Tema 1 - Escalas, condición y estabilidad

Curso de Física Computacional

M. en C. Gustavo Contreras Mayén

Facultad de Ciencias - UNAM

31 de enero de 2020

Objetivo

Al concluir el Tema 1, el alumno:

1. En el diseño de algoritmos para la solución numérica de problemas de la física, aplicará los conceptos de: *Condición*, *Estabilidad* y *Eficiencia*, apoyándose en la teoría de representación de números en la computadora y de la teoría de propagación de errores.

¿Qué es la física computacional?

La física computacional es una nueva manera de hacer investigación en física, próxima al experimento y a la teoría.

En el laboratorio se realizan mediciones en sistemas físicos reales (restringida a la factibilidad de recursos técnicos), y que luego los físicos teóricos explican esas mediciones mediante las teorías.

Áreas de investigación en la física

- Problemas que no tienen solución analítica.

Áreas de investigación en la física

- Problemas que no tienen solución analítica.
- Validar aproximaciones y hacer efectivas las teorías propuestas.

Áreas de investigación en la física

- Problemas que no tienen solución analítica.
- Validar aproximaciones y hacer efectivas las teorías propuestas.
- Comparar cuantitativamente teorías y mediciones experimentales.

Áreas de investigación en la física

- Problemas que no tienen solución analítica.
- Validar aproximaciones y hacer efectivas las teorías propuestas.
- Comparar cuantitativamente teorías y mediciones experimentales.
- Visualizar conjuntos de datos complejos.

Áreas de investigación en la física

- Problemas que no tienen solución analítica.
- Validar aproximaciones y hacer efectivas las teorías propuestas.
- Comparar cuantitativamente teorías y mediciones experimentales.
- Visualizar conjuntos de datos complejos.
- Control y medición de experimentos.

La física computacional aplicada

- Predicción del clima

La física computacional aplicada

- Predicción del clima
- Superconductividad

La física computacional aplicada

- Predicción del clima
- Superconductividad
- Genoma Humano

La física computacional aplicada

- Predicción del clima
- Superconductividad
- Genoma Humano
- Visión y lenguaje

La física computacional aplicada

- Predicción del clima
- Superconductividad
- Genoma Humano
- Visión y lenguaje
- Fusión nuclear

La física computacional aplicada

- Predicción del clima
- Superconductividad
- Genoma Humano
- Visión y lenguaje
- Fusión nuclear
- Oceanografía

La física computacional aplicada

- Predicción del clima
- Superconductividad
- Genoma Humano
- Visión y lenguaje
- Fusión nuclear
- Oceanografía
- Ciencia de los materiales

La física computacional aplicada

- Predicción del clima
- Superconductividad
- Genoma Humano
- Visión y lenguaje
- Fusión nuclear
- Oceanografía
- Ciencia de los materiales
- Diseño de semiconductores

La física computacional aplicada

- Predicción del clima
- Superconductividad
- Genoma Humano
- Visión y lenguaje
- Fusión nuclear
- Oceanografía
- Ciencia de los materiales
- Diseño de semiconductores
- Astrofísica relativista

La física computacional aplicada

- Predicción del clima
- Superconductividad
- Genoma Humano
- Visión y lenguaje
- Fusión nuclear
- Oceanografía
- Ciencia de los materiales
- Diseño de semiconductores
- Astrofísica relativista
- Sistemas de combustión

La física computacional aplicada

- Predicción del clima
- Superconductividad
- Genoma Humano
- Visión y lenguaje
- Fusión nuclear
- Oceanografía
- Ciencia de los materiales
- Diseño de semiconductores
- Astrofísica relativista
- Sistemas de combustión
- Estructura biológica

La física computacional aplicada

- Predicción del clima
- Superconductividad
- Genoma Humano
- Visión y lenguaje
- Fusión nuclear
- Oceanografía
- Ciencia de los materiales
- Diseño de semiconductores
- Astrofísica relativista
- Sistemas de combustión
- Estructura biológica
- Diseño de fármacos

