Lecture 04: Memory and Binary Systems

ITSC 3181 Introduction to Computer Architecture
https://passlab.github.io/ITSC3181/

Department of Computer Science
Yonghong Yan
yyan7@uncc.edu
https://passlab.github.io/yanyh/

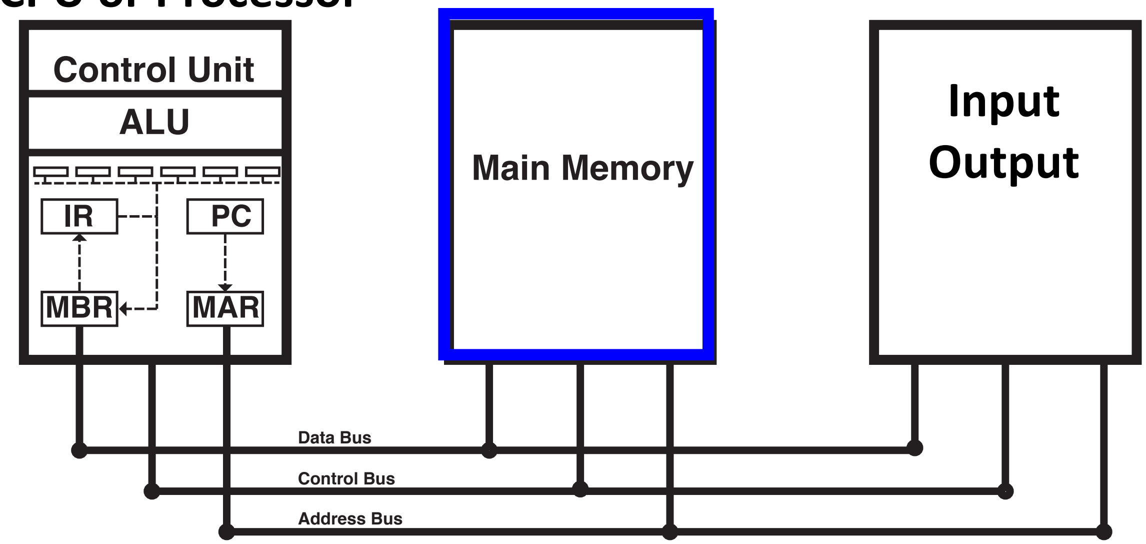
Lectures for Chapter 1 and C Basics
Computer Abstractions and Technology

- **Lecture 01: Chapter 1**
  - 1.1 – 1.4: Introduction, great ideas, Moore's law, abstraction, computer components, and program execution

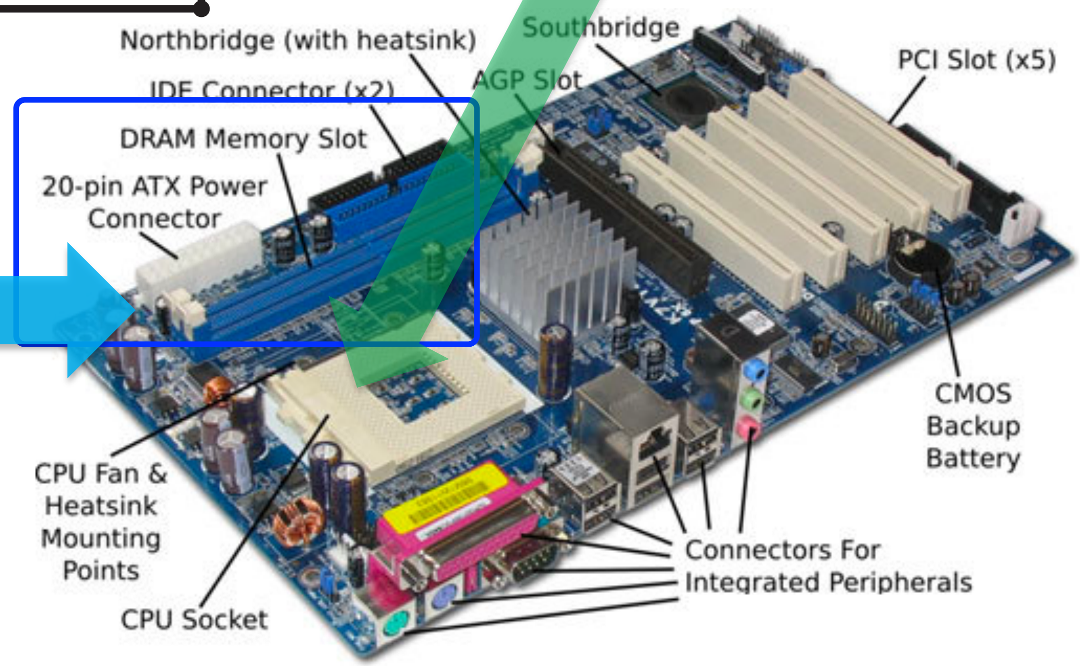
- **Lecture 02 - 03: C Basics; Compilation, Assembly, Linking and Program Execution**
- **Lecture 03 - 04: Chapter 1**
  - 1.6 – 1.7: Performance, power and technology trends
- ☛ **Lecture 04 - 05: Memory and Binary Systems**
- **Lecture 05:**
  - 1.8 - 1.9: Multiprocessing and benchmarking

