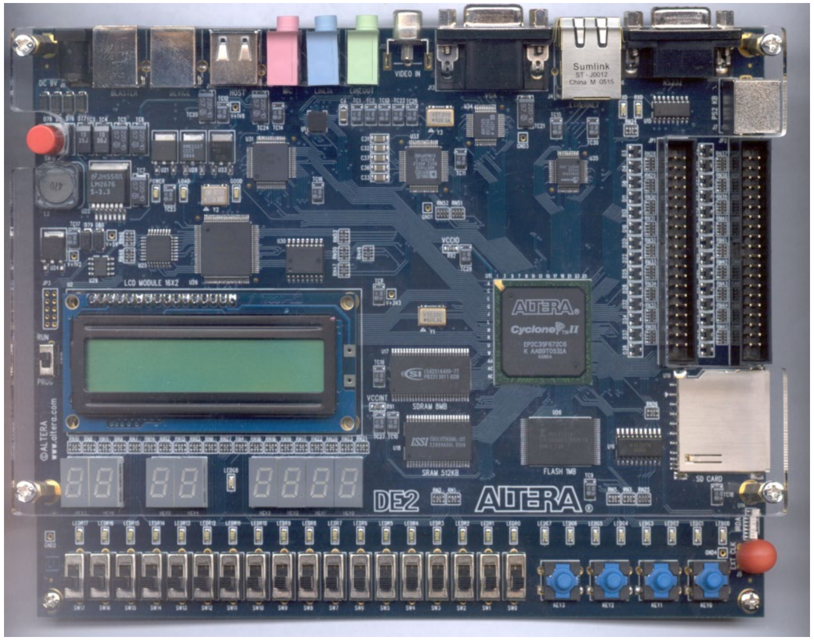
CPE 322
Digital Hardware Design Fundamentals

Electrical and Computer Engineering

UAH

Review of Basic Number Representation



Review of Number Representations

Unsigned Number

Positional Notation $\ldots K_2 K_1 K_0 \bullet K_{-1} K_{-2} \cdots$ *where* $K_i = symbols$

Integer Part *Fractional Part*

Radix Point

for base R

Number = $\ldots + K_2 R^2 + K_1 R^1 + K_0 R^0 + K_{-1} R^{-1} + K_{-2} R^{-2} + \ldots$

Review of Number Representations

Example: Unsigned Decimal Number

Positional Notation $\ldots K_2 K_1 K_0 \bullet K_{-1} K_{-2} \cdots$ *where* K_i *= digits 0 - 9*

Integer Part *Fractional Part*

Decimal Point

base R=10

$$\text{Number} = 123.45 =$$

$$1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$$

Review of Number Representations

Example: Unsigned Binary Number

Positional Notation $\ldots K_2 K_1 K_0 \bullet K_{-1} K_{-2} \cdots$ *where* K_i *= symbols 0 & 1*

Integer Part *Fractional Part*

Binary Point

base R=2

$$\text{Number} = 101.11_2$$

$$= 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}$$

$$= 5.75_{10}$$

Review of Number Representations

Unsigned Binary Number -- Finite Length (fixed point)

- Digital logic representation of number have a set number of bits.

