Representación de Números Reales

Organización del Computador 1
1er Cuatrimestre 2018

En un mundo de 0’s y 1’s…

- ¿Qué sabemos representar en forma de ceros y unos?

- Números naturales/magnitudes (notación sin signo)

- Números enteros (notación con signo, complemento, exceso, notación con signo)

Antes de continuar

- ¿Qué número queremos representar con este numeral (tira de símbolos)?

74,312

a) ¿setenta y cuatro coma trescientos doce?

b) ¿setenta y cuatro coma tres uno dos ?

c) ¿ambas opciones son correctas

d) ¿ninguna opción es correcta

Volviendo a la primaria

- **Números enteros:** la posición de cada símbolo refiere a una potencia de 10

- **Números fraccionarios:** en realidad pasa lo mismo.

10^1	10^0		10^{-1}	10^{-2}	10^{-3}
7	4	,	3	1	2
7×10	4×1		$3 \times 0,1$	$1 \times 0,01$	$2 \times 0,001$

En general

- **Si la base es b**: Utilizamos b símbolos para representar los números entre el 0 y (b-1).

- Los numerales se interpretan como:

$$(\mathtt{a}_n \mathtt{a}_{n-1} \cdots \mathtt{a}_0 \ , \ \mathtt{a}_{-1} \cdots \mathtt{a}_{-k+1} \mathtt{a}_{-k})_b = \sum_{i=-k}^{n} \mathtt{a}_i \cdot b^i$$

¿Cuál es la distancia entre 2 números **consecutivos**?

Finitud

- Con k bits puedo representar un intervalo de números enteros:
  - Notación sin signo, complemento a 2, etc.
- Con k bits, ¿puedo representar un intervalo de números reales?

Valores aproximados

- **Truncamiento**: Me olvido de los dígitos menos significativos del valor y lo aproximo por el menor valor representable,
  - Ejemplo: $(0{,}1254)_{10} \rightarrow (0{,}12)_{10}$
- **Redondeo**: Elijo el valor aproximado en función de redondear las cifras:
  - Ejemplo: $(0{,}1254)_{10} \rightarrow (0{,}125)_{10} \rightarrow (0{,}13)_{10}$

Representación Punto Fijo con Signo

- Representamos la parte entera con signo usando alguno de los métodos presentados para números enteros
    - complemento a 2, signo+magnitud, exceso a N, etc.)
- Representamos la parte fraccionaria a través de un simple cambio de base a binario.

Representación Punto Fijo con Signo

- ¿Qué elementos debo definir?
  - Cantidad de bits para parte entera y cantidad de bits para parte fraccionaria
  - Codificación de números enteros
  - Ubicación de cada parte en el numeral

Cambiemos de Base

- Ejemplos:
  - ¿Cómo representamos 1/3 en base 3? ¿y -1/3?
  - ¿… 1/16 en base 2? ¿y -1/16?
  - ¿… 0.25141 en base 4?

Representación Punto Fijo con Signo

- Ejemplo:

Fracción Entero

- Fracción: 2 bits
- Parte Entera: 4 bits (complemento a 2)

Algunos problemitas

- La distancia conocida desde la Tierra hasta los confines del universo es $4{,}6 \times 10^{26}$ metros
- La masa de un protón es $1{,}67 \times 10^{-27}$ kg
- ¿Qué problemas tengo para representar en mi computadora estas dos magnitudes con punto fijo?

Notación científica

- Se utiliza para expresar magnitudes muy grandes y muy pequeñas en el mismo formato.

$$\pm f \times 10^{e}$$

- donde:
  - *f* es un número fraccionario
  - *e* es un número entero
  - ¿Ejemplos?

