Lecture 2
Data Representation

CPS310

Computer Organization II

WINTER 2022

© Dr. A. Sadeghian

Sign & Magnitude

The leftmost bit (MSB) is the sign (0 = positive, 1 = negative) and

the remaining bits are the magnitude

Example:

$+25_{10}$ = 0 001 1001$)_2$

-25_{10} = 1 001 1001$)_2$

$+ 12_{10}$ = 0000 1100$)_2$

$- 12_{10}$ = 1111 0011$)_2$

Two representations for zero:

$+0 = 00000000_2$, $-0 = 10000000_2$

Largest number is +127, smallest number is -127_{10}, using an 8-bit representation

1's Complement

The leftmost bit (MSB) is the sign (0 = positive, 1 = negative). Negative form of a number is obtained by complementing each bit (from 0 to 1 or from 1 to 0)

This goes both ways, converting between positive and negative numbers

Example:

$+25_{10}$ = 0 001 1001$)_2$

-25_{10} = 1 110 0110$)_2$

$+12_{10}$ = 0000 1100$)_2$

-12_{10} = 1111 0011$)_2$

Two representations for zero: $+0 = 00000000_2$, $-0 = 11111111_2$

Largest number is $+127_{10}$, smallest number is -127_{10}, using an 8-bit representation

2's Complement

The leftmost bit is the sign (0 = positive, 1 = negative). Negative of a number is obtained by adding 1 to the one's complement negative

This goes both ways, converting between positive and negative numbers

Example (recall that -25_{10} in one's complement is $1110\ 0110)_2$:

$+ 25_{10} = 0\ 001\ \ 1001)_2$

$- 25_{10} = 1\ 110\ \ 0111)_2$

$+ 12_{10} = 0000\ \ 1100)_2$
$- 12_{10} = 1111\ \ 0100)_2$

One representation for zero: $+0 = 00000000_2$, $-0 = 00000000_2$

Largest number is $+127_{10}$, smallest number is -128_{10}, using an 8-bit representation

Excess (Biased)

- Positive and negative representations of a number are obtained by adding ***a bias*** to the number

- Example:
- Excess-3, add 3 to the number
- Excess-127, add 127 to the number

Excess-Bias

The effect is that numerically smaller numbers have smaller bit patterns, simplifying comparisons for floating point exponents (see next table)

Example:

Excess-128 [2^7] adds 128 to the 2's complement rep, ignoring any carry out of the most significant bit:

- **$+12_{10} = 1000\ 1100_2$**
- **$-12_{10} = 0111\ 0100_2$**

+12:	**0000 1100**	**-12:**	**1111 0100**
128:	**1000 0000**	**128:**	**1000 0000**
----------------------------		**----------------------------**	
	1000 1100		**(1) 0111 0100**

- One representation for zero: $+0 = 10000000_2$, $-0 = 10000000_2$
- Largest number is $+127_{10}$, smallest number is -128_{10}, using an 8-bit representation

3-Bit Signed Integer Representations

Decimal	Unsigned	Sign–Mag.	1's Comp.	2's Comp.	Excess 4
7	111	–	–	–	–
6	110	–	–	–	–
5	101	–	–	–	–
4	100	–	–	–	–
3	011	011	011	011	111
2	010	010	010	010	110
1	001	001	001	001	101
+0	000	000	000	000	100
-0	–	100	111	000	100
-1	–	101	110	111	011
-2	–	110	101	110	010
-3	–	111	100	101	001
-4	–	–	–	100	000

ASCII Character Code - American Standard Code for Information Interchange

- **ASCII is a 7-bit code, commonly stored in 8-bit bytes**
- **“A” is at 41_{16}. To convert upper case letters to lower case letters, add 20_{16}. Thus “a” is at $41_{16} + 20_{16} = 61_{16}$**
- **The character “5” at position 35_{16} is different than the number 5. To convert character-numbers into number-numbers, subtract 30_{16}: $35_{16} - 30_{16} = 5$**

