Data Representation

Lecture 2

A little Bit of Mathematics

- Since young, we been learn to write numbers as

  1 2 3 4 5 6 7 8 9 0

- The actual way to write them is

  1_{10} 2_{10} 3_{10} 4_{10} 5_{10} 6_{10} 7_{10} 8_{10} 9_{10} 0_{10}

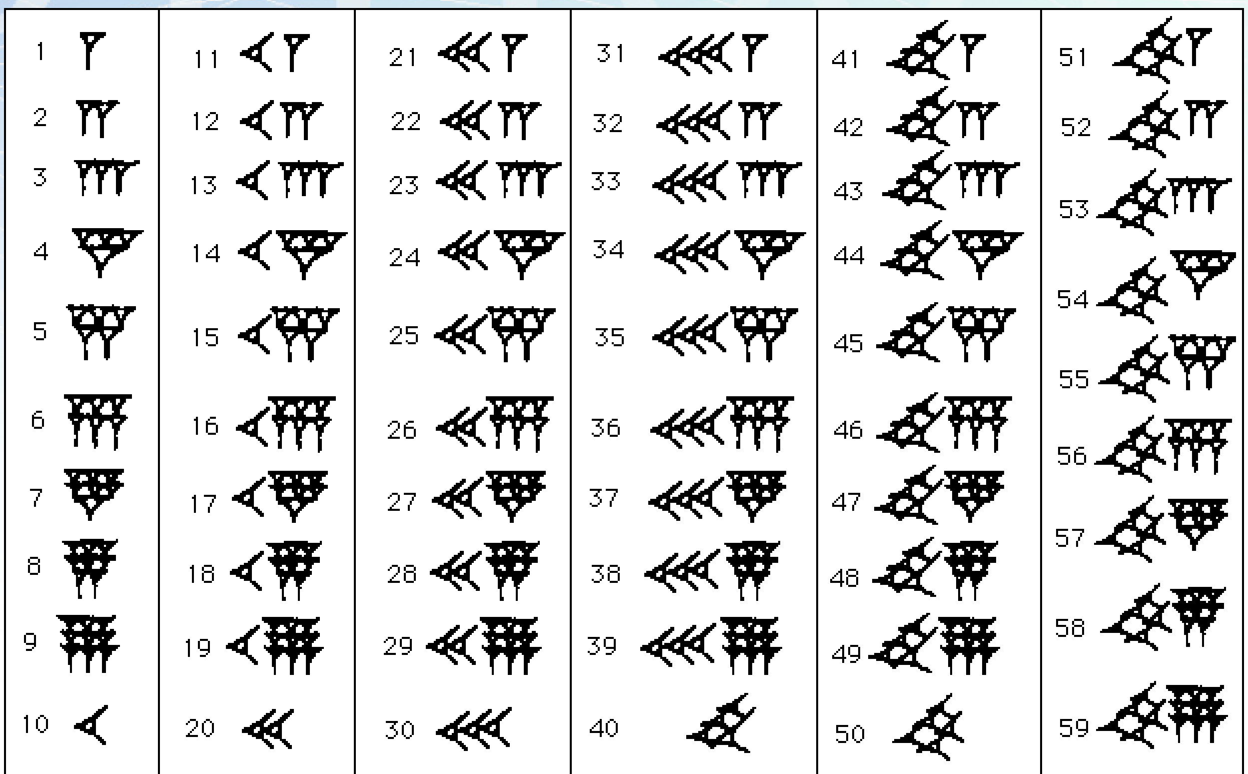
A little Bit of Mathematics

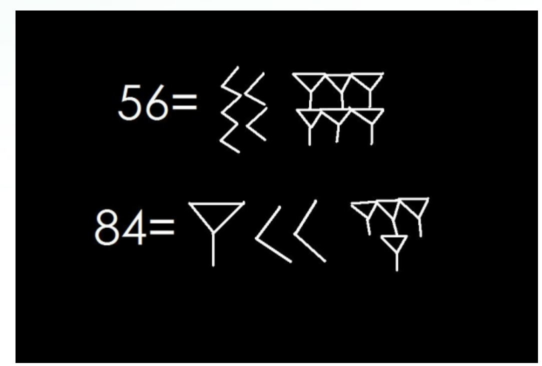
1_{10} 2_{10} 3_{10} 4_{10} 5_{10} 6_{10} 7_{10} 8_{10} 9_{10} 0_{10}