Main Memory (DRAM) of a Computer

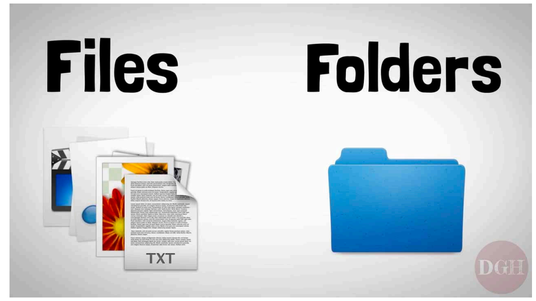
CPU or Processor



CPU is also called a chip.



Everything is Data Stored in Files



- Source code, executable, object are all files
  - Files: Hello.c, sum_full.c, sum
  - Folder: ., .., /home/yanyh, etc
- Compiler, OS kernel, etc are all stored as files
  - gcc, vmlinuz-4.4.0-104-generic
- Information about files/folders and data are also files
  - Metadata

- **Files need to be loaded to memory in order to be processed**
  - ./hello: load the file hello and execute it
  - ls: load the file ls, which is the command ls, and execute it. The ls command lists the files in the specified folder.

Loading a file for a command to Memory

- To load a file from disk into memory

- Loading: To execute a file, e.g.
  - `yanyh@vm:~/sum$ ./sum 1000000`
    - ./ is to specify the path of sum file
  - To execute any linux command, e.g. “ls, cd”, etc.
  - Right-click an app icon to execute the app
- The runtime instance of an executable is called a “**process**”
  - It occupies memory,
  - It uses resources (files, sockets, driver, etc).
  - It executes its threads (machine instructions).
  - See the processes of the system using “ps” command, Windows “task manager”, and Mac OS X “Activity Monitor”

Memory and States

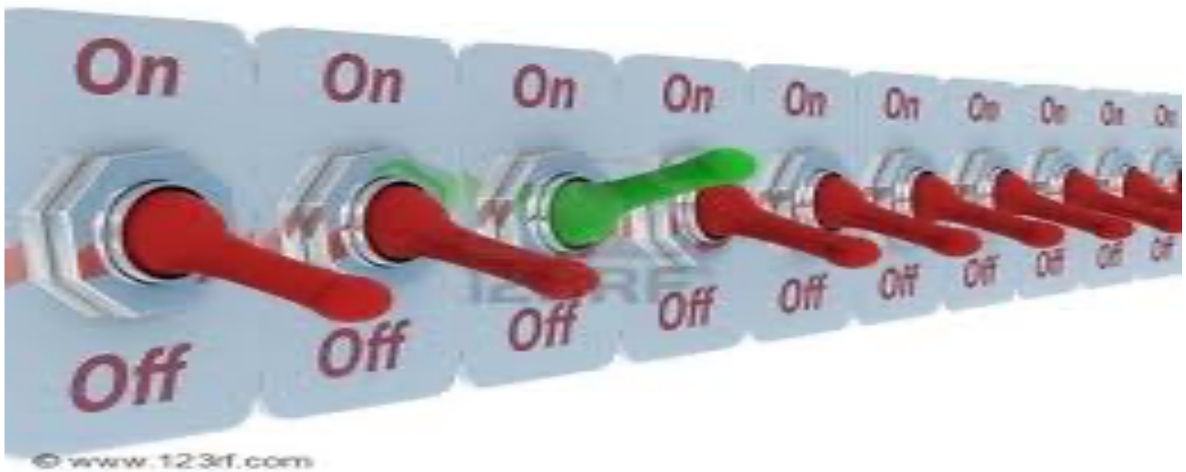
- A memory device is a gadget that helps you record information and recall the information at some later time.
- The minimum unit of memory is like an electrical switch

- The electrical switch can be in one of these 2 states:

> - off (we will call this state 0)
> - on (we will call this state 1)

Memory Cells Used In A Computer

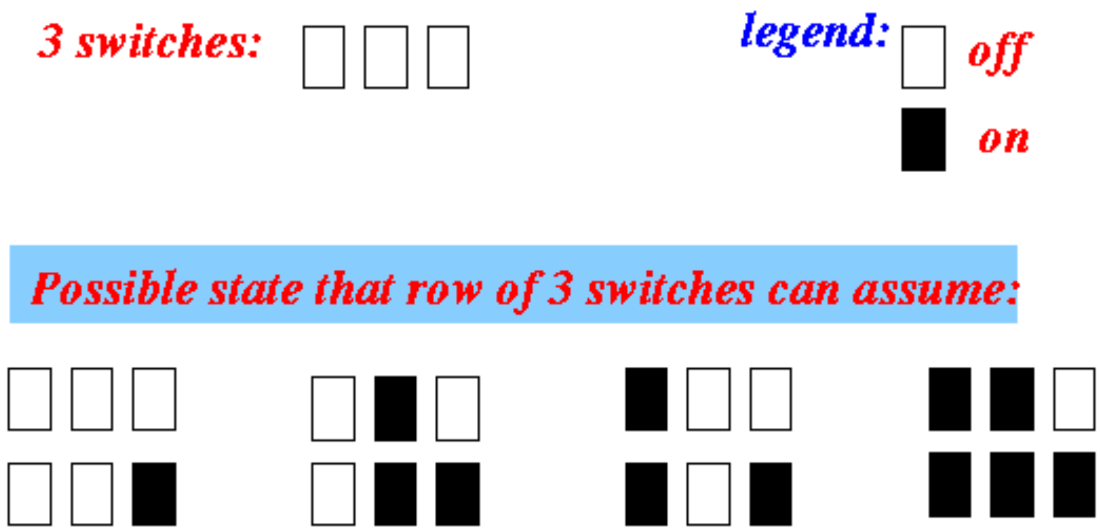
- *One* switch can be in one of 2 states
- A row of *n* switches:



can be in one of 2^n states !

Memory Cells Used In A Computer (cont.)

- Example: row of 3 switches

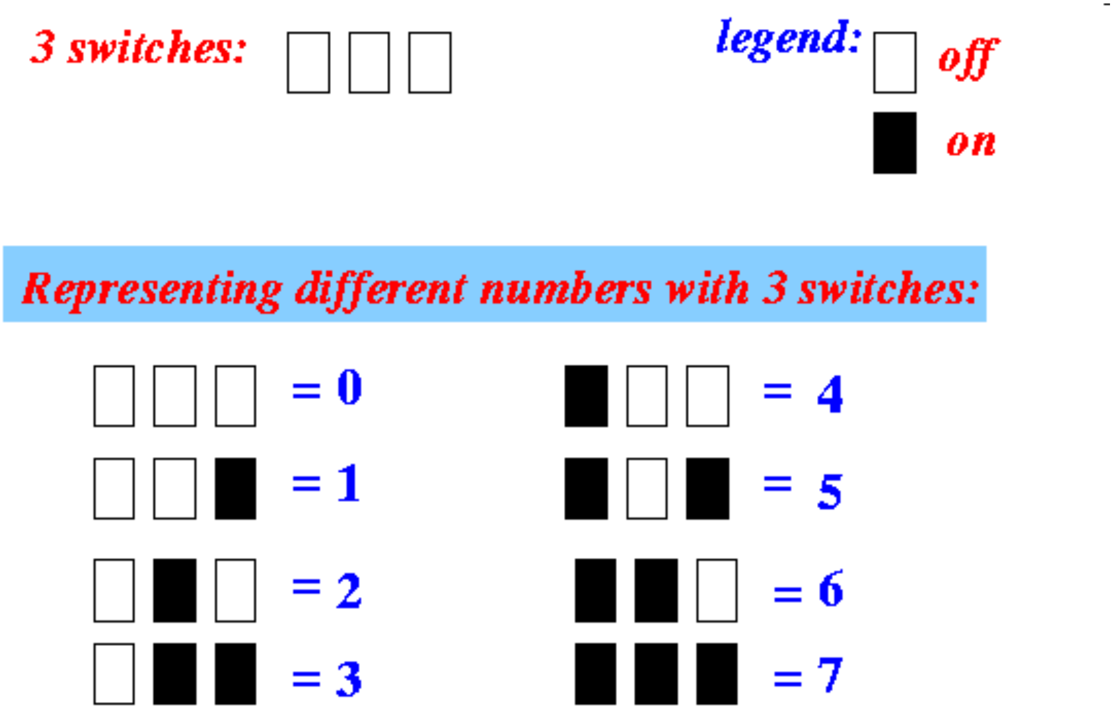


- A row of 3 switches can be in one of $2^3 = 8$ states.
- The 8 possible states are given in the figure above.