La física computacional aplicada

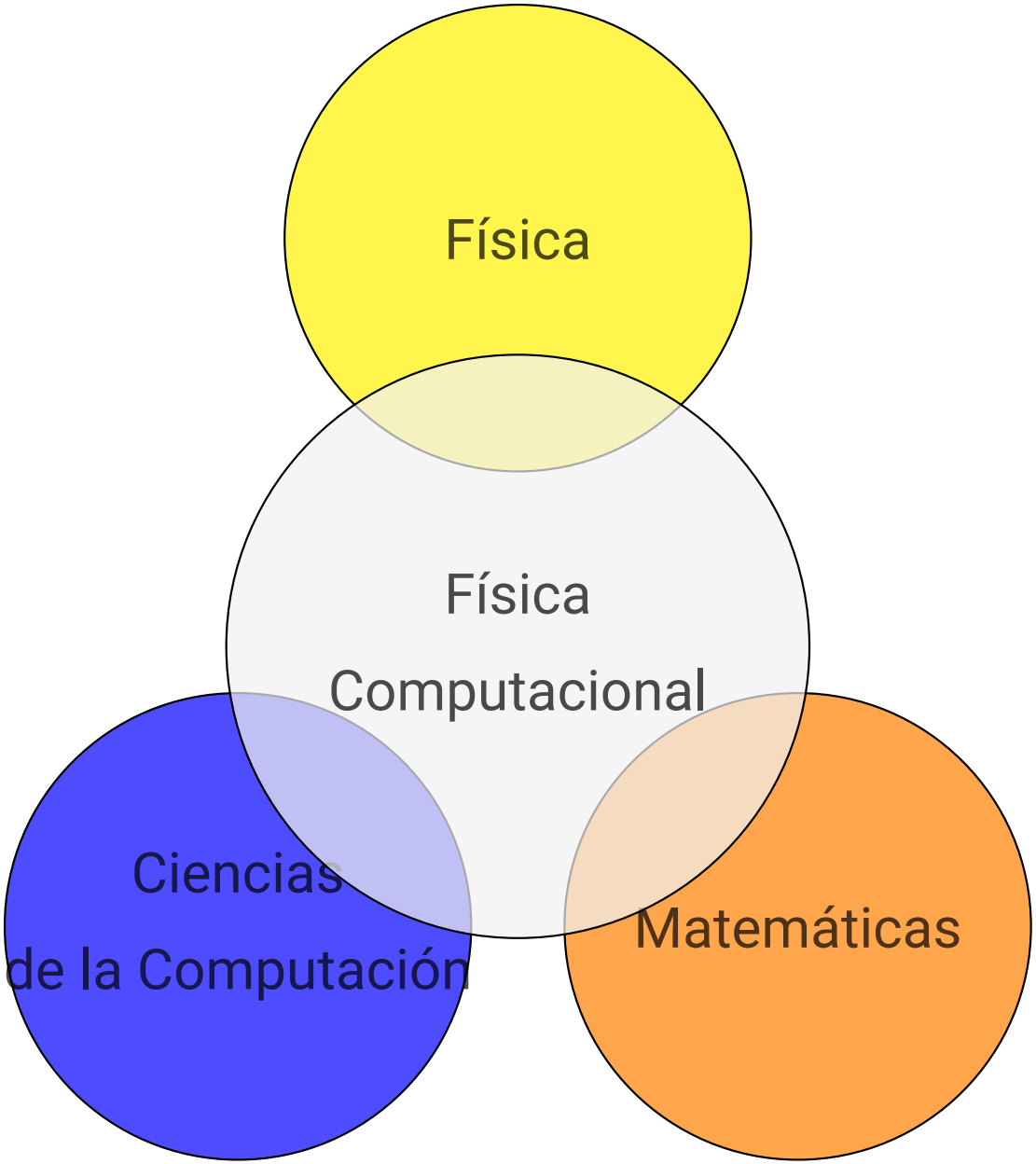
- Predicción del clima
- Superconductividad
- Genoma Humano
- Visión y lenguaje
- Fusión nuclear
- Oceanografía
- Ciencia de los materiales
- Diseño de semiconductores
- Astrofísica relativista
- Sistemas de combustión
- Estructura biológica
- Diseño de fármacos
- Turbulencia

