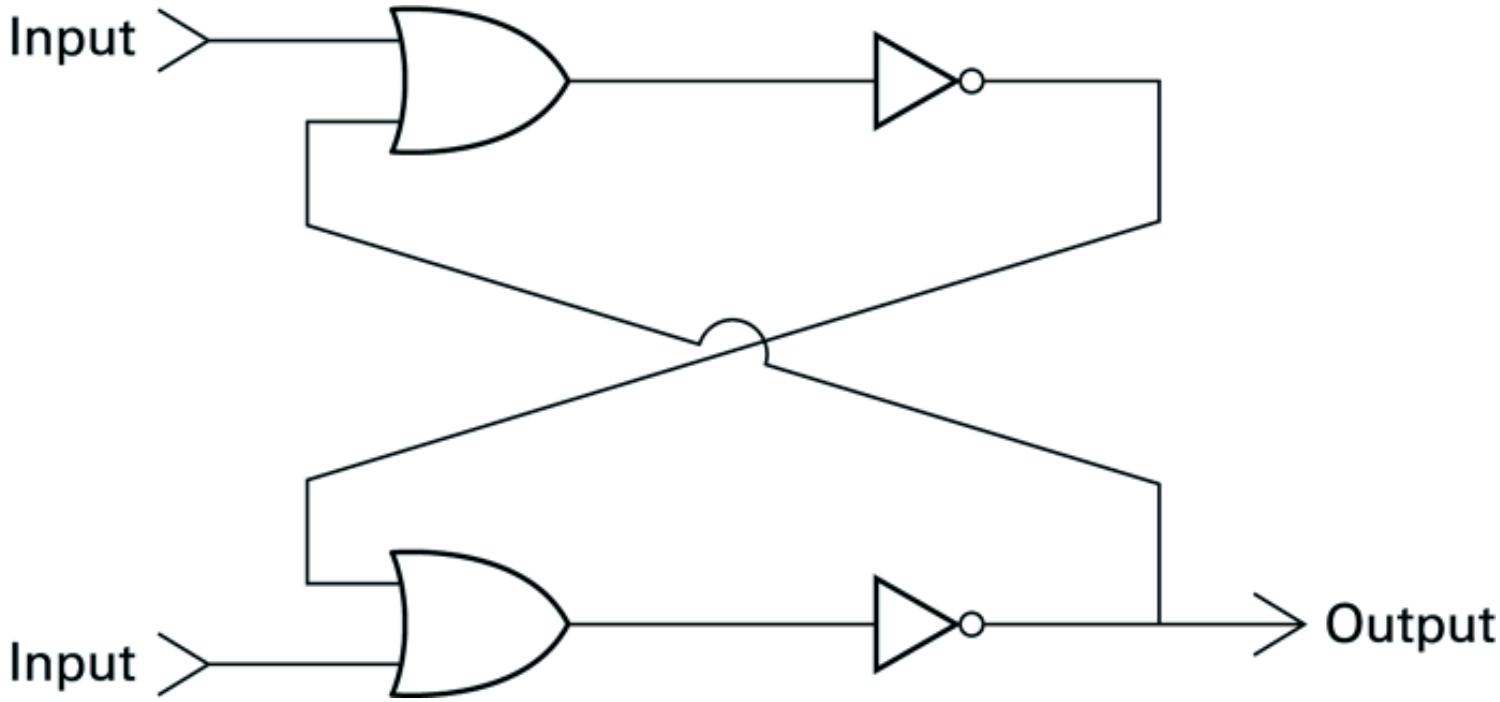
Week 4th

Prepared by Dr Syed Khaldoon Khurshid

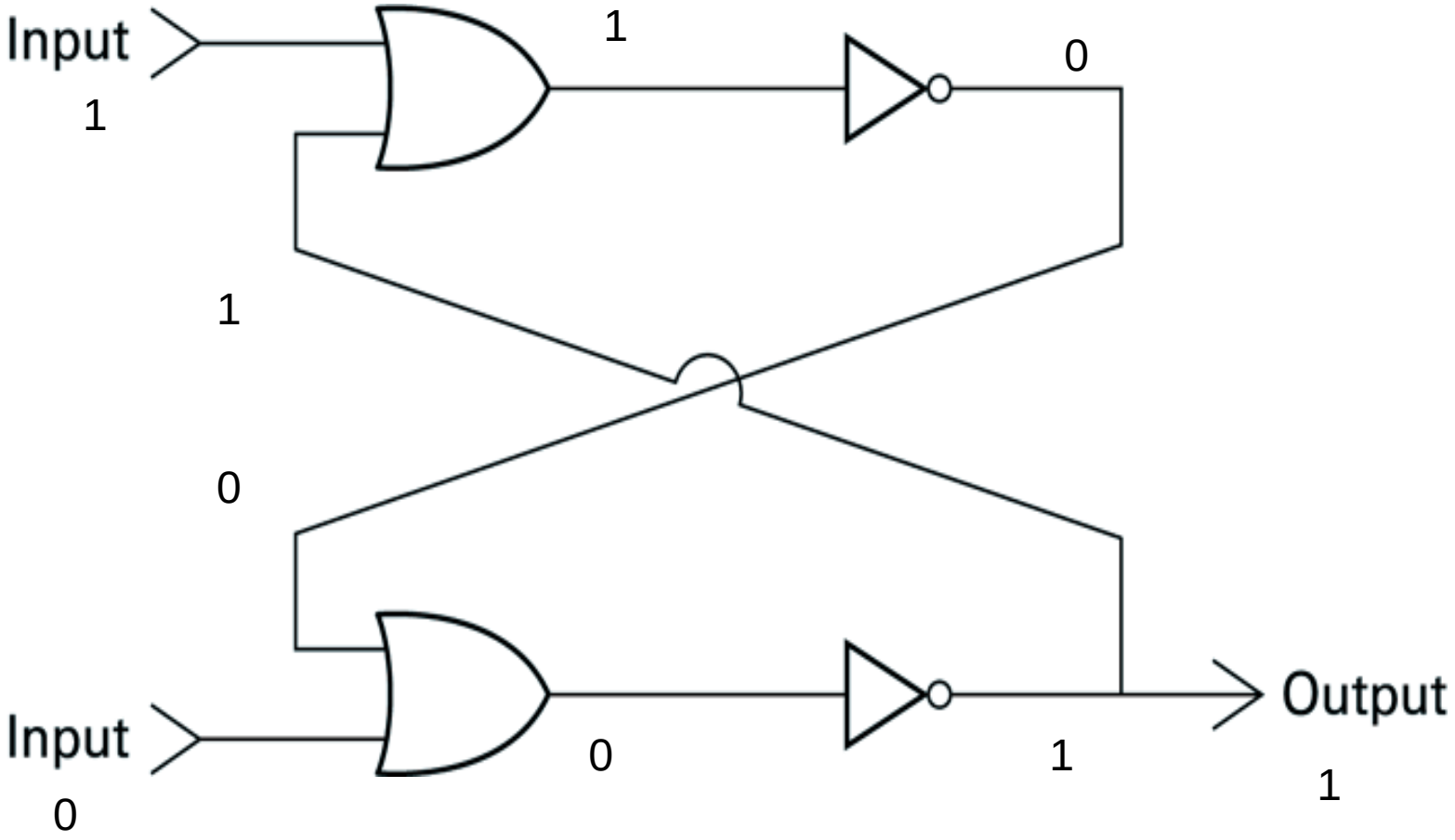
Another way of constructing Flip-Flop

Assignment: **Write outputs & sequence of steps on the basis of following inputs?**

