FLOATING POINT OPERATION

Floating Point Number representation

* The location of the fractional point is not fixed to a certain location
* The range of the representable numbers is wide

$$F = EM$$

m_n	$e_k e_{k-1} \ldots e_0$	$m_{n-1} m_{n-2} \ \ldots \ m_0 . m_{-1} \ \ldots \ m_{-m}$
sign	exponent	mantissa

- Mantissa
  Signed fixed point number, either an integer or a fractional number

- Exponent
  Designates the position of the radix point

Floating Point Number Representation

Example

sign			sign	
0	.1234567		0	04
	mantissa		exponent	

==> $+.1234567 \times 10^{+04}$

Note:

In Floating Point Number representation, only Mantissa(M) and Exponent(E) are explicitly represented. The Radix(R) and the position of the Radix Point are implied.

Example

A binary number +1001.11 in 16-bit floating point number representation (6-bit exponent and 10-bit fractional mantissa)

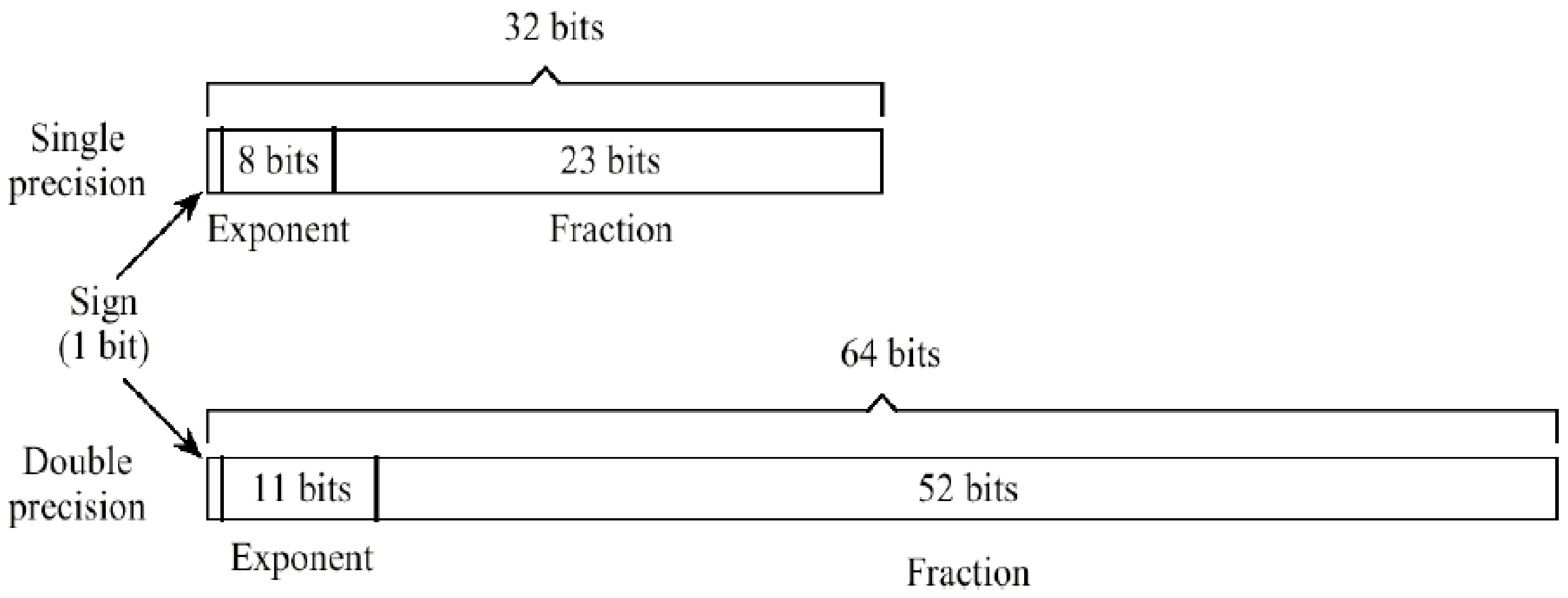
	0	0 00100	100111000
	Sign	Exponent	Mantissa
or	0	0 00101	010011100