La física computacional aplicada

- Predicción del clima
- Superconductividad
- Genoma Humano
- Visión y lenguaje
- Fusión nuclear
- Oceanografía
- Ciencia de los materiales
- Diseño de semiconductores
- Astrofísica relativista
- Sistemas de combustión
- Estructura biológica
- Diseño de fármacos
- Turbulencia
- Recuperación de petróleo y gas

La física computacional aplicada

- Predicción del clima
- Superconductividad
- Genoma Humano
- Visión y lenguaje
- Fusión nuclear
- Oceanografía
- Ciencia de los materiales
- Diseño de semiconductores
- Astrofísica relativista
- Sistemas de combustión
- Estructura biológica
- Diseño de fármacos
- Turbulencia
- Recuperación de petróleo y gas
- Cromodinámica cuántica


Física
Física
Computacional
Ciencias
de la Computación
Matemáticas

Alcance del curso

El curso está diseñado para ofrecer una introducción a los métodos numéricos aplicados a la física.

Se da un punto de referencia para continuar profundizando de manera particular en temas específicos.

Alcance del curso

El desarrollo de las habilidades de programación, están en función del tiempo dedicado al trabajo fuera de la clase, pero consideramos que se abren un panorama diferente para abordar ya sea un servicio social, tesis o proyecto de trabajo para un posgrado.

Representación de números en la computadora

Las computadoras son herramientas muy poderosas, pero son tienen un alcance finito.

Un problema que se presenta en el diseño de computadora es cómo representar un número arbitrario usando una cantidad finita de espacio de memoria y cómo tratar con las limitaciones que surgen de esta representación.

Como consecuencia de que las memorias de la computadora se basan en un estado de magnetización o electrónico de un giro que apunta hacia arriba o hacia abajo, las unidades más elementales de memoria de la computadora son los dos enteros binarios (bits) `0` y `1`.

Rango de representación

Esto significa que la forma en que se almacenan los números en memoria, es como cadenas largas de ceros y unos, es decir, de modo binario.

De tal manera N bits puede almacenar números enteros en el rango $[0, 2^N]$, pero debido a que el signo del número entero está representado por el primer bit (un bit cero para números positivos), el rango real disminuye a $[0, 2^{N-1}]$.

Otras bases de operación

La representación binaria de números a través de ceros y unos, funciona y opera bien para las computadoras, pero no para los usuarios.

Es por ello que las cadenas binarias se convierten en números octal, decimal o hexadecimal antes de que los resultados se presenten a los usuarios.

Los números octales y hexadecimales son también oportunos porque en la conversión no se pierde precisión.

Pero no todo queda bien ya que nuestras reglas decimales de aritmética no funcionan para ellos.

Desventaja de la aritmética binaria

La conversión a números decimales hace que los números sean más fáciles de trabajar, pero a menos que el número sea una potencia de 2, el proceso conduce a una disminución en la precisión.

Longitud de palabra

Una descripción de un sistema informático particular indica normalmente la *longitud de la palabra*, es, el número de bits utilizados para almacenar un número.

La longitud se expresa en bytes, donde

$$1 \text{ byte} \equiv 1 \text{ B} \equiv 8 \text{ bit}$$

Unidades de medida

Tanto la memoria como el almacenamiento se mide en bytes, kilobytes, megabytes, gigabytes, terabytes y petabytes (10^{15}).

No debemos de confundirnos al elegir las unidades de medida, ya que el prefijo *kilo*, no siempre equivale a 1000

$$1\,\mathrm{k} = 1\,\mathrm{kB} = 2^{10}\mathrm{B} = 1024\,\text{byte}$$

La memoria de las primeras computadoras usaban palabras de 8 bits, esto implicaba que el mayor entero era $2^7 = 128$, (7 debido a 1 bit para el signo).

Si queríamos almacenar un número más grande, tanto el hardware como el software se diseñaban para generar un desbordamiento (overflow).

Usando 64 bits, se permiten enteros en el rango $1 - 2^{63} \simeq 10^{19}$.