- The "10" is called the **base** or **radix**
- Likely motivated by counting with ten fingers
- We call the system of base 10, **decimal system**

A little Bit of Mathematics

- Other bases had been used in the past
- Babylonian system uses base **60**





Data Representation

- In Computers, we use 4 different base numbers for our work
  - Decimal System
  - Binary System
  - Octal System
  - Hexadecimal System

Decimal System

Lecture 3

Data Representation

Decimal System

- Also called **base 10** and **Denary**
- Numerical base most widely used by modern civilization
- **Positional notation system** for representing numbers
  - Representation consists of a string of digits

Decimal System

– Each digit increases in numeric weight by **10** from right to left

– Example

Decimal Number 327, positional notation is:

$$327 = \underbrace{(3x10^2)}_{\text{Hundred (100)}} + \underbrace{(2x10^1)}_{\text{ten (10)}} + \underbrace{(7x10^0)}_{\text{unit (1)}}$$

weight value for the right-most digit

Decimal System

- What is the positional notation of decimal number 1234?

Positional Notation is

$$1234 = 1 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0$$

Binary System

Lecture 3

Data Representation

Binary System

- How is data stored on hard disk?
- Data in computers are represented by **0 and 1**
- Hence the name **binary**, base **2**

Binary System

- Each binary digit is called a **bit**
  - **Bi**nary Digi**t**
- Byte is a string (i.e. a set / group) of **8 bits**

- In mathematics, using International System of Units (SI)
  - 1000 = kilo, k
  - 1 kb = ?

Binary System

- 1kb = 1000 bytes by definition using SI units
- However in Computer Science, we are always based on multiple of 2
  - Hence 1kb = 1024 bytes (2^{10})

- But for storage devices, most hardware manufacturers use 1kb = 100 bytes.
  - That's why they appear smaller in size when we use them

Binary System

- What is the positional notation of binary number 101?

- Notice the difference between decimal
  - 10 is replaced by 2

Binary Number 101 positional notation is:

$$101 = \underbrace{(1x2^2)}_{(4)} + \underbrace{(0x2^1)}_{(2)} + \underbrace{(1x2^0)}_{(1)}$$

(1) ← weight value

MSB & LSB

- About MSB & LSB
- Stands for Most Significant Bit & Least Significant Bit
  - A way to tell which is front which is back
  - Important for a computer to know this
  - Leads to Endianness

MSB LSB

10001010101

Endian

- The ordering of bytes

11010100	01100101	or	01100101	11010100

Could be easily misinterpreted

- 2 key forms : Big-Endian & Little-Endian
- Will be elaborated on later in the course. For now, just keep this in mind
- Know that today's computers are generally either big or little endian

Hexadecimal System

Lecture 3

Data Representation

Hexadecimal System

- Radix of **16**
- Widely used for computer systems
  - Useful and convenient external representation for internal contents

