Today

Data Representation

Decimal

Binary.

Hexadecimal / Octal

Why?

0X500 / 0x500

Digits → 0, 1, 2, ... 9, A, B, C, D, E, F

need not be uppercase.

$$\underbrace{1110}_{E}\ \underbrace{1111}_{F}{}_{2}$$

$$EF_{16}$$

Less work for the interpretor.

Octal groups 3 bits together

Representation Numbers

1) Unsigned encoding (≥ 0) Non-negative numbers
2) 2's complement encoding (signed numbers)
3) floating point encoding (real numbers)

Data Sizes

short a; (or) short int a;

C type	32-bit	64-bit
char	1	1
short [int]	2	2
int	4	4
long [int]	4	8
long long [int]	8	8
float	4	4
double	8	8
pointer	4	8

(In bytes)

Why 4 byte and 8 byte units for pointers?

These are the 'word sizes' of these processors.

Word Size → The size of each of the processors registers.

32-bit processor

0000 ... 32 0s to 1111 32 1s

0 to 2^32 - 1

4 GB.

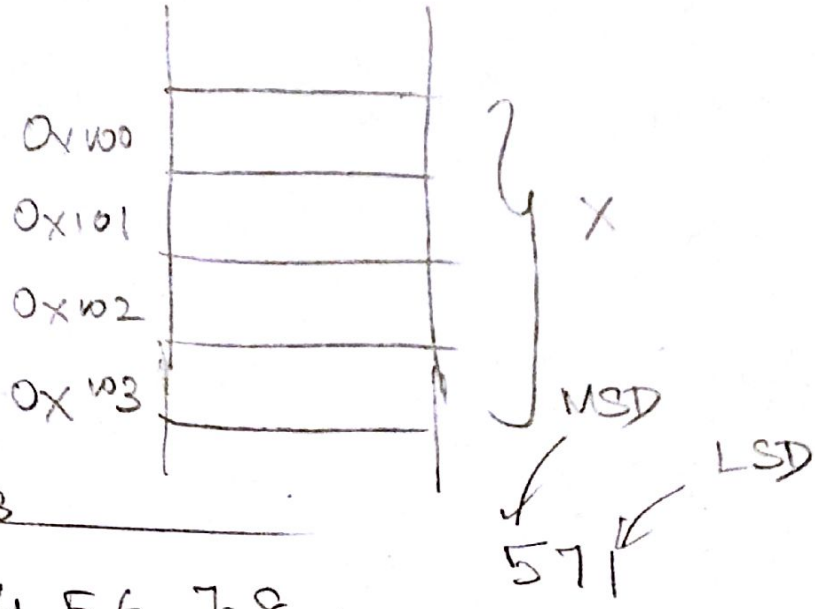
64 bit processor

0 to 2^64 - 1

Addressing & Byte Ordering

```
int x;

&x = 0x100;
```



Endian-ness

Little Endian

0x100	78
0x101	56
0x102	34
0x103	12

Least Significant Byte gets stored first.
(Most Intel machines)

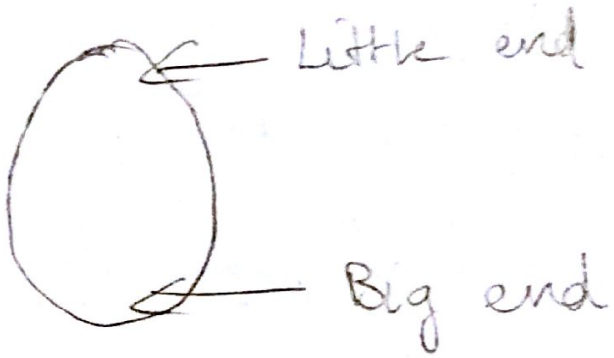
Big Endian

0x100	12
0x101	34
0x102	56
0x103	78

Most Significant byte comes first
(Most IBM machines)

Bi-endian - machines that can be configed to either.

Gulliver's Travels



Problem

① 

Big E — Big Endian (Network byte order) — Little E

htons → host to network short
ntohs → network to host short
htonl → " long
ntohl → " long.

(2) Machine C

64 94 04 08

↓

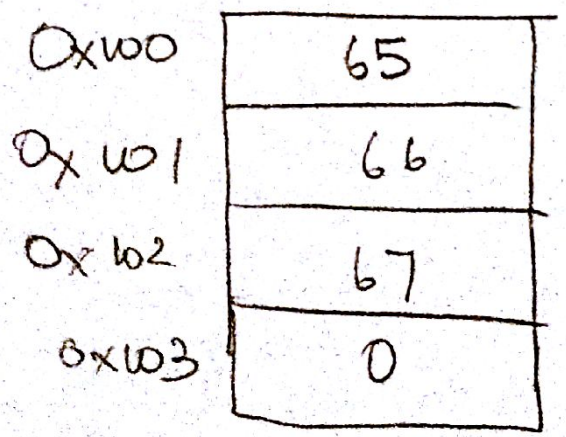
0x 08 04 94 64

↓

0x 8 0 4 9 4 64

Strings in C (char arrays)

```
char *str = "ABC";
```



This is not a group They are individual characters.

0 - 127

There are other formats Unicode (UTF 16)

Bitwise Operators

Bit Vector → String of bits (0s and 1s)
Array

→ Work at a bit level

$X = 12_{10} = 1100_2$

$y = 10_{10} = 1010_2$

AND

x & y

```
  1 1 0 0
  1 0 1 0
---------
  1 0 0 0   → 8
---------
```

Logical operator (x && y).

x = 0
y = 10
x && y
= 0

OR

x | y

```
  1 1 0 0
  1 0 1 0
---------
  1 1 1 0   → 14
---------
```

XOR

x ^ y

```
  1 1 0 0
  1 0 1 0
---------
  0 1 1 0   → 6
---------
```

NOT

~x 1100

0011 → 3

Logical NOT

!x (12)

= 0

x = 0

!x → 1

Shift operators

Left Shift operator <<

x = 00001100 → 12

x << 1 00011000 → 24

x << 2 00110000 → 48

Right Shift operator >>

x = 00001100 → 12

x >> 2 00000011 → 3

Right shift is a bit operation

	①	②
	0110 0011	1001 0101
x >> 4 (logical)	0000 0110	0000 1001
	unsigned data	
x >> 4 (arithmetic)	0000 0110	1111 1001
		(signed data)

Integer Representation

I Unsigned encoding

$$B2U_w(\vec{x}) = \sum_{i=0}^{w-1} x_i \cdot 2^i$$

Maps strings of 0s and 1s to non-negative integers.

$$B2U_4([0001]) = 0\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0 = 1$$

(3, 2, 1, 0)

$B2U_4\left([1010]\right) = 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$

$= 8 + 0 + 2 + 0$

$= 10$

1111 $\longrightarrow$ 15

For a vector of length W, the value ranges from 0 to $2^W - 1$

0 to $2^4 - 1$ $(= 15)$.