Lecture 5 In-Class Worksheet

1 Learning Objectives

This worksheet is based on Patt & Patel textbook section 2.7. After completing this lesson, you will know how to:

- [S05-1] Convert numbers to and from normalized floating point representation.
- [S05-2] Write a sequence of bits using hexadecimal notation and determine the sequence of bits written using hexadecimal notation.
- [S05-3] Encode a character using ASCII.
- [S05-4] Identify the character encoded in ASCII.

2 Floating Point Representation

A floating point number consists of a *sign* bit, *exponent* bits, and *mantissa* bits. The sign bit encodes whether the number is positive or negative, the exponent encodes the power of two multiplier, and the mantissa bits encode the fractional number to be multiplied by the power of two. The IEEE 754 floating point standard defines a 32-bit floating point representation that consists of the sign bit, followed by 8 bits used for the exponent, followed by 23 bits used for the mantissa. These are encoded as follows:

- **Sign.** The number is positive if the sign bit is 0, and negative otherwise.
- **Exponent.** The exponent of the power of two is encoded as an 8-bit unsigned integer, where the exponent is obtained by subtracting 127 from the unsigned integer.
- **Mantissa.** The mantissa is encoded by storing the 23 bits after the binary point (excluding the 1 to the left of the binary point).

The value of a number encoded using IEEE 754 32-bit floating point is given by the following formula.

Numeric value of a number in IEEE 754 32-bit floating point representation

$$\text{Value} = (-1)^S \times 1.M \times 2^{E-127} \tag{1}$$

The notation $1.M$ is shorthand for appending the bits of the mantissa, from the most significant bit to the least, after the binary point. Mathematically, this represents: $1 + M \cdot 2^{-23}$, if we take interpret M as an unsigned integer. IEEE 754 reserves the exponent code $E = 0$ for denormalized numbers and the code $E = 255$ for infinities (positive and negative) as

well as special values called Not a Number (NaN) that indicate an error. (NaN is produced, for example, when you divide 0 by 0.)

> **Q1.** What value is represented by the following bits using IEEE 754 32-bit floating point representation?
>
> 1 01001100 10110000000000000000000

From $S = 1$, we know that the sign is negative. From $E = 01001100$, we know that the exponent is $1001100_2 - 127_10 = 76 - 127 = -51$. From $M = 10110000000000000000000$, we know that the mantissa is 1.1011_2, which is $1 + \frac{1}{2} + \frac{1}{8} + \frac{1}{16} = 1.6875$. Combining these, the value represented is: -1.6875×2^{-51}.

> **Q2.** What is the IEEE 754 32-bit floating point representation of 3.125?

We need to convert the number to *normalized* binary representation. Start by converting to binary:

$$\begin{aligned} 3.125 &= 2 + 1 + 0.125 \\ &= 2 + 1 + \frac{1}{8} \\ &= 2^1 + 2^0 + 2^{-3}. \end{aligned}$$

The normalized number must be in the range $[1, 2)$, so divide by 2 to get 1.1001_2. This gives $M = 10010000000000000000000$ and $E = 1 + 127 = 128$. The 32 bits of the IEEE 754 representation are: 0 10000000 10010000000000000000000.

Not all values can be expressed exactly using IEEE 754 floating point. In particular, values that cannot be represented as a rational number whose denominator is a power of two cannot be represented perfectly. For example, to represent 3.14, we would need to represent $3\frac{14}{100} = \frac{157}{50}$ using a rational number with a denominator that is a power of two. This is not possible; the best we can do is approximate.

> ★ **Q3.** Find the best approximation of 3.14 using IEEE 754 32-bit floating point representation.

First, normalize 3.14 to obtain 1.57×2^1. We would like to find an integer number M that minimizes:

$$\left|(1 + M \cdot 2^{-23}) - 1.57\right| = \left|\frac{M}{2^{23}} - \frac{57}{100}\right|.$$

Equivalently, an M that minimizes:

$$\begin{aligned}|M - 57 \cdot 2^{23}/100| &= |M - 57 \cdot 8388608/100| \\ &= |M - 4781506.56|,\end{aligned}$$

which gives $M = 4781507 = 10010001111010111000011$. Putting it all together, the 32 bits of the IEEE 754 representation are: 0 10000000 10010001111010111000011.

3 Hexadecimal Notation

Hexadecimal notation is a way of representing groups of four bits using the digits 0–9 and the letters A–F. Converting between binary and hexadecimal does not require any arithmetic because every hexadecimal digit represents 4 bits in the binary representation. To convert from binary to hexadecimal, group bits into groups of four, and replace each group of 4 bits with the corresponding hexadecimal digit using the table on the right. Similarly, to convert from hexadecimal to binary, replace each hexadecimal digit with the corresponding 4-bit binary string.