Notación científica normalizada

- Si no restringimos los valores de f tenemos múltiples representaciones de un mismo valor:
  - - 194,3 x 10^2
  - - 1,943 x 10^4
  - - 194300 x 10^-3
  - - 0,1943 x 10^5
- La notación científica normalizada restringe *f* tal que:

$$0{,}1 <= f < 1$$

Punto flotante (float)

- **Intuición**: Usar la misma idea que usa la notación científica normalizada

±f x 2^e

- e es un número entero
- f es un número fraccionario normalizado entre
  - 1<= |f| < 2

Punto Flotante

- ¿Qué necesito para definir una codificación de punto flotante?
  - Cantidad de bits de exponente (e)
  - Codificación del exponente
  - Cantidad de bits de mantisa (f)
  - Codificación de la mantisa

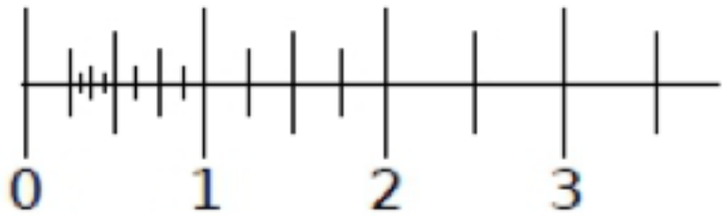
Punto Flotante

- Algunas preguntas:
  - ¿Necesito almacenar el “2” de 2^e?
  - ¿Si todas las mantisas comienzan con “1,” necesito almacenarlo?

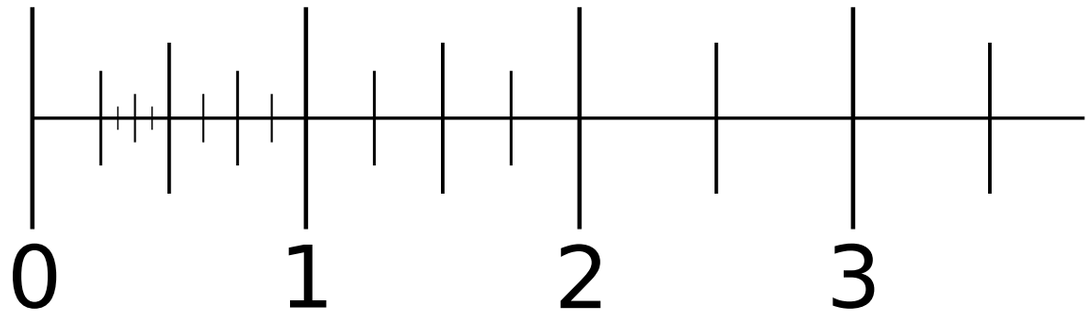
Punto Flotante - Ejemplo

- Exponente: 2 bits exceso a 2
- Mantisa: 2 bits parte fraccionaria, “1.” implícito
- Potencia base: 2

$exp \backslash mant$	00	01	10	11
00	0,25	0,3125	0,375	0,4375
01	0,5	0,625	0,75	0,875
10	1	1,25	1,5	1,75
11	2	2,5	3	3,5



Punto Flotante - Ejemplo



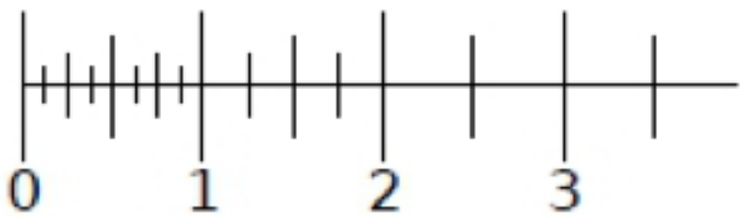
- Con notación totalmente normalizada se representa un intervalo muy denso alrededor de valores no muy recurrentes.

- ¿Cómo podemos representar valores más útiles?

