COMPUTERS AND PROGRAMMING

Steven M. Rubin

Lesson 7:

Computer Numbers and Other Data
(a curiosity lesson)

INTRODUCTION TO CURIOSITY

- Ignore the details, just sit back and listen
- See what's inside these machines
- Understand how computers think
- Appreciate the complexity

- Today's lesson:
  - Find out how numbers and other data are stored

BINARY

Decimal	Binary
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111
16	10000

Bit: a single binary digit (0/1)

Byte: 8 bits, from 00000000 to 11111111 (0 to 255)

HEXADECIMAL

Decimal	Binary	Hexadecimal
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10

Hexadecimal from Reboot

One byte can be expressed as two hexadecimal digits

Example: $2D_{16}$ is 0010 1101 and 00101101_2 is 45_{10}

POWERS OF TWO

$2^0 = 1$ binary: 1

$2^1 = 2$ binary: 10

$2^2 = 2x2 = 4$ binary: 100

$2^3 = 2x2x2 = 8$ binary: 1000

$2^4 = 2x2x2x2 = 16$ binary: 10000

…

$2^9 = 512$ binary: 1000000000

$2^{10} = 1024$ binary: 10000000000

"MEGA" IS NOT A MILLION

Kilobyte	2^{10} (1024)	$\approx$ 1,000
Megabyte	2^{20} (1,048,576)	$\approx$ 1,000,000
Gigabyte	2^{30} (1,073,741,824)	$\approx$ 1,000,000,000
Terabyte	2^{40} (1,099,511,627,776)	$\approx$ 1,000,000,000,000

Beyond that: Petabyte (2^{50}), Exabyte (2^{60}),
Zettabyte (2^{70}), Yottabyte (2^{80})

Approximately, 2^{10} is 10^3 (thousand)
Approximately, 2^{20} is 10^6 (million)
Approximately, 2^{30} is 10^9 (billion)
Approximately, 2^{40} is 10^{12} (trillion)

Megabyte from Reboot

DISK DRIVES CHEAT

- Drive claims to has 1 terabyte
- Fine print says it has 1,000,000,000,000 bytes
- That is NOT a terabyte, it's a trillion bytes
  - Terabyte is 1,099,511,627,776
  - Missing about 99 gigabytes of capacity!
- But they are excused
  - Comes close enough
  - Low-level formatting uses data

NAMES FOR SMALL NUMBERS

- Milli is a thousandth (precisely)
- Micro is a millionth
- Nano is a billionth
- Pico is a trillionth
- Beyond that: femto, atto,

  zepto, yocto
- Admiral Grace Hopper carried a nanosecond (11.8 inches)

WOMEN IN COMPUTING

- Ada Lovelace (mathematician)
  - First programmer

- Grace Hopper (PhD in Mathematics)
  - Made COBOL, first computer language

- Anita Borg (PhD in Computer Science)
  - Founded Inst. for Women in Technology
    - Now called AnitaB.org
    - Yearly Grace Hopper Celebrations

- Amy Lansky (PhD in Computer Science)
  - Concurrency and A.I., married Steven Rubin

INTEGERS USE BINARY

- What about negative numbers?
  - Sign is in the high bit
  - System is called 2's complement
- Negation in 2's complement:
  - Flip all bits and add 1
  - Works in both directions
  - Addition and subtraction are easy
- Can store more negative than positive
- Overflow creates random values (7+3 = -6)

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1

JAVA INTEGERS

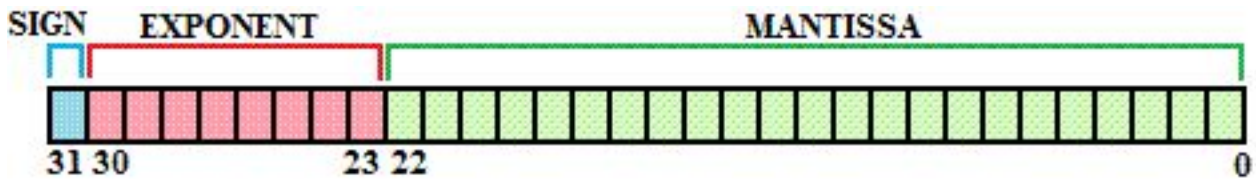
- "int" is 4 bytes (32 bits)
  - This is why it holds ±2 billion
  - Gigabyte, $2^{30} = 1,073,741,824$
  - So $2^{31} = 2,147,483,648$
  - 32^{nd} bit is for the sign, so this is the limit
- "long" is 8 bytes (64 bits, ±9 quintillion)
- "short" is 2 bytes (16 bits, ±32,768)
- "byte" is 1 byte (8 bits, ±127)
- 4-bit integer jokingly called a "nibble" (not Java)