Dec	*Hex*	*Bin*	*Dec*	*Hex*	*Bin*
0	0	0000	8	8	1000
1	1	0001	9	9	1001
2	2	0010	10	A	1010
3	3	0011	11	B	1011
4	4	0100	12	C	1100
5	5	0101	13	D	1101
6	6	0110	14	E	1110
7	7	0111	15	F	1111

> **Q4.** Express 1011001101011101 using hexadecimal notation.

Start by grouping the bits into groups of four: 1011 0011 0101 1101. Then replace each group with the corresponding hexadecimal digit from the table above to get B35D.

> **Q5.** Find the sequence of bits expressed as CAFE in hexadecimal notation.

Replace each hexadecimal digit with the corresponding 4-bit binary string, to get

1100 1010 1111 1110.

4 ASCII

The American Standard Code for Information Interchange (ASCII) is a code for representing the Latin alphabet, decimal digits, and common punctuation using 7 bits. The code is given in the appendix to this worksheet. Converting to and from ASCII is just a matter of looking up the corresponding code in the table.

Q6. What is the ASCII code for the upper-case letter E? the lower-case letter e?

The ASCII code of `E` is $69_{10} = 45_{16}$, and of `e` is $101_{10} = 65_{16}$.

Q7. What character is represented by ASCII code 43_{16}? 52_{10}?

43_{16} is the upper-case character `C`. 52_{10} is the character `4`. The *character* `4` is different from the *number* 4, which is represented using either unsigned or two's complement binary representation and represents a quantity. The character `4` is an unit of text, like the letter `C` or the punctuation symbol `?`.

Q8. You are given an ASCII code of a character representing a decimal digit. The code x is given in 8-bit two's complement representation. What two's complement value can you add to x to get the numeric value of the encoded digit? For example, if $x = 52$, the encoded digit is `4`. We would like to add a two's complement value to x to get the number 4.

We note that the characters `0` to `9` representing decimal digits 0 to 9 have ASCII codes 48 to 57, respectively. This means we need to map 48 to 0, 49 to 1, ..., and 57 to 9. because the mapping is consecutive, we can find number represented by the digit with ASCII code x by subtracting 48. To subtract 48, we can add -48, which has two's complement representation `11010000`.

Appendix

Decimal

0	nul	1	soh	2	stx	3	etx	4	eot	5	enq	6	ack	7	bel
8	bs	9	ht	10	nl	11	vt	12	np	13	cr	14	so	15	si
16	dle	17	dc1	18	dc2	19	dc3	20	dc4	21	nak	22	syn	23	etb
24	can	25	em	26	sub	27	esc	28	fs	29	gs	30	rs	31	us
32	sp	33	!	34	"	35	#	36	$	37	%	38	&	39	’
40	(	41	)	42	*	43	+	44	,	45	-	46	.	47	/
48	0	49	1	50	2	51	3	52	4	53	5	54	6	55	7
56	8	57	9	58	:	59	;	60	<	61	=	62	>	63	?
64	@	65	A	66	B	67	C	68	D	69	E	70	F	71	G
72	H	73	I	74	J	75	K	76	L	77	M	78	N	79	O
80	P	81	Q	82	R	83	S	84	T	85	U	86	V	87	W
88	X	89	Y	90	Z	91	[	92	\	93	]	94	^	95	_
96	‘	97	a	98	b	99	c	100	d	101	e	102	f	103	g
104	h	105	i	106	j	107	k	108	l	109	m	110	n	111	o
112	p	113	q	114	r	115	s	116	t	117	u	118	v	119	w
120	x	121	y	122	z	123	{	124	|	125	}	126	~	127	del

Hexadecimal

00	nul	01	soh	02	stx	03	etx	04	eot	05	enq	06	ack	07	bel
08	bs	09	ht	0a	nl	0b	vt	0c	np	0d	cr	0e	so	0f	si
10	dle	11	dc1	12	dc2	13	dc3	14	dc4	15	nak	16	syn	17	etb
18	can	19	em	1a	sub	1b	esc	1c	fs	1d	gs	1e	rs	1f	us
20	sp	21	!	22	"	23	#	24	$	25	%	26	&	27	’
28	(	29	)	2a	*	2b	+	2c	,	2d	-	2e	.	2f	/
30	0	31	1	32	2	33	3	34	4	35	5	36	6	37	7
38	8	39	9	3a	:	3b	;	3c	<	3d	=	3e	>	3f	?
40	@	41	A	42	B	43	C	44	D	45	E	46	F	47	G
48	H	49	I	4a	J	4b	K	4c	L	4d	M	4e	N	4f	O
50	P	51	Q	52	R	53	S	54	T	55	U	56	V	57	W
58	X	59	Y	5a	Z	5b	[	5c	\	5d	]	5e	^	5f	_
60	‘	61	a	62	b	63	c	64	d	65	e	66	f	67	g
68	h	69	i	6a	j	6b	k	6c	l	6d	m	6e	n	6f	o
70	p	71	q	72	r	73	s	74	t	75	u	76	v	77	w
78	x	79	y	7a	z	7b	{	7c	|	7d	}	7e	~	7f	del