LinkedIn: *https://linkedin.com/in/mrunalnshah*
GitHub: *https://github.com/mrunalnshah*
**Email : ** *mrunalnshah2883@gmail.com*

Myself, Mrunal.

Mrunal Nirajkumar Shah

bit

Building Block of Computers

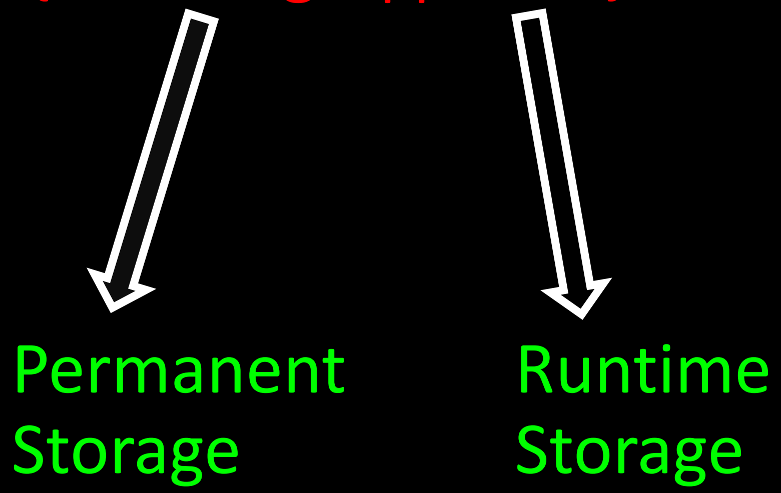
Random Access Memory

We use RAM to store runtime data.

Data Structure

What is Data Structure?

