 *Conversion entre virgule flottante et entier*

Il doit être possible d'effectuer des conversions entre tous les formats en virgule flottante supportés. La conversion vers des entiers doit s'effectuer par arrondi comme spécifié dans l'article 4. Les conversions entre formats de nombres entiers en virgule flottante et formats de nombres entiers doit être exacte, à moins qu'une exception ne se produise comme il est spécifié en 7.1.

5.5 *Arrondi de nombres en virgule flottante vers une valeur entière*

Il doit être possible d'effectuer une opération d'arrondi sur un nombre en virgule flottante pour obtenir une valeur entière dans le même format virgule flottante. L'arrondi doit être conforme à la spécification de l'article 4, avec la condition que lors d'un arrondi au plus près, si la différence entre l'opérande non arrondi et le résultat arrondi est exactement un demi, le résultat arrondi est pair.

5.6 *Conversion binaire-décimale*

La conversion entre des chaînes décimales dans un format au moins et des nombres binaires en virgule flottante dans tous les formats de base supportés doit être offerte pour tous les nombres couvrant les domaines spécifiés dans le tableau 2. Les entiers M et N des tableaux 2 et 3 sont tels que les chaînes décimales ont pour valeur $\pm M \times 10^{\pm N}$. En entrée, les zéros de queue doivent être ajoutés ou ôtés de M (dans les limites spécifiées dans le tableau 2) de manière à minimiser N. Lorsque la destination est une chaîne décimale, il y a lieu que son digit le moins significatif soit positionné selon les spécifications du format concernant l'arrondi.

Lorsque l'entier M a une valeur située en dehors du domaine spécifié dans les tableaux 2 et 3, c'est-à-dire lorsque $M \geq 10^9$ pour la simple précision et 10^{17} pour la double précision, le réalisateur peut, en option altérer tous les digits significatifs situés après le neuvième en simple précision et après le dix-septième en double précision en leur donnant d'autres valeurs, en général 0.

5.2 *Square root*

The square root operation shall be provided in all supported formats. The result is defined and has a positive sign for all operands ≥ 0, except that $\sqrt{-0}$ shall be -0. The destination format shall be at least as wide as the operand's. The result shall be rounded as specified in clause 4.

5.3 *Floating-point format conversions*

It shall be possible to convert floating-point numbers between all supported formats. If the conversion is to a narrower precision, the result shall be rounded as specified in clause 4. Conversion to a wider precision shall be exact. There is no exception.

5.4 *Conversions between floating-point and integer*

It shall be possible to convert between all supported floating-point formats and all supported integer formats. Conversion to integer shall be effected by rounding as specified in clause 4. Conversions between floating-point integers and integer formats shall be exact unless an exception arises as specified in 7.1.

5.5 *Round floating-point number to integral value*

It shall be possible to round a floating-point number to an integral valued floating-point number in the same format. The rounding shall be as specified in clause 4, with the understanding that when rounding to nearest, if the difference between the unrounded operand and the rounded result is exactly one half, the rounded result is even.

5.6 *Binary ↔ decimal conversion*

Conversion between decimal strings in at least one format, and binary floating-point numbers in all supported basic formats, shall be provided for numbers throughout the ranges specified in Table 2. The integers M and N in Tables 2 and 3 are such that the decimal strings have values $\pm M \times 10^{\pm N}$. On input, trailing zeros shall be appended to or stripped from M (up to the limits specified in Table 2) in order to minimize N. When the destination is a decimal string, its least significant digit should be located by format specifications for purposes of rounding.

When the integer M lies outside the range specified in Tables 2 and 3, i.e. when $M \geq 10^9$ for single or 10^{17} for double, the implementor may, at his option, alter all significant digits after the ninth for single and seventeenth for double, to other decimal digits, typically 0.

Les conversions doivent être arrondies correctement, comme spécifié dans l'article 4 pour les opérandes inclus dans les domaines spécifiés dans le tableau 3. Sinon, dans le cas d'arrondi au plus près, l'erreur du résultat converti ne doit pas excéder de plus de 0,47 unités du digit le moins significatif du résultat l'erreur qui résulterait de l'application des spécifications d'arrondi de l'article 4, pour autant qu'il n'y ait aucun dépassement de capacité concernant l'exposant. Pour les modes d'arrondi orientés, l'erreur doit avoir le signe correct et ne doit pas dépasser 1,47 unités de plus faible poids.

Les conversions doivent être monotones. C'est-à-dire que l'augmentation de la valeur d'un nombre binaire exprimé en virgule flottante ne doit pas se traduire par une diminution de sa valeur après conversion en une chaîne décimale; et une augmentation de la valeur d'une chaîne décimale ne doit pas se traduire par une diminution de sa valeur après conversion en un nombre binaire en virgule flottante.

Lorsqu'une opération d'arrondi au plus près est effectuée, la conversion de binaire à décimal et réciproquement doit être l'identité tant que la chaîne décimale possède la précision maximale spécifiée dans le tableau 2, c'est-à-dire 9 digits en simple précision et 17 digits en double précision*.

Si la conversion décimale-binaire donne lieu à un dépassement supérieur ou inférieur, la réponse sera comme spécifié dans l'article 7. Il convient que les conditions de dépassement supérieur et inférieur, les non-nombres et les valeurs infinies apparaissant lors des conversions binaire à décimal soient signalées à l'utilisateur par des chaînes appropriées. Cette norme ne donne aucune information à propos du traitement des non-nombres codés sous forme de chaînes décimales.

Pour éviter des incohérences, il convient que les procédures utilisées pour la conversion binaire-décimale fournissent les mêmes résultats, que la conversion soit effectuée lors de la traduction du langage (interprétation, compilation ou assemblage) ou lors de l'exécution du programme (exécution et entrée/sortie interactive).