00	NUL	10	DLE	20	SP	30	0	40	@	50	P	60	`	70	p
01	SOH	11	DC1	21	!	31	1	41	A	51	Q	61	a	71	q
02	STX	12	DC2	22	"	32	2	42	B	52	R	62	b	72	r
03	ETX	13	DC3	23	#	33	3	43	C	53	S	63	c	73	s
04	EOT	14	DC4	24	$	34	4	44	D	54	T	64	d	74	t
05	ENQ	15	NAK	25	%	35	5	45	E	55	U	65	e	75	u
06	ACK	16	SYN	26	&	36	6	46	F	56	V	66	f	76	v
07	BEL	17	ETB	27	'	37	7	47	G	57	W	67	g	77	w
08	BS	18	CAN	28	(	38	8	48	H	58	X	68	h	78	x
09	HT	19	EM	29	)	39	9	49	I	59	Y	69	i	79	y
0A	LF	1A	SUB	2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
0B	VT	1B	ESC	2B	+	3B	;	4B	K	5B	[	6B	k	7B	{
0C	FF	1C	FS	2C	,	3C	<	4C	L	5C	\	6C	l	7C	|
0D	CR	1D	GS	2D	-	3D	=	4D	M	5D	]	6D	m	7D	}
0E	SO	1E	RS	2E	.	3E	>	4E	N	5E	^	6E	n	7E	~
0F	SI	1F	US	2F	/	3F	?	4F	O	5F	_	6F	o	7F	DEL

NUL	Null	FF	Form feed	CAN	Cancel
SOH	Start of heading	CR	Carriage return	EM	End of medium
STX	Start of text	SO	Shift out	SUB	Substitute
ETX	End of text	SI	Shift in	ESC	Escape
EOT	End of transmission	DLE	Data link escape	FS	File separator
ENQ	Enquiry	DC1	Device control 1	GS	Group separator
ACK	Acknowledge	DC2	Device control 2	RS	Record separator
BEL	Bell	DC3	Device control 3	US	Unit separator
BS	Backspace	DC4	Device control 4	SP	Space
HT	Horizontal tab	NAK	Negative acknowledge	DEL	Delete
LF	Line feed	SYN	Synchronous idle		
VT	Vertical tab	ETB	End of transmission block		

Extended Binary Coded Decimal Interchange Code

EBCDIC Character Code

- EBCDIC is an 8-bit code.

00	NUL	20	DS	40	SP	60	–	80		A0		C0	{	E0	\
01	SOH	21	SOS	41		61	/	81	a	A1	~	C1	A	E1	
02	STX	22	FS	42		62		82	b	A2	s	C2	B	E2	S
03	ETX	23		43		63		83	c	A3	t	C3	C	E3	T
04	PF	24	BYP	44		64		84	d	A4	u	C4	D	E4	U
05	HT	25	LF	45		65		85	e	A5	v	C5	E	E5	V
06	LC	26	ETB	46		66		86	f	A6	w	C6	F	E6	W
07	DEL	27	ESC	47		67		87	g	A7	x	C7	G	E7	X
08		28		48		68		88	h	A8	y	C8	H	E8	Y
09		29		49		69		89	i	A9	z	C9	I	E9	Z
0A	SMM	2A	SM	4A	¢	6A	‘	8A		AA		CA		EA	
0B	VT	2B	CU2	4B		6B	,	8B		AB		CB		EB	
0C	FF	2C		4C	<	6C	%	8C		AC		CC		EC	
0D	CR	2D	ENQ	4D	(	6D	_	8D		AD		CD		ED	
0E	SO	2E	ACK	4E	+	6E	>	8E		AE		CE		EE	
0F	SI	2F	BEL	4F	|	6F	?	8F		AF		CF		EF	
10	DLE	30		50	&	70		90		B0		D0	}	F0	0
11	DC1	31		51		71		91	j	B1		D1	J	F1	1
12	DC2	32	SYN	52		72		92	k	B2		D2	K	F2	2
13	TM	33		53		73		93	l	B3		D3	L	F3	3
14	RES	34	PN	54		74		94	m	B4		D4	M	F4	4
15	NL	35	RS	55		75		95	n	B5		D5	N	F5	5
16	BS	36	UC	56		76		96	o	B6		D6	O	F6	6
17	IL	37	EOT	57		77		97	p	B7		D7	P	F7	7
18	CAN	38		58		78		98	q	B8		D8	Q	F8	8
19	EM	39		59		79		99	r	B9		D9	R	F9	9
1A	CC	3A		5A	!	7A	:	9A		BA		DA		FA	|
1B	CU1	3B	CU3	5B	$	7B	#	9B		BB		DB		FB	
1C	IFS	3C	DC4	5C	.	7C	@	9C		BC		DC		FC	
1D	IGS	3D	NAK	5D	)	7D	'	9D		BD		DD		FD	
1E	IRS	3E		5E	;	7E	=	9E		BE		DE		FE	
1F	IUS	3F	SUB	5F	¬	7F	"	9F		BF		DF		FF	