Normal Form

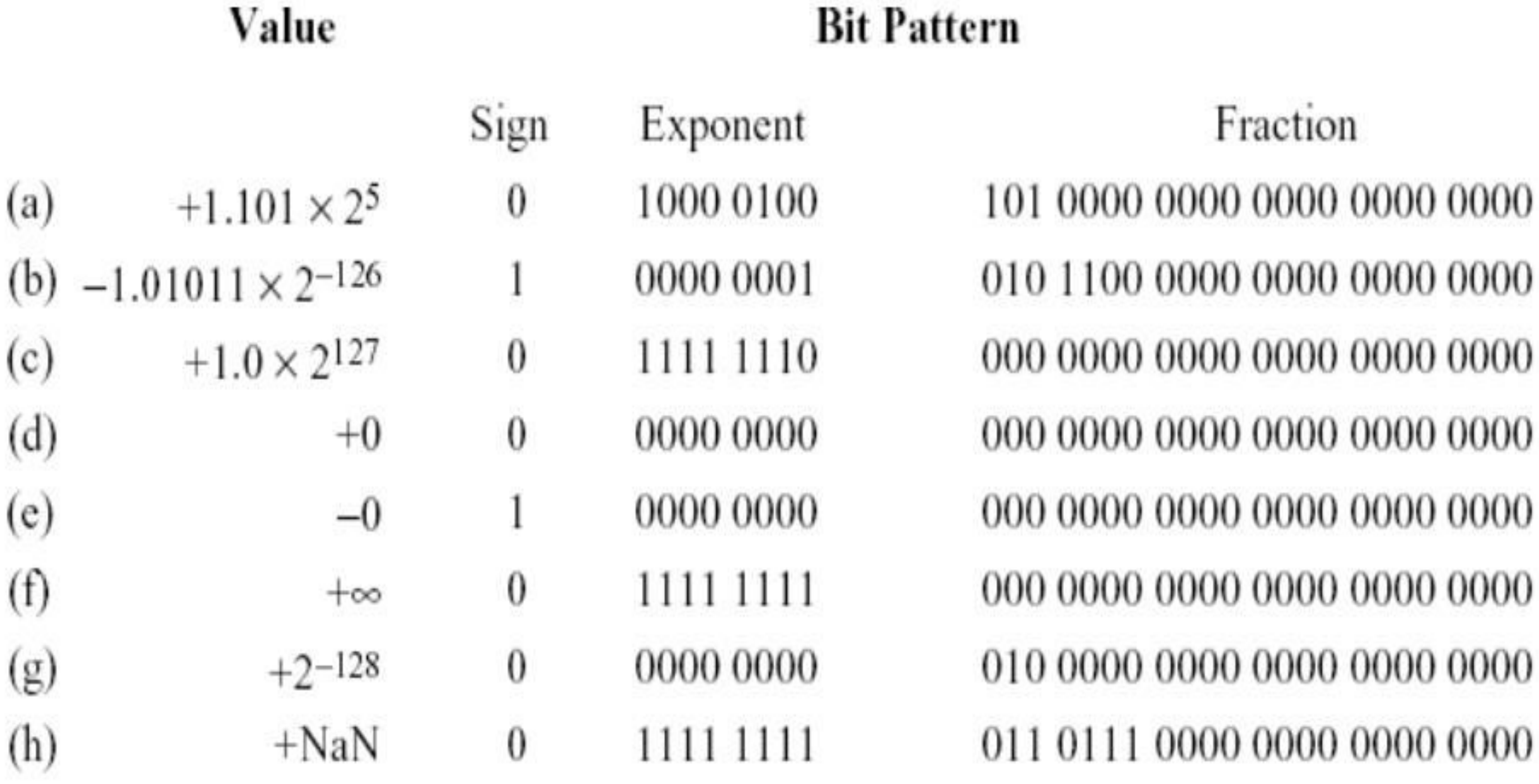
- There are many different floating point number representations of the same number
  $\rightarrow$ Need for a unified representation in a given computer