Que podría en apariencia ser una rango mucho más grande, pero no lo es cuando se compara con el rango de escalas que tenemos en el mundo real: del radio del universo al radio de un protón, tenemos aproximadamente 10^{41} órdenes de magnitud.

Tipos de notación

Los números reales se representan en computadoras en notación de punto fijo o punto flotante.

La notación de punto fijo se puede utilizar para números con un número fijo de lugares luego del punto decimal (raíz) o para números enteros. Tiene como ventaja que se utiliza la aritmética complementaria de dos y por tanto, almacenar enteros exactamente.[1].

[1]Para ampliar el tema, revisa la guía de operaciones binarias

Números de punto fijo

En la representación de punto fijo con N bits y con un formato de complemento de dos, un número se representa como

$$N_{\text{fijo}} = \text{signo} \times (\alpha_n\, 2^n + \alpha_{n-1}\, 2^{n-1} + \ldots + \\ + \alpha_0\, 2^0 + \ldots + \alpha_{-m}\, 2^{-m}) \tag{1}$$

donde $n + m = N - 2$.

Números de punto fijo

Es decir, 1 bit se utiliza para almacenar el signo, con los restantes $(N-1)$ bits se usan para almacenar los α_i valores (se entienden las potencias de 2).

Los valores particulares para N, m y n son dependientes de la máquina.

Rango de los enteros

Los enteros se representan típicamente con 4 bytes (32 bits) de longitud, en el rango

$$-2147483648 \leq 4 - \text{B entero} \leq 2147483647$$

Ventaja de la representación

Una ventaja de la representación (**??**) es que se puede contar con todos los números de punto fijo para tener el mismo error absoluto de 2^{-m-1} (el término se deja fuera del extremo derecho de (**??**).

Desventaja de la representación

La correspondiente desventaja es que los números pequeños (aquellos para los cuales la primera cadena de valores de α son ceros) tienen grandes errores relativos.

Utilidad de los enteros

Dado que en el mundo real los errores relativos tienden a ser más importantes que los absolutos, los números enteros se utilizan principalmente para fines de conteo y en aplicaciones especiales.

Números de punto flotante

En la mayoría de los cálculos científicos se utilizan números de punto flotante de doble precisión (64 bits = 8 Bytes).

La representación en las computadoras de números en punto flotante de números es una versión binaria de lo que comúnmente se conoce como notación científica.

Notación científica

Por ejemplo, la velocidad de la luz en notación científica

$$c = +2.997\,924\,58 \times 10^{8}\,\mathrm{m\,s^{-1}}$$

y en notación de ingeniería

$+0.299\,792\,458 \times 10^{9}\,\mathrm{m\,s^{-1}}$ ó $0.299\,795\,498 \times 10^{9}\,\mathrm{m\,s^{-1}}$

Notación científica

Por ejemplo, la velocidad de la luz en notación científica

$$c = +2.997\,924\,58 \times 10^{8}\,\mathrm{m\,s^{-1}}$$

y en notación de ingeniería

$+0.299\,792\,458 \times 10^{9}\,\mathrm{m\,s^{-1}}$ ó $0.299\,795\,498 \times 10^{9}\,\mathrm{m\,s^{-1}}$

En cada uno de estos casos, el número al frente se denomina **mantisa** y contiene nueve cifras significativas. La potencia 10 a la que se eleva se llama **exponente**, con el signo más $(+)$ incluido, como recordatorio de que estos números pueden ser negativos.