Tableau 2 - Domaine de conversion décimale

Format	Décimal-binaire		Binaire-décimal	
	$M_{max.}$	$N_{max.}$	$M_{max.}$	$N_{max.}$
Simple précision	$10^9 - 1$	99	$10^9 - 1$	53
Double précision	$10^{17} - 1$	999	$10^{17} - 1$	340

* Les propriétés spécifiées pour les conversions résultent des limites d'erreur qui dépendent du format (simple ou double précision) et du nombre de chiffres décimaux qui interviennent; la valeur 0,47 mentionnée correspond à la limite pour le pire des cas. Pour une discussion détaillée de ces bornes d'erreur et des algorithmes de conversion économiques qui utilisent le format étendu, se référer à "Accurate Yet Economical Binary ↔ Decimal conversions" par Jerome T. Coonen (Ph.D. Dissertation, Université de Californie, Berkeley).

Conversions shall be correctly rounded as specified in clause 4 for operands lying within the ranges specified in Table 3. Otherwise, for rounding to nearest, the error in the converted result shall not exceed by more than 0.47 units in the destination's least significant digit the error that would be incurred by the rounding specifications of clause 4, provided that exponent over/underflow does not occur. In the directed rounding modes the error shall have the correct sign and shall not exceed 1.47 units in the last place.

Conversions shall be monotonic. That is, increasing the value of a binary floating-point number shall not decrease its value when converted to a decimal string; and increasing the value of a decimal string shall not decrease its value when converted to a binary floating-point number.

When rounding to the nearest, conversion from binary to decimal and back to binary shall be the identity as long as the decimal string is carried to the maximum precision specified in Table 2, namely, 9 digits for single and 17 for double.*

If decimal to binary conversion over/underflows, the response is as specified in clause 7. Over/underflow and NaNs and infinities encountered during binary to decimal conversion should be indicated to the user by appropriate strings. This standard says nothing about dealing with NaNs encoded in decimal strings.

To avoid inconsistencies, the procedures used for binary ↔ decimal conversion should give the same results regardless of whether the conversion is performed during language translation (interpretation, compilation or assembly) or during program execution (run-time and interactive input/output).

Table 2 - Decimal conversion ranges

Format	Decimal to binary		Binary to decimal	
	Max. *M*	Max. *N*	Max. *M*	Max. *N*
Single	$10^9 - 1$	99	$10^9 - 1$	53
Double	$10^{17} - 1$	999	$10^{17} - 1$	340

* The properties specified for conversions are implied by error bounds that depend on the format (single or double) and the number of decimal digits involved; the 0.47 mentioned is a worst-case bound only. For a detailed discussion of these error bounds and economical conversion algorithms that exploit the extended format, see "Accurate Yet Economical Binary ↔ Decimal Conversions" by Jerome T. Coonen (Ph.D. Dissertation, University of California, Berkeley).

Tableau 3 - Domaine d'arrondi exact pour la conversion décimale

Format	Décimal-binaire		Binaire-décimal	
	$M_{max.}$	$N_{max.}$	$M_{max.}$	$N_{max.}$
Simple précision	$10^9 - 1$	13	$10^9 - 1$	13
Double précision	$10^{17} - 1$	27	$10^{17} - 1$	27

5.7 *Comparaison*

Il doit être possible de comparer des nombres en virgule flottante dans tous les formats supportés, même si les formats des opérandes sont différents. Les comparaisons sont exactes et ne créent jamais de dépassement de capacité. Il existe quatre relations possibles, qui s'excluent mutuellement: "plus petit que", "égal", "plus grand que" et "non ordonné". Ce dernier cas survient lorsqu'un des opérandes au moins est un non-nombre. Chaque non-nombre doit donner "non ordonné" comme résultat de la comparaison avec tout élément y compris lui-même. Les comparaisons doivent ignorer le signe de zéro (ainsi +0 = -0).

Le résultat d'une comparaison doit être fourni selon l'une des deux manières suivantes: soit comme un code condition identifiant l'une des quatre relations indiquées ci-dessus, ou comme une réponse binaire (vrai-faux) à un prédicat qui identifie la comparaison spécifique désirée. En supplément à la réponse vrai-faux, une condition d'exception d'opération invalide (voir 7.1) doit être signalée quand, comme indiqué dans la dernière colonne du tableau 4, des opérandes "non ordonnés" sont comparés en utilisant un des prédicats mettant en oeuvre "<" ou ">" mais pas "?". Le symbole "?" signifie ici "non ordonné".

Le tableau 4 montre les vingt-six prédicats fonctionnellement distincts identifiés, dans la première colonne, au moyen de trois notations: ad hoc, FORTRAN et mathématique. Il montre comment ils sont obtenus à partir des quatre codes conditions et indique quels prédicats sont la cause d'une exception d'opération invalide lorsque la relation est "non ordonné". Les entrées V et F indiquent si le prédicat est vrai ou faux lorsque la relation correspondante est vérifiée.

Table 3 - Correctly rounded decimal conversion range

Format	Decimal to binary		Binary to decimal	
	Max. *M*	Max. *N*	Max. *M*	Max. *N*
Single	$10^9 - 1$	13	$10^9 - 1$	13
Double	$10^{17} - 1$	27	$10^{17} - 1$	27

5.7 *Comparison*

It shall be possible to compare floating-point numbers in all supported formats, even if the operands' formats differ. Comparisons are exact and never overflow nor underflow. Four mutually exclusive relations are possible: "less than," "equal," "greater than," and "unordered". The last case arises when at least one operand is NaN. Every NaN shall compare "unordered" with everything, including itself. Comparisons shall ignore the sign of zero (so +0 = -0).

The result of a comparison shall be delivered in one of the two following ways: either as a condition code identifying one of the four relations listed above, or as a true-false response to a predicate that names the specific comparison desired. In addition to the true-false response, an invalid operation exception (see 7.1) shall be signalled when, as indicated in the last column of Table 4, "unordered" operands are compared, using one of the predicates involving "<" or ">" but not "?". (Here the symbol "?" signifies "unordered".)