- *the most significant position of the mantissa contains a non-zero digit*

IEEE-754 Floating Point Formats



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)	+NaN	0	1111 1111	011 0111 0000 0000 0000 0000

IEEE-754 Conversion Example

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

- Step #1: Convert to target base. -12.625$_{10}$ = -1100.101$_2$
- Step #2: Normalize. -1100.101$_2$ = -1.100101$_2$ $\times 2^3$
- Step #3: Fill in bit fields. Sign is negative, so sign bit is 1. Exponent is in excess 127 (not excess 128!), so exponent is represented as the

unsigned integer 3 + 127 = 130. Leading 1 of significant is hidden, so

final bit pattern is:

1 1000 0010 . 1001 0100 0000 0000 0000 000

Character Representation ASCII

ASCII (American Standard Code for Information Interchange) Code

MSB (3 bits)

LSB (4 bits)

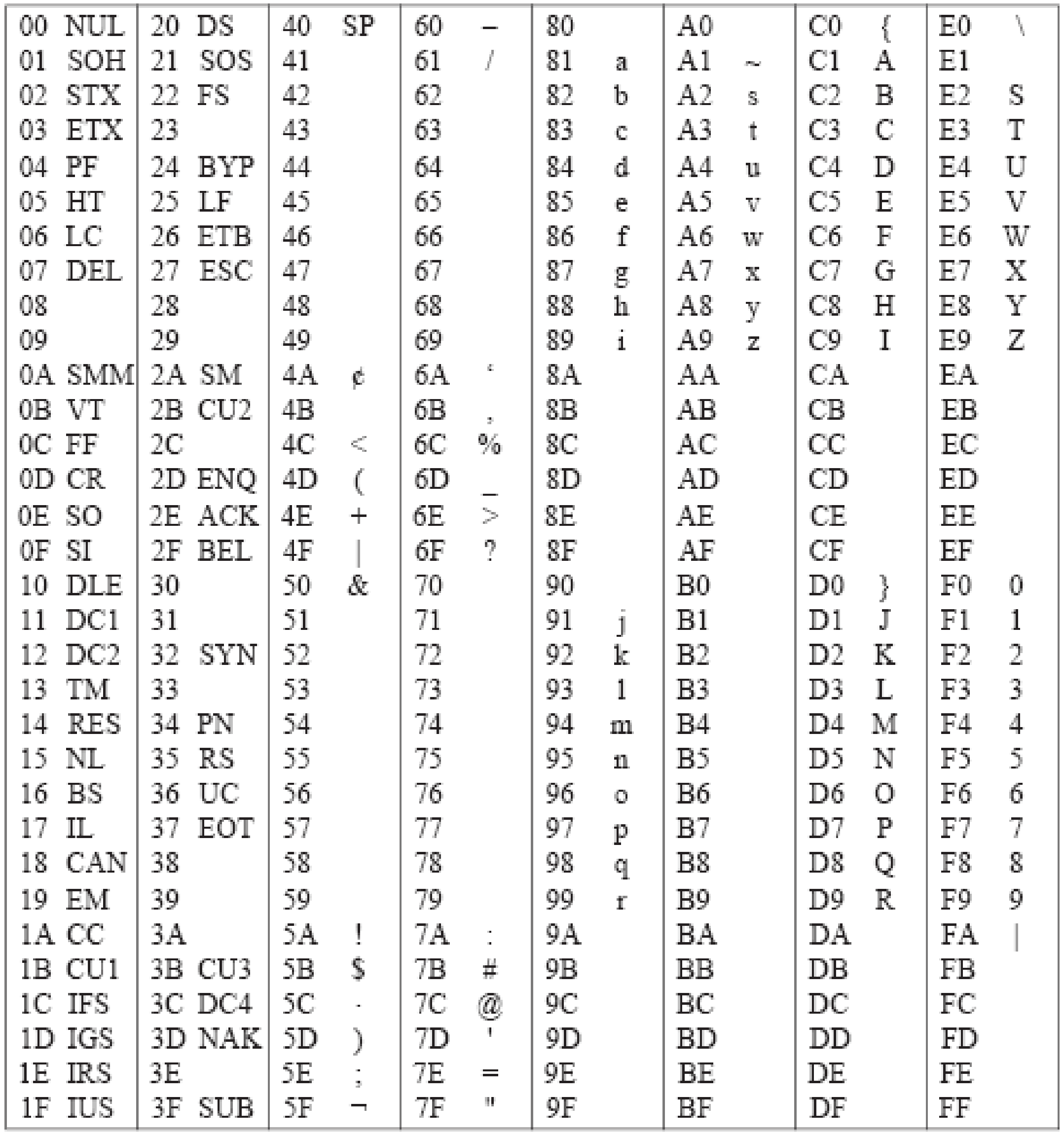
	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	‘	P
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	“	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	‘	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	I	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	m	n	~
F	SI	US	/	?	O	n	o	DEL