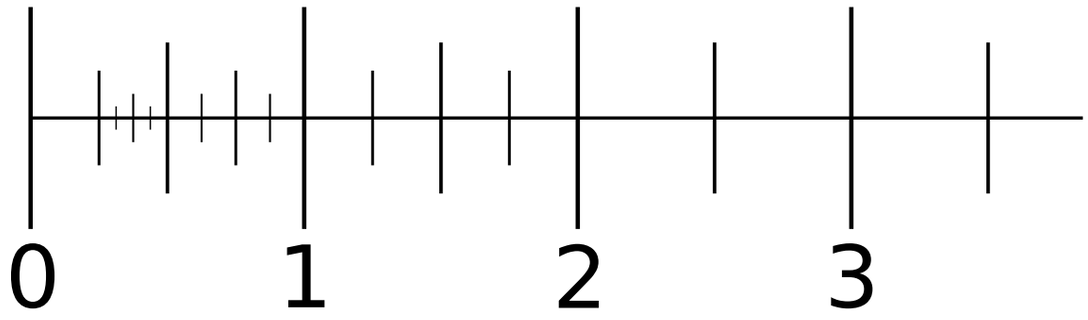
Punto Flotante - Des-normalización

- **Des-normalización selectiva:** El exponente “00” (representa el -2) es utilizado para representar números sin “1.” implícito.

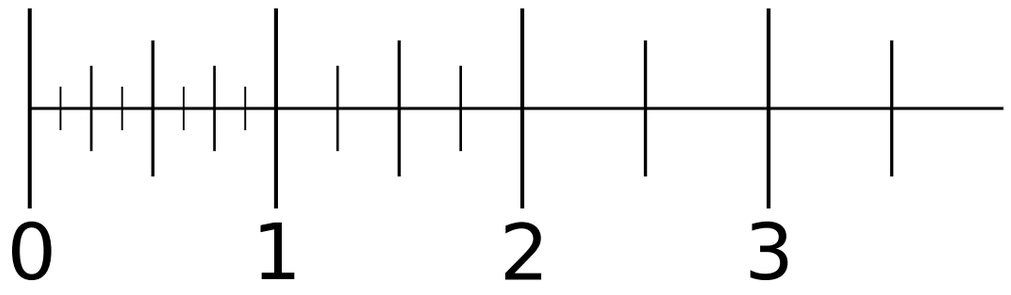
$exp \backslash mant$	00	01	10	11
00	0	0,0625	0,125	0,1875
01	0,5	0,625	0,75	0,875
10	1	1,25	1,5	1,75
11	2	2,5	3	3,5



Punto Flotante



Solo Normalizados

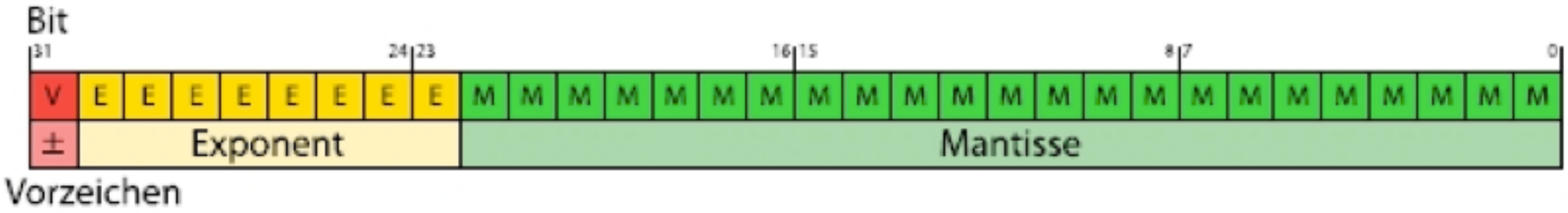


Usando Normalizados y Des-Normalizados

- La des-normalización selectiva una mejor distribución de los valores a representar