STX	Start of text	RS	Reader Stop	DC
DLE	Data Link Escape	PF	Punch Off	DC
BS	Backspace	DS	Digit Select	DC
ACK	Acknowledge	PN	Punch On	CU
SOH	Start of Heading	SM	Set Mode	CU
ENQ	Enquiry	LC	Lower Case	CU
ESC	Escape	CC	Cursor Control	SY
BYP	Bypass	CR	Carriage Return	IF
CAN	Cancel	EM	End of Medium	EC
RES	Restore	FF	Form Feed	ET
SI	Shift In	TM	Tape Mark	NA
SO	Shift Out	UC	Upper Case	SM
DEL	Delete	FS	Field Separator	SC
SUB	Substitute	HT	Horizontal Tab	IG
NL	New Line	VT	Vertical Tab	IR
LF	Line Feed	UC	Upper Case	IU

Unicode Character Code

- **Unicode is a 16-bit code.**

0000	NUL	0020	SP	0040	@	0060	`	0080	Ctrl	00A0	NBS	00C0	À	00E0	à
0001	SOH	0021	!	0041	A	0061	a	0081	Ctrl	00A1	¡	00C1	Á	00E1	á
0002	STX	0022	"	0042	B	0062	b	0082	Ctrl	00A2	¢	00C2	Â	00E2	â
0003	ETX	0023	#	0043	C	0063	c	0083	Ctrl	00A3	£	00C3	Ã	00E3	ã
0004	EOT	0024	$	0044	D	0064	d	0084	Ctrl	00A4	¤	00C4	Ä	00E4	ä
0005	ENQ	0025	%	0045	E	0065	e	0085	Ctrl	00A5	¥	00C5	Å	00E5	å
0006	ACK	0026	&	0046	F	0066	f	0086	Ctrl	00A6	¦	00C6	Æ	00E6	æ
0007	BEL	0027	'	0047	G	0067	g	0087	Ctrl	00A7	§	00C7	Ç	00E7	ç
0008	BS	0028	(	0048	H	0068	h	0088	Ctrl	00A8	¨	00C8	È	00E8	è
0009	HT	0029	)	0049	I	0069	i	0089	Ctrl	00A9	©	00C9	É	00E9	é
000A	LF	002A	*	004A	J	006A	j	008A	Ctrl	00AA	ª	00CA	Ê	00EA	ê
000B	VT	002B	+	004B	K	006B	k	008B	Ctrl	00AB	«	00CB	Ë	00EB	ë
000C	FF	002C	,	004C	L	006C	l	008C	Ctrl	00AC	¬	00CC	Ì	00EC	ì
000D	CR	002D	-	004D	M	006D	m	008D	Ctrl	00AD	–	00CD	Í	00ED	í
000E	SO	002E	.	004E	N	006E	n	008E	Ctrl	00AE	®	00CE	Î	00EE	î
000F	SI	002F	/	004F	O	006F	o	008F	Ctrl	00AF	¯	00CF	Ï	00EF	ï
0010	DLE	0030	0	0050	P	0070	p	0090	Ctrl	00B0	°	00D0	Đ	00F0	¶
0011	DC1	0031	1	0051	Q	0071	q	0091	Ctrl	00B1	±	00D1	Ñ	00F1	ñ
0012	DC2	0032	2	0052	R	0072	r	0092	Ctrl	00B2	²	00D2	Ò	00F2	ò
0013	DC3	0033	3	0053	S	0073	s	0093	Ctrl	00B3	³	00D3	Ó	00F3	ó
0014	DC4	0034	4	0054	T	0074	t	0094	Ctrl	00B4	´	00D4	Ô	00F4	ô
0015	NAK	0035	5	0055	U	0075	u	0095	Ctrl	00B5	µ	00D5	Õ	00F5	õ
0016	SYN	0036	6	0056	V	0076	v	0096	Ctrl	00B6	¶	00D6	Ö	00F6	ö
0017	ETB	0037	7	0057	W	0077	w	0097	Ctrl	00B7	·	00D7	×	00F7	÷
0018	CAN	0038	8	0058	X	0078	x	0098	Ctrl	00B8	¸	00D8	Ø	00F8	ø
0019	EM	0039	9	0059	Y	0079	y	0099	Ctrl	00B9	¹	00D9	Ù	00F9	ù
001A	SUB	003A	:	005A	Z	007A	z	009A	Ctrl	00BA	º	00DA	Ú	00FA	ú
001B	ESC	003B	;	005B	[	007B	{	009B	Ctrl	00BB	»	00DB	Û	00FB	û
001C	FS	003C	<	005C	\	007C	|	009C	Ctrl	00BC	1/4	00DC	Ü	00FC	ü
001D	GS	003D	=	005D	]	007D	}	009D	Ctrl	00BD	1/2	00DD	Ý	00FD	Þ
001E	RS	003E	>	005E	^	007E	~	009E	Ctrl	00BE	3/4	00DE	ý	00FE	þ
001F	US	003F	?	005F	_	007F	DEL	009F	Ctrl	00BF	¿	00DF	§	00FF	ÿ