FLOATS ARE COMPLEX

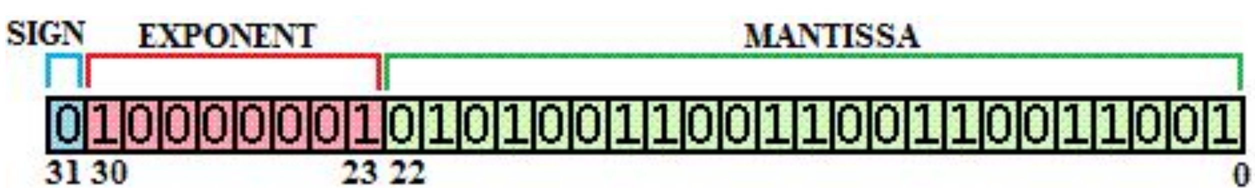
- Numbers can be viewed as mantissa and exponent
  - So 5.3 would be 0.53×10^1 (mantissa=0.53, exponent=1)
  - Very big and very small numbers work well
    - 75,000,000 (75 million) is 0.75×10^8
    - 0.000000006 (6 billionths) is 0.6×10^{-8}
  - It's common to "normalize" the mantissa
    - Make it's value be from 0 to 1
  - In computers, the exponent is a power of 2, not 10

FLOATS IN COMPUTERS

- The Java "float" is 4-bytes (32-bits)



  - Number has value Mantissa x 2^{Exponent}
  - Sign bit is for mantissa which has decimal point at the left
  - Exponent is signed by subtracting "bias" (127)
  - Example: 5.3_{10} is $101.0100110011001100110011_2$
    - Normalize (shift decimal left 2 times): 1.0101001100110011001100110011
      - So exponent is 2, plus bias of 127 = 129 or 10000001_2
    - Sign is 0, mantissa is 01010011001100110011001

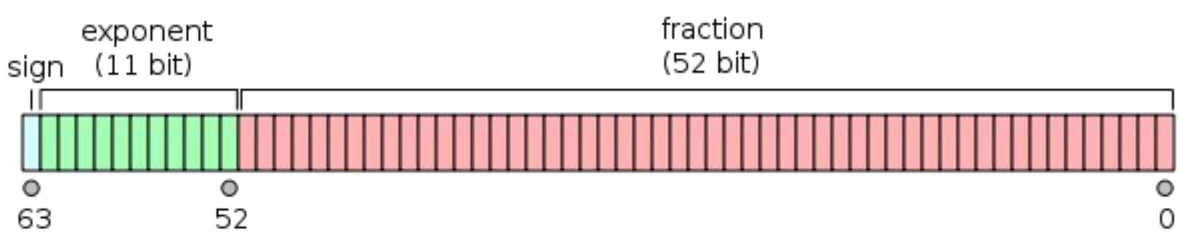


FLOATS ARE INACCURATE

- Conversion from base 10 to base 2 is not precise
  - Not enough bits in the mantissa
  - 32-bit floats have about 6 digits of precision
  - 75,000,000 has 2 digits of precision (75)
  - 0.000000006 has 1 digit of precision (6)
  - 75000000.000000006 has 17 digits of precision
    - Much more than a float can hold
    - So 75,000,000 + 0.000000006 = 75,000,000
- Entire subfield of Computer Science studies this
  - Called Numerical Analysis

JAVA DOUBLES

- Java "double" has 8 bytes (64-bits)
  - Three more exponent bits, much longer mantissa
  - Good for about 16 digits of precision
  - Still can't represent 75000000.000000006



INFINITE PRECISION

- Need more? Java can do it
- BigDecimal objects can store anything
  - Implemented with an array of digits
  - Arithmetic done laboriously, can be slow

WHAT ABOUT CHARACTERS?