Table 4 exhibits the twenty-six functionally distinct useful predicates named, using three notations in the first column: ad hoc, FORTRAN-like, and mathematical. It shows how they are obtained from the four condition codes and tells which predicates cause an invalid operation exception when the relation is "unordered". The entries T and F indicate whether the predicate is true or false when the respective relation holds.

Tableau 4 - Prédicats et relations

Prédicats			Relations				Exceptions
Ad hoc	FORTRAN	Math	Plus grand que	Plus petit que	Egal	Non ordonné	Invalide si non ordonné
=	.EQ.	=	F	F	V	F	Non
?<>	.NE.	≠	V	V	F	V	Non
>	.GT.	>	V	F	F	F	Oui
>=	.GE.	≥	V	F	V	F	Oui
<	.LT.	<	F	V	F	F	Oui
<=	.LE.	≤	F	V	V	F	Oui
?	non ordonné		F	F	F	V	Non
							Oui
<>	.LG.		V	V	F	F	
<=>	.LEG.		V	V	V	F	Oui
?>	.UG.		V	F	F	V	Non
?>=	.UGE.		V	F	V	V	Non
?<	.UL.		F	V	F	V	Non
?<=	.ULE.		F	V	V	V	Non
?=	.UE.		F	F	V	V	Non
NON(>)			F	V	V	V	Oui
NON(>=)			F	V	F	V	Oui
NON(<)			V	F	V	V	Oui
NON(<=)			V	F	F	V	Oui
NON(?)			V	V	V	F	Non
NON(<>)			F	F	V	V	Oui
NON(<=>)			F	F	F	V	Oui
NON(?>)			F	V	V	F	Non
NON(?>=)			F	V	F	F	Non
NON(?<)			V	F	V	F	Non
NON(?<=)			V	F	F	F	Non
NON(?=)			V	V	F	F	Non

Remarquer que les prédicats s'associent par paires, chacun étant la négation logique de l'autre; l'application d'un préfixe comme "NON" pour effectuer la négation d'un prédicat dans le tableau 4 échange le sens des valeurs vrai et faux des entrées associées, mais laisse la dernière colonne inchangée*.

Les réalisations qui offrent des prédicats doivent fournir les 6 premiers prédicats du tableau 4 et il convient qu'elles fournissent aussi le septième, ainsi que le moyen d'effectuer la négation logique des prédicats.

---

* Il peut sembler qu'il existe deux façons d'écrire la négation logique d'un prédicat, l'une utilisant "NON" de manière explicite et l'autre inversant l'opérateur relationnel. Par exemple, la négation logique de (X=Y) peut s'écrire soit NON(X=Y) ou (X?<>Y); dans ce cas, les deux expressions sont fonctionnellement équivalentes à (X≠Y). Cependant, cette coïncidence ne se produit pas pour les autres prédicats. Par exemple, la négation logique de (X<Y) est seulement NON(X<Y); le prédicat inverse (X?>=Y) est différent en ce sens qu'il ne signale pas d'exception d'opération invalide lorsque X et Y sont "non ordonnés".

Table 4 - Predicates and relations

Predicates			Relations				Exception
Ad hoc	FORTRAN	Math	Greater than	Less than	Equal	Unordered	Invalid if unordered
=	.EQ.	=	F	F	T	F	No
?<>	.NE.	≠	T	T	F	T	No
>	.GT.	>	T	F	F	F	Yes
>=	.GE.	≥	T	F	T	F	Yes
<	.LT.	<	F	T	F	F	Yes
<=	.LE.	≤	F	T	T	F	Yes
?	unordered		F	F	F	T	No
							Yes
<>	.LG.		T	T	F	F	
<=>	.LEG.		T	T	T	F	Yes
?>	.UG.		T	F	F	T	No
?>=	.UGE.		T	F	T	T	No
?<	.UL.		F	T	F	T	No
?<=	.ULE.		F	T	T	T	No
?=	.UE.		F	F	T	T	No
NOT(>)			F	T	T	T	Yes
NOT(>=)			F	T	F	T	Yes
NOT(<)			T	F	T	T	Yes
NOT(<=)			T	F	F	T	Yes
NOT(?)			T	T	T	F	No
NOT(<>)			F	F	T	T	Yes
NOT(<=>)			F	F	F	T	Yes
NOT(?>)			F	T	T	F	No
NOT(?>=)			F	T	F	F	No
NOT(?<)			T	F	T	F	No
NOT(?<=)			T	F	F	F	No
NOT(?=)			T	T	F	F	No

Note that predicates come in pairs, each a logical negation of the other; applying a prefix like "NOT" to negate a predicate in Table 4 reverses the true/false sense of its associated entries, but leaves the last column's entry unchanged.*

Implementations that provide predicates shall provide the first six predicates in Table 4 and should provide the seventh, as well as a means of logically negating predicates.

---

* There may appear to be two ways to write the logical negation of a predicate, one using "NOT" explicitly and the other reversing the relational operator. For example, the logical negation of (X=Y) may be written either NOT(X=Y) or (X?<>Y); in this case, both expressions are functionally equivalent to (X≠Y). However, this coincidence does not occur for the other predicates. For instance, the logical negation of (X<Y) is just NOT(X<Y); the reversed predicate (X?>=Y) is different in that it does not signal an invalid operation exception when X and Y are "unordered".

6. Infini, non-nombres et zéro signé

6.1 *Arithmétique de l'infini*

L'arithmétique de l'infini doit être considérée comme le cas limite de l'arithmétique réelle, avec des opérandes de valeur aussi grande que l'on veut, lorsqu'une telle limite existe. Les infinis doivent s'interpréter au sens affine, c'est-à-dire que $-\infty <$ (tout nombre fini) $< +\infty$.

L'arithmétique sur ∞ est toujours exacte et, pour cette raison, ne doit signaler aucune exception, sauf pour les opérations invalides spécifiées pour ∞ en 7.1. Les cas d'exception qui relèvent de ∞ sont signalés seulement dans les cas suivants:

1) ∞ est créé a partir d'opérandes finis par dépassement de capacité (voir 7.3), ou par une division par zéro (voir 7.2), avec le déroutement correspondant désactivé, ou

2) ∞ est un opérande invalide (voir 7.1).

