Phần 1:

Chú thích: tiêu chuẩn = tiêu chuẩn IEEE-754

**3.Floating point Format:**

3.1 Tổng quan

3.1.1 Format

ĐỊnh dạng số thực dấu phẩy động (Floating point format) dùng để biểu diễn một tập hợp con hữu hạn các số thực (xem 3.2). Các định dạng được đặc trưng bởi cơ số (radix), độ chính xác (precision) và phạm vi số mũ (exponent range) và mỗi định dạng có thể đại diện cho một tập hợp dữ liệu dấu phẩy động duy nhất (xem 3.3).

Tất cả các định dạng có thể được hỗ trợ dưới định dạng số học (arithmetic format); nghĩa là, chúng có thể được sử dụng để biểu diễn các toán hạng dấu phẩy động hoặc kết quả cho các phép toán được mô tả trong các phần sau.

Các fixed-width encoding cụ thể cho các định dạng nhị phân và thập phân được định nghĩa trong phần này cho một tập hợp con các định dạng (xem 3.4 và 3.5). Các interchange format này được xác định theo kích thước của chúng (xem 3.6) và có thể được sử dụng để chuyển đổi dữ liệu floating point giữa các triển khai.

Có 5 định dạng cơ bản sau:

3 định dạng nhị phân (binary format), với độ dài encoding lần lượt là 32,64 và 128 bits

2 định dạng thập phân (decimal format), với độ dài encoding lần lượt là 64 và 128 bits

Các format số học bổ sung được khuyến nghị để mở rộng các định dạng cơ bản này (xem 3.7).

Việc lựa chọn định dạng nào trong số các định dạng của tiêu chuẩn này để hỗ trợ là do ngôn ngữ lập trình tự xác định, hoặc nếu tiêu chuẩn ngôn ngữ liên quan là im lặng hoặc trì hoãn việc triển khai, thì việc triển khai sẽ tự xác định định dạng. Tên được sử dụng cho các định dạng trong tiêu chuẩn này không nhất thiết phải là tên được sử dụng trong môi trường lập trình.

3.1.2 Conformance

Việc triển khai phù hợp với bất kỳ định dạng được hỗ trợ nào sẽ cung cấp các phương tiện để khởi tạo định dạng đó và sẽ cung cấp các chuyển đổi giữa định dạng đó và tất cả các định dạng được hỗ trợ khác.

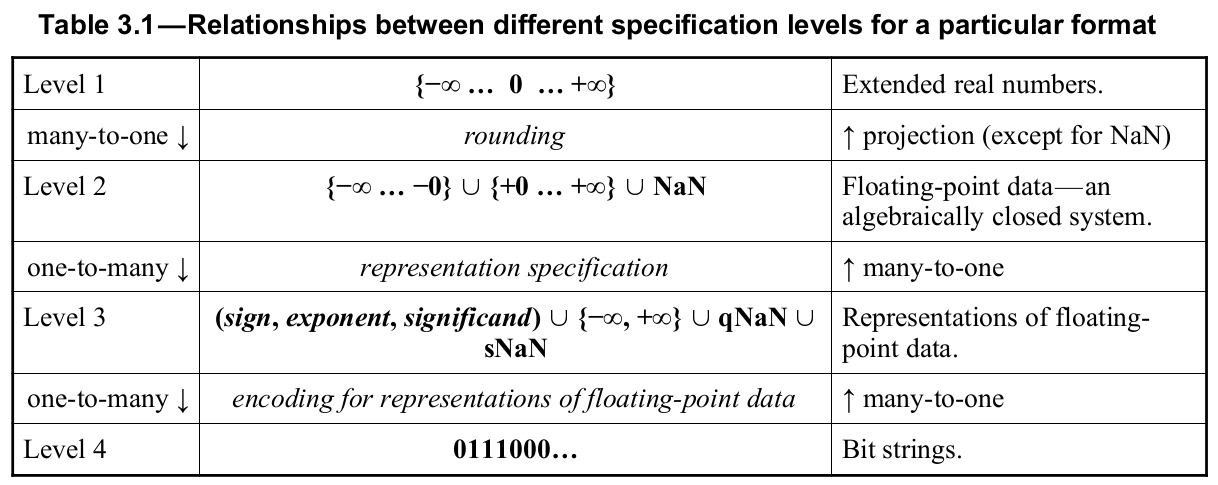
Việc triển khai phù hợp với định dạng số học được hỗ trợ sẽ cung cấp tất cả các phép toán của tiêu chuẩn được xác định trong Điều 5, cho định dạng đó.

Việc triển khai phù hợp với interchange format được hỗ trợ sẽ cung cấp phương tiện để đọc và ghi định dạng đó bằng cách sử dụng một encoding cụ thể cho định dạng đó.

Môi trường lập trình tuân theo tiêu chuẩn này, trong một cơ số cụ thể, bằng cách triển khai một hoặc nhiều định dạng cơ bản của cơ số đó dưới dạng cả định dạng số học được hỗ trợ và interchange format được hỗ trợ.

3.2. Các level cụ thể

Tính toán dấu phẩy động là một phép gần đúng có hệ thống của số học thực, như được minh họa trong Bảng 3.1. Tính toán dấu phẩy động chỉ có thể biểu diễn một tập hợp con hữu hạn của các số thực liên tục. Do đó, các tính chất nhất định của số học thực, chẳng hạn như tính kết hợp của phép cộng, không phải lúc nào cũng phù hợp với Floating tính toán dấu phẩy động.



Cấu trúc toán học làm cơ sở cho số học trong tiêu chuẩn này là số thực mở rộng, tức là tập hợp các số thực cùng với âm và dương vô cùng. Đối với một định dạng nhất định, quá trình làm tròn (xem Điều 4) ánh xạ một số thực mở rộng thành một số dấu phẩy động có trong định dạng đó. Một số thực dấu phẩy động, có thể là số 0 có dấu, số hữu hạn khác 0, vô cực có dấu hoặc NaN (Not a Number), có thể được ánh xạ tới một hoặc nhiều biểu diễn của dữ liệu dấy phẩy động ở một định dạng.

Các biểu diễn của dữ liệu dấu phẩy động ở định dạng bao gồm

* bộ ba (sign, exponent, significand); trong cơ số b, số thực dấu phẩy động được biểu diễn bởi một bộ ba là
* qNaN, sNaN

Một mã hóa ánh xạ một biểu diễn của dữ liệu dấy phẩy động thành một chuỗi bit. Một dữ liệu dấu phẩy động có thể được mã hóa thành nhiều hơn một chuỗi bit. Mã hóa NaN nên được sử dụng để lưu trữ thông tin chẩn đoán hồi cứu (xem 6.2).

Floating point datum: số ít của floating point data

**3.3 Tập hợp dữ liệu dấu phẩy động**

Điều khoản phụ này chỉ định các tập hợp dữ liệu dấu phẩy động có thể biểu diễn trong tất cả các định dạng dấu phẩy động; mã hóa cho các biểu diễn cụ thể của dữ liệu dấu phẩy động ở các định dạng thay thế được định nghĩa trong 3.4 và 3.5, và các tham số cho các định dạng thay thế được xác định trong 3.6.

Tập hợp các số dấu phẩy động hữu hạn có thể biểu diễn trong một định dạng cụ thể được xác định bởi các tham số số nguyên sau:

― b = the radix (cơ số), 2 or 10

― p: số chữ số trong significand (precision)

― emax: lũy thừa maximum e

― emin: lũy thừa minimun e

emin bắt buộc là 1 − emax trong tất cả định dạng.

Giá trị của các tham số này cho mỗi định dạng cơ bản được cho trong Bảng 3.2, trong đó mỗi định dạng được xác định bằng cơ số của nó và số bit trong mã hóa của nó. Các ràng buộc về các tham số này đối với các định dạng chính xác mở rộng và có thể mở rộng (extended and extendable) được đưa ra trong 3.7.

Trong mỗi định dạng, các dữ liệu dấu phẩy động sau sẽ được biểu diễn:

Số thực dấu phẩy động có dấu zero and non-zero có dạng,

trong đó

― s = 0 hoặc 1.

― e là số nguyên bất kì thỏa emin ≤ e ≤ emax.

― m là một số được biểu diễn bằng một chuỗi chữ số có dạng d 0 • d 1 d 2 ...d p −1 với là chữ số nguyên (vì thế 0 ≤ m < b).

― 2 số vô hạn, +∞ and −∞.

― 2 NaNs, qNaN (quiet) and sNaN (signaling).

Đây là những biểu diễn floating point duy nhất.

Trong mô tả ở trên, significand m được xem ở dạng khoa học, với dấu thập phân ngay sau chữ số đầu tiên. Trong một số mục đích để thuận tiện thì người ta cũng xem significand là một số nguyên; trong trường hợp đó các số dấu phẩy động hữu hạn được mô tả như vậy:

Số thực dấu phẩy động có dấu bằng 0 hoặc khác 0 có dạng , trong đó

― s is 0 or 1.