Hexadecimal System

- What happens after F?
  - $16_{10} = 10_{16}$

Hexadecimal System

Hex	Dec	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011

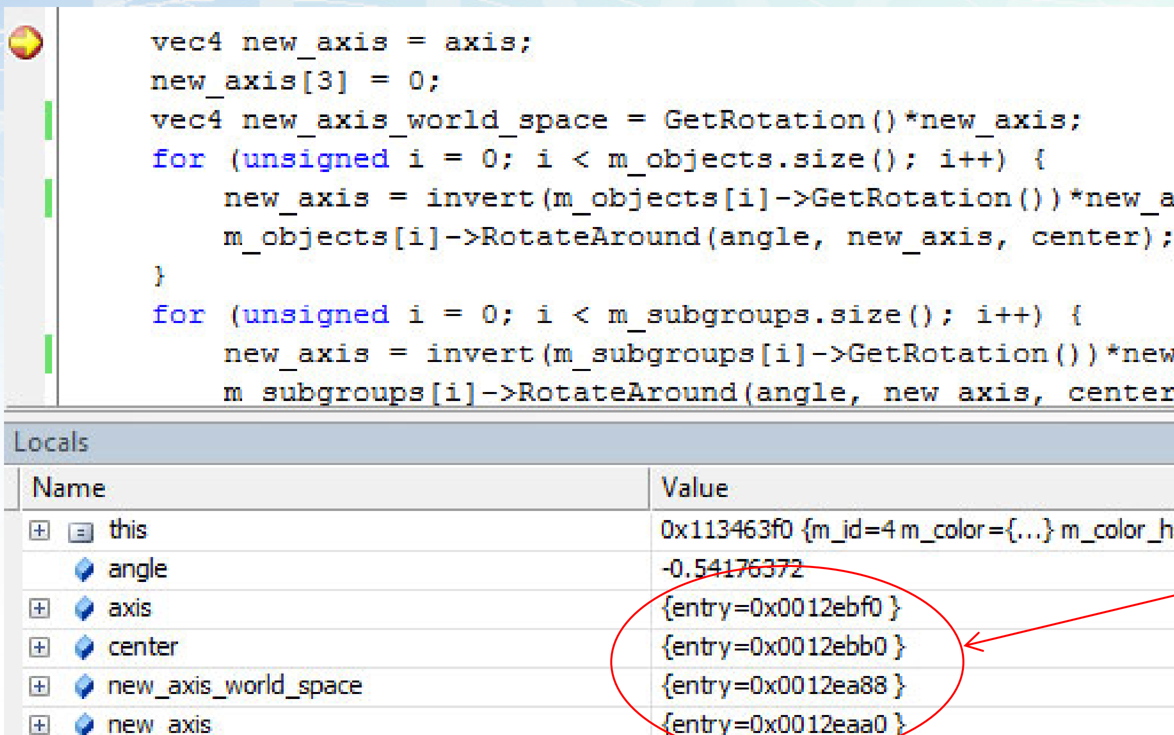
Hex	Dec	Binary
4	4	0100
5	5	0101
6	6	0110
7	7	0111

Hex	Dec	Binary
8	8	1000
9	9	1001
A	10	1010
B	11	1011

Hex	Dec	Binary
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Usage of Hexadecimal