m+f-1 m+f-2 . . . . 1 0

$$K_{m-1}K_{m-2}\ldots K_2K_1K_0 \bullet K_{-1}K_{-2}\ldots K_{-(f-1)}K_f$$

MSB *Integer Part* *Fractional Part* *LSB*

Binary Point

where K_i = symbols 0 & 1

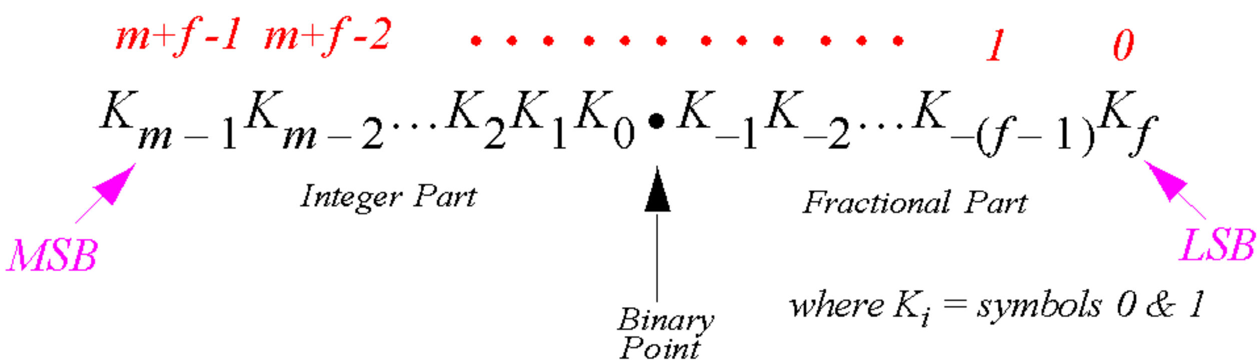
- In general, there are *m* bits to represent the integer (magnitude) portion of the number, *f* bits used to represent the fraction portion of the number.

Review of Number Representations

Signed Binary Number -- Finite Length (fixed point)

- Two's Complement is the most common signed binary number format.

- Two's complement uses standard positional notation but with the most significant bit having a negative weighting



Review of Number Representations

Signed Binary Number -- Finite Length (fixed point)

m+f-1   m+f-2   1   0

$$K_{m-1}K_{m-2}\ldots K_2K_1K_0 \bullet K_{-1}K_{-2}\ldots K_{-(f-1)}K_f$$

MSB   *Integer Part*   *Binary Point*   *Fractional Part*   *LSB*

where K_i = symbols 0 & 1

$$-K_{m-1}2^{m-1}+K_{m-2}2^{m-2}+\ldots+K_0 2^0+K_{-1}2^{-1}+\ldots+K_{-(f-1)}2^{-(f-1)}+K_{-f}2^{f}$$

MSB negative weighting   *LSB*

Integers are just the special case where *f = 0*

Review of Number Representations

Two's Complement Representation of Integers:

- Range: $2^{m-1}-1$ positive numbers

  2^{m-1} negative numbers

  1 representation of zero

MSB => sign bit: 0 = positive (or zero) 1 = negative

Review of Number Representations

Two's Complement Representation of Integers:

- To represent positive or zero numbers, simply enter binary value

  i.e. for m=8 then $15_{10} = 00001111_2$

- To represent negative numbers, first enter the binary form of the numbers absolute value, then complement each bit and add 1.

  i.e. for m=8 then -15_{10} is found by

  comp(00001111) + 1 = 11110000 + 1

  $= 11110001_2$

Review of Number Representations

Signed binary number – Finite Length (floating point):

The format for an IEEE 32-bit floating-point number is shown below

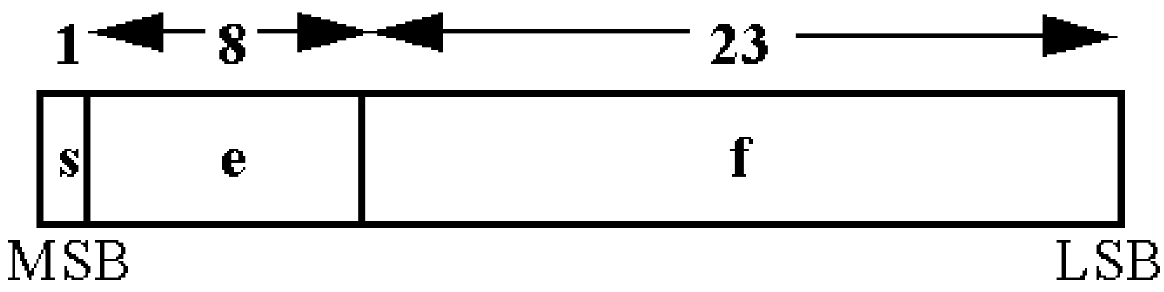
Floating Point Number = $-1^{s} \times 1.f \times 2^{e-b}$

where:

s=sign bit (0=positive mantissa, 1=negative mantissa)

e=exponent biased by b (b= 127 for IEEE 32-bit format)

f=fractional mantissa



Review of Number Representations

Signed binary number – Finite Length (floating point):