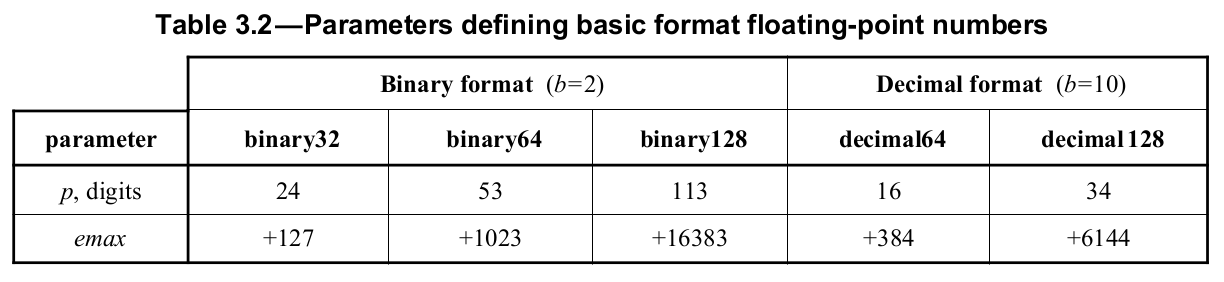
― q là số nguyên thỏa emin ≤ q + p − 1 ≤ emax.

― c là một số được biểu diễn bằng một chuỗi chữ số có dạng d 0 d 1 d 2 ...d p −1 where là chữ số nguyên thỏa (vì thế c là số nguyên thỏa mãn ).

Cách xem significand như một số nguyên c, với số mũ q tương ứng của nó, mô tả chính xác cùng một tập hợp các số dấu phẩy động không và khác 0 như cách xem ở dạng khoa học. (Với số thực dấu phẩy động hữu hạn, e = q + p − 1 và m = )

Số floating point bình thường dương nhỏ nhất là và lớn nhất là . Số floating point khác 0 có độ lớn nhỏ hơn được gọi là subnormal bởi vì độ lớn của chúng nằm giữa 0 và số bình thường nhỏ nhất. Chúng luôn có ít hơn p chữ số có nghĩa. Mọi số dấu phẩy động hữu hạn đều là bội số của số subnormal nhỏ nhất .

Đối với một số dấu phẩy động có giá trị bằng 0, dấu s cung cấp thêm thông tin. Mặc dù tất cả các định dạng đều có các biểu diễn riêng biệt cho +0 và −0, dấu của số 0 có ý nghĩa trong một số trường hợp, chẳng hạn như phép chia cho số 0, nhưng không có ý nghĩa trong các trường hợp khác (xem 6.3). Các định dạng thay thế nhị phân chỉ có một biểu diễn cho mỗi +0 và −0, nhưng các định dạng thập phân có nhiều. Trong tiêu chuẩn này, 0 và ∞ được viết không dấu khi dấu không quan trọng.



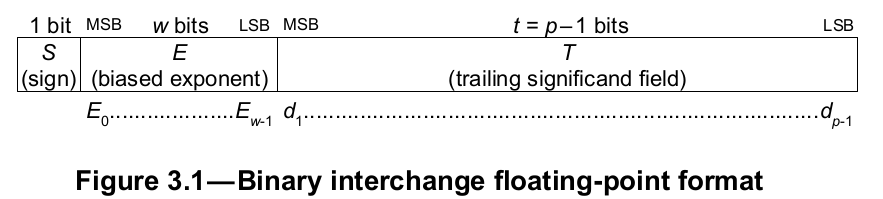
**Định dạng thay thế nhị phân (Binary interchange format encodings)**

Mỗi số dấu phẩy động chỉ có một mã hóa (encoding) ở định dạng thay thế nhị phân. Để làm cho mã hóa là duy nhất, theo các tham số trong 3.3, giá trị của significand m được tối đa hóa bằng cách giảm e cho đến khi e = emin hoặc m ≥ 1, tức là biến đổi lại biểu diễn thành . Khi đó m thành .

Sau khi quá trình này được thực hiện, nếu e = emin và 0 <m < 1, số dấu phẩy động là subnormal. Các số subnormal (và số 0) được mã hóa bằng một giá trị lũy thừa bias riêng.

Các biểu diễn của dữ liệu ở định dạng thay thế nhị phân được mã hóa duy nhất thành k bit trong ba trường sau được sắp xếp theo thứ tự như trong Hình 3.1:

* 1 bit dấu S
* w-bit bias exponent E = e + bias: phần mũ, cơ số là 2
* (t=p-1) bit significand theo sau T = ; bit đầu tiên của significand, được mã hóa đơn giản trong biased exponent E.



Các giá trị của k, p, t, w và bias cho các định dạng thay thế nhị phân được liệt kê trong Bảng 3.5 (xem 3.6).

Phạm vi của E của mã hóa phải bao gồm:

* mọi số nguyên từ 1 đến , để mã hóa các số bình thường
* giá trị đặt trước là 0 để mã hóa ± 0 và các số subnormal
* giá trị đặt trước là để mã hóa và NaN

Biểu diễn r của dữ liệu dấu phẩy động và giá trị v của dữ liệu dấu phẩy động được biểu diễn, được suy ra từ các trường cấu thành như sau:

* Nếu và , thì r là qNaN hoặc sNan và v là NaN bất kể S và dùng để phân biệt riêng giữa qNaN và sNaN (xem 6.2.1)
* Nếu và , thì r và v =
* Nếu , thì r là

giá trị của số dấu phẩy động tương ứng là ; do đó các số normal có bit đầu tiên của significand là 1

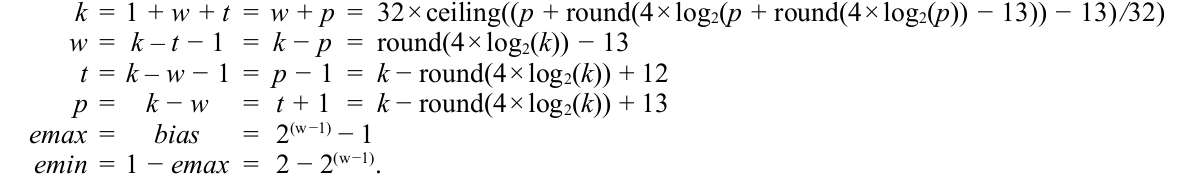
* Nếu và , thì là

giá trị của số dấu phẩy động tương ứng là; do đó các số subnormal có bit đầu tiên của significand là 0

* Nếu và , thì r là và (số 0 có dấu, xem 6.3)

Trong các định dạng thay thế nhị phân, tất cả các mã hóa số và NaN đều là canonical encoding.

Chú ý: Khi k là 64 hoặc bội của 32 và >= 128, đối với mã hóa này, tất cả những điều sau đây đúng:



**Decimal interchange format encodings**

Cohort

Không giống như ở định dạng floating point nhị phân, ở định dạng floating point thập phân, một số có thể có nhiều biểu diễn. Tập hợp các biểu diễn mà một số floating point ánh xạ tới được gọi là cohort của số dấu phẩy động; các thành viên của một cohort là các đại diện riêng biệt của cùng một số dấu phẩy động. Ví dụ: nếu c là bội của 10 và q nhỏ hơn giá trị lớn nhất cho phép của nó, thì (s, q, c) và (s, q + 1, c / 10) là hai biểu diễn cho cùng một số dấu phẩy động và là thành viên của cùng một cohort.

Mặc dù bằng nhau về số lượng, các thành viên khác nhau của một cohort có thể được phân biệt bằng các phép toán dành riêng cho số thập phân (xem 5.3.2, 5.5.2 và 5.7.3). Các cohort cuar các số dấu phẩy động khác nhau có thể có số lượng thành viên khác nhau. Nếu biểu diễn của một số hữu hạn khác 0 có n chữ số thập phân từ chữ số khác 0 có nghĩa nhất của nó đến chữ số khác 0 có nghĩa nhỏ nhất, thì biểu diễn của cohort sẽ có tối đa p - n + 1 thành viên trong đó p là số chữ số có nghĩa trong định dạng.