- Understanding Hexadecimal is very useful in Visual Studio

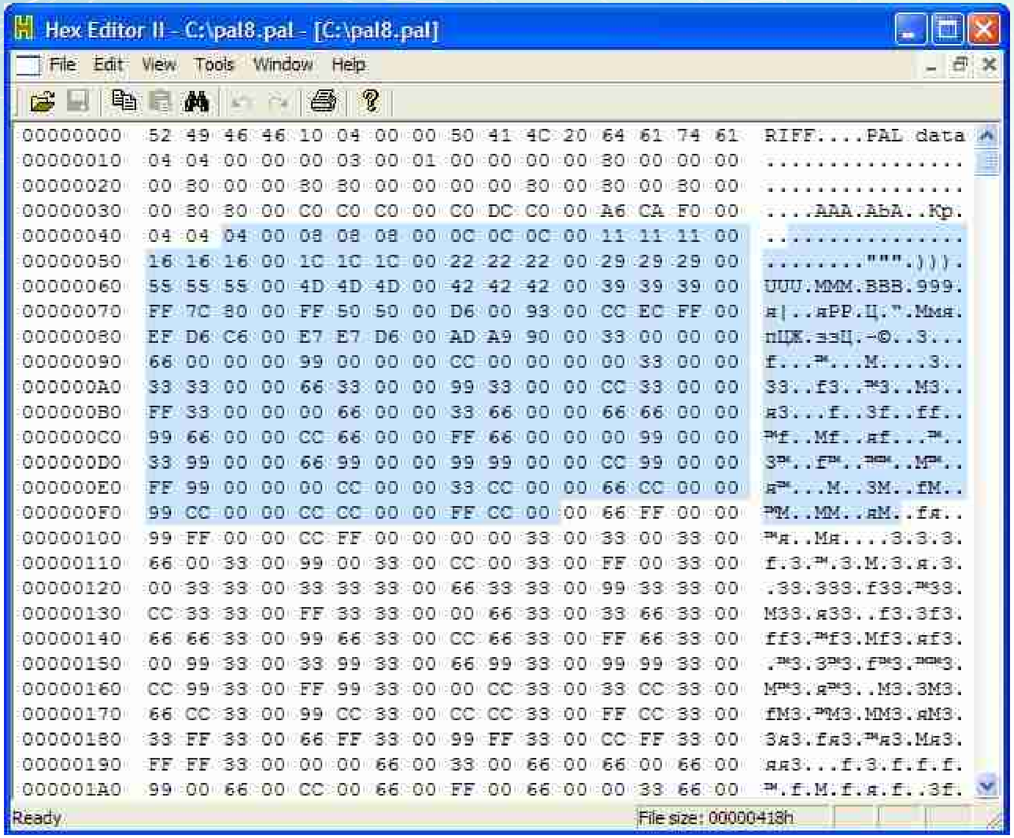


Many values listed are in hex (numbers starting with 0x)

Why Hexadecimal is Important for You

- In old school days, this is how we ‘edit’ savegames to cheat ☺

- Again, using hexadecimals!



Octal Numeral System

Lecture 3

Data Representation

Octal

- Radix of **8**
- Used most often in Unix for file permission
  - read up on chmod command in Unix
- Advantage over hexadecimal is that it does not require extra symbols as digits
- Originally widely used in computing for digital displays
  - Binary display were too complex
  - Decimal requires complex hardware to convert radices
  - Hexadecimal requires more numerals

Prefix

- In C Programming
  - Octal Numeral System
    - 0o
  - Hexadecimal
    - 0x

Conversion of Bases

Lecture 3

Data Representation

Conversion of Bases

- First Step:
  - **Convert the number to decimal first**
  - **It's hard to visualize numbers in other bases due to our past education and normal usage**

Formula

- To convert any number from the base **n**,
  say $M_{k-1}M_{k-2}M_{k-3}\dots\ M_{0(n)}$
  (e.g binary **1111** where there are 4 characters in the string hence

  k = 4 and n = 2

- Decimal = $M_{k-1} \times n^{k-1} + M_{k-2} \times n^{k-2} + \dots + M_1 \times n^1 + M_0 \times n^0$

  (e.g $1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 15$)

Conversion of Bases

- Second Step:
  - **Divide the number by the base you wish to convert to**
  - **The remainder will be the value to put at the left side of the answer**

- **Repeat 2nd step till the result from division is 0.**

Conversion of Bases

- Example: Convert 2012 to hex

Step	Num	Result	Remainder	Hex
0	**2012**	**125**	**12**	**C**
1	**125**	**7**	**13**	**DC**
2	**7**	**0**	**7**	**7DC**

Hence result is $\mathbf{7DC_{16}}$

Conversion of Bases

- Let's convert it back to decimal

$$
\begin{aligned}
7DC_{16} &= 7_{16} \times 16_{10}^{2} + D_{16} \times 16_{10}^{1} + C_{16} \times 16_{10}^{0} \\
&= 7_{10} \times 16_{10}^{2} + 13_{10} \times 16_{10}^{1} + 12_{10} \times 16_{10}^{0} \\
&= 7_{10} \times 256_{10} + 13_{10} \times 16_{10} + 12_{10} \times 1_{10} \\
&= 1792_{10} + 208_{10} + 12_{10} \\
&= 2012_{10}
\end{aligned}
$$

Conversion of Bases

- Shortcut!
- For our industry, binary and hexadecimal is very common
- We can do direct conversion easily without much effort
  - Because $16 = 2^4$

Conversion of Bases

- We can think of
  - 4 bits convert to 1 hex digit

- For example

00101011 = 2B

Character Representation

Lecture 3
Data Representation

Character Representation

- A typical 'string' of characters expressed in a form usable by a computer

h	e	l	l	o	!	\0

- Computer doesn't store each character as itself

  i.e. h = h (This is meaningless to a computer)