6.2 *Opérations avec des non-nombres*

Deux types de non-nombres, indicateur et muet, doivent être supportés dans toutes les opérations. Les non-nombres indicateurs autorisent des valeurs pour les variables non initialisées et des améliorations de type arithmétique (telles que les infinités complexes-affines ou les domaines d'amplitude extrême) qui sont en dehors de l'objet de cette norme. Il y a lieu que les non-nombres muets, par des moyens laissés à la discrétion du réalisateur, fournissent des informations de diagnostic rétrospectif héritées des données et résultats invalides ou non disponibles. La propagation des informations de diagnostic nécessite que l'information contenue dans les non-nombres soit préservée à travers les opérations arithmétiques et les conversions de format en virgule flottante.

Les non-nombres indicateurs doivent être des opérandes réservés qui signalent les conditions d'exception d'opération invalide (voir 7.1) pour chaque opération indiquée dans l'article 5. La décision de signaler une exception d'opération invalide lors de la copie d'un non-nombre indicateur sans changement de format est laissée au réalisateur comme option.

Toute opération portant sur un non-nombre indicateur ou toute opération invalide (voir 7.1) doit, si aucun déroutement ne se produit et si un résultat en virgule flottante est attendu, fournir un non-nombre muet comme résultat.

Toute opération portant en entrée sur un ou deux non-nombres, aucun n'étant un non-nombre indicateur, ne doit pas signaler d'exception, mais, si un résultat en virgule flottante est attendu, doit fournir comme résultat un non-nombre muet, qui appartiendrait à un des non-nombres d'entrée. Remarquer qu'il est possible que les conversions de format ne puissent fournir le même non-nombre. Les non-nombres muets ont des effets semblables aux non-nombres indicateurs qui ne fournissent pas de résultat en virgule flottante; ces opérations, à savoir la comparaison et la conversion à un format dépourvu de non-nombres, sont détaillés en 5.4, 5.6, 5.7 et 7.1.

6. Infinity, NaNs, and signed zero

6.1 *Infinity arithmetic*

Infinity arithmetic shall be construed as the limiting case of real arithmetic with operands of arbitrarily large magnitude, when such a limit exists. Infinities shall be interpreted in the affine sense, that is, $-\infty <$ (every finite number) $< +\infty$.

Arithmetic on ∞ is always exact and therefore shall signal no exceptions, except for the invalid operations specified for ∞ in 7.1. The exceptions that do pertain to ∞ are signalled only when

1) ∞ is created from finite operands by overflow (see 7.3) or division by zero (see 7.2), with the corresponding trap disabled, or

2) ∞ is an invalid operand (see 7.1).

6.2 *Operations with NaNs*

Two different kinds of NaN, signalling and quiet, shall be supported in all operations. Signalling NaNs afford values for uninitialized variables and arithmetic-like enhancements (such as complex-affine infinities or extremely wide range) that are not the subject of the standard. Quiet NaNs should, by means left to the implementor's discretion, afford retrospective diagnostic information inherited from invalid or unavailable data and results. Propagation of the diagnostic information requires that information contained in the NaNs be preserved through arithmetic operations and floating-point format conversions.

Signalling NaNs shall be reserved operands that signal the invalid operation exception (see 7.1) for every operation listed in clause 5. Whether copying a signalling NaN without a change of format signals the invalid operation exception is the implementor's option.

Every operation involving a signalling NaN or invalid operation (see 7.1) shall, if no trap occurs and if a floating-point result is to be delivered, deliver a quiet NaN as its result.

Every operation involving one or two input NaNs, none of them signalling, shall signal no exception but, if a floating-point result is to be delivered, shall deliver as its result a quiet NaN, which should be one of the input NaNs. Note that format conversions might be unable to deliver the same NaN. Quiet NaNs have effects similar to signalling NaNs on operations that do not deliver a floating-point result; these operations, namely comparison and conversion to a format that has no NaNs, are discussed in 5.4, 5.6, 5.7 and 7.1.

6.3 *Bit de signe*

Cette norme ne parle pas du signe d'un non-nombre. Dans les autres cas, le signe d'un produit ou d'un quotient s'obtient par le "OU exclusif" des signes des opérandes; et le signe de la somme, ou de la différence $x - y$ considérée comme la somme $x + (-y)$, diffère de l'un des signes des termes au plus. Ces règles doivent s'appliquer même lorsque les opérandes ou les résultats sont zéro ou l'infini.

Lorsque la somme de deux opérandes de signes contraires (ou la différence de deux opérandes de même signe) est exactement nulle, le signe de cette somme (ou différence) doit être "+" dans tous les modes d'arrondi, sauf l'arrondi vers $-\infty$, auquel cas ce signe doit être "-". Cependant, $x + x = x - (-x)$ conserve le même signe que x même lorsque x est nul.

A l'exception près que $\sqrt{-0}$ doit avoir la valeur -0, toute racine carrée valide doit avoir un signe positif.

7. Exceptions

Il y a cinq cas d'exceptions qui doivent être signalés lorsqu'ils sont détectés. Cette signalisation peut consister à positionner un indicateur d'état, exécuter un déroutement, ou éventuellement effectuer les deux. A chaque exception, il convient qu'un déroutement soit associé sous contrôle de l'utilisateur, comme spécifié dans l'article 8. La réponse par défaut à une exception doit être de poursuivre sans déroutement. Cette norme spécifie les résultats devant être fournis selon les deux cas, déroutement et non-déroutement. Dans certains cas, le résultat est différent lorsqu'un déroutement est activé.

Pour chaque type d'exception, la réalisation doit fournir un indicateur d'état qui doit être mis à un lors de toute occurrence de l'exception correspondante lorsqu'aucun déroutement ne se produit. Cet indicateur doit être remis à zéro seulement à la demande de l'utilisateur. Ce dernier doit pouvoir tester et modifier les indicateurs d'état individuellement et, de plus, il y a lieu qu'il puisse sauvegarder et restaurer les cinq indicateurs en bloc.