Special Case Numbers

Type	Sign	Exp	Exp+Bias	Exponent	Significand	Value
Zero	0	-127	0	0000 0000	000 0000 0000 0000 0000 0000	0.0
One	0	0	127	0111 1111	000 0000 0000 0000 0000 0000	1.0
Minus One	1	0	127	0111 1111	000 0000 0000 0000 0000 0000	−1.0
Smallest denormalized number	*	-127	0	0000 0000	000 0000 0000 0000 0000 0001	$\pm 2^{-23} \times 2^{-126} = \pm 2^{-149} \approx \pm 1.4 \times 10^{-45}$
"Middle" denormalized number	*	-127	0	0000 0000	100 0000 0000 0000 0000 0000	$\pm 2^{-1} \times 2^{-126} = \pm 2^{-127} \approx \pm 5.88 \times 10^{-39}$
Largest denormalized number	*	-127	0	0000 0000	111 1111 1111 1111 1111 1111	$\pm(1-2^{-23}) \times 2^{-126} \approx \pm 1.18 \times 10^{-38}$
Smallest normalized number	*	-126	1	0000 0001	000 0000 0000 0000 0000 0000	$\pm 2^{-126} \approx \pm 1.18 \times 10^{-38}$
Largest normalized number	*	127	254	1111 1110	111 1111 1111 1111 1111 1111	$\pm(2-2^{-23}) \times 2^{127} \approx \pm 3.4 \times 10^{38}$
Positive infinity	0	128	255	1111 1111	000 0000 0000 0000 0000 0000	$+\infty$
Negative infinity	1	128	255	1111 1111	000 0000 0000 0000 0000 0000	$-\infty$
Not a number	*	128	255	1111 1111	non zero	NaN
* Sign bit can be either 0 or 1 .						

Review of Number Representations

Binary Coded Decimal (BCD) Numbers:

- Encoding where each 'digit' (0-9) is represented by its own 4-bit binary sequence (0000_2—1001_2)
- Advantage: Easy to convert to the decimal digits needed for I/O devices and in some cases faster decimal calculations
  - Disadvantage: Requires more bits than an equivalent binary representation. Also requires more complex circuitry for arithmetic operations.

Review of Number Representations

Binary Coded Decimal (BCD) Numbers:

example:

for m=8 bit encoding:

01111000_2 => 0111 1000 => 78_{10}

Review of Number Representations

Conversion of Binary to Octal Numbers (and conversely) :

Converting binary numbers to Octal Numbers can be done by inspection

Each octal digit corresponds to 3 bits.

Just begin at the binary replace each group of three bits with the corresponding octal digit (assume leading and lagging 0's).

example:

$011010111110.0011_2 =$

$011\ 010\ 111\ 110\ .\ 001\ 100 =$

$3\quad 2\quad 7\quad 6\ .\ 1\quad 4_8$

Review of Number Representations

Conversion of Binary to Hexadecimal Numbers (and conversely) :

Converting binary numbers to Hexadecimal Numbers can also be done by inspection

Each hex digit corresponds to 4 bits.

Just begin at the binary point and replace each group of four bits with the corresponding hexadecimal symbol (0-F).

example:

$011010111110.0011_2 =$

$0110\ 1011\ 1110\ .\ 0011 =$

$6\quad B\quad E\ .\ 3_{16}$

Review of Number Representations

- Note: Conversion from binary to hexadecimal (or octal) is so easy hexadecimal and octal are often considered to be short-hand notation form binary!