- Instead a table is used to map the meaning of character to a number that computer understands

  i.e. ASCII Table

ASCII

Dec	Hx	Oct	Char	
0	0	000	NUL	(null)
1	1	001	SOH	(start of heading)
2	2	002	STX	(start of text)
3	3	003	ETX	(end of text)
4	4	004	EOT	(end of transmission)
5	5	005	ENQ	(enquiry)
6	6	006	ACK	(acknowledge)
7	7	007	BEL	(bell)
8	8	010	BS	(backspace)
9	9	011	TAB	(horizontal tab)
10	A	012	LF	(NL line feed, new line)
11	B	013	VT	(vertical tab)
12	C	014	FF	(NP form feed, new page)
13	D	015	CR	(carriage return)
14	E	016	SO	(shift out)
15	F	017	SI	(shift in)
16	10	020	DLE	(data link escape)
17	11	021	DC1	(device control 1)
18	12	022	DC2	(device control 2)
19	13	023	DC3	(device control 3)
20	14	024	DC4	(device control 4)
21	15	025	NAK	(negative acknowledge)
22	16	026	SYN	(synchronous idle)
23	17	027	ETB	(end of trans. block)
24	18	030	CAN	(cancel)
25	19	031	EM	(end of medium)
26	1A	032	SUB	(substitute)
27	1B	033	ESC	(escape)
28	1C	034	FS	(file separator)
29	1D	035	GS	(group separator)
30	1E	036	RS	(record separator)
31	1F	037	US	(unit separator)

Dec	Hx	Oct	Html	Chr
32	20	040		Space
33	21	041	!	!
34	22	042	"	"
35	23	043	#	#
36	24	044	$	$
37	25	045	%	%
38	26	046	&	&
39	27	047	'	'
40	28	050	(	(
41	29	051	)	)
42	2A	052	*	*
43	2B	053	+	+
44	2C	054	,	,
45	2D	055	-	-
46	2E	056	.	.
47	2F	057	/	/
48	30	060	0	0
49	31	061	1	1
50	32	062	2	2
51	33	063	3	3
52	34	064	4	4
53	35	065	5	5
54	36	066	6	6
55	37	067	7	7
56	38	070	8	8
57	39	071	9	9
58	3A	072	:	:
59	3B	073	;	;
60	3C	074	<	<
61	3D	075	=	=
62	3E	076	>	>
63	3F	077	?	?

Dec	Hx	Oct	Html	Chr
64	40	100	@	@
65	41	101	A	A
66	42	102	B	B
67	43	103	C	C
68	44	104	D	D
69	45	105	E	E
70	46	106	F	F
71	47	107	G	G
72	48	110	H	H
73	49	111	I	I
74	4A	112	J	J
75	4B	113	K	K
76	4C	114	L	L
77	4D	115	M	M
78	4E	116	N	N
79	4F	117	O	O
80	50	120	P	P
81	51	121	Q	Q
82	52	122	R	R
83	53	123	S	S
84	54	124	T	T
85	55	125	U	U
86	56	126	V	V
87	57	127	W	W
88	58	130	X	X
89	59	131	Y	Y
90	5A	132	Z	Z
91	5B	133	[	[
92	5C	134	\	\
93	5D	135	]	]
94	5E	136	^	^
95	5F	137	_	_

Dec	Hx	Oct	Html	Chr
96	60	140	`	`
97	61	141	a	a
98	62	142	b	b
99	63	143	c	c
100	64	144	d	d
101	65	145	e	e
102	66	146	f	f
103	67	147	g	g
104	68	150	h	h
105	69	151	i	i
106	6A	152	j	j
107	6B	153	k	k
108	6C	154	l	l
109	6D	155	m	m
110	6E	156	n	n
111	6F	157	o	o
112	70	160	p	p
113	71	161	q	q
114	72	162	r	r
115	73	163	s	s
116	74	164	t	t
117	75	165	u	u
118	76	166	v	v
119	77	167	w	w
120	78	170	x	x
121	79	171	y	y
122	7A	172	z	z
123	7B	173	{	{
124	7C	174			|
125	7D	175	}	}
126	7E	176	~	~
127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Character Set / Character Encoding

- The ASCII Table is a good example of a character set
- Each character mapped to a number
- Number can be expressed as decimal, binary, hexadecimal
- Limitations:
  - ASCII limited to 128 characters
  - Can't express complex character languages on computers e.g.日本語