Control Character Representation (ASCII)

NUL	Null	DC1	Device Control 1
SOH	Start of Heading (CC)	DC2	Device Control 2
STX	Start of Text (CC)	DC3	Device Control 3
ETX	End of Text (CC)	DC4	Device Control 4
EOT	End of Transmission (CC)	NAK	Negative Acknowledge (CC)
ENQ	Enquiry (CC)	SYN	Synchronous Idle (CC)
ACK	Acknowledge (CC)	ETB	End of Transmission Block (CC)
BEL	Bell	CAN	Cancel
BS	Backspace (FE)	EM	End of Medium
HT	Horizontal Tab. (FE)	SUB	Substitute
LF	Line Feed (FE)	ESC	Escape
VT	Vertical Tab. (FE)	FS	File Separator (IS)
FF	Form Feed (FE)	GS	Group Separator (IS)
CR	Carriage Return (FE)	RS	Record Separator (IS)
SO	Shift Out	US	Unit Separator (IS)
SI	Shift In	DEL	Delete
DLE	Data Link Escape (CC)		

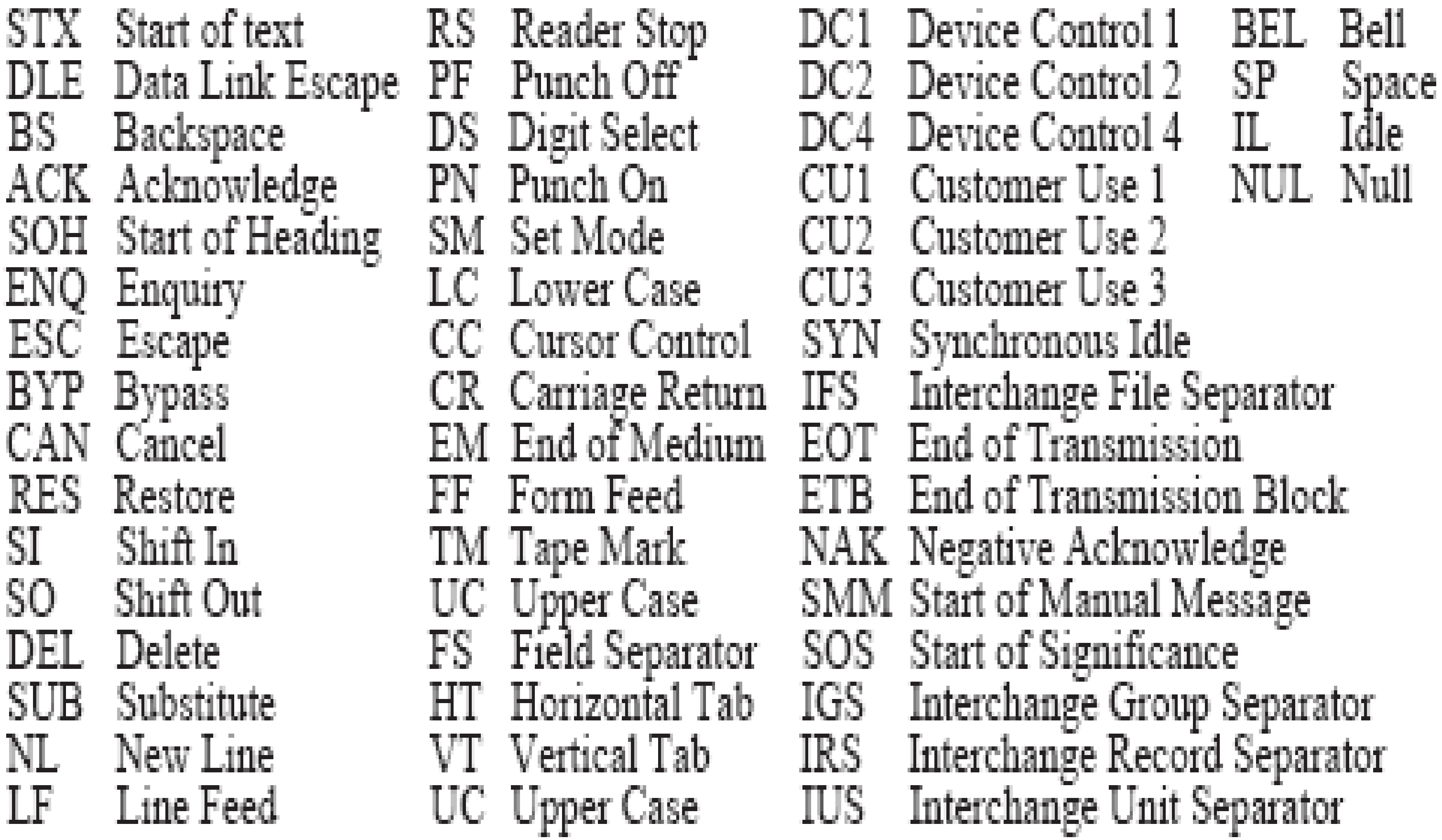
(CC) Communication Control
(FE) Format Effector
(IS) Information Separator

