COMP2611: Computer Organization

Data representation

Overview

- You will learn the following in this tutorial:
  - Data representations
    - Base conversions and two's complement
    - IEEE 754 single/double precision floating point numbers
    - ASCII encoding

Data representation

Data representation

- Base conversions, two's complement
- IEEE 754 floating point format,
- ASCII representation of characters

Exercises

Two's complement representation

- Computers use two's complement representation scheme for signed numbers
- Positive numbers are represented as usual
- Negative numbers are converted as follows:
  - Each bit in the number is flipped (i.e., 0 -> 1 and 1 -> 0) to get its one's complement
  - Add 1 to the one's complement to get the two's complement
- Bit 31 is the sign bit. (bit 31 is 0 for +ve numbers, and 1 for –ve numbers)
- Example : -5 ($0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0101_{(2)}$)
  - Flip all bits to get its one's complement

    $1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1010_{(2)}$
  - Add1 to the one's complement to get the two's complement

    $1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1011_{(2)}$

Two's complement representation

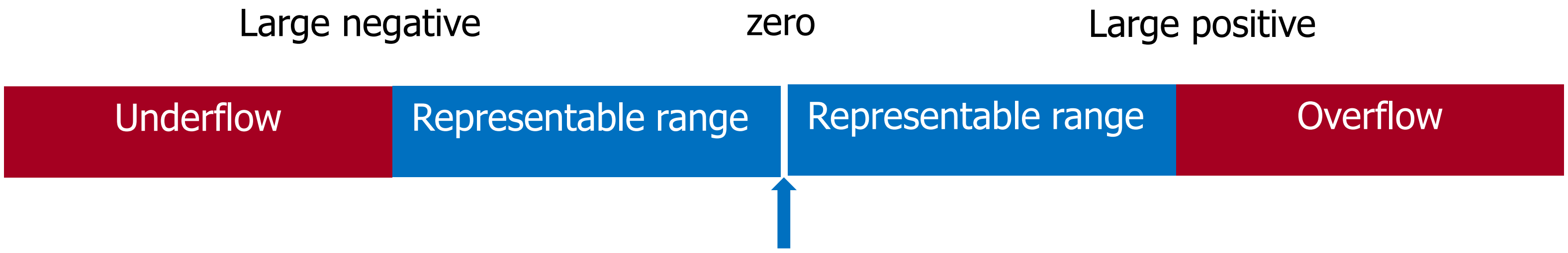
- To convert a two's complement number back to its decimal form:
  - If the sign bit is zero, convert the binary number directly to decimal format
  - Otherwise, flip the bits in the number and add 1 to the inverted number
  - Convert the result to decimal format
  - Put a –ve sign to the number
- Example : Convert the two's complement number $1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1011_{(2)}$ to decimal format
  - Sign bit =1, flip all bits:

    $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0100_{(2)}$
  - Add 1 to the inverted number:

    $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0101_{(2)}$
  - Convert the result to decimal format and add a –ve sign:

    $-5_{(10)}$

- **Overflow** (signed integer)
  - The value is bigger than the largest integer that can be represented
- **Underflow** (signed integer)
  - The value is smaller than the smallest integer that can be represented



Conversion of bases

- Example : Convert $.625_{(10)}$ to the binary format:

  $0.625 \times 2 = 1.25$ LHS of decimal pt=1 RHS =0.25

  $0.25 \times 2 = 0.5$ LHS of decimal pt=0 RHS =0.5

  $0.5 \times 2 = 1$ LHS of decimal pt=1 RHS = 0 done!

  $0.625_{(10)} => 0.101_{(2)}$

  Most significant digit

  Least significant digit

- Example : Convert $101.101_{(2)}$ to the decimal format:

  $(1x2^{2}) + (0x2^{1}) + (1x2^{0}) + (1x2^{-1}) + (0x2^{-2}) + (1x2^{-3}) = 5.625_{(10)}$

  $101.101_{(2)} => 5.625_{(10)}$

The IEEE 754 single precision floating point format

- The IEEE 754 standard uses 32 bits to represent single precision floating point numbers.

