Objectives - Thu 3/7/2019

- Endiannness
- x/
- Bit operations

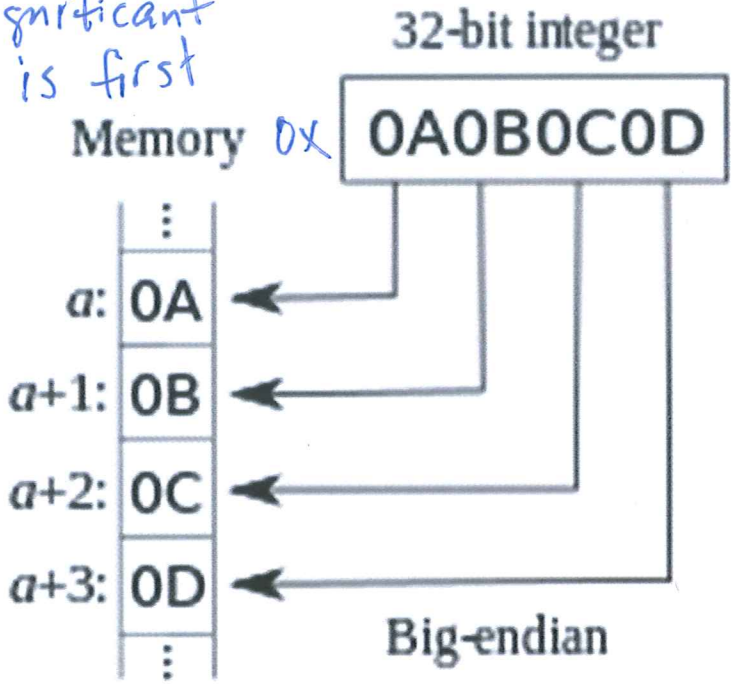
x/ units

	# of bytes	# of bits	# of hex digits
bit	1/8	1	
nibble (1 hex digits)	1/2	4	1
byte	1	8	2
half-word	2	16	4
word (4 bytes)	4	32	8
giant	8	64	16

8888

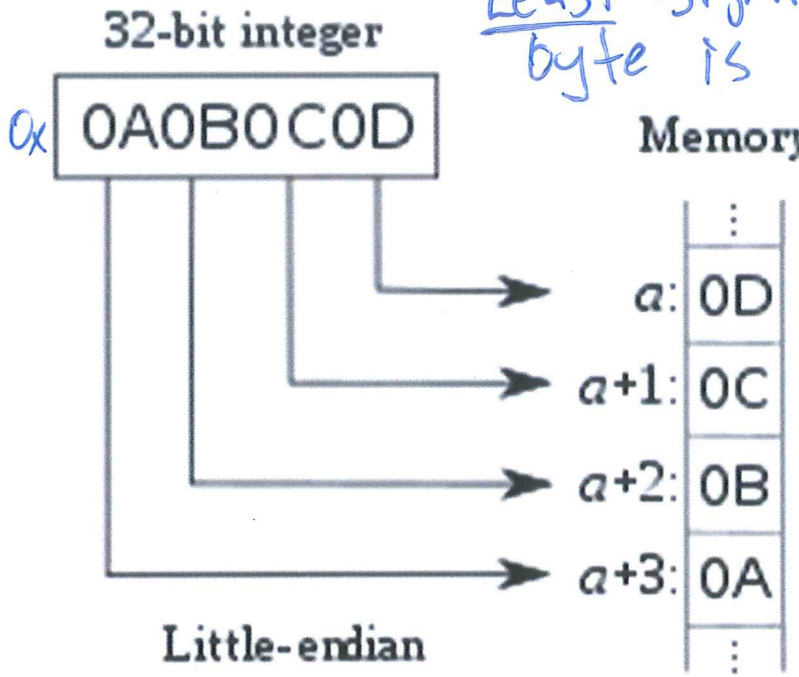
Big endian

Most significant byte is first



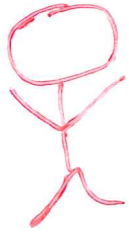
Little endian

Least significant byte is first



OpenRISC

IBM z/ series



humans (us)