Les seules exceptions qui peuvent coïncider sont "inexact avec dépassement" et "inexact avec dépassement inférieur".

7.1 *Opérations invalides*

L'exception d'opération invalide est signalée lorsqu'un des opérandes est invalide pour l'opération à exécuter. Le résultat, lorsque l'exception se produit sans déroutement, doit être un non-nombre muet (voir 6.2), pourvu que la destination ait un format virgule flottante. Les opérations invalides sont les suivantes:

1) Toute opération sur un non-nombre indicateur (voir 6.2).

2) Addition ou soustraction: soustraction en valeur absolue de valeurs infinies comme $(+\infty) + (-\infty)$.

3) Multiplication $0 \times \infty$.

6.3 *The sign bit*

This standard does not interpret the sign of a NaN. Otherwise the sign of a product or quotient is the "Exclusive OR" of the operands' signs; and the sign of a sum, or of a difference $x - y$ regarded as a sum $x + (-y)$, differs from at most one of the addends' signs. These rules shall apply even when operands or results are zero or infinite.

When the sum of two operands with opposite signs (or the difference of two operands with like signs) is exactly zero, the sign of that sum (or difference) shall be "+" in all rounding modes except round towards $-\infty$, in which mode that sign shall be "-". However, $x + x = x - (-x)$ retains the same sign as x even when x is zero.

Except that $\sqrt{-0}$ shall be -0, every valid square root shall have a positive sign.

7. Exceptions

There are five types of exceptions that shall be signalled when detected. The signal entails setting a status flag, taking a trap, or possibly doing both. With each exception should be associated a trap under user control, as specified in clause 8. The default response to an exception shall be to proceed without a trap. This standard specifies results to be delivered in both trapping and non-trapping situations. In some cases the result is different if a trap is enabled.

For each type of exception the implementation shall provide a status flag that shall be set on any occurrence of the corresponding exception when no corresponding trap occurs. It shall be reset only at the user's request. The user shall be able to test and to alter the status flags individually, and should further be able to save and restore all five at one time.

The only exceptions that can coincide are inexact with overflow and inexact with underflow.

7.1 *Invalid operations*

The invalid operation exception is signalled if an operand is invalid for the operation to be performed. The result, when the exception occurs without a trap, shall be a quiet NaN (see 6.2) provided the destination has a floating-point format. The invalid operations are:

1) Any operation on a signalling NaN (see 6.2).

2) Addition or subtraction: magnitude subtraction of infinities like $(+\infty) + (-\infty)$.

3) Multiplication $0 \times \infty$.

4) Division 0/0 ou ∞/∞.

5) Calcul du reste: x REM y, où y est nul ou x est infini.

6) Racine carrée: si l'opérande est négatif.

7) Conversion d'un nombre binaire en virgule flottante en un format entier ou décimal lorsqu'un dépassement de capacité, la présence de valeurs infinies ou de non-nombres empêchent une représentation fidèle dans ce format, et que ce fait ne peut pas être signalé autrement, et

8) comparaison utilisant des prédicats impliquant "<" ou ">", sans "?", lorsque les opérandes sont "non ordonnés" (voir 5.7, tableau 4).

7.2 *Division par zéro*

Si le diviseur est nul et le dividende un nombre fini non nul, alors l'exception division par zéro doit être signalée. Le résultat, lorsqu'aucun déroutement n'a lieu, doit être la valeur ∞ avec le signe correct (voir 6.3).

7.3 *Dépassement de capacité*

L'exception de dépassement de capacité doit être signalée lorsque la valeur du plus grand nombre fini représentable dans le format de destination est dépassée par le résultat théorique en virgule flottante avec arrondi qui serait obtenu si le domaine de l'exposant était illimité (article 4). Le résultat lorsqu'aucun déroutement ne se produit, doit être déterminé par le mode d'arrondi et le signe du résultat intermédiaire comme suit:

1) Dans le mode d'arrondi au plus près, tous les dépassements donnent le résultat ∞ avec le signe du résultat intermédiaire.

2) Dans le mode d'arrondi vers 0, tous les dépassements donnent comme résultat le plus grand nombre fini du format avec le signe du résultat intermédiaire.

3) Dans le mode d'arrondi vers -∞, tous les dépassements positifs donnent comme résultat le plus grand nombre fini du format et les dépassements négatifs donnent -∞ comme résultat.

4) Dans le mode d'arrondi vers +∞, tous les dépassements négatifs donnent comme résultat le nombre fini négatif le plus petit du format et les dépassements positifs donnent +∞ comme résultat.

Les dépassements donnant lieu à un déroutement doivent fournir à la routine de traitement de déroutement associée et pour toutes les opérations sauf les conversions, le résultat obtenu en divisant le résultat infiniment précis par 2^{α} puis en effectuant l'arrondi. L'ajustement d'excédent α est de 192 pour le format simple précision, de 1 536 en double précision, et de $3 \times 2^{n-2}$ dans le format étendu où n repré-

4) Division 0/0 or ∞/∞.

5) Remainder x REM y, where y is zero or x is infinite.

6) Square root if the operand is less than zero.

7) Conversion of a binary floating-point number to an integer or decimal format when overflow, infinity, or NaN precludes a faithful representation in that format and this cannot otherwise be signalled; and

8) Comparison via predicates involving "<" or ">", without "?", when the operands are "unordered" (see 5.7, Table 4).

7.2 *Division by zero*

If the divisor is zero and the dividend is a finite non-zero number, then the division by zero exception shall be signalled. The result, when no trap occurs, shall be a correctly signed ∞ (see 6.3).

7.3 *Overflow*

The overflow exception shall be signalled whenever the destination format's largest finite number is exceeded in magnitude by what would have been the rounded floating-point result (clause 4) were the exponent range unbounded. The result, when no trap occurs, shall be determined by the rounding mode and the sign of the intermediate result as follows:

1) Round to nearest carries all overflows to ∞ with the sign of the intermediate result.