NUL	Null	SOH	Start of heading	CAN	Cancel	SP	Space
STX	Start of text	EOT	End of transmission	EM	End of medium	DEL	Delete
ETX	End of text	DC1	Device control 1	SUB	Substitute	Ctrl	Control
ENQ	Enquiry	DC2	Device control 2	ESC	Escape	FF	Form feed
ACK	Acknowledge	DC3	Device control 3	FS	File separator	CR	Carriage return
BEL	Bell	DC4	Device control 4	GS	Group separator	SO	Shift out
BS	Backspace	NAK	Negative acknowledge	RS	Record separator	SI	Shift in
HT	Horizontal tab	NBS	Non-breaking space	US	Unit separator	DLE	Data link escape
LF	Line feed	ETB	End of transmission block	SYN	Synchronous idle	VT	Vertical tab

Floating Point Numbers

- Any floating point number can be shown using:
    1. ***Sign***
    2. ***Exponent***
    3. ***Significand (Mantissa)***

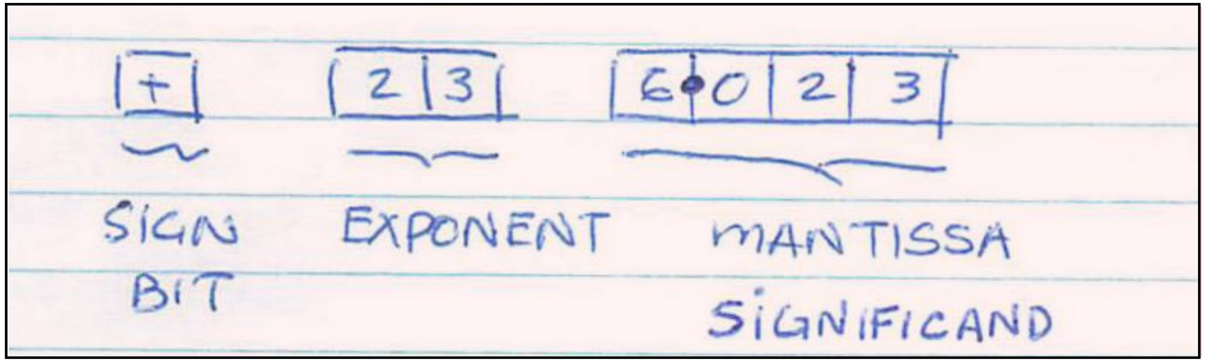
For example:

❶ ❷ ❸

+ 6.023 x 10^{23}

1. Sign: **+**
2. Significand: **6.023**
3. Exponent: **23**

❶ ❸ ❷



Floating Point Number

RANGE:

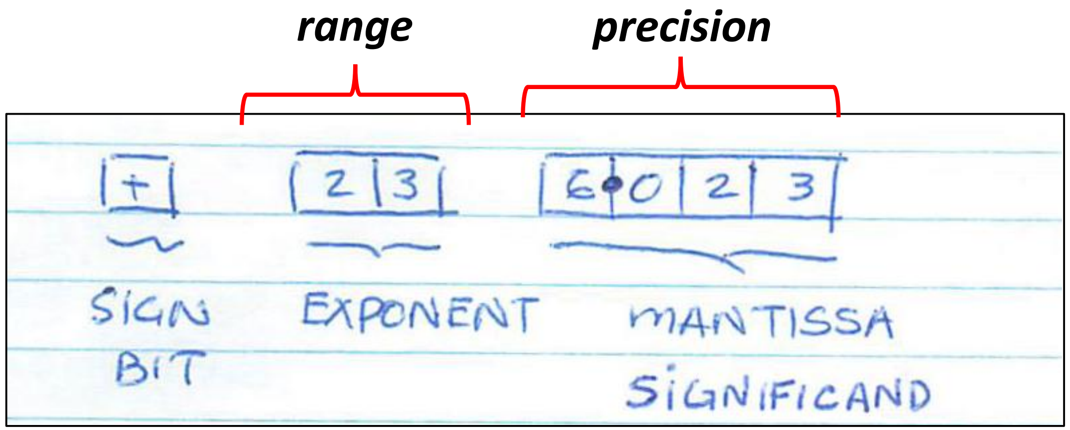
Depends on the number of digits used for exponent and the base used (10, 2, …)

PRECISION:

Depends on the number of digits used for the significand

Note:

Do not need to store decimal point as long as it is always in a fixed place



Floating Point Number Representation

A floating point number can be represented in a number of different ways:

3584.1

3584.1×10^0

358.41×10^1

35.841×10^2

3.5841×10^3 ← ***this is the normalized representation***

How can we fix this?

Normalize it

Normalization of a binary representation

- Move the radix point to the right of the leftmost <u>non-zero</u> digit
- Change the exponent accordingly

+ 110100

+ 110100 $\times 2^{0}$

+ 11010 $\times 2^{1}$

+ 1101 $\times 2^{2}$

+ 110. 1 $\times 2^{3}$

+ 11. 01 $\times 2^{4}$

+ 1. 101 $\times 2^{5}$ ← ***this is the normalized representation***

The Hidden Bit

- In binary representation, the leftmost bit of a normalized significand (mantissa) is always **1**
- So we really do not need to store this bit
- Known as the hidden bit or hidden **1**