The EBCDIC character code, shown with hexadecimal indices



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	

The EBCDIC control character representation



STX	Start of text	RS	Reader Stop	DC1	Device Control 1	BEL	Bell
DLE	Data Link Escape	PF	Punch Off	DC2	Device Control 2	SP	Space
BS	Backspace	DS	Digit Select	DC4	Device Control 4	IL	Idle
ACK	Acknowledge	PN	Punch On	CU1	Customer Use 1	NUL	Null
SOH	Start of Heading	SM	Set Mode	CU2	Customer Use 2		
ENQ	Enquiry	LC	Lower Case	CU3	Customer Use 3		
ESC	Escape	CC	Cursor Control	SYN	Synchronous Idle		
BYP	Bypass	CR	Carriage Return	IFS	Interchange File Separator		
CAN	Cancel	EM	End of Medium	EOT	End of Transmission		
RES	Restore	FF	Form Feed	ETB	End of Transmission Block		
SI	Shift In	TM	Tape Mark	NAK	Negative Acknowledge		
SO	Shift Out	UC	Upper Case	SMM	Start of Manual Message		
DEL	Delete	FS	Field Separator	SOS	Start of Significance		
SUB	Substitute	HT	Horizontal Tab	IGS	Interchange Group Separator		
NL	New Line	VT	Vertical Tab	IRS	Interchange Record Separator		
LF	Line Feed	UC	Upper Case	IUS	Interchange Unit Separator		