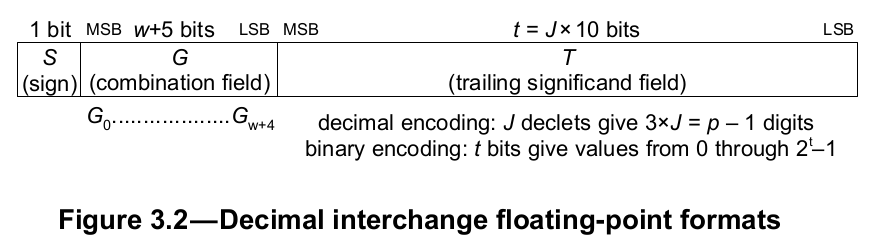
Ví dụ: số dấu floating point có một chữ số có thể có tối đa p cách biểu diễn khác nhau trong khi số dấu phẩy động gồm chữ số p không có số 0 ở cuối chỉ có một cách biểu diễn. (Một số dấu phẩy động gồm n chữ số có thể có ít hơn p - n + 1 thành viên trong nhóm thuần tập của nó nếu nó gần các điểm cực trị của phạm vi lũy thừa của định dạng.) Số 0 có cohort lớn hơn nhiều: cohort +0 chứa một biểu diễn cho mỗi số mũ, cũng như cohort của −0.

Đối với số học thập phân, ngoài việc chỉ định một kết quả số, các phép toán số học cũng chọn một thành viên của cohort của kết quả theo 5.2. Các ứng dụng thập phân có thể sử dụng các nhóm thông tin bổ sung được chuyển tải

Encodings

Các biểu diễn của dữ liệu dấu phẩy động ở định dạng trao đổi thập phân được mã hóa thành k bit trong ba trường sau, có bố cục chi tiết và mã hóa chuẩn (ưu tiên) được mô tả bên dưới.

* 1 bit dấu S
* w+5 bit trường kết hợp G mã hóa phân loại, nếu dữ liệu được mã hóa là số hữu hạn, thì là exponent q và 4 significand bit (1 hoặc 3 trong số đó được ngụ ý)
* t bit significand theo sau T chứa Jx10 bit và chứa số lượng lớn significand. Khi trường này được kết hợp với bit significand đầu tiên từ trường kết hợp G, format encodes tổng cộng p = 3xJ+1 chữ số thập phân.



Các giá trị của k, p, t, w và bias cho các định dạng hoán đổi thập phân được liệt kê trong Bảng 3.6 (xem 3.6).

Biểu diễn r của dữ liệu floating point và giá trị v của dữ liệu floating point được biểu diễn, được suy ra từ các trường cấu thành như sau:

* Nếu là 1111, thì v là NaN bất kể S. Hơn nữa, nếu G5 là 1, thì r là sNaN; ngược lại r là qNaN. Các bit còn lại của G được bỏ qua, và T là payload có thể được dùng để chuẩn đoán các loại NaN khác nhau

Payload được mã hóa tương tự với số hữu hạn mô tả ở mục dưới đây, với G được xem là toàn bit 0. Payload tương ứng với significad của số hữu hạn, thực hiện như là số nguyên với giá trị lớn nhất là 10^(3J) -1, và trường exponent được bỏ qua (xem như tất cả là 0). NaN ở dạng mã hóa ưu tiên (canonical) nếu bit G6 đến Gw+4 là 0 và mã hóa của payload là canonical.

* Nếu G0 -> G4 là 11110 thì r và v = . Giá trị của các bit còn lại trong G, và T được bỏ qua. Hai canonical encoding cho số vô hạn có bit G5 -> Gw+4 là 0 và T là 0
* Với số hữu hạn, r là (S,E-bias,C) và , với C là kết hợp giữa bit đầu tiên của significand từ G và singificand T, và E cũng được mã hóa trong G. Việc mã hóa trong các trường này phụ thuộc vào việc triển khai sử dụng mã hóa thập phân hay nhị phân cho significand.

1. Nếu việc triển khai sử dụng mã hóa thập phân cho significand, thì w bit có nghĩa nhỏ nhất của số mũ là G 5 đến G w +4. Hai bit significand của số mũ thiên vị và chuỗi chữ số thập phân d 0 d 1 ... d p −1 của dấu nghĩa và được hình thành từ các bit G 0 đến G 4 và T như sau:

\* Nếu 5 bit đầu của G là 110xx hoặc 1110x, thì bit đầu tiên của significand là d0 = 8 + G4, giá trị của nó sẽ là 8 hoặc 9, và bit đầu của bias exponent sẽ là 2G2 + G3, có thể nhận giá trị là 0,1,2.

\* Nếu 5 bit đầu của G là 0xxxx hoặc 10xxx, bit d0 sẽ là 4G2+2G3+G4, giá trị range 0->7, và bit đầu của bias exponent là 2G0+G1, giá trị là 0,1 hoặc 2. Kết quả là nếu T = 0 và 5 bit đầu của G là 00000, 01000 hoặc 10000 thì v =

p −1 = 3 × J d 1 ... d p −1 chữ số thập phân được mã hóa bởi T, chứa các phần tử J được mã hóa ở dạng thập phân dày đặc.

Một canonical significand chỉ có các canon declets như được thể hiện trong Bảng 3.3 và 3.4. Các phép toán tính toán chấp nhận tất cả 1024 declet có thể có trong toán hạng. Ngoại trừ các hoạt động tính toán quiet (xem 5.5), các hoạt động tính toán chỉ tạo ra 1000 canon declet.

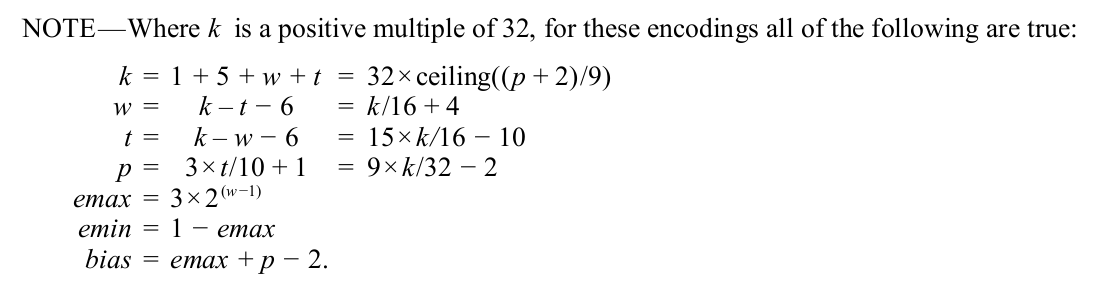
2. Ngoài ra, nếu việc triển khai sử dụng mã hóa nhị phân cho significand thì:

\* Nếu G 0 và G 1 cùng là một trong 00, 01 hoặc 10, thì bias exponent E được hình thành từ G0 đến Gw+1 và significand được hình thành từ bit Gw+2 đến hết (bao gồm cả T)

\* Nếu G 0 và G 1 cùng với nhau là 11 và G 2 và G 3 cùng là một trong 00, 01 hoặc 10, thì bias expornent E được hình thành từ G 2 đến G w +3 và significand được hình thành bằng cách thêm tiền tố 4 bit (8 + G w +4) đến T.

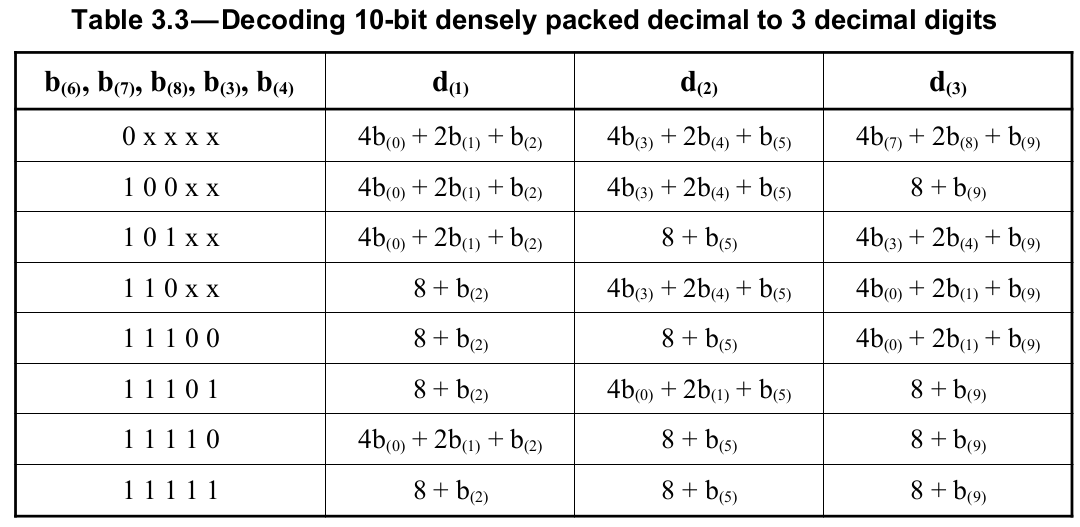
Giá trị lớn nhất của significand mã hóa nhị phân bằng giá trị của significand được mã hóa thập phân tương ứng; nghĩa là, 10^(3 × J + 1) −1 (hoặc 10^(3 × J) −1 khi T được sử dụng làm payload của một NaN). Nếu giá trị vượt quá giá trị lớn nhất, significand c là không chính tắc và giá trị được sử dụng cho c bằng 0.