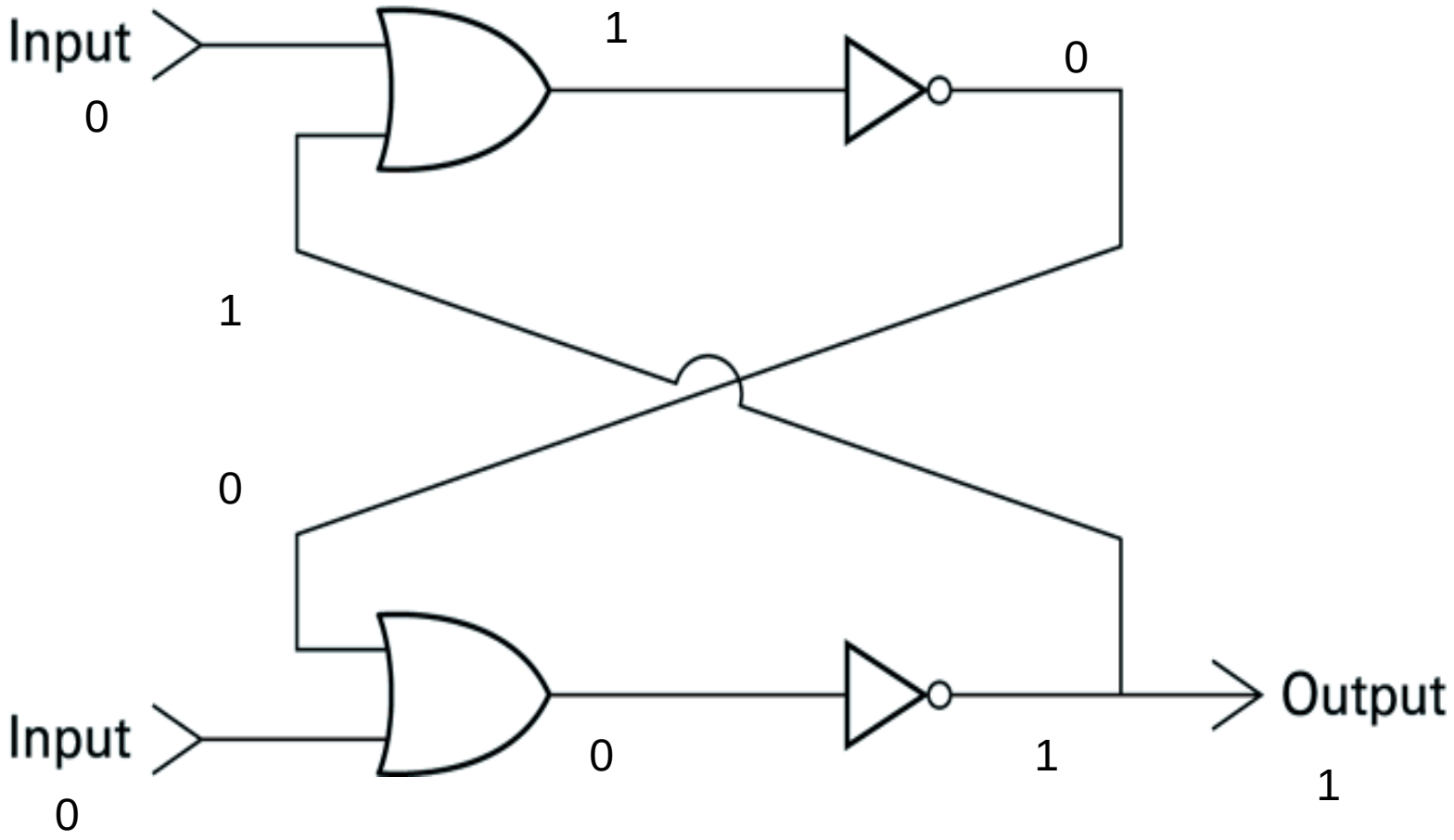
- **upper input = 1 and lower input = 0**
- **upper input = 0 and lower input = 0**



Solution:



Solution:



1.4 Representing Information as Bit Patterns

- Now that we know how to store single bits, we can consider how *information* can be encoded as *bit patterns*
- Different encoding systems exist for different types of information
  - numbers, text, images, sound, …
- Encoding systems more and more standardized
  - American National Standards Institute (ANSI)
  - International Organization for Standardization (ISO)

1.4 Representing Text

- Each symbol represented by a unique bit *pattern*
- Text represented by long *stream of patterns*
- Today's standard coding system:
  - ASCII (American Standard Code for Information Interchange)
  - Bit patterns of length 7 (generally extended by 1 bit)
  - See ASCII-table in Appendix A.

01001000	01100101	01101100	01101100	01101111	00101110
H	**e**	**l**	**l**	**o**	**.**

ASCII

- 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.**
- ASCII was actually designed for use with teletypes and so the descriptions are somewhat obscure.

ASCII

- **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**'.
- In the next slide is the ASCII character table and this includes descriptions of the first 32 non-printing characters.