References

Text Book

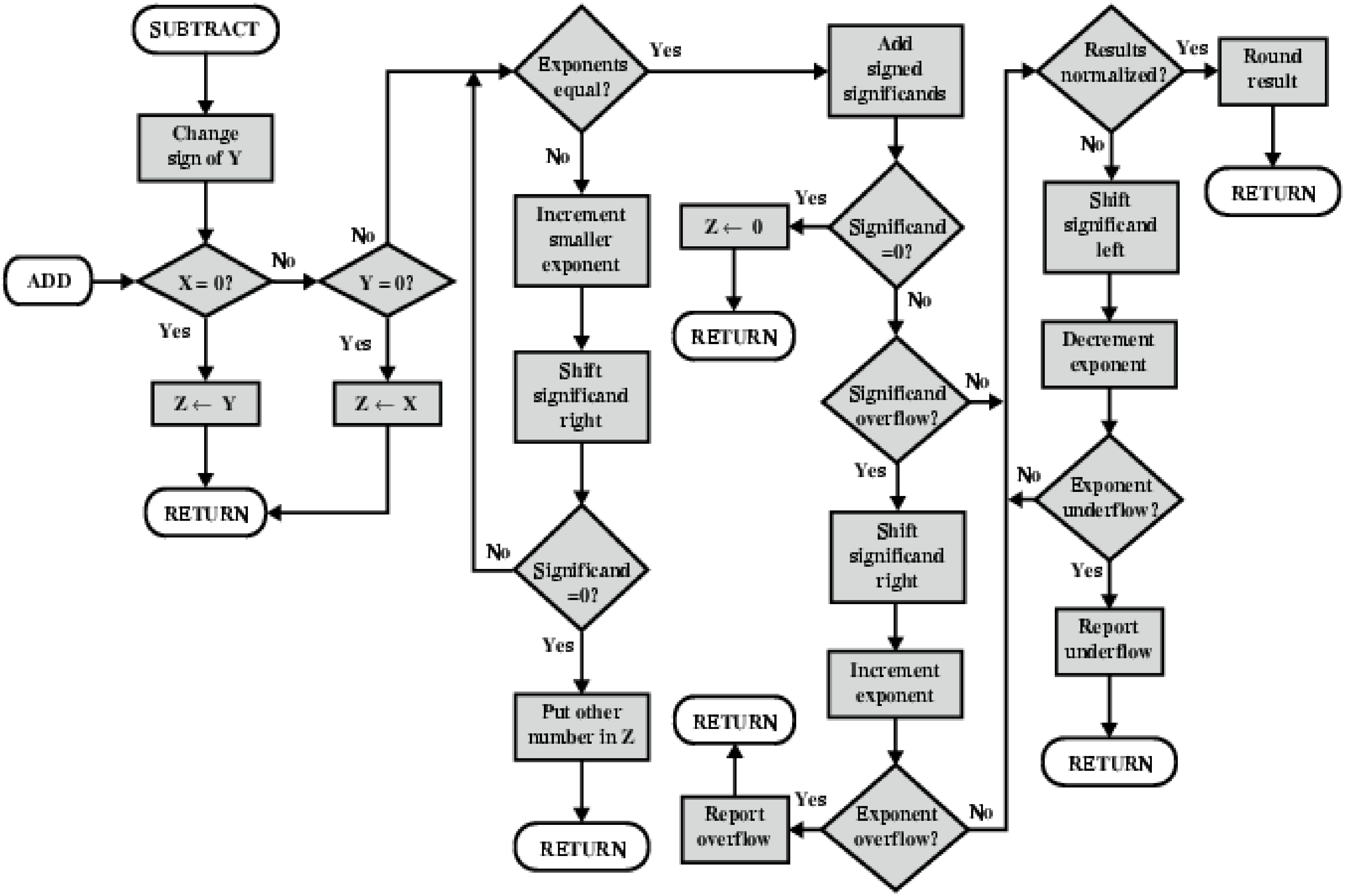
- M. M. Mano, Computer System Architecture, Prentice-Hall,2004
- William Stallings “Computer Organization and architecture” Prentice Hall, 7th edition, 2006

Floating Point Operations

FP Arithmetic +/-

- Check for zeros
- Align Mantissa (adjusting exponents)
- Add or subtract Mantissa’s
- Normalize the result