Almacenamiento

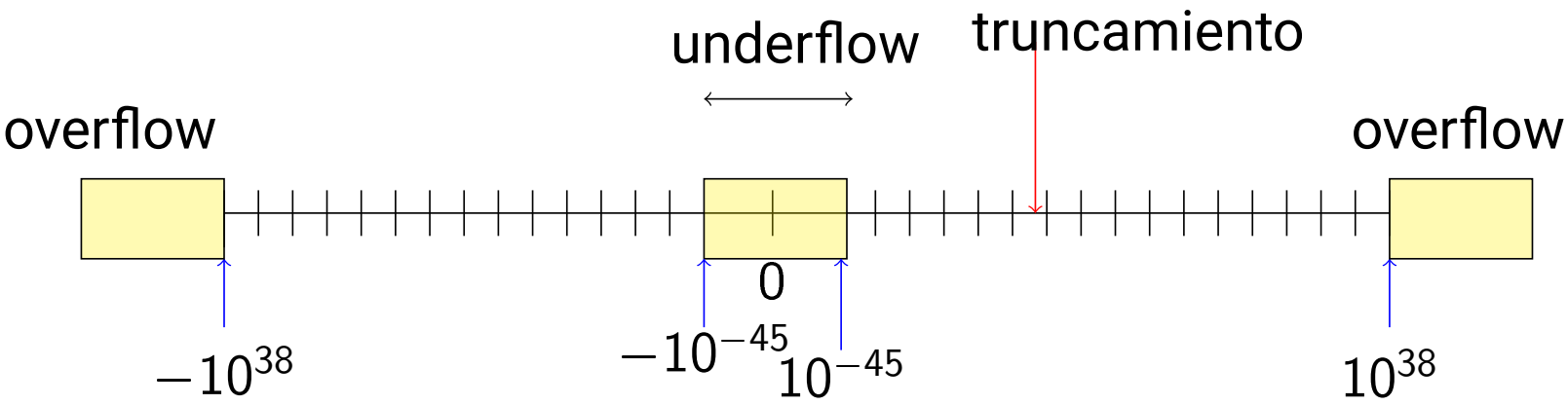
Los números de punto flotante se almacenan en la computadora como una concatenación del bit de signo, el exponente y la mantisa.

Debido a que sólo se almacena un número finito de bits, el conjunto de números de punto flotante que el equipo puede almacenar exactamente, **números de máquina**, es mucho menor que el conjunto de los números reales.

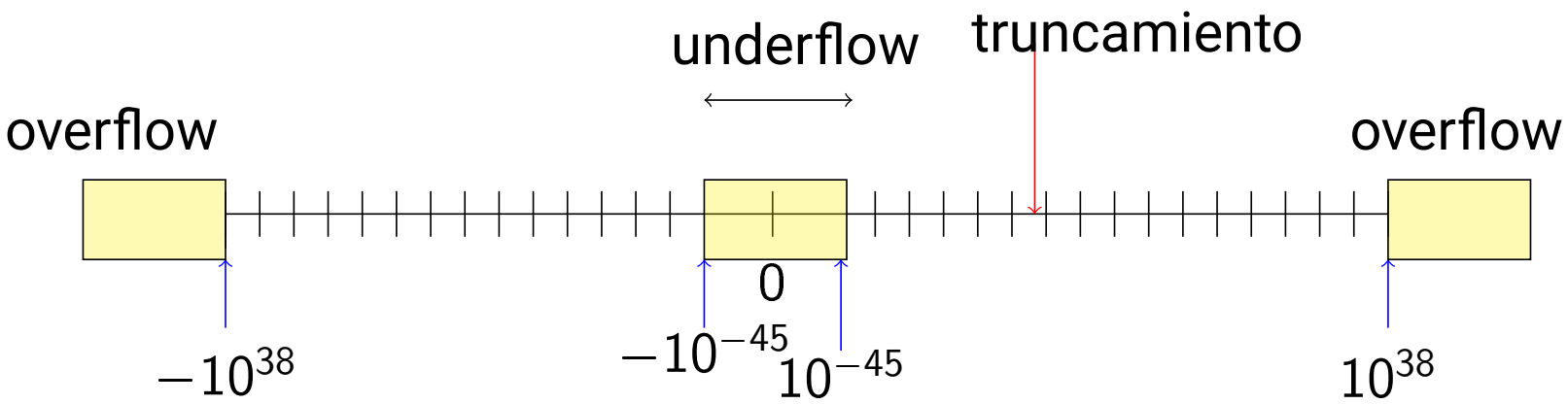
Errores de almacenamiento: overflow

Los números de máquina tienen un máximo y un mínimo.