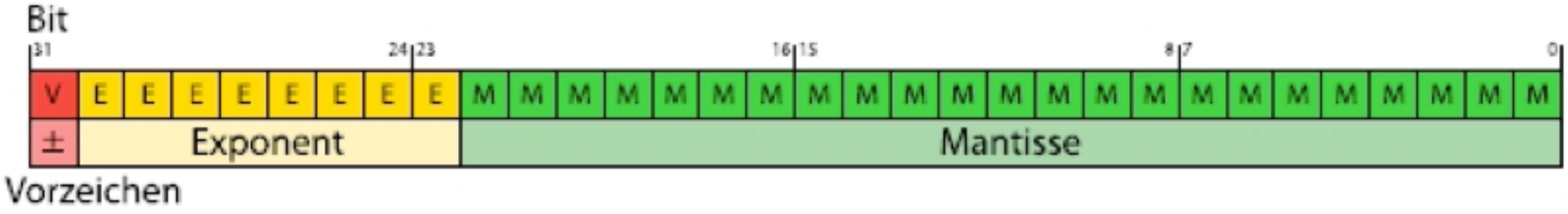
IEEE 754

- Es un standard para uniformizar la codificación de números enteros entre computadoras y programas.



- 1 bit signo (de la fracción). Si b=1 es negativo.
- 8 bits exponente
- 23 bits mantisa
- **Pregunta:** ¿Cómo se interpretan?

IEEE 754: Números Normalizados

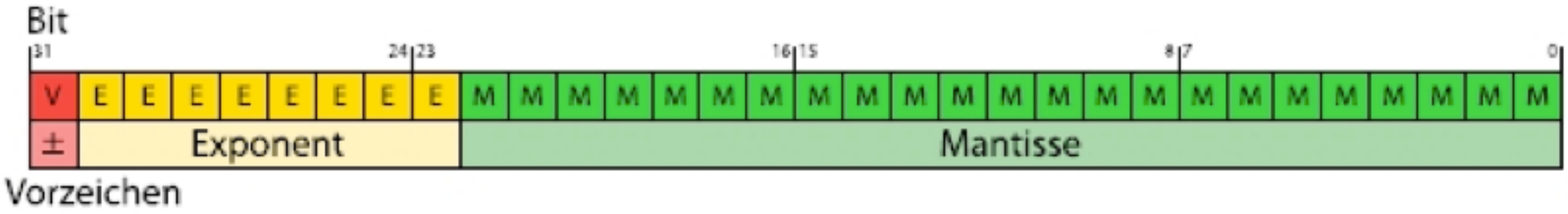


- Si el exponente NO ES **00000000** NI **11111111** entonces:
  - el exponente se decodifica usando **exceso 127**
  - la mantisa tiene un “1.” implícito
  - el bit de signo es el signo del número

$$\pm 1.f \times 2^{e}$$

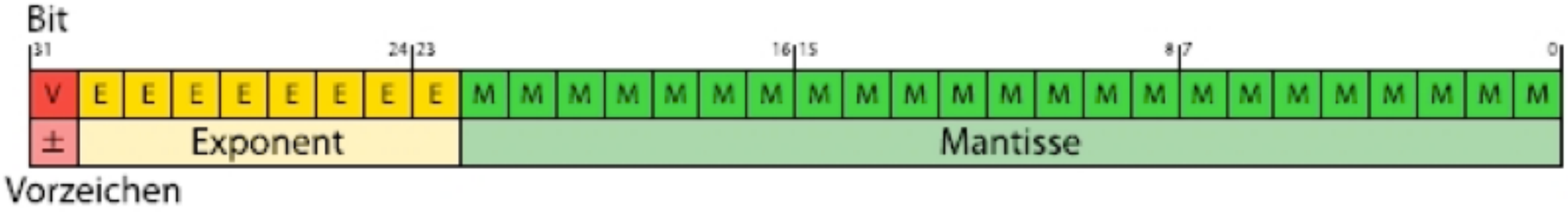
- **Pregunta:** ¿Cuál es el número positivo más chico que puedo representar?