Review of Number/Character Representations

ASCII Table and Description

ASCII stands for American Standard Code for Information Interchange. Computers can only understand numbers, so an ASCII code is the numerical representation of a character such as 'a' or '@' or an action of some sort. ASCII was developed a long time ago and now the non-printing characters are rarely used for their original purpose. Below is the ASCII character table and this includes descriptions of the first 32 non-printing characters. ASCII was actually designed for use with teletypes and so the descriptions are somewhat obscure. If someone says they want your CV however in ASCII format, all this means is they want 'plain' text with no formatting such as tabs, bold or underscoring - the raw format that any computer can understand. This is usually so they can easily import the file into their own applications without issues. Notepad.exe creates ASCII text, or in MS Word you can save a file as 'text only'

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Review of Number/Character Representations

- Other encodings of characters and numbers are also used. Examples, EBSIDIC, Gray codes, etc.

Gray Code					Binary			
2^3	2^2	2^1	2^0	Position	2^3	2^2	2^1	2^0
0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	1
0	0	1	1	2	0	0	1	0
0	0	1	0	3	0	0	1	1
0	1	1	0	4	0	1	0	0
0	1	1	1	5	0	1	0	1
0	1	0	1	6	0	1	1	0
0	1	0	0	▶ 7 ◀	0	1	1	1
1	1	0	0	▶ 8 ◀	1	0	0	0
1	1	0	1	9	1	0	0	1
1	1	1	1	10	1	0	1	0
1	1	1	0	11	1	0	1	1
1	0	1	0	12	1	1	0	0
1	0	1	1	13	1	1	0	1
1	0	0	1	14	1	1	1	0
1	0	0	0	15	1	1	1	1

Floating Point Representations

- A floating point binary number consists of three parts:
  - sign (S), exponent (E), and mantissa (M).
  - Each (S, E, M) pattern uniquely identifies a floating point number.
- For each bit pattern, its IEEE floating-point value is derived as:
  - value = $(-1)^S * M * \{2^E\}$, where $1.0 \leq M < 10.0_B$
- S=0 results in a positive number and S=1 a negative number.

Floating Point Representations (Mantissa part, M)

- Specifying that $1.0_B \leq M < 10.0_B$ makes the mantissa value for each floating point number unique.
  - For example, the only one mantissa value allowed for 0.5_D is M =1.0
    - $0.5_D = 1.0_B * 2^{-1}$
  - Neither $10.0_B * 2^{-2}$ nor $0.1_B * 2^{0}$ qualifies
- Because all mantissa values are of the form 1.XX…, one can omit the “1.” part in the representation.
  - The mantissa value of 0.5_D in a 2-bit mantissa is 00, which is derived by omitting “1.” from 1.00.
  - Mantissa without the implied 1 is called the *fraction*

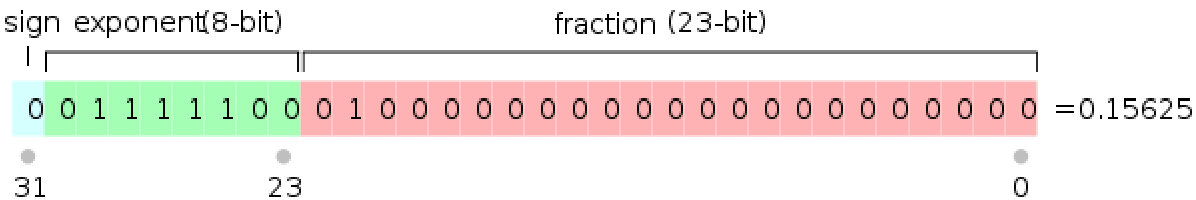
Floating Point Representations (Exponent, E)

- In an n-bits exponent representation, $2^{n-1}-1$ is added to its 2's complement representation to form its excess representation.
  - See Table for a 3-bit exponent representation
- A simple unsigned integer comparator can be used to compare the magnitude of two FP numbers
- Symmetric range for +/- exponents (case where Exponent is all 1's is reserved)

2's complement	Actual decimal	Excess-3
000	0	011
001	1	100
010	2	101
011	3	110
100	**(reserved pattern)**	**111**
101	-3	000
110	-2	001
111	-1	010

Floating Point Representations IEEE 754 Format

- **Single Precision**
  - **1 bit sign, 8 bit exponent (bias-127), 23 bit fraction**



- **Double Precision**
  - **1 bit sign, 11 bit exponent (1023-bias), 52 bit fraction**