- Originally stored using ASCII (7-bit code)

ASCII American Standard Code for Information Interchange

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

INTERNATIONAL CHARACTERS

- ASCII is limited to the English alphabet
- UTF is broader (Unicode Transmission Format)
  - Starts with ASCII, adds more bytes if 8th bit is on

UTF-8 Encoding

Bits	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
7	0XXXXXX					
11	110XXXXX	10XXXXXX				
16	1110XXXX	10XXXXXX	10XXXXXX			
21	11110XXX	10XXXXXX	10XXXXXX	10XXXXXX		
26	111110XX	10XXXXXX	10XXXXXX	10XXXXXX	10XXXXXX	
31	1111110X	10XXXXXX	10XXXXXX	10XXXXXX	10XXXXXX	10XXXXXX

Character	UTF-8
A	41
c	63
Ö	C3 B6
亜	E4 BA 9C
𝄞	F0 9D 84 9E

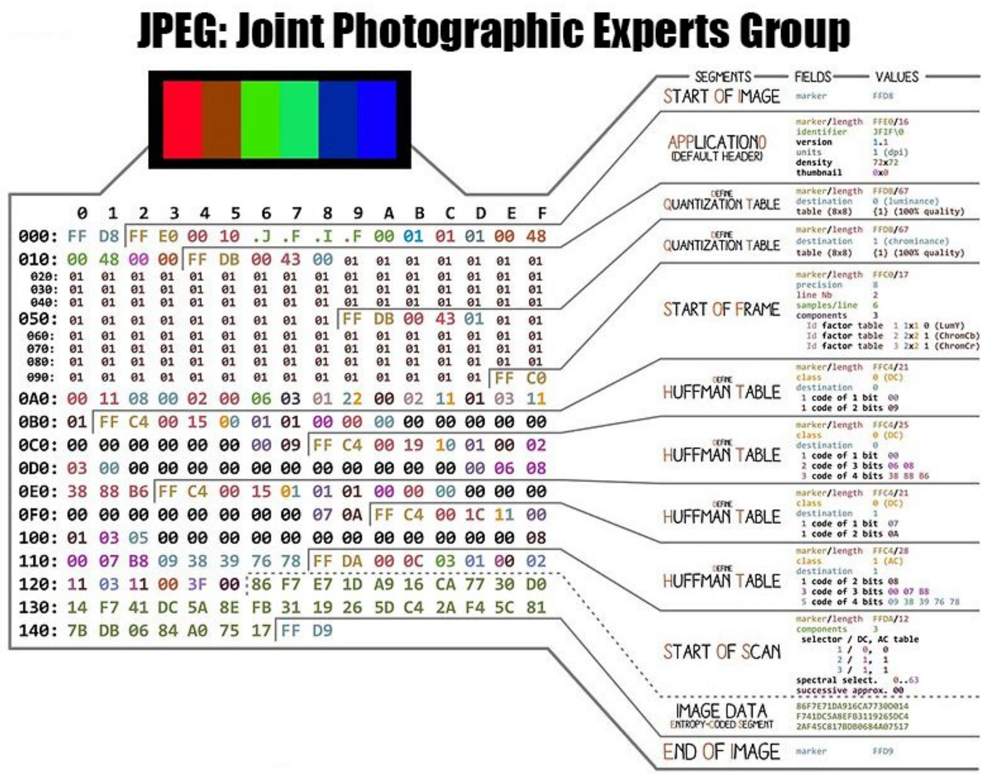
JAVA CHARACTERS

- UTF usually uses only 2 or 3 bytes
- Inside Java, characters are stored in 2-bytes

```
char capA = 'A';
int capAValue = (int)capA;    // set to 65
```