31	30	22	0
S	Exponent	significand	
1 bit	**8 bits**	**23 bits**	

- S : sign bit (0 positive, 1 negative)
- Exponent : 8-bit field, bias = 127
- Significant : 23-bit field

Exercise: Convert $-5.625_{(10)}$ to the single precision floating point format:

The IEEE 754 double precision floating point format

- The IEEE 754 standard uses 64 bits to represent double precision floating point numbers.

63	62	51	0
S	Exponent	significand	
1 bit	**11 bits**	**52 bits**	

- S : sign bit (0 positive, 1 negative)
- Exponent : 11-bit field, bias = 1023
- Significant : 52-bit field

Exercise: Convert $-5.625_{(10)}$ to the double precision floating point format:

IEEE 754 special cases

- IEEE 754 Single precision format:

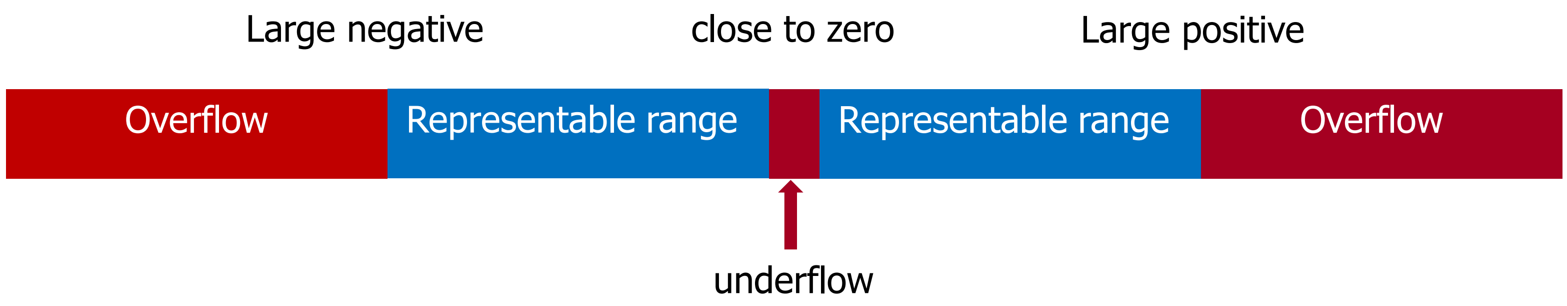
Significand \ Exponent	0	**1 - 254**	255
0	0	$(-1)^S \times (1.F) \times (2)^{E-127}$	$(-1)^S \times (\infty)$
$\neq$ 0	$(-1)^S \times (0.F) \times (2)^{-126}$		non-numbers e.g. 0/0 , $\sqrt{-1}$

- IEEE 754 Double precision format:

Significand \ Exponent	0	**1 - 2046**	2047
0	0	$(-1)^S \times (1.F) \times (2)^{E-1023}$	$(-1)^S \times (\infty)$
$\neq$ 0	$(-1)^S \times (0.F) \times (2)^{-1022}$		non-numbers e.g. 0/0 , $\sqrt{-1}$

- **Overflow** (floating-point)
  - A positive exponent becomes too large to fit in the exponent field

- **Underflow** (floating-point)
  - A negative exponent becomes too large to fit in the exponent field



Representing text– the ASCII codes

- The American Standard Code for Information Interchange (ASCII)
- ASCII is a character encoding scheme for encoding text in 8 bits
- The list of the first 128 characters are shown below

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Data representation

Data representation

- Base conversions, two’s complement
- IEEE 754 floating point format,
- ASCII representation of characters

Exercises

Question 1: Given the bit pattern 1000 0000 0100 0110 0000 0000 0000 0000

- What is the value if this is a 2’s complement representation?
- What if the pattern is an unsigned interger?
- What if it is an IEEE single precision number?
- What if it represents 4 ASCII characters (assume bits 31-24, 23-16, 15-8, 7-0 store the characters, and ASCII value of 128 is the symbol ‘€’).

Question 2: Assume the bit pattern 1001 1100 follows the IEEE-like floating point representation format

S	Exponent	significand
1 bit	3 bits	4 bits

- What is the bias of the exponent?
- What value is the given pattern representing?
- What is the range of numbers that this IEEE-like floating point representation system can represent?
- What is the granularity of this representation system?

Conclusion

- We have reviewed:
  - Simple base conversions
  - IEEE 754 floating point format
  - ASCII character scheme.