ecegrid

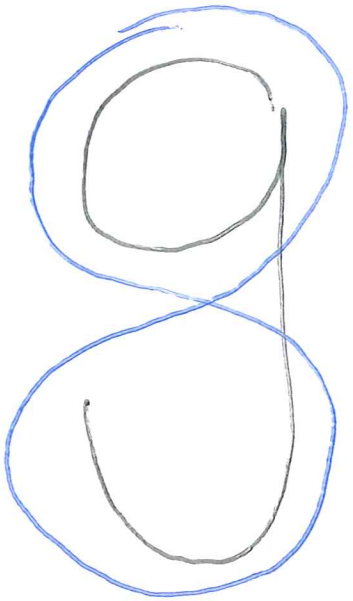
Your PC at home

Your phone

Your watch

Credit: https://en.wikipedia.org/wiki/Endianness

YOUR COMPUTER IS WIERD



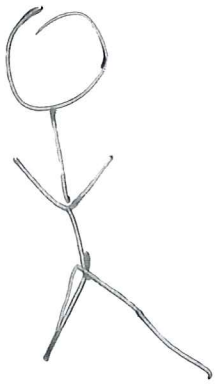
iant

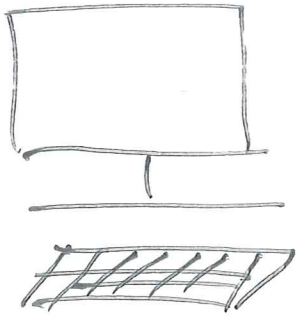
ggg8

Why

"sixty-five"

('A' == 0x41)





Unsigned int (4 byte o.o.p.)

0x	41	00	00	00

(hex)

00065

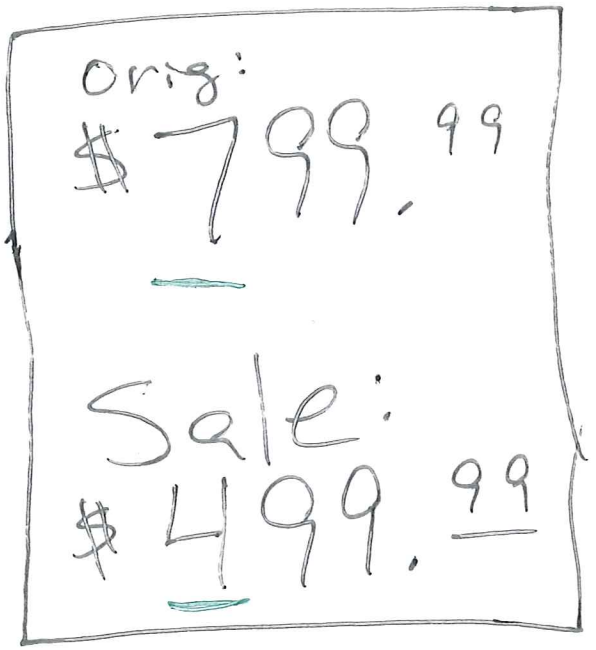
write only what we need

most significant byte

least significant digit

0x0a0b0c0d

1684496141



Little endian affects only within a single number (eg. int, address)

Arrays + struct objects are in the order in your code - not "strange"

```
struct Point {
    int x;   <- first in memory
    int y;
};
```

What gets "flipped"? (by little endian)

flipped (bytes)

multi byte numbers:

- int
- long
- address
- float

not flipped

- hex digits within a byte
- array elements
- struct fields (in order given in type definition)
- bits (within a byte)

As a C literal (human-readable)

0x 0a 0b 0c 0d

most significant byte

least significant byte