Exception và Recommend.

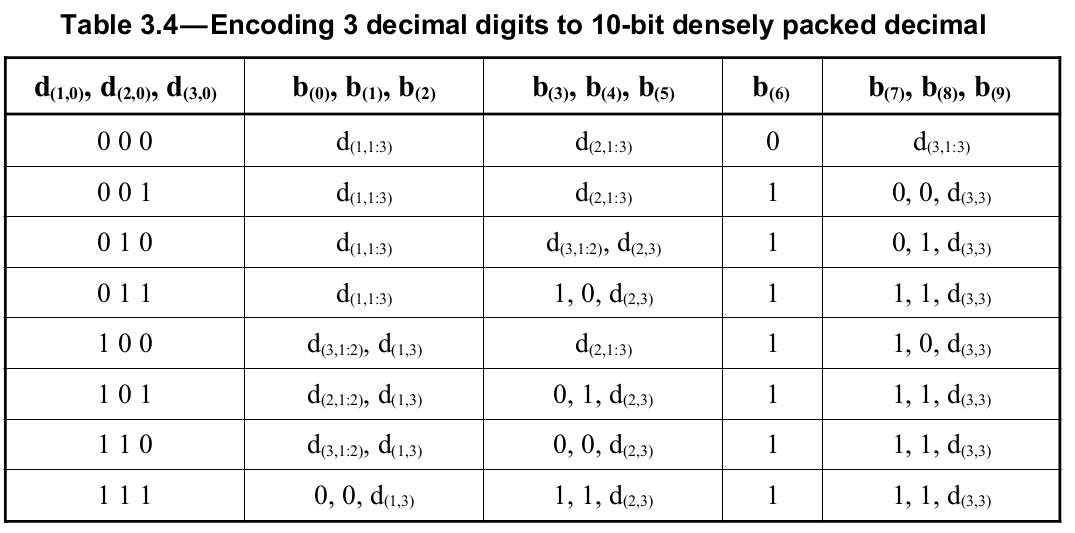
Các phép toán tính toán thường chỉ tạo ra các significand chính tắc (canonical) và luôn chấp nhận các non-canonical trong các toán hạng.

**Decoding densely packed decimal:**

Bảng 3.3 giải mã một declet, với 10 bit b (0) đến b (9), thành 3 chữ số thập phân d (1), d (2), d (3). Cột đầu tiên ở dạng nhị phân và "x" biểu thị một bit "không quan tâm". Do đó, tất cả 1024 mẫu 10 bit có thể có sẽ được chấp nhận và ánh xạ thành 1000 tổ hợp 3 chữ số có thể có với một số trùng lặp.



**Encoding densely packed decimal**: Bảng 3.4 mã hóa 3 chữ số thập phân d (1), d (2) và d (3), mỗi chữ số có 4 bit có thể được biểu thị bằng chỉ số con thứ hai d (1,0: 3), d ( 2,0: 3), và d (3,0: 3), trong đó bit 0 là MSB và bit 3 LSB, thành một declet, với 10 bit b (0) đến b (9). Hầu hết các hoạt động tính toán chỉ tạo ra 1000 mẫu 10 bit chuẩn được xác định bởi Bảng 3.4.



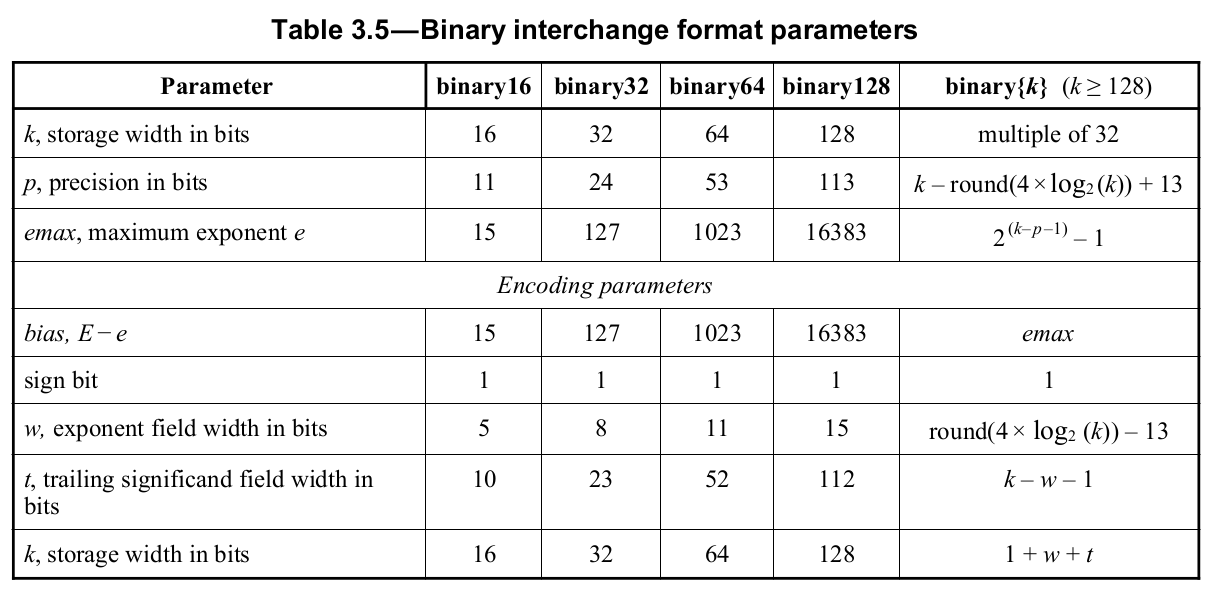
24 mẫu không chuẩn có dạng 01x11x111x, 10x11x111x hoặc 11x11x111x (trong đó “x” biểu thị một bit “không quan tâm”) không được tạo ra từ kết quả của một hoạt động tính toán. Tuy nhiên, như được liệt kê trong Bảng 3.3, các mẫu 24 bit này ánh xạ tới các giá trị trong phạm vi từ 0 đến 999. Mẫu bit trong trường significand theo sau của NaN có thể ảnh hưởng đến cách NaN được truyền (xem 6.2).

**3.6. Các thông số của định dạng thay thế**

Các định dạng thay thế hỗ trợ việc thay thế dữ liệu số thực dấu phẩy động giữa các lần triển khai. Trong mỗi cơ số, significand và phạm vi số mũ của định dạng thay đổi được xác định bởi kích thước của nó; Do đó, việc thay thế dữ liệu dấu phẩy động có kích thước nhất định luôn chính xác mà không có khả năng bị overflow hoặc underflow.

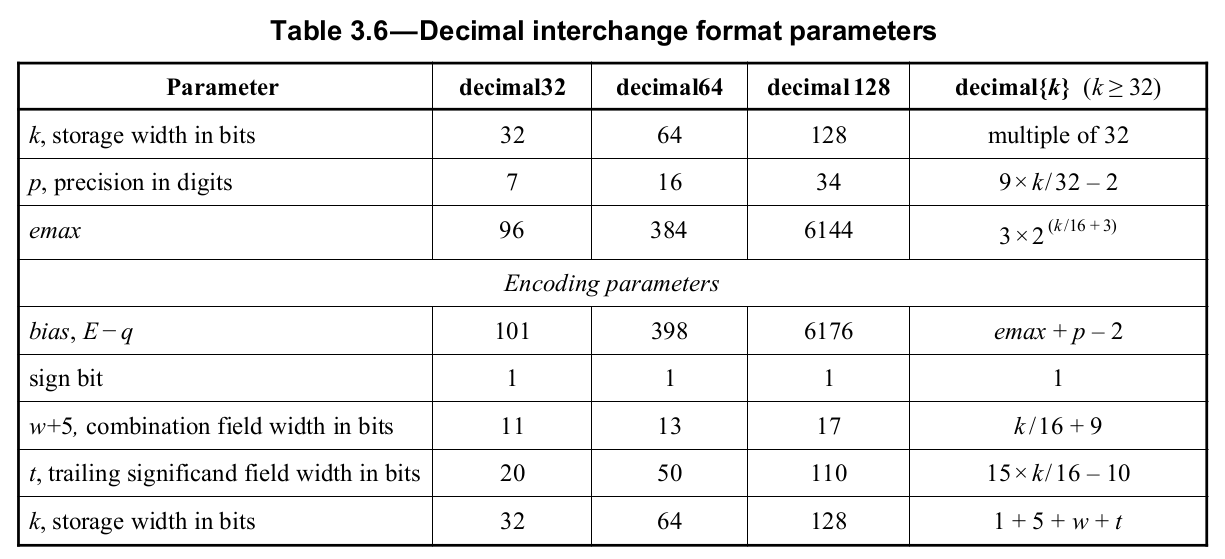
Tiêu chuẩn IEEE-754 xác định các định dạng thay thế nhị phân có độ rộng 16, 32, 64 và 128 bit và nói chung cho bất kỳ bội số nào của 32 bit và ít nhất 128 bit. Các định dạng decimal interchange được xác định cho bất kỳ bội số 32 bit nào trong số ít nhất 32 bit.