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

Extended ASCII Codes

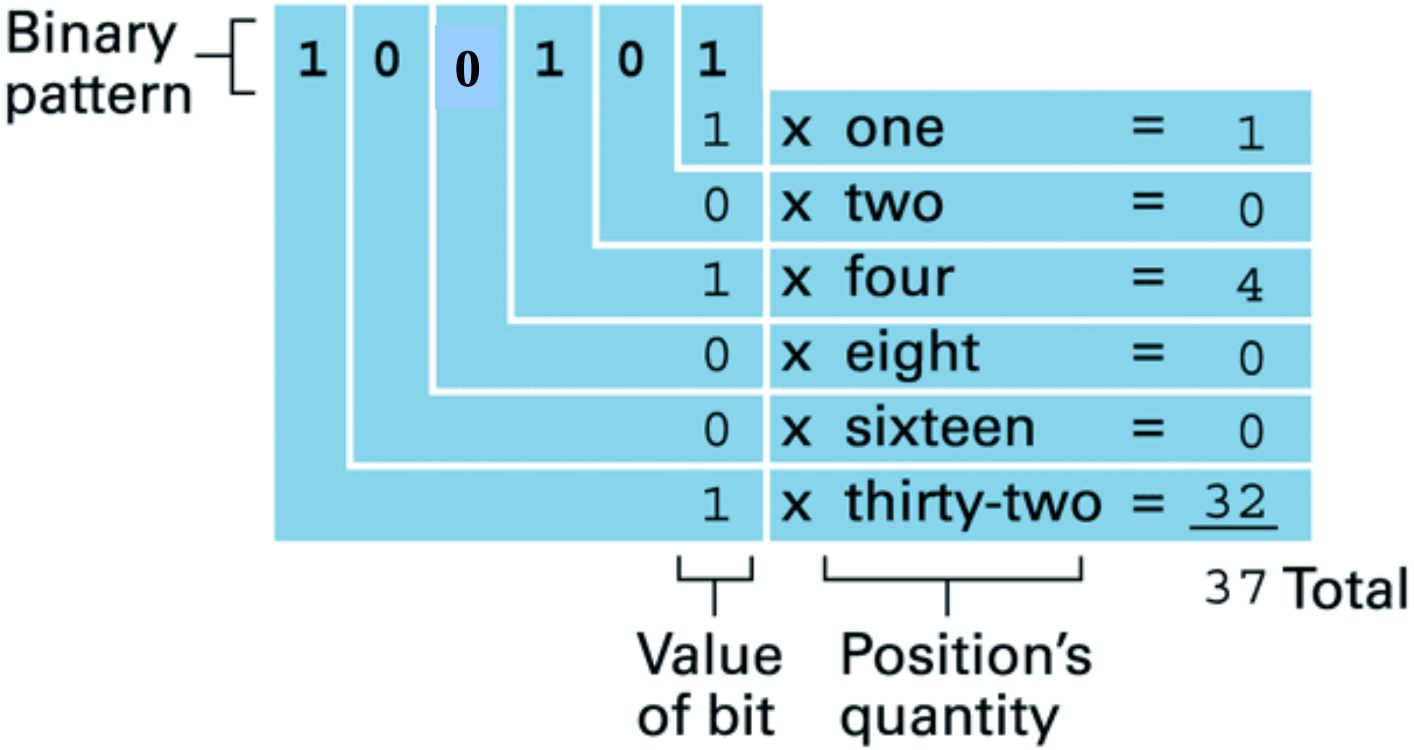
128	Ç	144	É	160	á	176	▒	192	└	208	╨	224	α	240	≡
129	ü	145	æ	161	í	177	▓	193	┴	209	╤	225	ß	241	±
130	é	146	Æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
131	â	147	ô	163	ú	179	|	195	├	211	╙	227	π	243	≤
132	ä	148	ö	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
133	à	149	ò	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
134	å	150	û	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
135	ç	151	ù	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
136	ê	152	ÿ	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
137	ë	153	Ö	169	⌐	185	╣	201	╔	217	┘	233	Θ	249	∙
138	è	154	Ü	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
139	ï	155	¢	171	½	187	╗	203	╦	219	█	235	δ	251	√
140	î	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	n
141	ì	157	¥	173	¡	189	╜	205	═	221	▌	237	φ	253	²
142	Ä	158	₧	174	«	190	╛	206	╬	222	▐	238	ε	254	■
143	Å	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	

Source: www.LookupTables.com

1.4 Representing Numbers

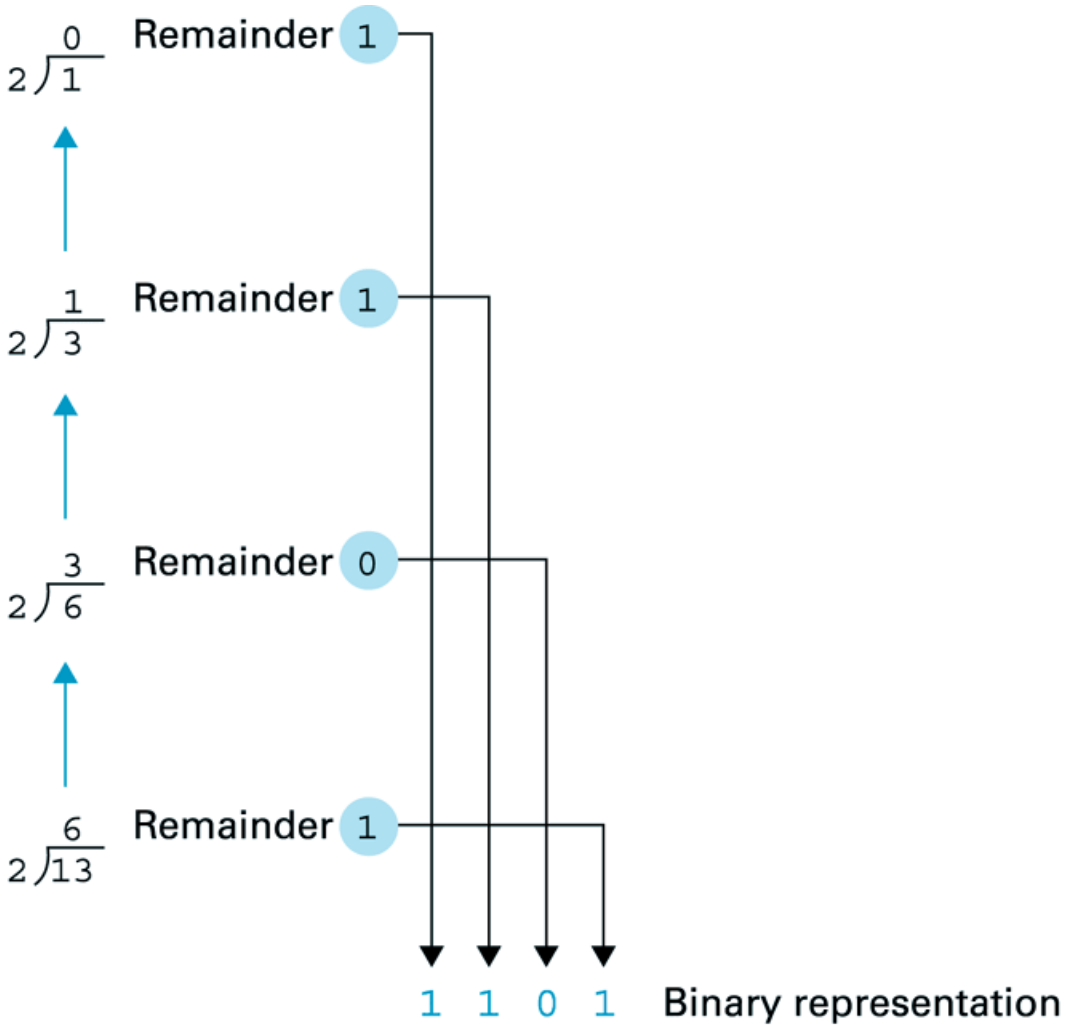
- **ASCII-encoding inefficient for numeric values**
- **Consider storing the value 25:**
  - **In ASCII: 00110010 00110101 (16 bits)**
  - Worse: largest 16-bit number would be 99
- **More efficient approach is to use *binary system***
  - uses digits 0 and 1, incl. factor 2 for all bit-positions
- **Compare decimal system**
  - uses digits 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9, incl. factor 10 for each decimal position

1.4 Decoding the Binary Representation 100101



– $1\times2^5 + 0\times2^4 + 0\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 = 37$

1.4 Obtaining the binary representation of 13



1.5 The Binary System: Addition

- Knowing how numeric values are encoded, we can consider how to do calculations
- Binary addition:

0	1	0	1
+0	+0	+1	+1
0	1	1	10

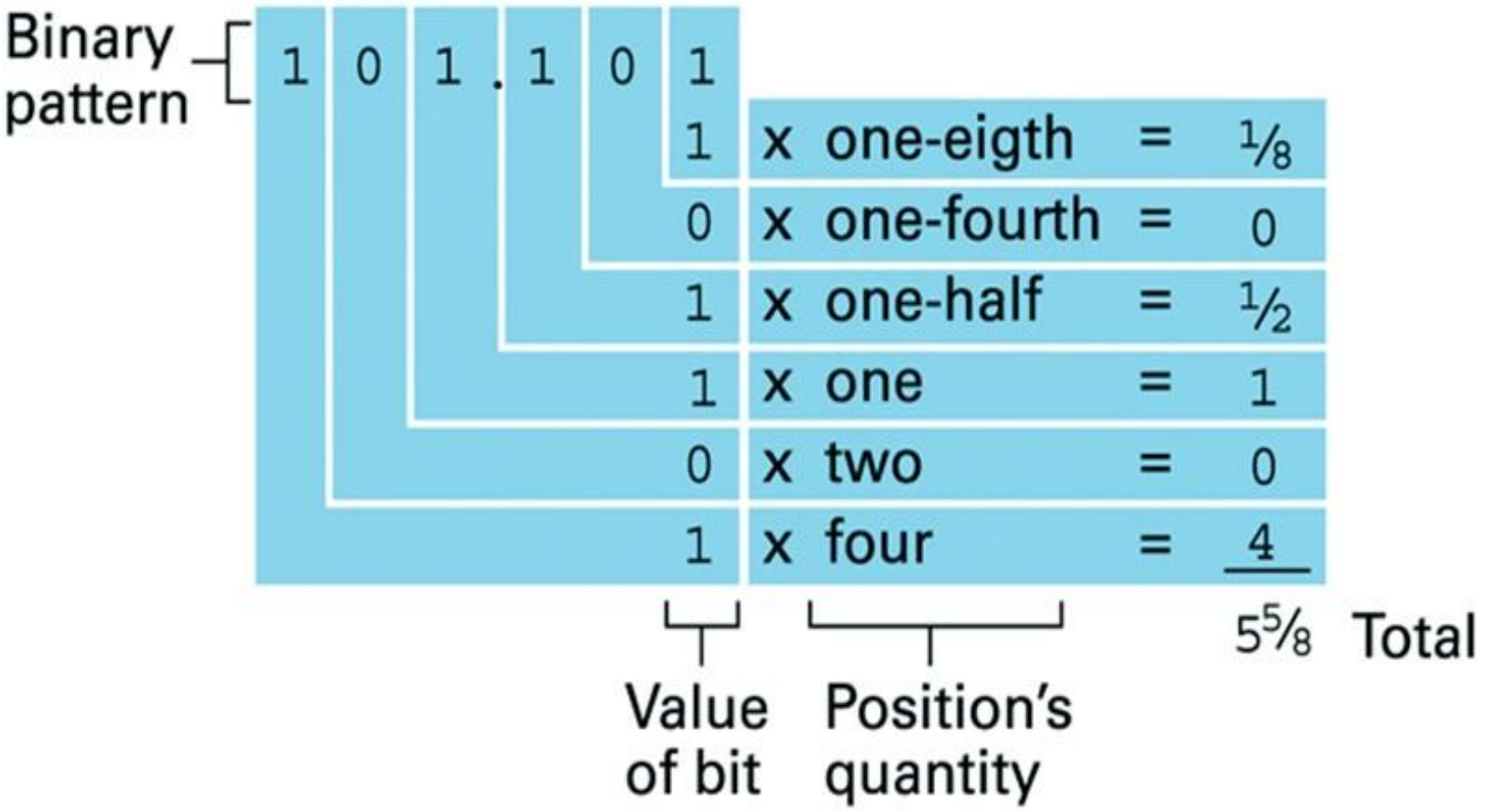
- Example:

```
  0 0 1 1 1 0 1 0
+ 0 0 0 1 1 0 1 1
-----------------
  0 1 0 1 0 1 0 1
```

(58 + 27 = 85)

1.5 Fractions in the Binary System

- Radix point has same role as in decimal system



Fraction Decimal to Fraction Binary Conversion

Convert .625 into Binary

Fraction Decimal to Fraction Binary Conversion: Convert .625 into Binary:

- In fact, there is a simple, step-by-step method for computing the binary expansion on the right-hand side of the point. We will illustrate the method by converting the decimal value .625 to a binary representation..
- **Step 1**: Begin with the decimal fraction and multiply by 2. The whole number part of the result is the first binary digit to the right of the point.
- Because **.625 x 2 = 1.25**, the first binary digit to the right of the point is a 1.

So far, we have **.625 = .1??? . . .**

Ref: http://cs.furman.edu/digitaldomain/more/ch6/dec_frac_to_bin.htm

Fraction Decimal to Fraction Binary Conversion: Convert .625 into Binary:

- **Step 2**: Next we disregard the whole number part of the previous result (the 1 in this case) and multiply by 2 once again. The whole number part of this new result is the *second* binary digit to the right of the point. We will continue this process until we get a zero as our decimal part or until we recognize an infinite repeating pattern.
- Because **.25 x 2 = 0.50**, the second binary digit to the right of the point is a 0.

So far, we have **.625 = .10?? . . .**

Fraction Decimal to Fraction Binary Conversion: Convert .625 into Binary:

- **Step 3**: Disregarding the whole number part of the previous result (this result was .50 so there actually is no whole number part to disregard in this case), we multiply by 2 once again. The whole number part of the result is now the next binary digit to the right of the point.
- Because **.50 x 2 = 1.00**, the third binary digit to the right of the point is a 1.

  So now we have **.625 = .101?? . . .**
- **Step 4**: In fact, we do not need a Step 4. We are finished in Step 3, because we had 0 as the fractional part of our result there. Hence the representation of **.625 = .101**

Ref: http://cs.furman.edu/digitaldomain/more/ch6/dec_frac_to_bin.htm

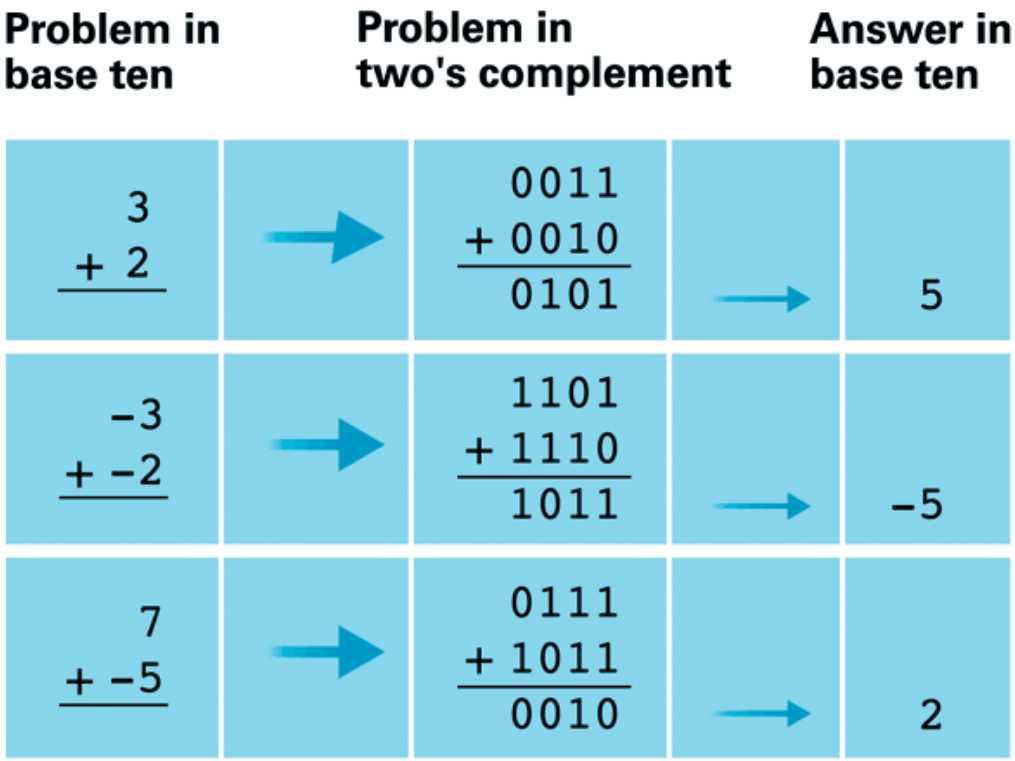
1.6 Storing Integers: Two's Complement Notation