$$+\,1.\,101 \quad \times 2^{5}$$

↑

Hidden 1

Hidden Bit Example

Example: a significand in the form of

Hidden bit

1. 11010

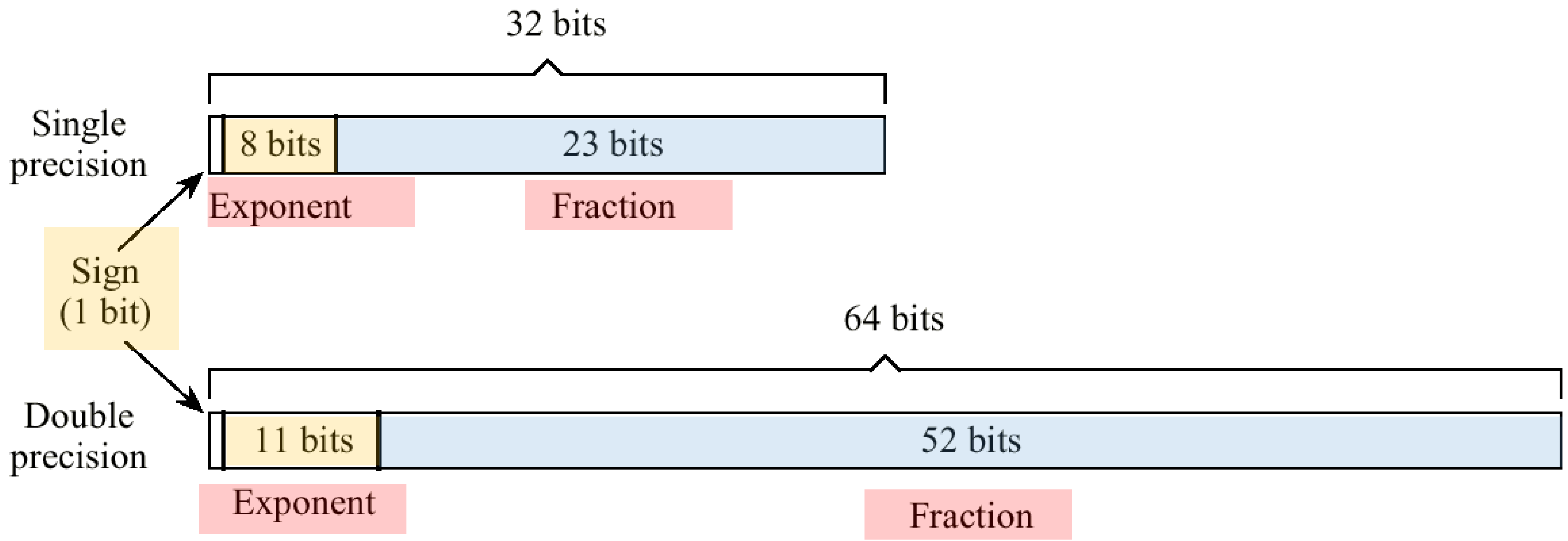
is stored as

11010

IEEE-754 Standard for Floating Point Numbers

IEEE 754 Standard supports 2 formats:

1. ***Single precision (32 bits)***
2. ***Double precision (64 bits)***



Single Precision Floating Point Number

- **32-bit representation**

  - Bit 31 (1 bit): **Sign bit**
  - Bit 30-23 (8 bits): **Exponent**
  - Bit 22-0 (23 bits): **Fraction**

- Exponent is represented in **EXCESS-127**

Single Precision Floating Point Number

- **0000 0000** and **1111 1111** have special meanings

- The most negative number should be (**0000 0000**), i.e., (-127),
- However, (**0000 0000**) has a special meaning
- Therefore, the most negative number is (-126)

- The most positive number is (**1111 1111**), i.e., (+128),
- However, (**1111 1111**) has a special meaning
- Therefore, the most positive number is (+127)

Example Single Precision

$+1.101 \times 2^{5}$

Sign: +

Exponent: **5**

Fraction: **101**

Exponent in EXCESS-127: (127 + 5 = 132) 1000 0100

Sign	Exponent	Fraction
0	**1000 0100**	**101 0000 0000 0000 0000 0000**

Example Single Precision

$- 1.01011 \times 2^{-126}$

Sign: **-**

Exponent: **-126**

Fraction: **01011**

Exponent in EXCESS-127: (127 + (-126) = 1) 0000 0001

1 0000 0001 010 1100 0000 0000 0000 0000

Example Single Precision

1.0×2^{127}

Sign: **+**

Exponent: **127**

Fraction: **0**

Exponent in EXCESS-127: (127 + 127 = 254) 1111 1110

0 1111 1110 000 0000 0000 0000 0000 0000

How to show ZERO?

- Exponent: all zero
- Fraction: all zero

- Depending on the sign bit, it can be +0 or -0
- **0 0000 0000 000 0000 0000 0000 0000 0000 (+ 0)**
- **1 0000 0000 000 0000 0000 0000 0000 0000 (- 0)**

How to show infinity?