Các tham số và cho mọi độ rộng của định dạng thay thế được thể hiện trong Bảng 3.5 cho các định dạng thay thế nhị phân và trong Bảng 3.6 cho các định dạng trao đổi thập phân. Các mã hóa cho các định dạng thay thế được mô tả trong 3.4 và 3.5.2; các tham số mã hóa cho mỗi chiều rộng định dạng trao đổi cũng được hiển thị trong Bảng 3.5 và 3.6.



Trong bảng 3.5, round( ) là: làm tròn đến số nguyên gần nhất.

Ví dụ binary256 sẽ có p = 237 và emax = 262143.



Ví dụ, decimal256 sẽ có p = 70 và emax = 1572864.

CHÚ THÍCH - Tiêu chuẩn IEEE-754 xác định việc biểu diễn dữ liệu riêng lẻ là các thực thể Cấp 4 theo khái niệm. Các ứng dụng trao đổi dữ liệu giữa các triển khai khác nhau phải giao tiếp các tham số nhất định mô tả các định dạng và bố cục của dữ liệu. Bên cạnh các vấn đề như thứ tự byte ảnh hưởng đến tất cả các thay thế dữ liệu, các tùy chọn triển khai nhất định được tiêu chuẩn này cho phép cũng phải được xem xét:

\* đối với các định dạng nhị phân, cách các sNaN được phân biệt với các qNaN

\* đối với các định dạng thập phân, cho dù mã hóa nhị phân hay thập phân được sử dụng.

Tiêu chuẩn IEEE-754 không xác định cách truyền đạt các thông số này.

**(Mở rộng và có thể mở rộng độ chính xác) Extended and extendable precisions**

Các định dạng mở rộng(extended) và có thể mở rộng(extendable) độ chính xác được khuyến nghị để mở rộng các giới hạn được sử dụng cho số học ngoài các định dạng cơ bản. Đặc biệt:

\*Một định dạng mở rộng độ chính xác là một định dạng số học mà mở rộng định dạng cơ bản với cả độ chính xác và phạm vi.

\*Một extandable precision format là một định dạng số học với độ chính xác và phạm vi được định nghĩa bởi người dùng

Các định dạng này được đặc trưng bởi các tham số b, p và , có thể khớp với các tham số của một định dạng thay thế và phải:

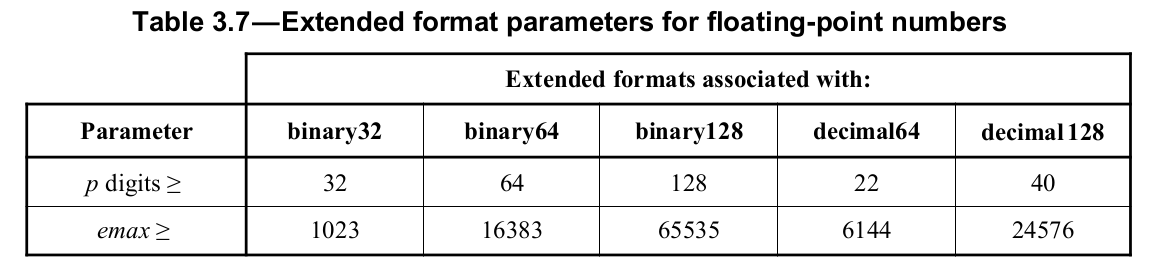
\*cung cấp các biểu diễn của dữ liệu dấu phẩy động được xác định trong 3.2 và 3.3

\*cung cấp các phép toán cho tiêu chuẩn đó, được định nghĩa ở phần 5, cho định dạng đó

Các mã hóa ở các định dạng này phải có chiều rộng cố định và có thể khớp với các mã hóa ở định dạng thay thế. Mỗi biểu diễn của một số dấu phẩy động có một mã hóa chính tắc duy nhất và có thể có các mã hóa không chính tắc. Mỗi NaN ở các định dạng này có một payload, có thể mã hóa thông tin chẩn đoán. Mỗi payload NaN có một mã hóa chính tắc và có thể có các mã hóa không chính tắc. Tất cả các khía cạnh khác của mã hóa cho các định dạng này là triển khai tự định nghĩa.

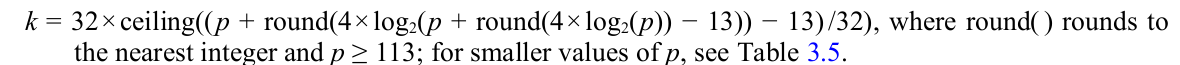
Các tiêu chuẩn ngôn ngữ phải xác định các cơ chế hỗ trợ extandable precision cho mỗi cơ số được hỗ trợ. Các tiêu chuẩn ngôn ngữ hỗ trợ extendable precision sẽ cho phép người dùng chỉ định p và emax. Các tiêu chuẩn ngôn ngữ cũng phải cho phép đặc tả extendable precision bằng cách chỉ định riêng p; trong trường hợp này emax phải được định nghĩa theo tiêu chuẩn ngôn ngữ là ít nhất 1000 × p khi p ≥ 237 bit ở định dạng nhị phân hoặc p ≥ 51 chữ số ở định dạng thập phân.

Các tiêu chuẩn hoặc triển khai ngôn ngữ phải hỗ trợ định dạng chính xác mở rộng mở rộng định dạng cơ bản rộng nhất được hỗ trợ trong cơ số đó. Bảng 3.7 quy định độ chính xác tối thiểu và phạm vi số mũ của định dạng độ chính xác mở rộng cho mỗi định dạng cơ bản.



CHÚ THÍCH 1: Đối với các extended format, phạm vi số mũ tối thiểu là của định dạng cơ bản rộng hơn tiếp theo, nếu có, trong khi precision tối thiểu là khoảng giữa định dạng cơ bản nhất định và định dạng cơ bản rộng hơn tiếp theo.

CHÚ THÍCH 2 - Để thay thế dữ liệu dấu phẩy động nhị phân, độ rộng k tính bằng bit của định dạng nhỏ nhất sẽ cho phép mã hóa một significand của ít nhất p bit được cho bởi:



Để trao đổi dữ liệu dấu phẩy động thập phân, độ rộng k tính bằng bit của định dạng nhỏ nhất sẽ cho phép mã hóa một precision ít nhất p chữ số được cho bởi:



Trong cả hai trường hợp, định dạng đã chọn có thể có precision lớn hơn (xem 3.4 và 3.5.2).

CHÚ THÍCH 3: Đối với các định dạng nhị phân, precision p phải ít nhất là 3, vì một số thuộc tính số không phù hợp với precision thấp hơn.

Tương tự, emax phải ít nhất là 2 để có thể hỗ trợ các phép toán trong 9.2

**7. Exceptions and default exception handling**

**7.1. Tổng quan về exception (ngoại lệ) và flags (cờ)**

Điều khoản này quy định năm loại ngoại lệ sẽ được báo hiệu khi chúng phát sinh; tín hiệu gọi xử lý mặc định hoặc thay thế cho ngoại lệ được báo hiệu. Đối với mỗi loại ngoại lệ, việc triển khai sẽ cung cấp một cờ trạng thái tương ứng.

Điều khoản này cũng chỉ định xử lý ngoại lệ không ngừng (non-stop exception) mặc định cho các tín hiệu ngoại lệ, nghĩa là cung cấp kết quả mặc định, tiếp tục thực thi và nâng cờ trạng thái tương ứng (ngoại trừ trường hợp exact underflow, xem 7.5). Phần 8 (ở sau) sẽ quy định các thuộc tính xử lý ngoại lệ thay thế cho các tín hiệu đó; một tiêu chuẩn ngôn ngữ (standard language) có thể chỉ định rằng một số thuộc tính đó được triển khai và sau đó xác định các phương tiện để người dùng kích hoạt chúng. Việc xử lý ngoại lệ kiểu mặc định hoặc thay thế cho một ngoại lệ cũng có thể báo hiệu các ngoại lệ khác (xem phần underflow và overflow, 7.4 và 7.5). Do đó, cờ trạng thái có thể được nâng lên bằng cách xử lý ngoại lệ mặc định hoặc thay thế; hoặc bằng hành động rõ ràng của người dùng (xem 5.7.4).