2) Round toward 0 carries all overflows to the format's largest finite number, with the sign of the intermediate result.

3) Round toward -∞ carries positive overflows to the format's largest finite number, and carries negative overflows to -∞.

4) Round toward +∞ carries negative overflows to the format's most negative finite number, and carries positive overflows to +∞.

Trapped overflows on all operations except conversions shall deliver to the trap handler the result obtained by dividing the infinitely precise result by 2^{α} and the rounding. The bias adjust α is 192 in the single, 1 536 in the double, and $3 \times 2^{n-2}$ in the extended format,

sente le nombre de bits du champ de l'exposant*. Les dépassements avec déroutements sur conversion de format binaire en virgule flottante doivent fournir à la routine de traitement de déroutement un résultat dans ce même format ou un format supérieur, avec éventuellement un ajustement de l'excédent de l'exposant, mais avec un arrondi correspondant à la précision de la destination. Les dépassements avec déroutements sur conversion décimale-binaire doivent fournir à la routine de traitement de déroutement un résultat dans le format le plus grand supporté, éventuellement avec un ajustement de l'excédent de l'exposant, mais avec un arrondi correspondant à la précision de la destination; lorsque le résultat a une valeur telle que l'exposant ne peut être ajusté, le dépassement doit fournir un non-nombre muet.

7.4 *Dépassement de capacité inférieur*

Deux événements corrélés contribuent à un dépassement inférieur. L'un est la création d'un résultat infinitésimal non nul de valeur comprise entre $\pm 2^{E_{min}}$ qui, à cause de sa petitesse, peut être la cause de quelque autre exception ultérieure, telle qu'un dépassement de capacité par division. L'autre est la perte considérable de précision lors de l'approximation par des nombres dénormalisés de tels nombres infinitésimaux. Le réalisateur peut choisir la manière dont ces événements sont détectés, mais doit détecter ces événements de la même manière pour toutes les opérations. La petitesse peut être détectée soit:

1) "après arrondi" lorsqu'un résultat non nul, calculé comme si le domaine de l'exposant était illimité, est compris strictement entre $\pm 2^{E_{min}}$,

ou

2) "avant arrondi" lorsqu'un résultat non nul, calculé comme si le domaine de l'exposant ainsi que la précision étaient tous deux illimités, serait compris strictement entre $\pm 2^{E_{min}}$,

Une perte de précision peut se détecter soit comme:

3) une perte de dénormalisation: lorsque le résultat fourni diffère de la valeur qui aurait été obtenue si le domaine de l'exposant avait été illimité,

ou

4) un résultat inexact: lorsque le résultat fourni diffère de la valeur qui aurait été obtenue si le domaine de l'exposant ainsi que la précision avaient été illimités. (C'est la condition appelée inexactitude en 7.5.)

---

* L'ajustement d'excédent est choisi de manière à décaler les valeurs correspondant aux dépassements supérieur et inférieur aussi près que possible du milieu du domaine de l'exposant, afin que, si on le désire, ces valeurs puissent être utilisées lors d'opérations ultérieures avec moins de risque de production de nouvelles exceptions.

where n is the number of bits in the exponent field.* Trapped overflow on conversion from a binary floating-point format shall deliver to the trap handler a result in that, or a wider format - possibly with the exponent bias adjusted - but rounded to the destination's precision. Trapped overflow on decimal to binary conversion shall deliver to the trap handler a result in the widest supported format, possibly with the exponent bias adjusted, but rounded to the destination's precision; when the result lies too far outside the range for the bias to be adjusted, a quiet NaN shall be delivered instead.

7.4 *Underflow*

Two correlated events contribute to underflow. One is the creation of a tiny nonzero result between $\pm 2^{E_{min}}$ which, because it is so tiny, may cause some other exception later such as overflow upon division. The other is extraordinary loss of accuracy during the approximation of such tiny numbers by denormalized numbers. The implementor may choose how these events are detected, but shall detect these events in the same way for all operations. Tininess may be detected either:

1) "after rounding": when a nonzero result computed as though the exponent range were unbounded would lie strictly between $\pm 2^{E_{min}}$,

or

2) "before rounding": when a nonzero result computed as though both the exponent range and the precision were unbounded would lie strictly between $\pm 2^{E_{min}}$.

Loss of accuracy may be detected as either:

3) a denormalization loss: when the delivered result differs from what would have been computed were exponent range unbounded;

or

4) an inexact result: when the delivered result differs from what would have been computed were both exponent range and precision unbounded. (This is the condition called inexact in 7.5.)

---

* The bias adjust is chosen to translate over/underflowed values as nearly as possible to the middle of the exponent range so that, if desired, they can be used in subsequent scaled operations with less risk of causing further exceptions.

Lorsqu'un déroutement associé à un dépassement de capacité n'est pas réalisé ou n'est pas activé (le cas par défaut), le dépassement de capacité doit être signalé (par l'intermédiaire de l'indicateur de dépassement inférieur) seulement lorsque la petitesse du résultat et la perte de précision ont été toutes deux détectées. La méthode de détection de la petitesse et de la perte de précision n'affecte pas le résultat fourni qui peut être zéro, dénormalisé ou $\pm 2^{E_{min}}$. Lorsqu'un déroutement pour dépassement de capacité inférieur a été réalisé et se trouve activé, le dépassement de capacité inférieur doit être signalé lorsque la petitesse est détectée quelle que soit la perte de précision. Pour toutes les opérations, sauf la conversion, les dépassements de capacité inférieurs avec déroutement doivent fournir à la routine de traitement de déroutement le résultat obtenu en multipliant le résultat théorique infiniment précis par 2^{α} puis en effectuant un arrondi. L'ajustement d'excédent α est de 192 en simple précision, de 1 536 en double précision, et de $3 \times 2^{n-2}$ en format étendu, ou n est le nombre de bits dans le champ de l'exposant*. Les dépassements de capacité inférieurs avec déroutement sur conversion doivent être traités de manière similaire au traitement des dépassements de capacité supérieurs sur conversion.