- In general: values of 32 bits
- Includes negative numbers
- Leftmost bit indicates the sign
  - *sign bit*
- Note:
  - Positive and negative numbers are identical from right to left up to & including first '1'; from there on are complements of one another

b. Using patterns of length four

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

1.6 Addition in two's complement notation



- Note: no circuitry for subtraction needed!
- Note: ***overflow*** **errors: 0101 + 0100 = 1001 (5 + 4 = -7)**

Chapter 1: Problem 23

Here's a message in ASCII. What does it say?

01010111 01101000 01100001 01110100 00100000 01100100
01101111 01100101 01110011 00100000 01101001 01110100
00100000 01110011 00110001 01111001 00111111

- Each block of 8 bits represents one character:
  - See ASCII table in Appendix A
  - Example: 01010111 = ‘W’
  - Message says: ‘What does it s1y?’
  - Note: 00110001 = ‘1’, while 01100001 = ‘a’…

Chapter 1: Problem 28

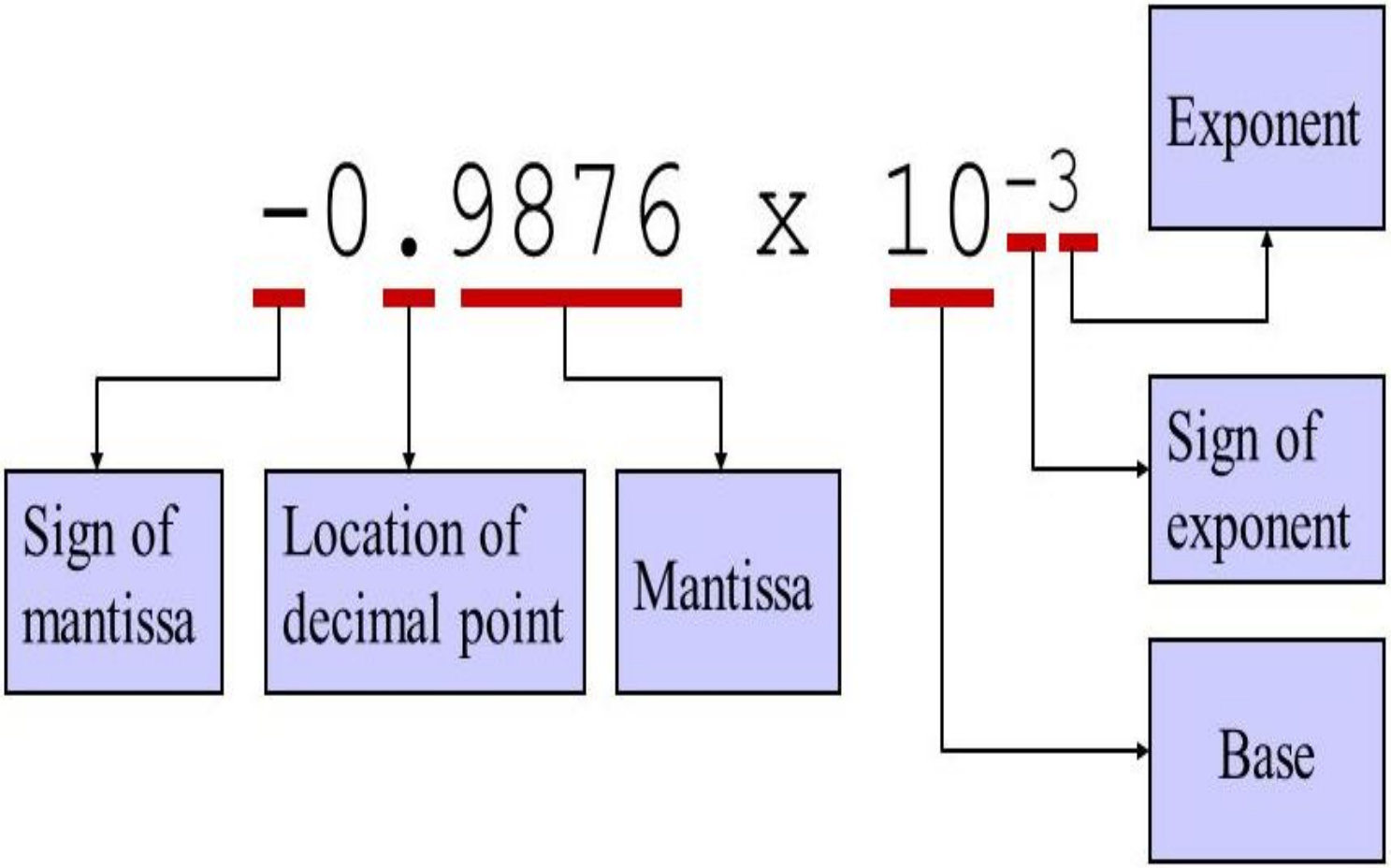
a. Write the number 14 by representing the 1 and 4 in ASCII.

b. Write the number 14 in binary representation.

- a. See ASCII Table in Appendix A:
  - 14 = 00110001 00110100

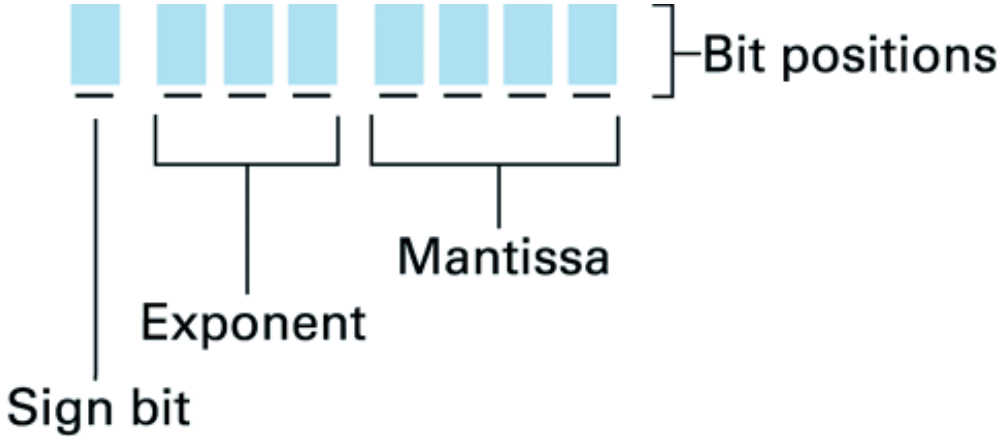
- b. In binary system each '1' represents a power of 2:
  - $14 = 8 + 4 + 2 = \mathbf{1}\times 2^3 + \mathbf{1}\times 2^2 + \mathbf{1}\times 2^1 + \mathbf{0}\times 2^0$ => binary: 1110