Si se supera el máximo, se produce una condición de error conocida como desbordamiento -overflow-.



Errores de almacenamiento: underflow



Si se cae por debajo del mínimo, se produce una condición de error conocida como subflujo -underflow-.

Manejo del overflow y del underflow

En este último caso, el software y el hardware se pueden configurar para que los overflow se pongan a cero sin que se le avise al usuario.

En contraste, los overflow suelen detener la ejecución.

Valor en memoria

La relación real entre lo que se almacena en la memoria y el valor de un número de punto flotante es algo indirecta, habiendo un número de casos especiales y relaciones utilizadas a lo largo de los años.

Valor en memoria

En el pasado, cada sistema operativo de computadora y cada lenguaje de programación contenían sus propios estándares para números de punto flotante.

Diferentes normas implican que el mismo programa que se ejecuta correctamente en diferentes equipos podría dar resultados diferentes.

Valor en memoria

A pesar de que los resultados eran sólo ligeramente diferentes, el usuario nunca podría estar seguro de si la falta de reproducibilidad de un caso de prueba se debía a la computadora particular que se está utilizando o a un error en la implementación del programa.

Estándar IEEE 754

En 1987, el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) y el American National Standards Institute (ANSI) adoptaron el estándar **IEEE 754** para la aritmética de punto flotante.

Cuando se sigue el estándar, puede esperar que los tipos de datos primitivos tengan la precisión y los intervalos dados en la siguiente tabla:

IEEE 754 para Tipos de Datos Primitivos

Nombre	Tipo	Bits	Byte	Rango
`boolean`	Lógico	1	$\frac{1}{8}$	`True` o `False`
`char`	String	16	2	`'u0000'` $\longleftrightarrow$ `'uFFFF'`
`byte`	Integer	8	1	$-128 \longleftrightarrow +127$
`short`	Integer	16	2	$-32768 \longleftrightarrow +32767$
`int`	Integer	32	4	checar en tablas
`long`	Integer	64	8	checar en tablas
`float`	Floating	32	4	checar en tablas
`double`	Flaoting	64	8	checar en tablas

Apego al IEEE 754

Cuando las computadoras y el software se apegan a este estándar, está garantizado que un programa producirá resultados idénticos en diferentes equipos.

Apego al IEEE 754

En realidad hay una serie de componentes en el estándar IEEE, y diferentes fabricantes de computadoras o procesadores pueden apegarse a sólo algunos de ellos.

Además, `python`puede no seguir a todos a medida que se desarrolla, pero probablemente con el tiempo lo haga.

Notación de punto flotante

Normalmente un número de punto flotante x se almacena como

$$x_f = (-1)^s \times 1.f \times 2^{e-\text{sesgo}} \tag{2}$$

esto es, con partes por separado para:

- El signo s.

Notación de punto flotante

Normalmente un número de punto flotante x se almacena como

$$x_f = (-1)^s \times 1.f \times 2^{e-\text{sesgo}} \tag{2}$$

esto es, con partes por separado para:

- El signo s.
- La parte fraccional de la mantisa f.

Notación de punto flotante

Normalmente un número de punto flotante x se almacena como

$$x_f = (-1)^s \times 1.f \times 2^{e-\text{sesgo}} \tag{2}$$

esto es, con partes por separado para:

- El signo s.
- La parte fraccional de la mantisa f.
- El valor del exponente e.

Almacenamiento en punto flontante

Todas las partes se almacenan en forma binaria y ocupan segmentos adyacentes de una sola palabra de 32 bits o dos palabras adyacentes de 32 bits para palabras dobles.

El signo s se almacena como un solo bit, con $s = 0$ ó 1 para un signo positivo o negativo.

Se usan ocho bits para almacenar el exponente e, lo que significa que e puede estar en el rango $0 \leq e \leq 255$.

Los extremos, $e = 0$ y $e = 255$, son casos especiales.