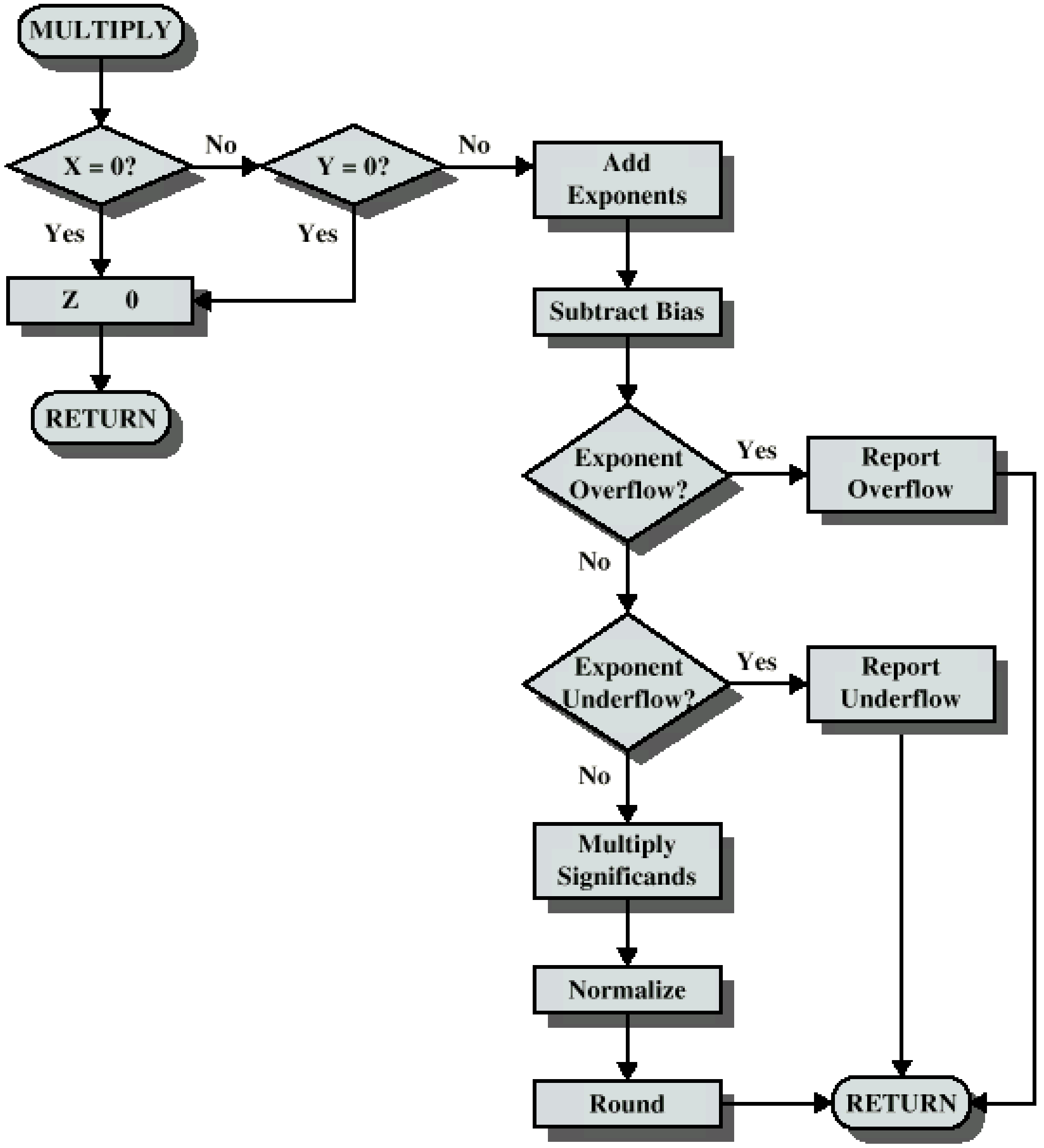
FP Addition & Subtraction Flowchart



Floating Point Multiplication

- Check for zero
- Add exponents
- Multiply Mantissa’s
- Normalize
- Round
- All intermediate results should be in double length storage