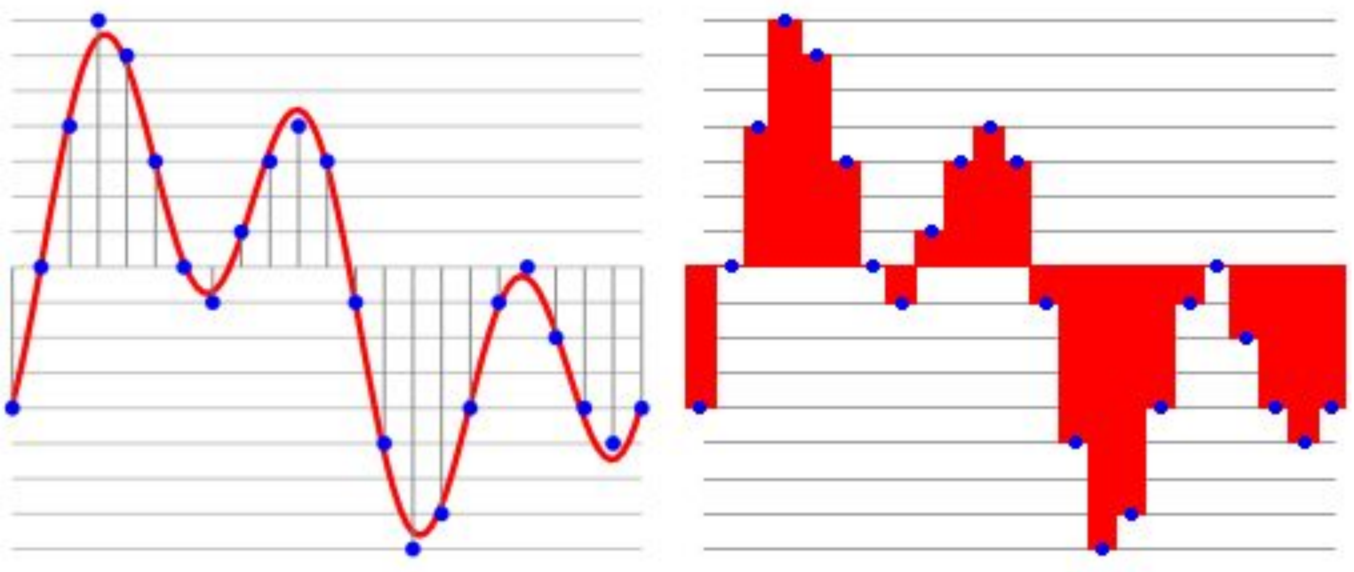
WHAT ABOUT PICTURES?

- Every container has its own format
- JPG is lossy, GIF and PNG are lossless
  - But GIF is limited and patented



WHAT ABOUT SOUND?

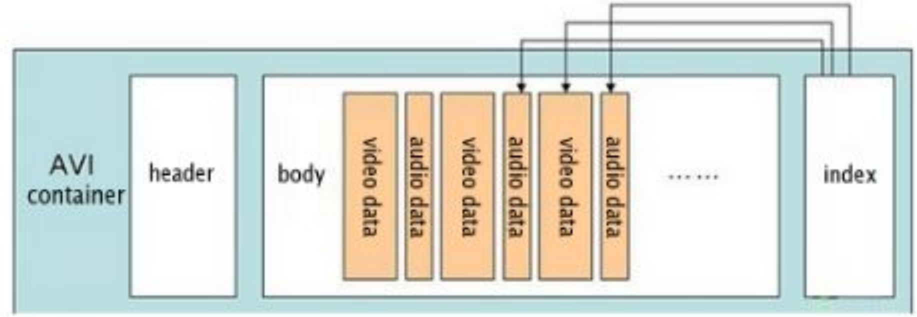
- Sound is digitized (thousands of times/second)
- Approximates the original sound waves



- Gets compressed (MP3 is lossy)

WHAT ABOUT VIDEO?

- Videos need audio too
- Video containers have both (plus subtitles, etc.)
  - AVI, MP4, MKV (Matroska)



NOW YOU KNOW

- Everything is reduced to bits
  - The only language a computer understands
- Good thing we don't program in binary!