Parts of Floating-point Notation



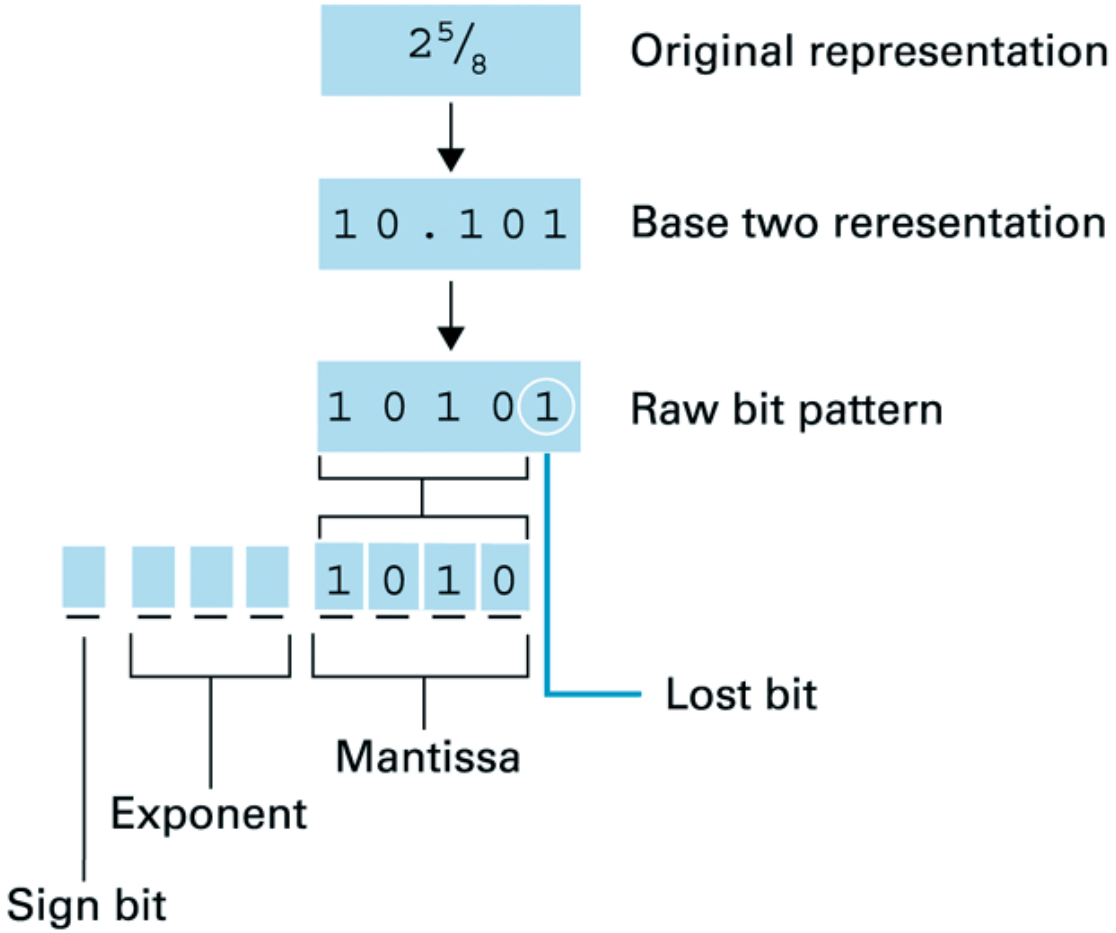
1.7 Storing Fractions: Floating-point Notation

- In contrast to integers, fractions require storage of the radix point
  - *Floating-point* notation



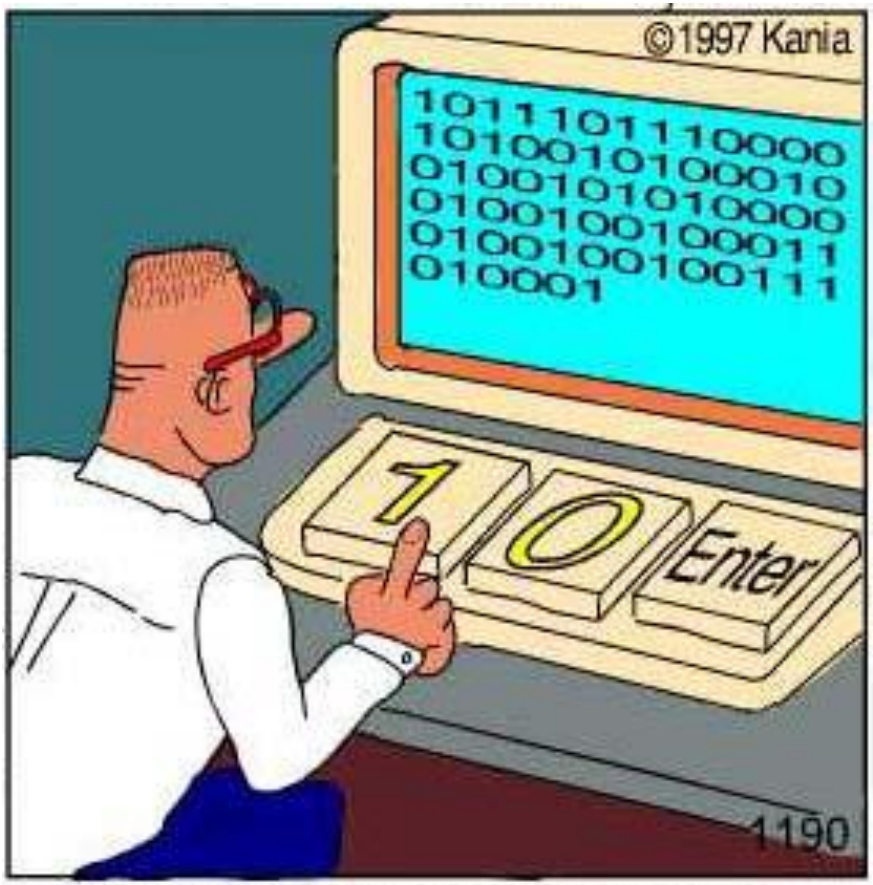
- **Example:** 1 110 1011

1.7 Truncation Errors: Coding the value 2 5/8



1.7 Truncation Errors (cont'd)

- Significance of truncation errors reduced by using larger mantissa & exponent fields (32bits)
- Problem of nonterminating expansion (e.g. 1/3)
  - worse in binary than in decimal system (e.g. 1/10)
- Interesting:
  - $2\ 1/2 + 1/8 + 1/8 = 2\ 1/2$
  - $1/8 + 1/8 + 2\ 1/2 = 2\ 3/4$
- When adding numbers, order may be important
  - rule: add smaller values first!