Structuring of Data {in Storage || RAM}



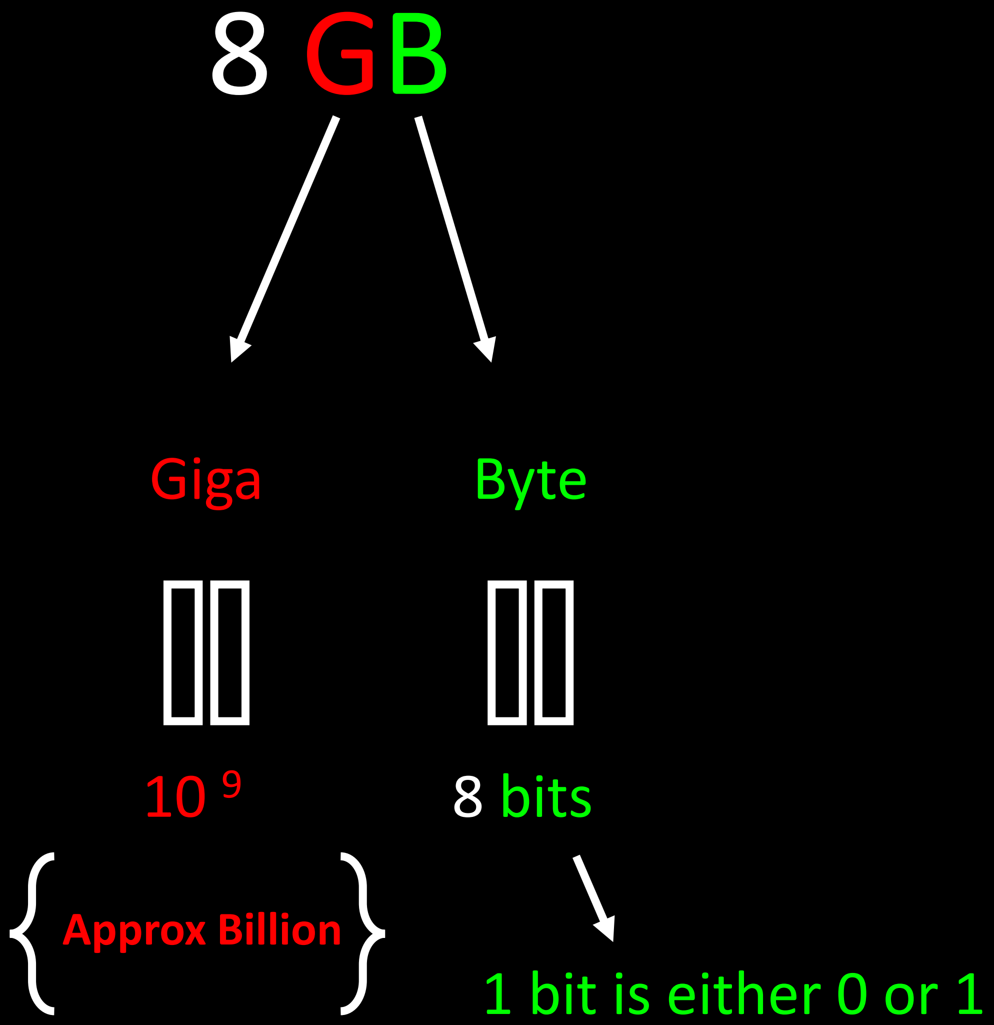
Permanent Storage

Runtime Storage

- This Data Structures are stored in RAM during Program Runtime.

- Examples of Data Structures are
  - int
    - 1, 2, 3, -5, …
  - float
    - 1.1, 3.8, …
  - char
    - a, A, b, c, … $, #, ₹, …
  - User built
    - Gender Class

It is very common for RAM to come in 4GB, 8GB, 12GB, 16GB, 32GB, 64GB, …



8 GB

Giga

▯▯

10^9

{ **Approx Billion** }

Byte

▯▯

8 bits

1 bit is either 0 or 1

bit

- A bit is either 0 or 1 in an Abstract form.

- In Reality:
  - 0 means Low Voltage
  - 1 means High Voltage

- Computers works with low and high voltages.

To Store, for example, {1, 7} in computer:

1

00000000 00000000 00000000 00000001

7

00000000 00000000 00000000 00000111

(integer: takes 4 bytes)

2^n … 64 32 16 8 4 2 1

“Bits are in the form of 2^n.”

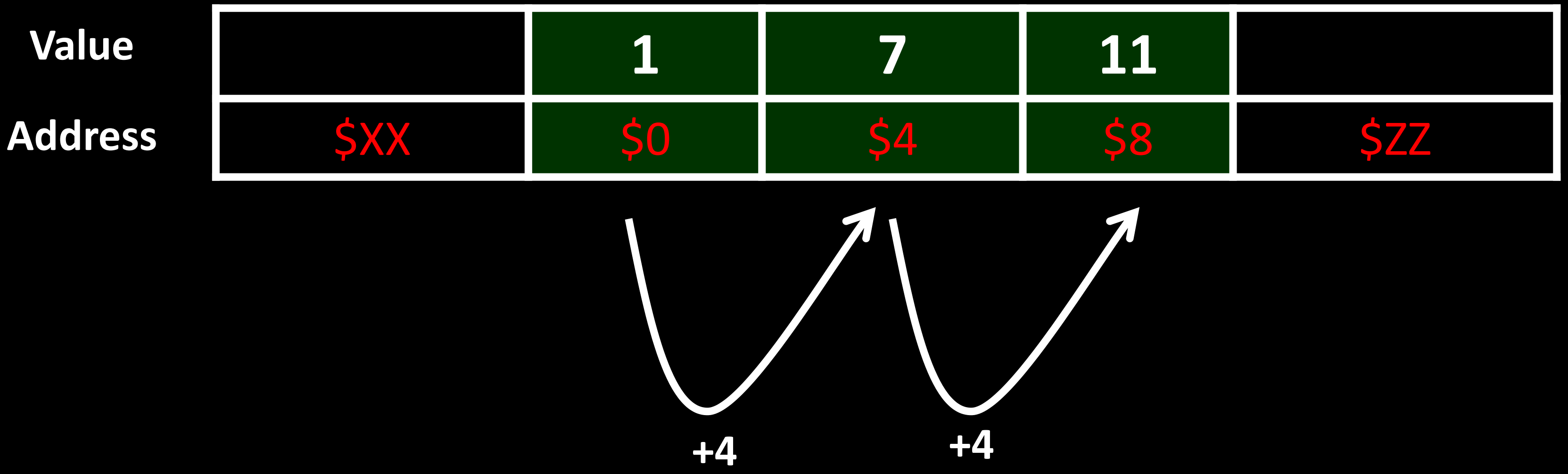
What is an Array?