IEEE 754: Números des-normalizados



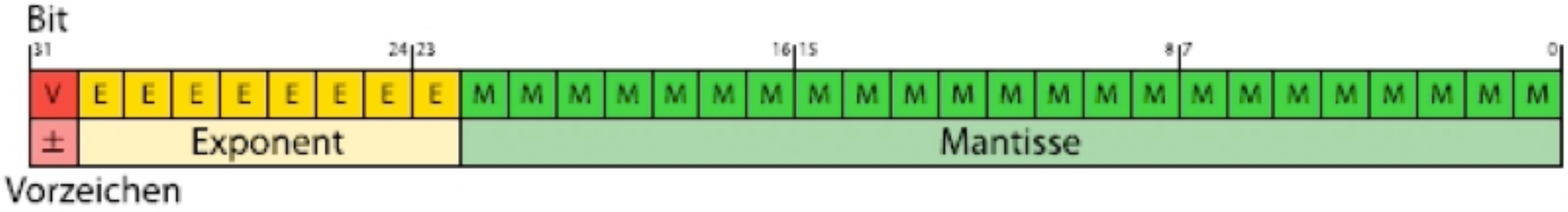
- Si el exponente ES **00000000** entonces:
  - el exponente se decodifica usando exceso 127
  - la mantisa tiene un “0.” implícito ±0.f x 2^e
  - el bit de signo es el signo del número
- **Pregunta**: ¿Cómo codificamos el cero?

IEEE 754: +0 y -0



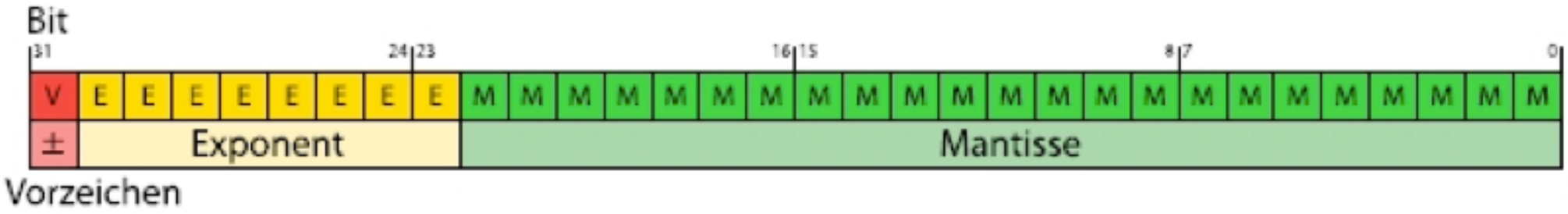
- Si el exponente es **00000000** y la mantisa es **00..00**, entonces:
  - Interpreto el valor como “Cero”.
  - Además, puedo leer el bit de signo para saber si es +0 o -0.
- **Pregunta**: ¿Para qué me puede servir el signo del cero?

IEEE 754: +Infinito y -Infinito



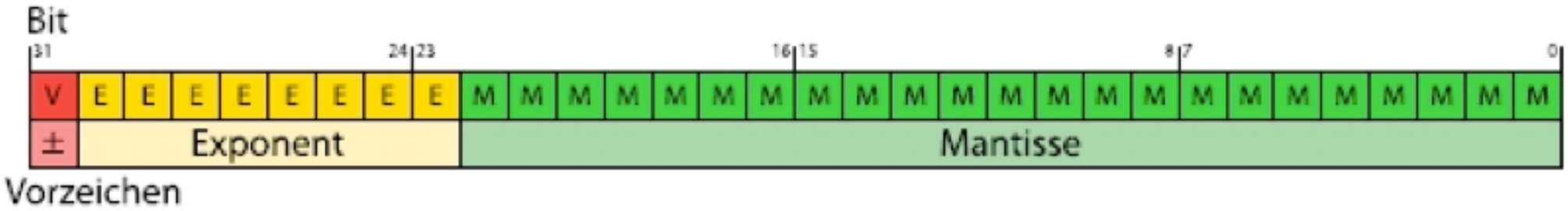
- Si el exponente es **1111 1111** y la mantisa es **00..00**, entonces:
  - Interpreto el valor como “INFINITO”.
  - Además, puedo leer el bit de signo para saber si es +INFINITO o -INFINITO.
- **Pregunta**: ¿Cuál es el resultado +Infinito x (-1)?