Floating Point Multiplication



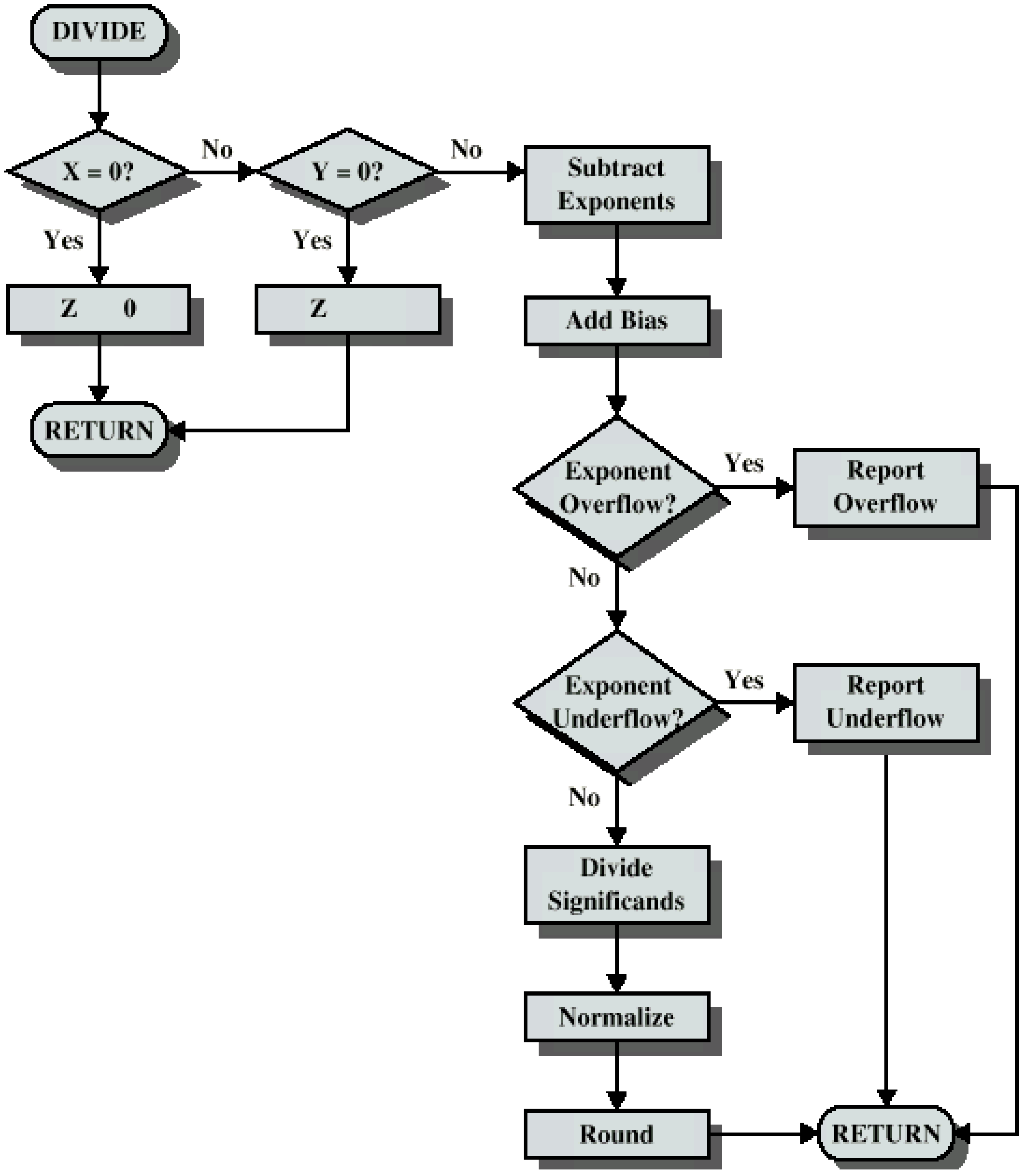
- Perform floating point multiplication:
  - 0 10000011 1100000…..
  - 0 11000000 1010000…..

  - 1 00000111 1000000…..
  - 0 11100000 1000000…..

Floating Point Division

- Check for zero
- Subtract exponents
- Divide Mantissa’s
- Normalize
- Round

Floating Point Division



References

Text Book

- William Stallings "Computer Organization and architecture" Prentice Hall, 7th edition, 2006
- http://courses.cs.tamu.edu/rabi/cpsc321/lectures/lec06.ppt