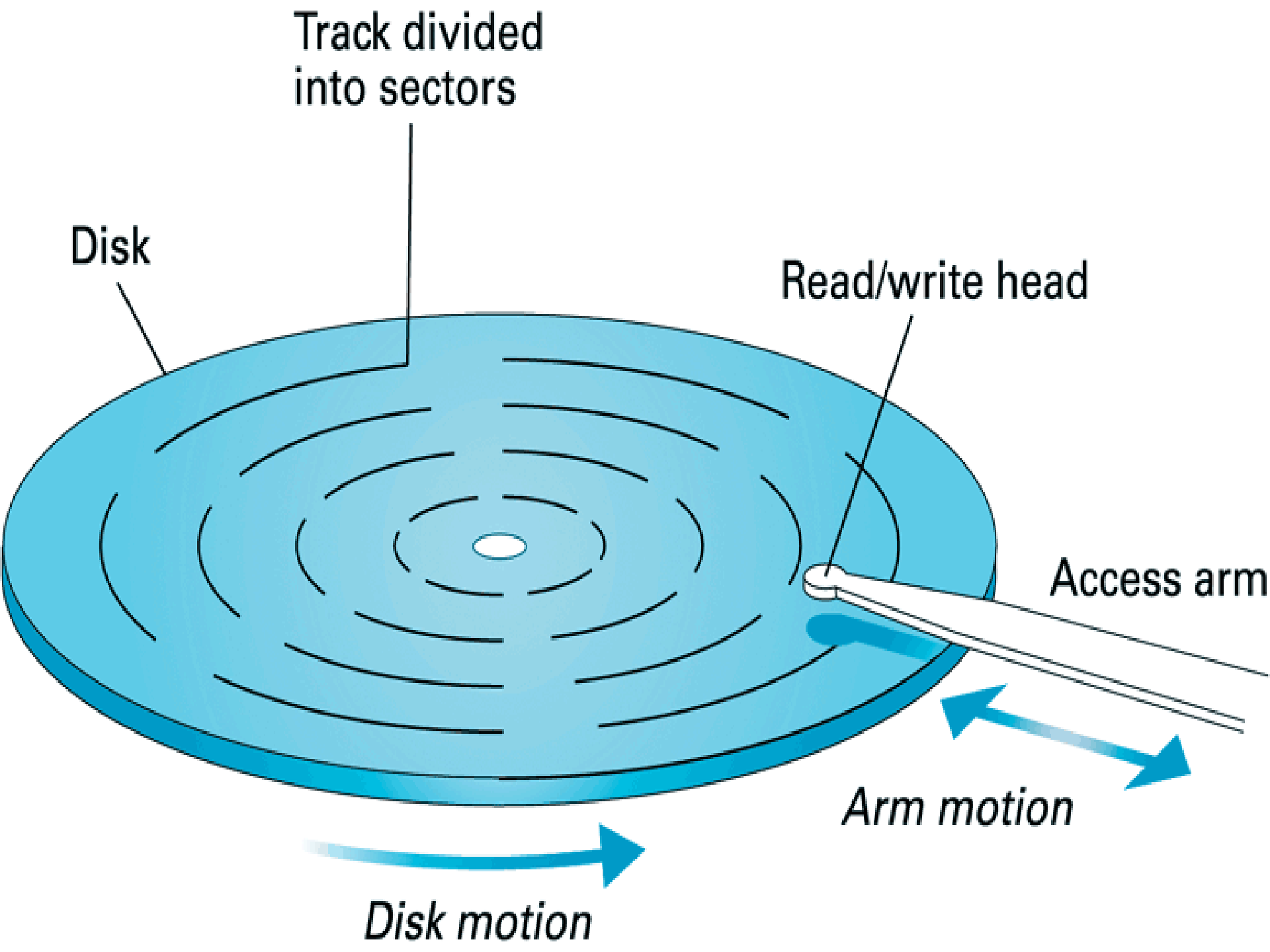
Real programmers code in binary.

Chapter 1: Conclusions

- Information stored as *streams of bits*
- Bit streams stored in main memory or on mass storage devices - each with different degree of random access (and thus: speed)
- Meaning of bit streams application dependent
- Standardized representations exist for (a.o):
  - text, numeric values, images, sounds, …
- For numeric values: overflow and truncation errors may make life difficult sometimes...

File Systems

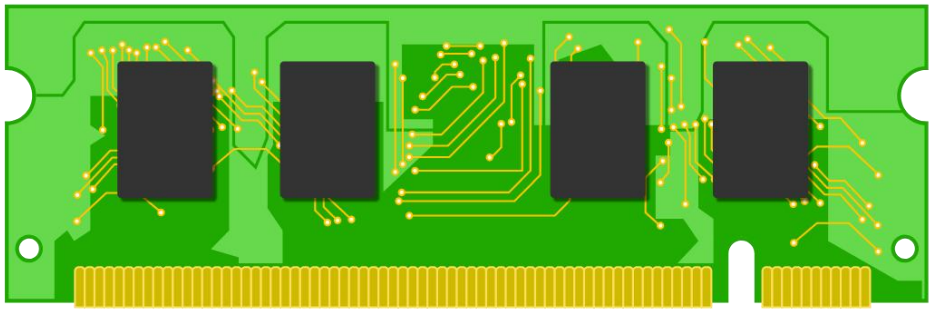
- **FAT 16, FAT 32 and NTFS**
  - **File Size = 9KB**
  - **FAT 16 cluster size = 4KB**
  - **FAT 32 cluster size = 2KB**
  - **NTFS cluster size = 1KB**
- **Services on hard disk**
  - **Fragmentation**
  - **Defragmentation**
  - **On Hard disk**


Track divided
into sectors
Disk
Read/write head
Access arm
Arm motion
Disk motion

Retrieving Data From A Computer

Prepared by Dr Syed Khaldoon Khurshid
Powered by Brilliant Application

- There are various devices that can store data on a computer. For example, hard disk drives **(HDD),** solid state drives **(SSD),** random access memory **(RAM)** modules, optical disks and drives, etc. We will go over key characteristics that are important from the software's point of view.




DVD

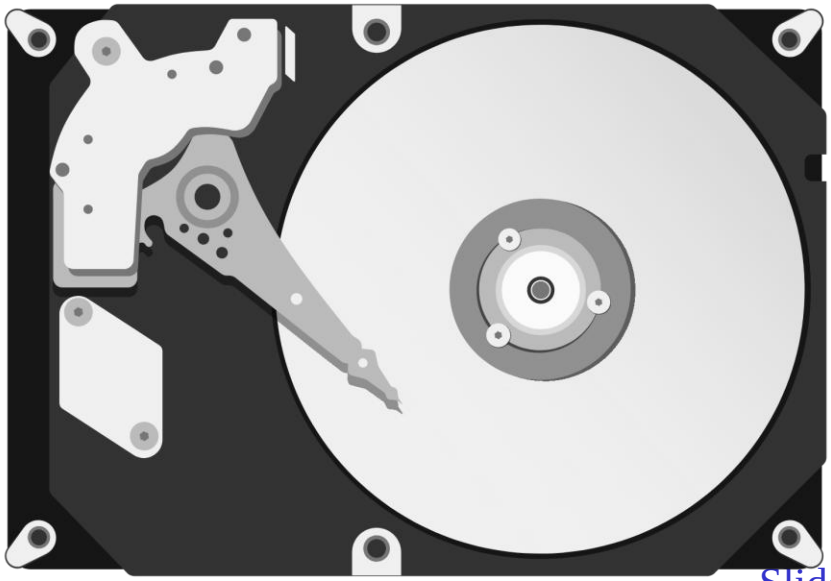
- CDs, DVDs, and Blu-rays are optical disks that can hold **(700MB–50 GB)** of data, and are often used to distribute music, movies, and game software. ROM disks such as **DVD-ROM** are read-only and the stored data cannot be changed or overwritten. Rewritable disks such as **DVD-RAM** disks are read-write and the stored data can be changed or overwritten.