Với xử lý ngoại lệ mặc định, cờ trạng thái được nâng lên thường chỉ ra rằng ngoại lệ tương ứng đã được báo hiệu và xử lý theo mặc định. Các ngoại lệ được xử lý mà không nâng cờ trạng thái chỉ trong trường hợp exa ct underflow. Cờ trạng thái được nâng lên mà không có ngoại lệ được báo hiệu chỉ khi theo yêu cầu của người dùng. Cờ trạng thái sẽ chỉ được hạ xuống theo yêu cầu của người dùng. Người sử dụng phải có thể kiểm tra và thay đổi các cờ trạng thái riêng lẻ hoặc chung, đồng thời phải lưu và khôi phục tất cả cùng một lúc (xem 5.7.4).

Một chương trình không kế thừa cờ trạng thái từ một source khác bắt đầu thực thi với tất cả cờ trạng thái được hạ xuống. Các tiêu chuẩn ngôn ngữ phải chỉ định các giá trị mặc định trong trường hợp không có bất kỳ thông số người dùng rõ ràng nào, điều chỉnh:

* Có tồn tại bất kỳ cờ cụ thể nào hay không (theo nghĩa là có thể kiểm tra được bằng các phương tiện phi lập trình như trình debugger) bên ngoài phạm vi mà chương trình dùng để đặt hoặc kiểm tra cờ đó một cách rõ ràng. Tức là ngoài phạm vi thì chỉ có thể kiểm tra không rõ ràng bằng debugger.
* Khi cờ có phạm vi lớn hơn trong một hàm được gọi, liệu và khi một sự kiện không đồng bộ, chẳng hạn như nâng hoặc hạ cờ trong một luồng hoặc trình xử lý tín hiệu khác, có ảnh hưởng đến cờ được kiểm tra trong hàm được gọi đó hay không; điều này bao gồm các sự kiện phát sinh từ sự không đồng bộ rõ ràng trong chương trình và cả các sự kiện phát sinh từ sự không đồng bộ gây ra bởi ngôn ngữ hoặc cách triển khai.
* Khi một cờ có phạm vi lớn hơn trong một hàm được gọi, liệu trạng thái của cờ đó có thể được xác định bằng các phương tiện phi lập trình (chẳng hạn như debugger) trong hàm được gọi đó hay không.
* Các cờ được nâng lên trong các hàm được gọi có nâng các cờ lên trong các hàm gọi hay không.
* Các cờ được nâng lên trong các hàm gọi có nâng các cờ lên trong các hàm được gọi hay không.
* Có cho phép hay không và nếu có thì có nghĩa là, để chỉ định rằng các cờ phải được duy trì trong trường hợp không có bất kỳ program statement rõ ràng nào khác:
  + Các cờ đứng tại điểm bắt đầu thực thi một chức năng cụ thể được kế thừa từ môi trường bên ngoài, thường là một chức năng gọi.
  + Khi trả về từ hàm được gọi hoặc kết thúc một hàm được gọi, các cờ đứng trong một hàm gọi là các cờ đứng trong hàm được gọi tại thời điểm trả về chương trình hoặc kết thúc chương trình.

Lời gọi của bất kỳ hoạt động nào theo yêu cầu của tiêu chuẩn này báo hiệu nhiều nhất một ngoại lệ; các ngoại lệ bổ sung có thể được báo hiệu bằng cách xử lý ngoại lệ mặc định hoặc bằng cách xử lý ngoại lệ thay thế cho ngoại lệ đầu tiên. Xử lý ngoại lệ mặc định cho overflow (xem 7.4) báo hiệu ngoại lệ không chính xác (inexact). Xử lý ngoại lệ mặc định cho underflow (xem 7.5) báo hiệu ngoại lệ không chính xác (inexact) nếu kết quả mặc định là không chính xác (inexact). Việc xử lý ngoại lệ mặc định cho không hợp lệ (invalid) (xem 7.2) do toán hạng NaN báo hiệu có thể báo hiệu một ngoại lệ hoạt động không hợp lệ (invalid) khác để so sánh bằng các vị từ (predicate - true/false) báo hiệu không có thứ tự.

Lời gọi thao tác restoreFlags hoặc raiseFlags (xem 5.7.4) có thể nâng bất kỳ tổ hợp cờ trạng thái nào. Lệnh gọi của bất kỳ thao tác nào khác theo yêu cầu của tiêu chuẩn này, khi tất cả các trường hợp ngoại lệ được xử lý theo mặc định, có thể nâng tối đa hai cờ trạng thái, đó là overflow với inexact (xem 7.4) hoặc underflow với inexact (xem 7.5).

Đối với các hoạt động tính toán được xác định trong tiêu chuẩn này, các ngoại lệ được định nghĩa dưới đây sẽ được báo hiệu khi và chỉ khi một số điều kiện nhất định phát sinh. Do đó, các hoạt động này sẽ được tính toán theo cách tránh các tín hiệu có thể quan sát được của người dùng về các ngoại lệ đối với các điều kiện khác, ngay cả khi hoạt động được thực hiện trong phần mềm sử dụng các hoạt động báo hiệu ngoại lệ khác. Các phép toán không được tiêu chuẩn này quy định, chẳng hạn như số học phức tạp hoặc một số hàm siêu việt nhất định, phải báo hiệu các trường hợp ngoại lệ theo các định nghĩa dưới đây đối với các phép toán được xác định trong tiêu chuẩn này, nhưng điều đó có thể không phải lúc nào cũng kinh tế. Việc báo hiệu các ngoại lệ cho các hoạt động không được quy định trong tiêu chuẩn này được định nghĩa bởi ngôn ngữ.

LƯU Ý - Người dùng không thể phát hiện được các tín hiệu bị dư thừa của một ngoại lệ khi thực hiện một thao tác trong xử lý ngoại lệ mặc định. Những tín hiệu dư thừa như vậy có thể được người dùng phát hiện trong thuộc tính recordException để xử lý ngoại lệ thay thế (được khuyến nghị) (xem 8.2) và, trong trường hợp các vị từ (predicate) báo hiệu không có thứ tự với toán hạng NaN báo hiệu, trong các thuộc tính xử lý ngoại lệ thay thế khác nếu sub-exception (xem 8.1) được hỗ trợ.

**7.2. Invalid operation**

Ngoại lệ invalid operation được báo hiệu khi và chỉ khi không có kết quả hữu ích có thể xác định được. Trong những trường hợp này, các toán hạng (operand) là không hợp lệ để thực hiện phép toán (operation).

Đối với các phép toán tạo ra kết quả ở định dạng dấu phẩy động, kết quả mặc định của một phép toán báo hiệu ngoại lệ hoạt động không hợp lệ phải là một qNaN và cung cấp một số thông tin chẩn đoán (xem 6.2). Các hoạt động này là:

a) bất kỳ hoạt động tính toán chung nào trên một (signaling) sNaN (xem 6.2), ngoại trừ một số chuyển đổi (xem 5.12)

b) phép nhân: nhân (0, ) hoặc nhân

c) fusedMultiplyAdd: fusedMultiplyAdd(0, ∞, c) or fusedMultiplyAdd(∞, 0, c) trừ trường hợp c là qNaN; nếu c là một qNaN thì ngoại lệ invalid operation có được báo hiệu hay không sẽ tùy thuộc vào cách triển khai.

d) addition or subtraction or fusedMultiplyAdd: phép trừ độ lớn của các số vô hạn, chẳng hạn như: addition(+ ∞, −∞)

e) phép chia (division): division (0,0) hoặc division

f) phép đồng dư (remainder): remainder(x, y), khi y là zero hoặc x là vô hạn và không phải là NaN

g) phép khai căn (square root): nếu toán hạng nhỏ hơn 0

h) lượng tử hóa (quantize) khi kết quả không phù hợp với định dạng đích hoặc khi một toán hạng là hữu hạn và toán hạng kia là vô hạn

Đối với các phép toán không tạo ra kết quả ở định dạng dấu phẩy động, các hoạt động báo hiệu ngoại lệ operation invalid là:

i) bất kỳ hoạt động tính toán báo hiệu (signaling-computational) nào trên sNaN (xem 6.2); sau đó, trong xử lý ngoại lệ mặc định, hoạt động được đánh giá với các qNaN thay cho các toán hạng sNaN để xác định kết quả, đối với các so sánh báo hiệu không có thứ tự (unordered-signaling) có thể báo hiệu một ngoại lệ invalid operation khác

j) chuyển đổi số dấu phẩy động sang định dạng số nguyên, khi giá trị nguồn là NaN, infinity hoặc một giá trị sẽ chuyển đổi thành số nguyên bên ngoài phạm vi của định dạng kết quả theo thuộc tính làm tròn hiện hành

k) so sánh bằng cách so sánh các vị từ báo hiệu không có thứ tự được liệt kê trong Bảng 5.2, khi các toán hạng không có thứ tự

l) logB(NaN), logB(∞), or logB(0) khi logBFormat là một định dạng số nguyên (xem 5.3.3).