7.5 *Inexactitude*

Si le résultat arrondi d'une opération n'est pas exact ou s'il donne lieu à un dépassement de capacité, alors l'exception d'inexactitude doit être signalée. Le résultat arrondi ou avec dépassement de capacité doit être fourni à la destination ou, si un déroutement sur inexactitude se produit, à la routine de traitement de cette exception.

8. Déroutements

Il convient qu'un utilisateur puisse disposer d'un déroutement pour chacun des cinq cas d'exception en spécifiant une routine de traitement de déroutement pour celui-ci. Il y a lieu qu'il puisse commander qu'une routine de traitement existante soit désactivée, sauvegardée ou restaurée. Il convient aussi qu'il puisse déterminer si une routine de traitement de déroutement spécifique a été activée. Lorsqu'une exception dont le déroutement a été désactivé est signalée, elle doit être traitée de la manière spécifiée dans l'article 7. Lorsqu'une exception dont le déroutement est activé est signalée, l'exécution du programme dans lequel l'exception s'est produite doit être suspendue, la routine de traitement de déroutement préalablement spécifiée par l'utilisateur doit être exécutée, et un résultat, si spécifié dans l'article 7, doit lui être fourni.

---

* Remarquer qu'un système dont le matériel sous-jacent déclenche toujours des déroutements sur dépassement de capacité inférieur, produisant un résultat arrondi, à l'exposant ajusté, doit indiquer si un tel résultat est arrondi en valeur absolue, de manière que le résultat correctement dénormalisé puisse être produit par le logiciel du système lorsque le déroutement de dépassement de capacité inférieur de l'utilisateur est désactivé.

When an underflow trap is not implemented or is not enabled (the default case) underflow shall be signalled (via the underflow flag) only when both tininess and loss of accuracy have been detected. The method for detecting tininess and loss of accuracy does not affect the delivered result which might be zero, denormalized or $\pm 2^{E_{min}}$. When an underflow trap has been implemented and is enabled, underflow shall be signalled when tininess is detected regardless of loss of accuracy. Trapped underflows on all operations except conversion shall deliver to the trap handler the result obtained by multiplying the infinitely precise result by 2^{α} and then rounding. The bias adjust α is 192 in the single, 1 536 in the double, and $3 \times 2^{n-2}$ in the extended format, where n is the number of bits in the exponent field.* Trapped underflows on conversion shall be handled analogously to the handling of overflows on conversion.

7.5 *Inexact*

If the rounded result of an operation is not exact or if it overflows without an overflow trap, then the inexact exception shall be signalled. The rounded or overflowed result shall be delivered to the destination, or, if an inexact trap occurs, to the trap handler.

8. Traps

A user should be able to request a trap on any of the five exceptions by specifying a handler for it. He should be able to request that an existing handler be disabled, saved or restored. He should also be able to determine whether a specific trap handler for a designated exception has been enabled. When an exception whose trap is disabled is signalled, it shall be handled in the manner specified in clause 7. When an exception whose trap is enabled is signalled, the execution of the program in which the exception occurred shall be suspended, the trap handler previously specified by the user shall be activated, and a result, if specified in clause 7, shall be delivered to it.

---

* Note that a system the underlying hardware of which always traps on underflow, producing a rounded, bias-adjusted result, must indicate whether such a result is rounded up in magnitude in order that the correctly denormalized result may be produced in system software when the user underflow trap is disabled.

8.1 *Routine de traitement de déroutement*

Il y a lieu qu'une routine de traitement de déroutement possède les possibilités d'un sous-programme qui peut rendre une valeur devant être utilisée à la place du résultat de l'opération ayant donné lieu à l'exception; ce résultat est indéfini, à moins qu'il ne soit fourni par la routine de traitement de déroutement. De la même manière, le ou les indicateurs correspondant aux exceptions signalées avec leurs déroutements associés activés peuvent être indéfinis, à moins qu'ils ne soient mis à un ou à zéro par la routine de traitement de déroutement.

Lorsqu'un système effectue un déroutement, il convient que la routine de traitement du déroutement soit capable de déterminer:

1) Quelles exceptions se sont produites lors de cette opération.

2) Le type d'opération qui était effectuée.

3) Le format de destination.

4) Lors des cas de dépassement de capacité supérieur, inférieur, et pour les exceptions d'inexactitude, le résultat correct arrondi, y compris l'information qui pourrait ne pas être compatible avec le format de destination.

5) Lors des cas d'exceptions pour opérations invalides et de division par zéro, les valeurs des opérandes.

8.2 *Précédence*

Lorsqu'ils sont activés, les déroutements pour dépassement de capacité supérieur et inférieur ont la priorité sur un déroutement pour inexactitude séparé.

8.1 *Trap handler*

A trap handler should have the capabilities of a subroutine that can return a value to be used in lieu of the exceptional operation's result; this result is undefined unless delivered by the trap handler. Similarly, the flag(s) corresponding to the exceptions being signalled with their associated traps enabled may be undefined unless set or reset by the trap handler.

When a system traps, the trap handler should be able to determine:

1) Which exception(s) occurred on this operation.

2) The kind of operation that was being performed.

3) The destination's format.

4) In overflow, underflow, and inexact exceptions, the correctly rounded result, including information that might not fit in the destination's format.

5) In invalid operations and divide by zero exceptions, the operand values.

8.2 *Precedence*

If enabled, the overflow and underflow traps take precedence over a separate inexact trap.

ANNEXE A

FONCTIONS ET PREDICATS RECOMMANDES

Cette annexe ne fait pas partie de la norme pour l'arithmétique binaire en virgule flottante, mais est incluse seulement pour information.

Les fonctions et prédicats suivants sont recommandés comme aidant à la portabilité des programmes vers différents systèmes, qui peut-être traitent de manière très différente l'arithmétique. Ils sont décrits de manière générique; c'est-à-dire les types des opérandes et des résultats sont considérés comme inhérents à ces opérandes. Les langages qui nécessitent un typage explicite auront des familles correspondantes de fonctions et de prédicats.