- Assume that there is a video game software distributed on DVD-ROM disks. From the following, what is correct regarding the typical use of optical disks when distributing video game software?

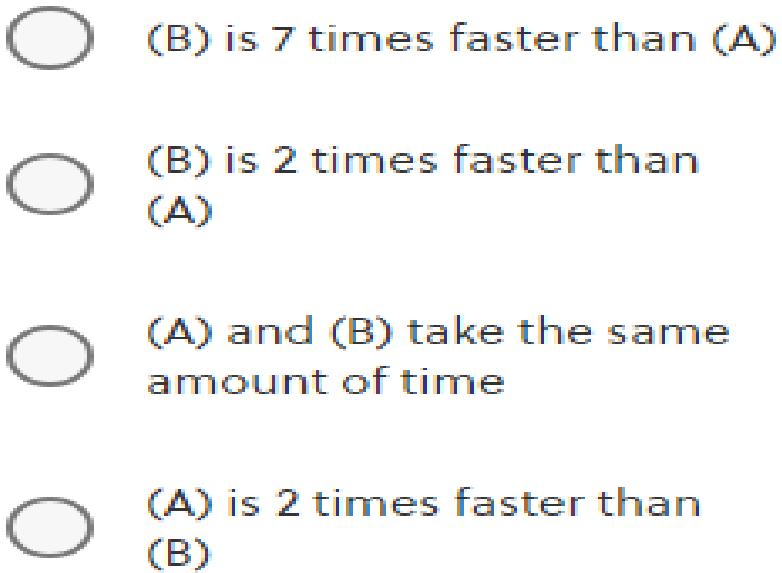
- ○ DVD-ROM disks have a high cost since they cannot be mass produced
- ○ DVD-ROM disks can be reused as blank media after finishing the game
- ○ Game results or play data can be saved on other writable media but not on the read-only DVD-ROM disk

- Correct answer: **Game results or play data can be saved on other writable media but not on the read-only DVD-ROM disk**
- The cost to produce optical disks is low considering the large amount of content that they can hold. Since DVD-ROM disks are read-only, the content stored on it cannot be changed. Game results or play data are often saved on other media such as memory cards, hard disk drives, etc.

- **HDDs are spinning magnetic disks** that can hold up to a few TB of data. They are often used as the main storage on computers to hold files and folders. The data access speed is fast for sequential accesses and becomes slower for random access. **Sequential access is reading or writing from a continuous chunk of data. Random access is reading or writing from different places on the disk.**



- **Given an HDD where the sequential access speed is 100 MB/s, which has an overhead of 10 ms when accessing a different place (random access overhead), and assuming that files are placed randomly within the drive and each file is stored continuously without any fragmentation, which of the following is correct regarding the speed of (A) and (B)?**
- (A) read 200 files, 50 KB each (10 MB total)
- (B) read 20 files, 500 KB each (10 MB total)



Correct answer: **(B) is 7 times faster than (A)**

$100\ \mathrm{MB/s}$ is $100000\ \mathrm{kB/s}$, so without accounting for random access time, the time to read a file of N kilobytes is $\frac{N}{100000}$. Random access time is $10\ \mathrm{ms}$, or $0.01\ \mathrm{s}$, so the total time to access a single file of size N is $\frac{N}{100000} + 0.01$.

For (A), accessing one 50 kB file including the random access overhead takes $0.01 + \frac{50}{100000} = 0.0105\ \mathrm{s}$.
200 files will be a total of $200 \times 0.0105 = 2.1\ \mathrm{s}$.

For (B), accessing one $500\ \mathrm{kB}$ file including the random access overhead takes $0.01 + \frac{500}{100000} = 0.015\ \mathrm{s}$.
20 files will be a total of $20 \times 0.015 = 0.3\ \mathrm{s}$.

Since $\frac{2.1}{0.3} = 7$, (B) is 7 times faster than (A).

- RAM is a semiconductor module that stores data within an integrated circuit that can hold up to a few GB of data. RAM is typically about 100 times faster than HDDs, and about 100 times more expensive for the same amount of data capacity. RAM does not have a significant speed decrease on random access. **RAM is a volatile memory which can only hold data when power is provided, and will lose data when power is removed.** Devices like HDDs and DVDs are non-volatile since they can hold data when power is removed.

- From the following, what is correct regarding volatility of devices?

We do not need to continuously provide power to RAM for it to hold data

The OS does not need to load data from HDD to RAM when starting the computer

We cannot expect data to remain on HDDs after turning the computer off

✓ We cannot expect data to remain on RAM after turning the computer off

Correct!

- Considering characteristics such as data capacity, access speed, volatility, read-only or read-write, the modern computer architecture uses HDDs or SSDs to store permanent data, and RAM for intermediate data**. Programs will load files from the HDD or SSD to RAM, use RAM to calculate, and store necessary data in files on the HDD or SSD.** RAM is accessed frequently from programs, and therefore it is called the “main memory”. In practical programming environments, when we use the word “memory”, we often mean RAM.

- Given RAM and HDD where the access speed is 10 GB/s and 100 MB/s respectively, we want to read a 3MB file 25 times. Ignoring random access overhead and the cost of writing to RAM, which answer is correct regarding the speed of (A) and (B)?
- (A) load the file from the HDD to RAM once, and read from RAM 25 times.
- (B) read directly from the HDD 25 times.

- (A) is 20 times faster than (B)
- (A) is 2 times faster than (B)
- (A) and (B) take the same amount of time
- (A) is slower than (B)

Explanation

Correct answer: (A) is 20 times faster than (B)

$10\ \mathrm{GB/s}$ is equivalent to $10000\ \mathrm{MB/s}$.

Loading a $3\ \mathrm{MB}$ file once from the $100\ \mathrm{MB/s}$ HDD takes $\frac{3}{100} = 0.03\ \mathrm{s}$, whereas
loading a $3\ \mathrm{MB}$ file once from the $10000\ \mathrm{MB/s}$ RAM takes $\frac{3}{10000} = 0.0003\ \mathrm{s}$.

(A) takes $1 \times 0.03 + 25 \times 0.0003 = 0.0375\ \mathrm{s}$, while
(B) takes $25 \times 0.03 = 0.75\ \mathrm{s}$.

$\frac{0.75}{0.0375} = 20$. Therefore, (A) is 20 times faster than (B).


Any
Questions