Casos especiales del estándar IIEEE

Nombre del número	Valores de s, e y f	Valor
Normal	$0 < e < 255$	$(-1)^s \times 2^{e-127} \times 1.f$
Subnormal	$e = 0, f \neq 0$	$(-1)^s \times 2^{e-126} \times 0.f$
Signo cero (± 0)	$e = 0, f = 0$	$(-1)^s \times 0.0$
$+\infty$	$s = 0, e = 255, f = 0$	**+INF**
$-\infty$	$s = 1, e = 255, f = 0$	**-INF**
Not a number	$s = u, e = 255, f \neq 0$	**NaN**

Números normales

Los números normales tienen un valor entre $0 < e < 255$, y con ellos la convención es suponer que el primer bit de la mantisa es un 1.

De modo que sólo se almacena la parte fraccional f después del punto binario.

Valores $\pm$ INF y NaN

Nótese que los valores $\pm$ INF y NaN no son números en el sentido matemático, es decir, son objetos que pueden ser manipulados o utilizados en cálculos para tomar límites

Más bien, son señales para la computadora y para el usuario de que algo ha ido mal y que el cálculo probablemente debería detenerse hasta resolver las cosas.

Los valores ± 0

En contraste, el valor -0 se puede utilizar en un cálculo sin problemas.

Algunos lenguajes pueden establecer variables no asignadas como -0 como una pista de que aún no se han asignado, pero no es lo más conveniente.

Precisión relativa en punto flotante

Debido a que la incertidumbre (error) está presente sólo en la mantisa y no en el exponente, las representaciones IEEE aseguran que todos los números normales de punto flotante tengan la misma precisión relativa.

El bit fantasma

Debido a que el primer bit se supone que es 1, no tiene que ser almacenado, y los diseñadores de computadoras sólo necesitan recordar que hay un *bit fantasma* allí para obtener un poco más de precisión.

Durante el procesamiento de números en un cálculo, el primer bit de un resultado intermedio puede llegar a ser cero, pero éste se cambia antes de que se almacene el número final.

Para repetir, en los casos normales, la mantisa real ($1.f$ en notación binaria) contiene un 1 implícito que precede al punto binario.

El número de sesgo

Con el fin de garantizar que el exponente e almacenado sea siempre positivo, se agrega un número fijo llamado sesgo al exponente real p, antes de que se almacene como exponente e.

El exponente real, que puede ser negativo, es

$$p = e - \text{sesgo} \tag{3}$$

Formatos básicos del IEEE

Hay dos formatos básicos para el IEEE de punto flotante, de precisión simple y de precisión doble.

Precisión simple

La precisión simple representa a los números de punto flotante de precisión simple (sencilla).

Ocupan 32 bits en total: con 1 bit para el signo, 8 bits para el exponente y 23 bits para la mantisa fraccional (lo que da una precisión de 24 bits cuando el bit fantasma es incluido)

Precisión doble

Es una abreviatura para los números de punto flotante de precisión doble.

Ocupan 64 bits en total: con 1 bit para el signo, 10 bits para el exponente y 53 bits para la mantisa fraccional (para precisión de 54 bits). Esto significa que los exponentes y mantisas para la precisión doble no son simplemente el doble de los floating.

Representación de 32 bits

Para ver el esquema en la práctica consideremos la representación de 32 bits:

	s	e		f	
Bit	31	30	23	22	0

Representación de 32 bits

Para ver el esquema en la práctica consideremos la representación de 32 bits:

	s	e		f	
Bit	31	30	23	22	0

El bit de signo s está en la posición 31, el exponente con sesgo, está en los bits $30 - 23$, y la mantisa fraccional f, en los bits $22 - 0$.

Ya que se usan 8 bits para almacenar el exponente e, y $2^8 = 256$, el exponente tiene un rango

$$0 \leq e \leq 255$$

Ya que se usan 8 bits para almacenar el exponente e, y $2^8 = 256$, el exponente tiene un rango

$$0 \leq e \leq 255$$

Con el sesgo = 127_{10}, el exponente completo queda como

$$p = e_{10} - 127$$

Para la precisión sencilla el exponente tiene el rango

$$-126 \leq p \leq 127$$

La mantisa f se almacena en los 23 bits con posición $22 - 0$. Para los números normales, esto es, para aquellos con $0 < e < 255$, f es la parte fraccional de la mantisa, y por tanto, el número representado en 32 bits es