Certaines fonctions ci-dessous, comme l'opération de copie $y := x$ sans changement de format, peuvent, selon l'option prise par le réalisateur, être traitées comme des opérations non arithmétiques qui ne signalent pas l'exception d'opération invalide pour les non-nombres indicateurs; les fonctions en question sont 1), 2), 6) et 7).

1) copysign(x,y) rend x avec le signe de y. Donc abs(x) = copysign(x,1,0), même si x est un non-nombre.

2) $-x$ est égal à x copié avec le signe contraire, et non $0 - x$; la distinction se rapporte au cas ou x est ± 0 ou un non-nombre. Par conséquent, ce serait une erreur d'utiliser le bit de signe pour distinguer entre les non-nombres indicateurs et les non-nombres muets.

3) scal$b$(y,N) rend $y \times 2^N$ pour les valeurs entières de N, sans calculer explicitement 2^N.

4) log$b$(x) rend l'exposant sans excédent de x, un entier signé dans le format de x, sauf dans les cas log$b$(non-nombre) qui est un non-nombre, log$b$(∞) qui vaut $+\infty$, et log$b$(0) qui vaut $-\infty$ et signale l'exception de division par zéro. Lorsque x est positif et fini, l'expression scal$b$(x,-log$b$(x)) a une valeur comprise strictement entre 0 et 2; elle est inférieure à 1 seulement lorsque x est dénormalisé.

5) nextafter(x,y) rend le voisin représentable de x dans la direction de y. Les cas spéciaux suivants peuvent se produire: si $x=y$, alors le résultat est x sans qu'aucune exception ne soit signalée; dans le cas contraire, si soit x ou y est un non-nombre muet alors le résultat est un ou l'autre des non-nombres d'entrée. Un dépassement de capacité inférieur est signalé lorsque x est fini mais que nextafter(x,y) est infini; un dépassement de capacité inférieur est signalé lorsque nextafter(x,y) a une valeur strictement incluse dans $\pm 2^{E_{min}}$; dans ces deux cas l'exception d'inexactitude est signalée.

APPENDIX A

RECOMMENDED FUNCTIONS AND PREDICATES

This appendix is not part of this standard for binary floating-point arithmetic, but is included for information only.

The following functions and predicates are recommended as aids to program portability across different systems, perhaps performing arithmetic very differently. They are described generically; that is, the types of the operands and results are inherent in the operands. Languages that require explicit typing will have corresponding families of functions and predicates.

Some functions below, like the copy operation $y := x$ without change of format may, at the implementor's option, be treated as non-arithmetical operations which do not signal the invalid operation exception for signalling NaNs; the functions in question are 1), 2), 6) and 7).

1) copysign(x,y) returns x with the sign of y. Hence abs(x) = copysign(x,1.0), even if x is NaN.

2) $-x$ is x copied with its sign reversed; not $0 - x$; the distinction is germane when x is ± 0 or NaN. Consequently, it would be a mistake to use the sign bit to distinguish signalling NaNs from quiet NaNs.

3) scal$b$(y,N) returns $y \times 2^N$ for integral values N, without computing 2^N.

4) log$b$(x) returns the unbiased exponent of x, a signed integer in the format of x, except that log$b$(NaN) is a NaN, log$b$(∞) is $+\infty$, and log$b$(0) is $-\infty$ and signals the division by zero exception. When x is positive and finite the expression scal$b$(x,$-$log$b$(x)) lies strictly between 0 and 2; it is less than 1 only when x is de-normalized.

5) nextafter(x,y) returns the next representable neighbour of x in the direction toward y. The following special cases arise: if $x=y$, then the result is x without any exception being signalled; otherwise, if either x or y is a quiet NaN, then the result is one or the other of the input NaNs. Overflow is signalled when x is finite but nextafter(x,y) is infinite; underflow is signalled when nextafter(x,y) lies strictly between $\pm 2^{E_{min}}$; in both cases, inexact is signalled.

6) finite(x) rend la valeur VRAI si $-\infty < x < +\infty$, et rend FAUX dans les autres cas.

7) isnan(x) ou de manière équivalente $x \neq x$, rend la valeur VRAI si x est un non-nombre, et rend FAUX dans les autres cas.

8) $x<>y$ est VRAI seulement si $x<y$ ou $x>y$, et se distingue de $x \neq y$, qui signifie NON($x=y$) (Tableau 4).

9) unordered(x,y), ou $x?y$ rend la valeur VRAI si x est non ordonné avec y, et rend FAUX dans les autres cas (Tableau 4).

10) class(x) indique dans laquelle des dix classes suivantes x se place: non-nombre indicateur, non-nombre muet, $-\infty$, nombre négatif normalisé non nul, nombre négatif dénormalisé, -0, +0, nombre positif dénormalisé, nombre positif normalisé non nul, $+\infty$. Cette fonction ne donne jamais lieu à une exception, même dans le cas de non-nombres indicateurs.

6) finite(x) returns the value TRUE if $-\infty < x < +\infty$, and returns FALSE otherwise.

7) isnan(x) or equivalently $x \neq x$ returns the value TRUE if x is a NaN, and returns FALSE otherwise.

8) $x<>y$ is TRUE only when $x<y$ or $x>y$, and is distinct from $x \neq y$, which means NOT($x=y$) (Table 4).

9) unordered(x,y), or $x?y$ returns the value TRUE if x is unordered with y, and returns FALSE otherwise (Table 4).

10) class(x) tells which of the following ten classes x falls into: signalling NaN, quiet NaN, $-\infty$, negative normalized nonzero, negative denormalized, -0, +0, positive denormalized, positive normalized nonzero, $+\infty$. This function is never exceptional, not even for signalling NaNs.

ICS 31.080

Typeset and printed by the IEC Central Office
GENEVA, SWITZERLAND