UNICODE

- Created to support a wider set of character representation
- Uses multiple bytes to describe a character
  - Versus ASCII which uses 1 byte per character
- A number of different encodings exist
  - UTF-1, UTF-7, UTF-8, UTF-EBCDIC, UTF-16, UTF-32

Most commonly used today

UTF-8

- Uses 1 to 4 bytes to represent a character
- Variable (can be 1-4 bytes)
  - Uses encoding to store in a variable way
- ASCII-compatible

UTF-8

Unicode	Byte-1	Byte-2	Byte-3	Byte-4	Example
U+0000 – U+007F	**0**xxxxxxx				U+0067 (103) Binary : 0110 0111 Encoded: 01100111
U+0080 – U+07FF	**110**yyyxx	**10**xxxxxx			U+05AA (1450) Binary : 0101 1010 1010 Encoded: [**110**10110] [**10**101010]
U+0800 – U+FFFF	**1110**yyyy	**10**yyyyxx	**10**xxxxxx		U+C123 (49443) Binary : 1100 00010010 0011 Encoded : [**1110**1100] [**10**000100] [**10**100011]
U+10000 – U+10FFFF	**11110**zzz	**10**zzyyyy	**10**yyyyxx	**10**xxxxxx	U+0D1B34 (858932) 0 1101 0001 1011 0011 0100 Encoded: [**11110**011] [**10**010001] [**10**101100] [**10**110100]