Numero normal de punto flotante $= (-1)^s \times 1.f \times 2^{e-12}$

Números subnormales

Los números subnormales tienen $e = 0$, $f \neq 0$; para estos números, f es la mantisa completa, por lo que el número representado en 32 bits es

$$\text{Numero subnormal } = (-1)^s \times 0.f \times 2^{e-126} \quad (4)$$

Representación de la mantisa

Los 23 bits $m_{22} - m_0$ que se utilizan para almacenar la mantisa de números normales, corresponde a la notación

$$\text{Mantisa } = 1.f = 1 + m_{22} \times 2^{-1} + m_{21} \times 2^{-2} + \ldots + \\ + m_0 \times 2^{-23} \tag{5}$$

Representación de la mantisa

Los 23 bits $m_{22} - m_0$ que se utilizan para almacenar la mantisa de números normales, corresponde a la notación

$$\text{Mantisa } = 1.f = 1 + m_{22} \times 2^{-1} + m_{21} \times 2^{-2} + \ldots + \\ + m_0 \times 2^{-23} \tag{5}$$

Para los números subnormales, se usa $0.f$

Ejemplo

Veamos un ejemplo: el mayor número positivo de punto flotante de precisión simple para una máquina de 32 bits, tiene el valor máximo para $e = 254$ y el valor máximo para f:

$$\begin{aligned} X_{max} &= 01111111011111111111111111111111 \\ &= (0)(11111110)(11111111111111111111111) \end{aligned} \tag{6}$$

donde se han agrupado los bits.

Se tienen entonces los siguiente valores

$$
\begin{aligned}
s &= 0 \\
e &= 11111110 = 254 \\
p &= e - 127 = 127 \\
f &= 1.11111111111111111111111 = 1 + 0.5 + 0.25 + . \\
&\rightarrow (-1)^s \times 1.f \times 2^{p=e-127} \simeq 2 \times 2^{127} \simeq 3.4 \times 10^{38}
\end{aligned}
\tag{7}
$$

Ejercicio a cuenta

Ahora te toca desarrollar las cuentas para evaluar el número positivo de punto flotante de precisión simple más pequeño subnormal, con $e = 0$, y con una mantisa:

$$0\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 001$$

Solución

Acomodamos los términos para un manejo más sencillo

$$s = 0$$

$$e = 0$$

$$p = e - 126 = -126$$

$$f = 0.0000\ 0000\ 0000\ 0000\ 0000\ 001 = 2^{-23}$$

$$\rightarrow (-1)^s \times 0.f \times 2^{p=e-126} = 2^{-149} \simeq 1.4 \times 10^{-45}$$

Resumen de la precisión simple

Podemos dejar como resumen que los números de precisión simple, tienen 6 ó 7 decimales de significancia y las magnitudes están en el rango

$$1.4 \times 10^{-45} \leq \text{ precision simple } \leq 3.4 \times 10^{38}$$

Representación de la precisión doble

Los números de precisión doble se almacenan en dos palabras de 32 bits, para un total de 64 bits (8) Bytes.

Representación de la precisión doble

Los números de precisión doble se almacenan en dos palabras de 32 bits, para un total de 64 bits (8) Bytes.

	s	e		f		f cont	
Bit	63	62	52	51	32	31	0

Representación de la precisión doble

El signo ocupa 1 bit, el exponente e ocupa 11 bits, la mantisa fraccional 52 bits.

Los valores se almacenan de manera contigua, en donde parte de la mantisa f se almacena en un palabra de 32 bits.

Valor del sesgo

Para los números de precisión doble, el valor del seso es mayor que en los de precisión simple

$$\text{sesgo } = 1111111111_2 = 1023_{10}$$

por lo que el exponente real es $p = e - 1023$.

Rangos en precisión doble

Los números en precisión doble tienen aproximadamente 16 decimales de precisión (1 parte en 2^{52}), y las magnitudes están en el rango

$$4.9 \times 10^{-324} \leq \text{ precision doble } \leq 1.8 \times 10^{308}$$