7.3. Chia cho 0

Ngoại lệ divideByZero phải được báo hiệu khi và chỉ khi một kết quả vô hạn chính xác được xác định cho một phép toán trên các toán hạng hữu hạn. Kết quả mặc định của DivideByZero phải là một ∞ và dấu của kết quả được xác định đúng theo phép toán:

* Đối với phép chia (division), khi số chia là 0 và số bị chia là một số hữu hạn khác 0, thì dấu của vô cùng là XOR của các dấu của các toán hạng (xem 6.3).
* Đối với logB (0) khi logBFormat là một định dạng dấu phẩy động, dấu của vô cùng là trừ (−∞).

7.4. Overflow

Ngoại lệ overflow sẽ được báo hiệu khi và chỉ khi số hữu hạn lớn nhất của định dạng đích bị vượt quá về độ lớn bởi kết quả dấu phẩy động làm tròn (xem Điều 4) khi phạm vi số mũ không bị giới hạn. Nói dễ hiểu là số mũ của phép làm tròn vượt quá .

Kết quả mặc định sẽ được xác định bởi thuộc tính hướng làm tròn (rounding-direction) và dấu của kết quả trung gian như sau:

a) roundTiesToEven và roundTiesToAway mang tất cả overflow sang ∞ bằng dấu của kết quả trung gian.

b) roundTowardZero mang tất cả overflow vào số hữu hạn lớn nhất của định dạng với dấu của kết quả trung gian.

c) roundTowardNegative mang positve overflow đến số hữu hạn lớn nhất của định dạng và mang negative overflow đến −∞.

d) roundTowardPositive mang negative overflow tới số hữu hạn âm nhất của định dạng và mang positve overflow đến + ∞.

Ngoài ra, trong xử lý ngoại lệ mặc định đối với overflow, cờ overflow phải được nâng lên và sau đó báo hiệu tiếp ngoại lệ không chính xác (inexact).

7.5. Underflow

Ngoại lệ underflow sẽ được báo hiệu khi phát hiện một kết quả nhỏ khác 0. Đối với các định dạng nhị phân, đây sẽ là:

a) after rounding (sau khi làm tròn) - khi kết quả khác 0 được tính như thể phạm vi số mũ không bị giới hạn nằm đúng giữa ± b emin, hoặc

b) before rounding - khi một kết quả khác 0 được tính như thể cả phạm vi số mũ và độ chính xác đều không bị giới hạn sẽ nằm đúng giữa ± b emin.

Người triển khai phải chọn cách phát hiện độ nhỏ, nhưng phải phát hiện độ nhỏ theo cùng một cách đối với tất cả các phép toán trong cơ số hai, bao gồm các phép toán chuyển đổi theo thuộc tính làm tròn nhị phân.

Đối với các định dạng thập phân, độ nhỏ được phát hiện trước khi làm tròn - khi kết quả khác 0 được tính như thể cả phạm vi số mũ và độ chính xác không bị giới hạn sẽ nằm giữa ± b emin.

Việc xử lý ngoại lệ mặc định cho underflow phải luôn mang lại kết quả làm tròn. Phương pháp phát hiện độ nhỏ không ảnh hưởng đến kết quả làm tròn được cung cấp, có thể là 0, dưới chuẩn hoặc ± b emin.

Ngoài ra, trong xử lý ngoại lệ mặc định cho underflow, nếu kết quả làm tròn là không chính xác (inexact) - nghĩa là, nó khác với những gì đã được tính toán khi cả phạm vi số mũ và độ chính xác không bị giới hạn - cờ undeflow sẽ được nâng lên và ngoại lệ không chính xác (inexact) (xem 7.6) sẽ được báo hiệu. Nếu kết quả làm tròn là chính xác, không có cờ nào được nâng lên và không có ngoại lệ không chính xác nào được báo hiệu. Đây là trường hợp duy nhất trong tiêu chuẩn này về việc xử lý mặc định tín hiệu ngoại lệ nhận tín hiệu ngoại lệ không nâng cờ tương ứng. Một tín hiệu underflow như vậy không có tác dụng quan sát được theo cách xử lý mặc định.

7.6 Inexact

Trừ khi được quy định khác (ví dụ: 5.8 và 5.9), một phép toán đưa ra kết quả số mà không báo hiệu có ngoại lệ nào khác sẽ báo hiệu không chính xác nếu kết quả làm tròn của nó khác với kết quả đã được tính toán khi cả dải số mũ và độ chính xác không bị giới hạn. Kết quả làm tròn sẽ được chuyển đến đích. Nôm na là ngoại lệ này xảy ra khi một kết quả chính xác y không được biểu diễn chính xác.

LƯU Ý - Việc xử lý ngoại lệ mặc định cho overflow sẽ nâng cờ tràn và báo hiệu không chính xác (inexact). Khi kết quả làm tròn không phải là exact subnormal, việc xử lý ngoại lệ mặc định cho underflow sẽ nâng cờ underflow và báo hiệu inexact. Khi tất cả các ngoại lệ này được xử lý theo mặc định, cờ inexact luôn được nâng lên khi cờ overflow hoặc cờ underflow được nâng lên.

8 Alternate exception handling attributes

8.1 Tổng quát

Các tiêu chuẩn ngôn ngữ phải xác định và yêu cầu triển khai cung cấp các phương tiện để người dùng liên kết các thuộc tính xử lý ngoại lệ thay thế với các khối (xem 4.1). Các trình xử lý ngoại lệ thay thế chỉ định danh sách các ngoại lệ và các hành động cần thực hiện cho mỗi ngoại lệ được liệt kê nếu nó được báo hiệu. Các tiêu chuẩn ngôn ngữ phải xác định danh sách ngoại lệ chứa bất kỳ tập hợp con nào trong số các ngoại lệ được liệt kê trong Điều 7: invalid operation, devideByZero, overflow, underflow hoặc inexact. Các tiêu chuẩn ngôn ngữ cũng nên xác định danh sách ngoại lệ chứa:

* AllExceptions: tất cả năm trường hợp ngoại lệ được liệt kê trong Điều khoản 7, hoặc
* bất kỳ tập hợp con nào của ngoại lệ phụ - các trường hợp con của các ngoại lệ trong Điều 7 (ví dụ: các trường hợp con của ngoại lệ invalid operation trong 7.2); hành động cho một ngoại lệ con được liệt kê được thực hiện khi và chỉ khi thao tác đó là một trường hợp cụ thể cho ngoại lệ con; tên ngoại lệ phụ được xác định theo ngôn ngữ.

Các tiêu chuẩn ngôn ngữ nên xác định tất cả các thuộc tính xử lý ngoại lệ thay thế của điều khoản này. Cụ thể, các tiêu chuẩn ngôn ngữ phải xác định ít nhất một thuộc tính xử lý ngoại lệ thay thế bị trì hoãn (xem 8.3) cho mỗi một trong năm trường hợp ngoại lệ được liệt kê trong Điều 7. Cú pháp và phạm vi cho các đặc tả của các giá trị thuộc tính đó là do ngôn ngữ xác định.

Thay đổi thuộc tính xử lý ngoại lệ không báo hiệu bất kỳ ngoại lệ nào.

8.2 Resuming alternate exception handling attributes

Việc liên kết thuộc tính xử lý ngoại lệ thay thế tiếp tục với một khối có nghĩa là: xử lý các ngoại lệ ngụ ý theo resuming attribute specified được chỉ định và tiếp tục thực thi khối được liên kết. Việc triển khai phải hỗ trợ các thuộc tính tiếp tục sau:

- default (raise flag):

Cung cấp xử lý ngoại lệ mặc định (xem Điều 7) trong khối liên kết mặc dù xử lý ngoại lệ thay thế có thể có hiệu lực trong phạm vi rộng hơn.

- raiseNoFlag

Cung cấp xử lý ngoại lệ mặc định (xem Điều 7) mà không cần nâng cờ trạng thái tương ứng.

- mayRaiseFlag:

Cung cấp cách xử lý ngoại lệ mặc định (xem Điều 7), ngoại trừ các ngôn ngữ xác định xem cờ có được nâng lên hay không. Các ngôn ngữ có thể trì hoãn việc triển khai để đạt được hiệu suất.

- recordException:

Cung cấp cách xử lý ngoại lệ mặc định (xem Điều 7) và ghi lại ngoại lệ tương ứng bất cứ khi nào Điều 7 quy định việc nâng cờ. Ghi lại ngoại lệ có nghĩa là lưu trữ mô tả về ngoại lệ, bao gồm các chi tiết do ngôn ngữ xác định có thể bao gồm phép toán và các toán hạng hiện tại cũng như vị trí của ngoại lệ. Các tiêu chuẩn ngôn ngữ xác định các phép toán để chuyển đổi các mô tả ngoại lệ sang và từ các chuỗi ký tự, và để kiểm tra, lưu và khôi phục các mô tả ngoại lệ.