IEEE 754: Not-a-Number (NaN)



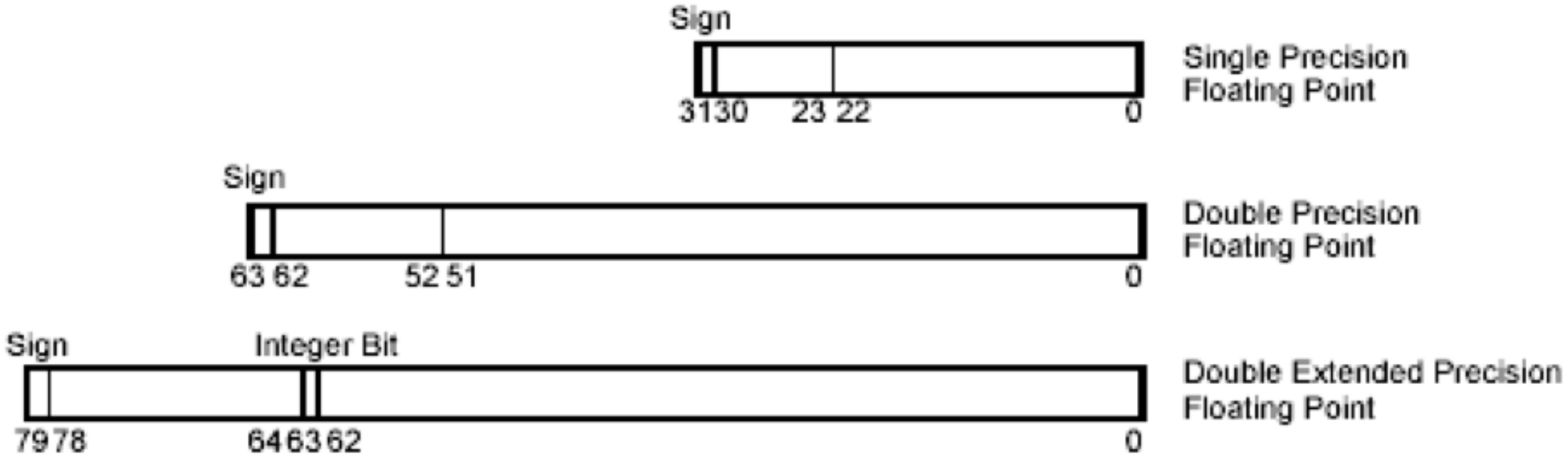
- Si el exponente es **1111 1111** y la mantisa NO es **00..00**, entonces:
  - una operación matemática produjo un valor que no es un número real.
- **Pregunta**: ¿Se les ocurren ejemplos?

IEEE 754: Not-a-Number (NaN)



- **Ejemplos**
  - 1/0
  - sqroot(-1)
  - log(0)
  - +INF x -INF
  - Overflow
  - Underflow

IEEE 754



- En C/Java:
  - Tipos float (32 bits) y double (64 bits)

Intel 8086+8087

- El procesador Intel 8086 podía extenderse con un co-procesador matemático (Intel 8087)
- Permitía ejecutar instrucciones para números en formato de punto flotante
- En siguientes versiones de Intel (ej: Pentium, etc) estas instrucciones se soportan nítidamente
- FPU: Floating Point Unit

Intel 8086+8087

- Registros:
  - posee una pila de 8 registros de 80 bits (`st0, st1,…,st7`)
  - posee registros para control (indicar precisión, redondeo, etc.) y estado (qué ocurrió en la última operación)
- Las instrucciones trabajan implícitamente con la pila

8087: Instrucciones

Instrucción	Descripción
`FINIT`	prepara la pila (desapila los valores anteriores) y resetea los flags y palabra de control
`FSTCW`	salva la actual palabra de control
`FLDCW`	carga una nueva palabra de control
`FLD src`	carga un valor desde una dirección
`FLD ST(i)`	apila el contenido del tope del stack+i

8087: Instrucciones

Instrucción	Descripción
`FST dst`	almacena el tope de la pila en dst
`FSTP dst`	desapila el tope y lo almacena en dst
`FLDPI`	apila la constante PI
`FLDZ`	apila la constante 0
`FLD1`	apila la constante 1.