Array is a contiguous block of data.

How arrays are stored in RAM?

1	7	11

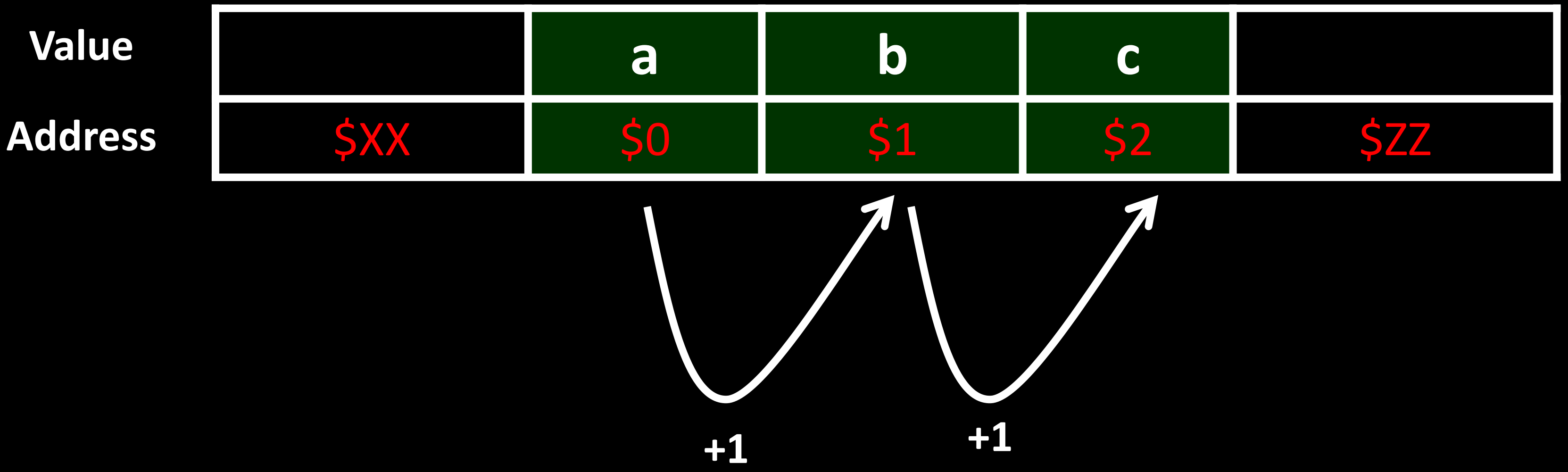
Most integers are stored in bytes of 4.



Value		1	7	11	
Address	$XX	$0	$4	$8	$ZZ

a	b	c

Most ASCII characters are stored in 1 byte.



Value		a	b	c	
Address	$XX	$0	$1	$2	$ZZ

Bit Operations

Manipulating the bits

AND	
0 **&** 0	0
0 **&** 1	0
1 **&** 0	0
1 **&** 1	1

OR	
0 **|** 0	0
0 **|** 1	1
1 **|** 0	1
1 **|** 1	1

XOR	
0 **^** 0	0
0 **^** 1	1
1 **^** 0	1
1 **^** 1	0

```
# AND

N = 1 & 1
```

```
# OR

N = 1 | 0
```

```
# XOR

N = 0 ^ 1
```

Truth Tables

NOT	
~0	1
~1	0

```
# NOT

N = ~1
```

Bit Shifting (<<, >>)

>> (Right Shift) – Shift the bit to the right, add 0 from the left

<< (Left Shift) – Shift the bit to the left, add 0 from the right.

- N = 11 (1011)

Operation	Bits
Load	1011
>> (Right Shift)	1011 >> 1 → 0101
<< (Left Shift)	0101 << 1 → 1010
>>	1010 >> 1 → 0101
>>	0101 >> 1 → 0010
>>	0010 >> 1 → 0001
>>	0001 >> 1 → 0000

Thanks

May the force be with you.