- substitute(x):

Có thể chỉ định cho bất kỳ ngoại lệ nào: thay thế kết quả mặc định của một hoạt động ngoại lệ đó bằng một biến hoặc biểu thức x. Thời gian và phạm vi mà x được đánh giá là do ngôn ngữ xác định.

- substituteXor(x):

Có thể chỉ định cho bất kỳ ngoại lệ nào phát sinh từ các phép toán nhân hoặc chia: như substitute (x), nhưng thay thế kết quả mặc định của một phép toán ngoại lệ đó bằng | x | và, nếu | x | không phải là NaN, lấy bit dấu từ XOR của các dấu của toán hạng.

- abruptUnderflow:

Khi underflow được báo hiệu vì một kết quả nhỏ khác 0 được phát hiện, thay thế kết quả mặc định bằng một số 0 cùng dấu hoặc kết quả làm tròn bình thường tối thiểu có cùng dấu, nâng cờ underflow và báo hiệu ngoại lệ không chính xác. Khi thuộc tính roundTiesToEven, roundTiesToAway hoặc roundTowardZero có thể áp dụng, độ lớn kết quả làm tròn sẽ bằng 0. Khi thuộc tính roundTowardPositive được áp dụng, độ lớn kết quả làm tròn phải là độ lớn bình thường tối thiểu cho các kết quả cực nhỏ dương tính và bằng không cho các kết quả cực nhỏ âm tính. Khi thuộc tính roundTowardNegative được áp dụng, độ lớn kết quả làm tròn phải là độ lớn bình thường tối thiểu cho các kết quả cực nhỏ âm và bằng không cho các kết quả cực nhỏ dương tính. Thuộc tính này không ảnh hưởng đến biểu diễn các toán hạng subnormal.

Đối với các phép toán số học tăng cường (9.5), trả về hai kết quả (phép toán (x y) và lỗi của nó, trong đó phép toán là +, - hoặc ×), nếu underflow được báo hiệu vì phép toán x y làm tròn bằng roundTiesTowardZero sẽ báo hiệu underflow, cả hai kết quả là 0 với dấu của phép toán x y. Nếu underflow được báo hiệu vì lỗi (phép toán (x y) - roundTiesTowardZero (phép toán (x y)) khác 0 và nằm đúng giữa ± b emin, kết quả lỗi mặc định được thay thế bằng 0 với dấu của (x operation y - roundTiesTowardZero (x operation y)). Những trường hợp này nâng cờ underflow và báo hiệu ngoại lệ không inexact.

8.3 Immediate and delayed alternate exception handling attributes

Liên kết xử lý ngoại lệ thay thế với một khối có nghĩa là: xử lý (các) ngoại lệ được chỉ định theo thuộc tính được chỉ định. Nếu ngoại lệ được chỉ định được báo hiệu thì, tùy thuộc vào ngoại lệ và thuộc tính xử lý ngoại lệ, việc thực thi khối được liên kết có thể bị hủy bỏ ngay lập tức hoặc có thể tiếp tục với xử lý mặc định. Trong trường hợp thứ hai, việc xử lý ngoại lệ bị trì hoãn và diễn ra khi khối liên kết kết thúc bình thường. Xử lý ngoại lệ trì hoãn là hoàn toàn xác định, trong khi giấy phép xử lý ngoại lệ ngay lập tức nhưng không yêu cầu triển khai để trade determinism đối với hiệu suất, bởi vì các kết quả trung gian được tính toán trong khối liên kết có thể không xác định.

Các tiêu chuẩn ngôn ngữ phải xác định và yêu cầu triển khai cung cấp các thuộc tính sau:

- Khối xử lý ngoại lệ thay thế ngay lập tức được liên kết với một khối: nếu ngoại lệ được chỉ định được báo hiệu, từ bỏ việc thực thi khối được liên kết càng sớm càng tốt và thực thi khối xử lý, sau đó tiếp tục thực thi ở nơi mà việc thực thi sẽ tiếp tục sau khi kết thúc bình thường của khối được liên kết, theo ngữ nghĩa của ngôn ngữ.

- Khối xử lý ngoại lệ thay thế bị trì hoãn được liên kết với một khối: nếu ngoại lệ được chỉ định được báo hiệu, hãy xử lý nó theo mặc định cho đến khi khối được liên kết kết thúc bình thường, sau đó thực thi khối xử lý, sau đó tiếp tục thực thi ở nơi quá trình thực thi sẽ tiếp tục sau khi kết thúc bình thường của khối được liên kết, theo ngữ nghĩa của ngôn ngữ.

- Chuyển giao ngay lập tức được liên kết với một khối: nếu ngoại lệ được chỉ định được báo hiệu, hãy chuyển quyền điều khiển càng sớm càng tốt; không có khả năng trở lại.

- Truyền trễ được liên kết đến một khối: nếu ngoại lệ được chỉ định được báo hiệu, hãy xử lý nó theo mặc định cho đến khi khối liên kết kết thúc bình thường, sau đó chuyển quyền điều khiển; không có khả năng trở lại.

Xử lý ngoại lệ thay thế ngay lập tức cho underflow sẽ được gọi khi underflow được báo hiệu, cho dù kết quả mặc định là exact hay inexact. Xử lý ngoại lệ thay thế bị trì hoãn cho underflow sẽ chỉ được gọi cho các tín hiệu underflow tương ứng với kết quả mặc định không chính xác mà cờ underflow sẽ được nâng lên.

CHÚ THÍCH 1: Việc xử lý ngoại lệ thay thế bị trì hoãn đối với một ngoại lệ được liệt kê trong Điều 7 (nhưng không phải ngoại lệ phụ) có thể được thực hiện bằng cờ trạng thái thử nghiệm. Tuy nhiên được thực hiện, cờ trạng thái tương ứng với ngoại lệ được chỉ định phải được lưu trước khi bắt đầu khối liên kết và sau đó được hạ xuống. Ở cuối khối được liên kết, cờ trạng thái hiện tại sẽ được lưu và cờ trạng thái đã lưu trước đó sẽ được khôi phục. Sau đó, cờ trạng thái đã lưu gần đây sẽ được kiểm tra để xác định xem có thực thi khối xử lý hay điều khiển chuyển giao hay không.

CHÚ THÍCH 2: Việc xử lý ngoại lệ thay thế ngay lập tức đối với một ngoại lệ có thể được thực hiện bằng bẫy hoặc đối với các ngoại lệ được liệt kê trong Điều 7 khác với quy trình dưới đây, bằng cách thử nghiệm các cờ trạng thái sau mỗi thao tác hoặc ở cuối khối liên kết. Do đó, đối với các trường hợp ngoại lệ được liệt kê trong Điều 7 khác với quy trình dưới đây, việc xử lý ngoại lệ ngay lập tức có thể được thực hiện với cơ chế tương tự như xử lý ngoại lệ chậm trễ, nếu không có cơ chế thực hiện tốt hơn. Bất kể được triển khai như thế nào, nếu ngoại lệ được chỉ định không được báo hiệu trong khối liên kết, thì cờ trạng thái tương ứng sẽ không được thay đổi. Nếu ngoại lệ được chỉ định được báo hiệu trong khối liên kết, gây ra việc thực thi khối xử lý hoặc chuyển quyền điều khiển, thì trạng thái của cờ trạng thái tương ứng có thể không xác định.

CHÚ THÍCH 3: Chuyển giao là một thành ngữ dành riêng cho ngôn ngữ để chỉ chuyển giao kiểm soát không thể tiếp tục. Các tiêu chuẩn ngôn ngữ có thể đưa ra một số thành ngữ chuyển giao như:

* + break: Bỏ qua khối được liên kết và tiếp tục thực thi trong đó quá trình thực thi sẽ tiếp tục sau khi kết thúc bình thường của khối được liên kết, theo ngữ nghĩa của ngôn ngữ.
  + throw exceptionName: Làm cho một exceptionName không được xử lý cục bộ mà được báo hiệu cho lần xử lý tiếp theo trong phạm vi, có lẽ là hàm đã gọi chương trình con hiện tại, theo ngữ nghĩa của ngôn ngữ đó. Người gọi có thể xử lý Tên ngoại lệ theo mặc định hoặc bằng cách xử lý thay thế, chẳng hạn như báo hiệu Tên ngoại lệ cho các chương trình con gọi ra cao hơn tiếp theo.
  + goto label: Nhảy; nhãn có thể là cục bộ hoặc toàn cục tùy theo ngữ nghĩa của ngôn ngữ.