8087: Instrucciones

Instrucción	Descripción
`FADD`	suma dos reales
`FSUB`	resta dos reales
`FMUL`	multiplica dos reales
`FDIV`	divide dos reales
`FREM`	resto de dividir dos reales

8087: Instrucciones

Instrucción	Descripción
`FCOM`	comparación
`FCOMP`	comparación y des-apila el tope
`FTST`	compara contra el 0
`FSQRT`	raíz cuadrada
`FTAN`	tangente del tope de la pila
`FSIN`	seno del tope de la pila
`FCOS`	coseno del tope de la pila

FPU: Ejemplo

- Calcular la superficie de un círculo usando números IEEE 754 de 32 bits (pi x R^2)

- El Radio está almacenado en la dirección de memoria **RADIO**

- **SUPERFICIE** es la dirección donde hay que almacenar la superficie calculada

FPU: Ejemplo

```
FINIT                  # Inicializa el stack

FLD RADIO              # Carga R al tope de la pila

FLD ST(0)              # Duplica R

FMUL                   # Computa R^2

FLDPI                  # Apila PI

FMUL                   # Computa PI x R^2

FSTP SUPERFICIE        # Desapila el contenido
```

FROM: Dr. Thomas R. Nicely
Professor of Mathematics
Lynchburg College
1501 Lakeside Drive
Lynchburg, Virginia 24501-3199

Phone: 804-522-8374
Fax: 804-522-8499
Internet: nicely@acavax.lynchburg.edu

TO: Whom it may concern

RE: Bug in the Pentium FPU

DATE: 30 October 1994

It appears that there is a bug in the floating point unit (numeric coprocessor) of many, and perhaps all, Pentium processors.

In short, the Pentium FPU is returning erroneous values for certain division operations. For example,

1/824633702441.0

is calculated incorrectly (all digits beyond the eighth significant digit are in error). This can be verified in compiled code, an ordinary spreadsheet such as Quattro Pro or Excel, or even the Windows calculator (use the scientific mode), by computing

(824633702441.0)*(1/824633702441.0),

which should equal 1 exactly (within some extremely small rounding error; in general, coprocessor results should contain 19 significant decimal digits). However, the Pentiums tested return

0.9999999996274709702

Pentium Bug (1994)

- El procesador Pentium tenía un error de diseño (ie circuitos digitales) que provocaba un cálculo erróneo de división

- Al principio, INTEL negó la falla

- Luego, negó la importancia de la falla

- Finalmente, INTEL debió hacer un recall de los procesadores Pentium afectados.

  - Se estima que tuvo un costo de U$S 400 millones

Pentium FDIV Bug

- El error estaba en la circuito que ejecutaba la instrucción FDIV

- El algoritmo de división (SRT) usa una tabla de división

  - Esta tabla tenía valores incorrectos (5 de 1066)

Pentium FDIV Bug

Resumen

- Representación de números reales
  - Punto Fijo
  - Punto Flotante (IEEE 754)
- Set de instrucciones Intel 8087
  - Co-procesador matemático
  - Intel FDIV bug

Bibliografía

- Capítulo 2 del Libro de Null (Leer solo los temas que vimos en clase)

- Datasheet Intel 8087 (http://archive.pcjs.org/pubs/pc/datasheets/8087-FPU.pdf)

- Programming with the x87 FPU (http://home.agh.edu.pl/~amrozek/x87.pdf)

- The Pentium Chip Story: An Internet Learning Experience (http://www.emery.com/1e/pentium.htm)