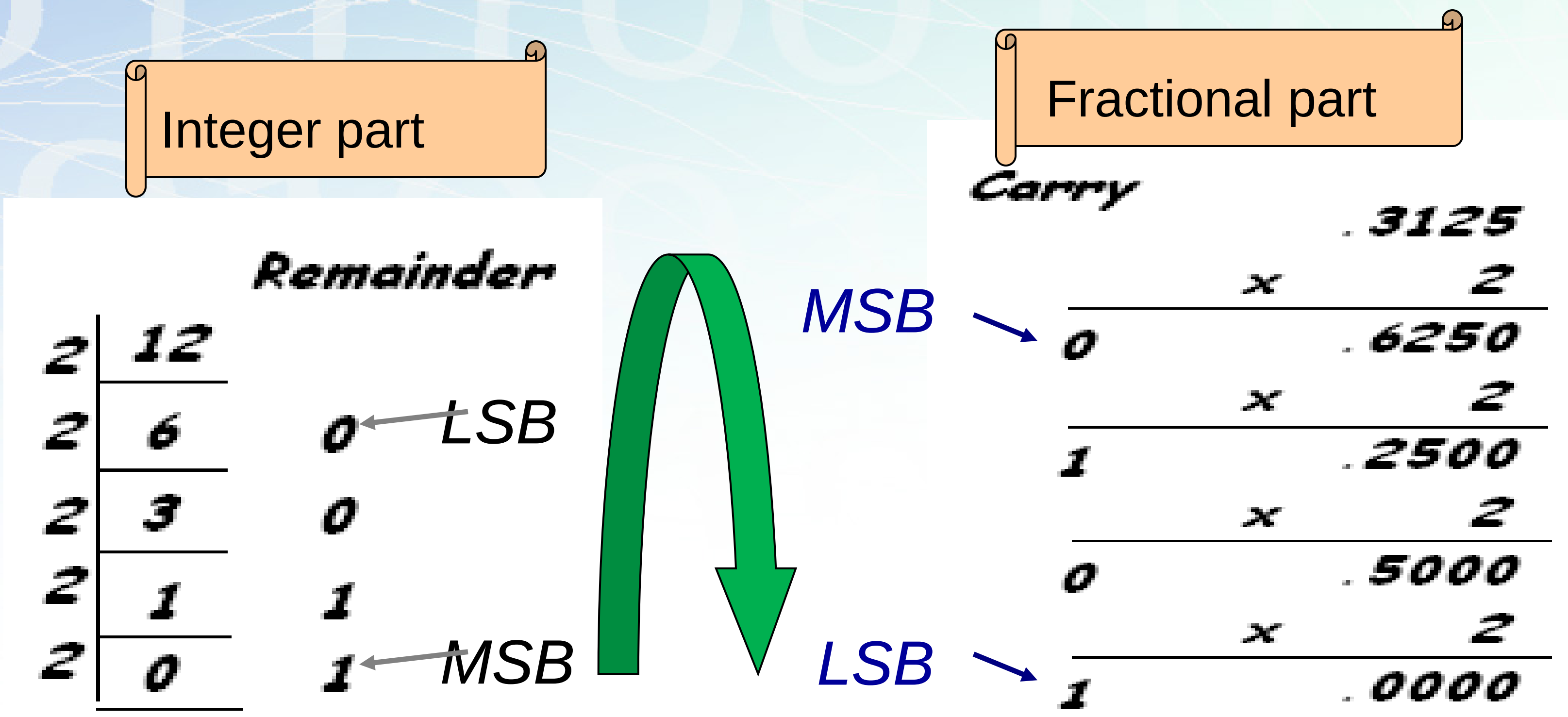
UCS

- Universal Character Set
- aka ISO 10646
- An international standard to encompass a truly universal set of characters
- However, UCS != Unicode
  - Some inherent differences
  - E.g. Unicode includes additional rules & specifications
  - For details, do some reading on your own

Conversions involving decimals

(i.e. non-whole number)

- Decimal to Binary
  - ➢Decimal: 12.3125 1100.0101

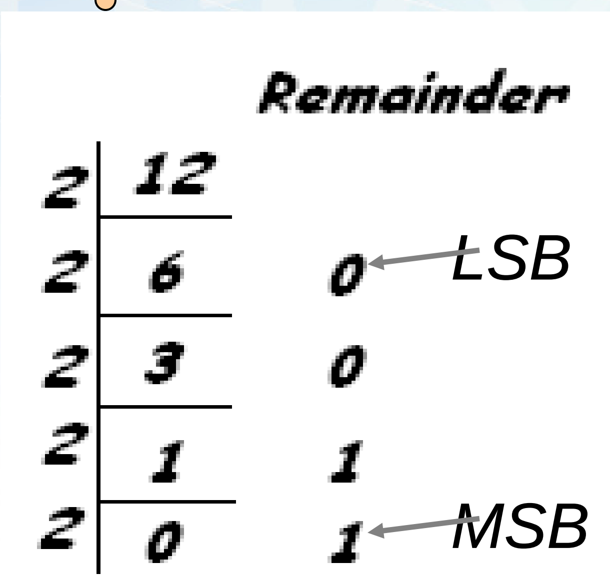


What if .0000 can't be achieved?
(endless remainder)

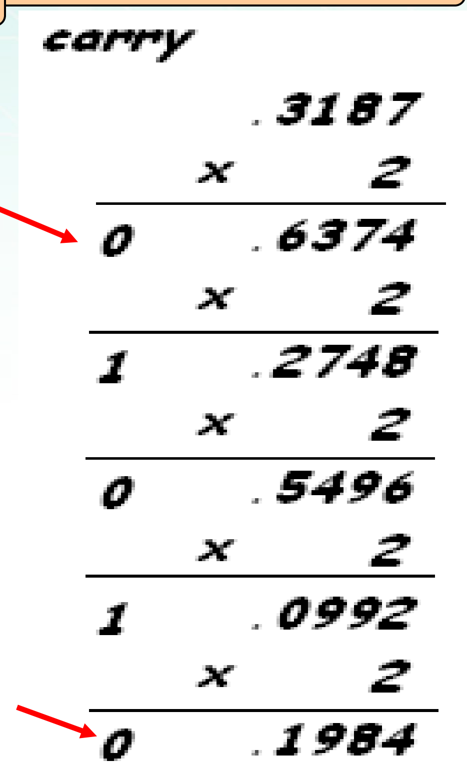
- Decimal to Binary
  - Decimal: 12.3187 ⇨ 1100.01010…

Stop at 5 numbers after the decimal point (unless question states otherwise)

Integer part



Fractional part



Q&A

Lecture 3

Data Representation

END

Lecture 3

Data Representation