- Infinity: overflow – Division by zero
- Exponent: all one
- Fraction: all zero

- Depending on the sign bit, it can be $+\infty$ or $-\infty$
- **0 1111 1111 000 0000 0000 0000 0000 0000 (+ ∞)**
- **1 1111 1111 000 0000 0000 0000 0000 0000 (- ∞)**

How to show NaN?

- NaN: Not a Number – zero divided by zero

- Exponent: all one
- Fraction: non-zero value

- +NaN
- 0 **1111 1111** 011 0111 0000 0000 0000 0000

Another Example

$+\ 2^{-128}$

Sign: 0

Exponent: -128

Fraction: 0

EXESS-127 rep: -128 + 127 = -1 (not valid)

Double Precision Floating Point Number

- 64-bit representation

- Bit 63 (1 bit): **Sign bit**
- Bit 62-52 (11 bits): **Exponent**
- Bit 51-0 (52 bits): **Fraction**

- Exponent is represented in **EXCESS-1023**

Double Precision

+ 2^{-128}

Sign: **0**

Exponent: **-128**

Fraction: **0**

Exponent in EXCESS-1023: (1023+(-128)= 895) 011 0111 1111

0 011 0111 1111 0000 … 0000

52 bits

IEEE-754 Examples

	Value	Bit Pattern		
		Sign	Exponent	Fraction
(a)	$+1.101 \times 2^{5}$	0	1000 0100	101 0000 0000 0000 0000 0000
(b)	-1.01011×2^{-126}	1	0000 0001	010 1100 0000 0000 0000 0000
(c)	$+1.0 \times 2^{127}$	0	1111 1110	000 0000 0000 0000 0000 0000
(d)	$+0$	0	0000 0000	000 0000 0000 0000 0000 0000
(e)	-0	1	0000 0000	000 0000 0000 0000 0000 0000
(f)	$+\infty$	0	1111 1111	000 0000 0000 0000 0000 0000
(g)	$+2^{-128}$	0	0000 0000	010 0000 0000 0000 0000 0000
(h)	$+\text{NaN}$	0	1111 1111	011 0111 0000 0000 0000 0000
(i)	$+2^{-128}$	0	011 0111 1111	0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

Fraction Conversion – Decimal to Binary

Repetitive Multiplication

Example: $0.6875)_{10}$ to Binary

- $0.6875 \times 2 = \mathbf{1}.375$
- $0.375 \times 2 = \mathbf{0}.75$
- $0.75 \times 2 = \mathbf{1}.5$
- $0.5 \times 2 = \mathbf{1}.0$
- $0.6875)_{10} = \boxed{0.\mathbf{1011})_2}$

-1 -2 -3 -4

$0.1\ \ 0\ \ 1\ \ 1)_2 = 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4}$

$= 0.5 + 0 + 0.125 + 0.0625 = 0.6875)_{10}$

IEEE-754 Conversion From Decimal to Binary

- Represent **-12.625_{10}** in single precision IEEE-754 format

Step #1 – Convert to target base:

$$\mathbf{-\ 12.625_{10} = -\ 1100.101_2}$$

Step #2 – Normalize:

$$\mathbf{-\ 1100.101_2 = -\ 1.100101_2 \times 2^3}$$

IEEE-754 Conversion Example

Step #3 - Fill in bit fields:

Sign: **1**

Exponent: 3 + 127 = 130 **1000 0010**

Fraction: **100101**

1 1000 0010 1001 0100 0000 0000 0000 0000 000

Addition - review

- **2 positive numbers**

```
  0 0
  00001010    (+10)10
+ 00010111    (+23)10
  00100001    (33)10
```

- **1 positive and 1 negative numbers**

```
  1 1
  00000101    (+5)10
+ 11111110    (-2)10
  00000011
(1) DISCARD CARRY
```

- **2 negative numbers**

```
  1 1
  11111111    (-1)10
+ 11111100    (-4)10
  11111011    (-5)10
(1) DISCARD
```

Overflow

- Overflow: error has happened

When?

- Numbers are of the same sign but the result is of opposite sign
- If two numbers are of opposite sign the overflow would not occur