Representing Numbers Using a Row of Switches

- We can represent each number using a different state of the switches.

Example:



The *Binary Number System*

- The binary number system uses 2 digits to encode a number:

  - 0 = represents no value
  - 1 = represents a unit value

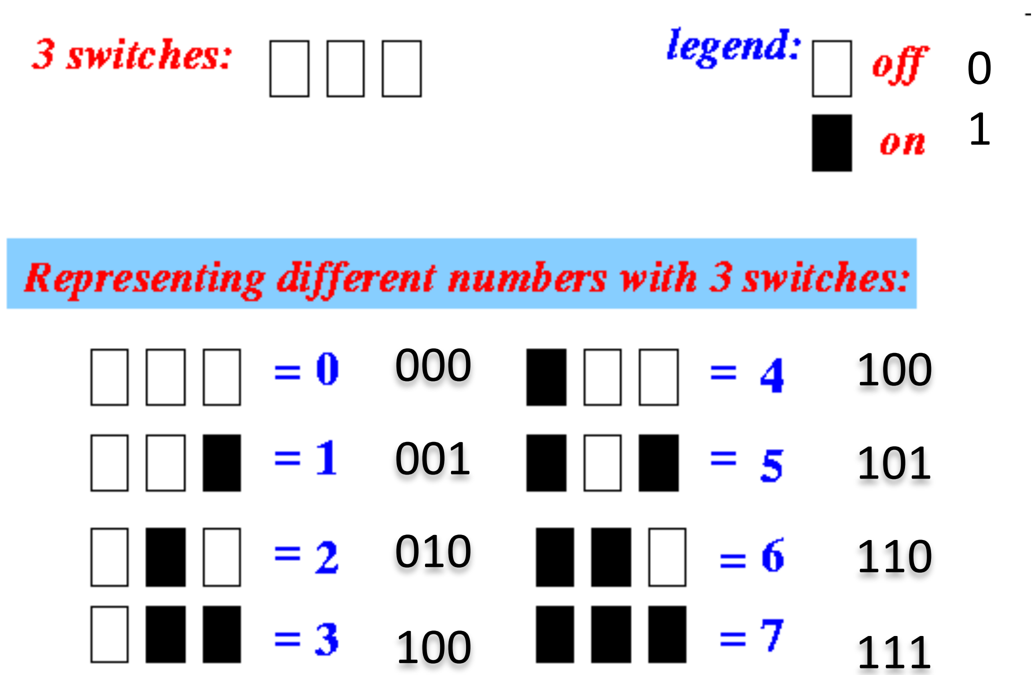
- That means that you can *only* use the digits 0 and 1 to write a *binary number*

  – Example: some binary numbers

  - 0
  - 1
  - 10
  - 11
  - 1010
  - and so on.

The *Binary Number System*

- The different states of these 3 switches represent the numbers 0-7 using the binary number system:



The *Binary Number* System

- The value that is *encoded (represented)* by a binary number is computed as follows:

Binary number	Value encoded by the binary number
$d_{n-1}\ d_{n-2}\ \ldots\ d_1\ d_0$	$d_{n-1} \times 2^{n-1} + d_{n-2} \times 2^{n-2} + \ldots + d_1 \times 2^1 + d_0 \times 2^0$

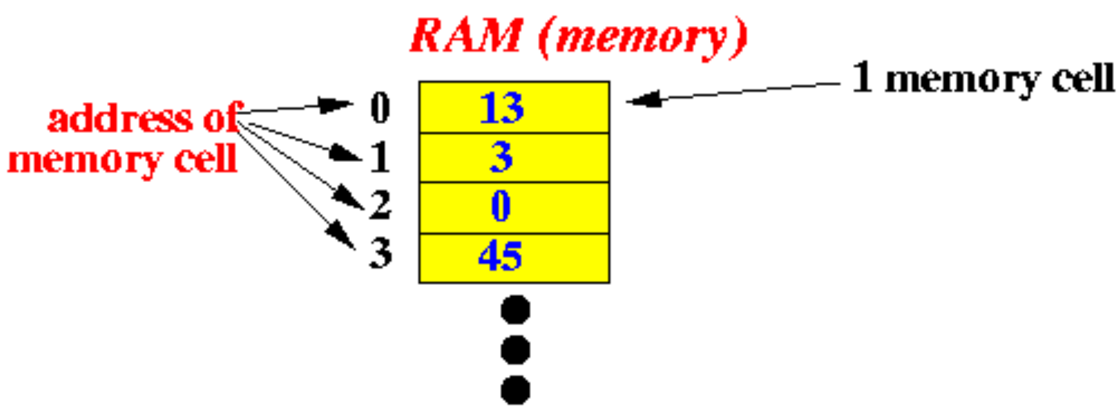
The *Binary Number* System

Example:

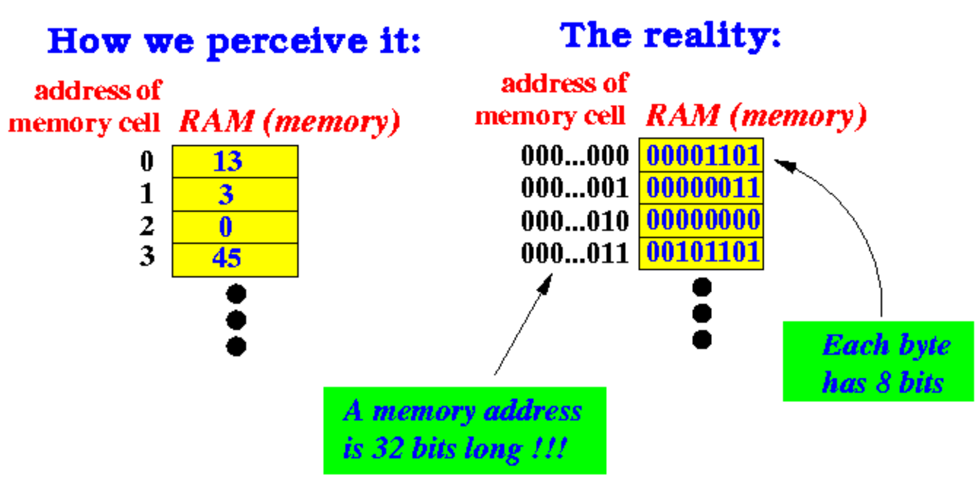
Binary number	Value encoded by the binary number
0	$0 \times 2^0 = 0$
1	$1 \times 2^0 = 1$
10	$1 \times 2^1 + 0 \times 2^0 = 2$
11	$1 \times 2^1 + 1 \times 2^0 = 3$
1010	$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 8 + 2 = 10$

Memory and Binary Number in a Computer

- Computer memory consists of multiple memory cells and each cells stores a number



- The computer system uses the binary number encoding to store the number



Memory and Binary Number in a Computer (cont.)

- *Note*: the address is also expressed as a binary number

A computer can have over 4,000,000,000 bytes (4 Gigabytes) of memory.

So we need a 32 bites to express the address

Combining Adjacent Memory Cells

- A byte has 8 bits and therefore, it can store:
  - $2^8 = 256$ different patterns

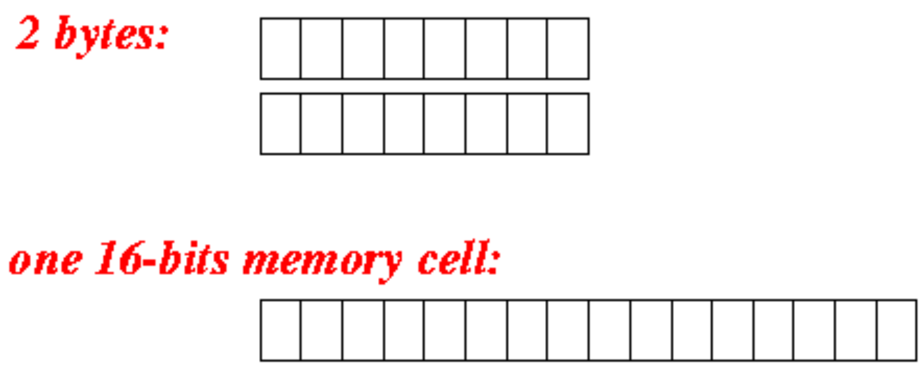
> - 00000000 = 0
> - 00000001 = 1
> - 00000010 = 2
> - 00000011 = 3
> - ...
> - 11111111 = 255

- Therefore, one byte can store one of 256 possible values
- You can store the number 34 into a byte,
- But you cannot store the number 456, the value is out of range)

Combining Adjacent Memory Cells (cont.)

- The computer can combine adjacent bytes (memory cells) and use it as a larger memory cell

  Schematically:

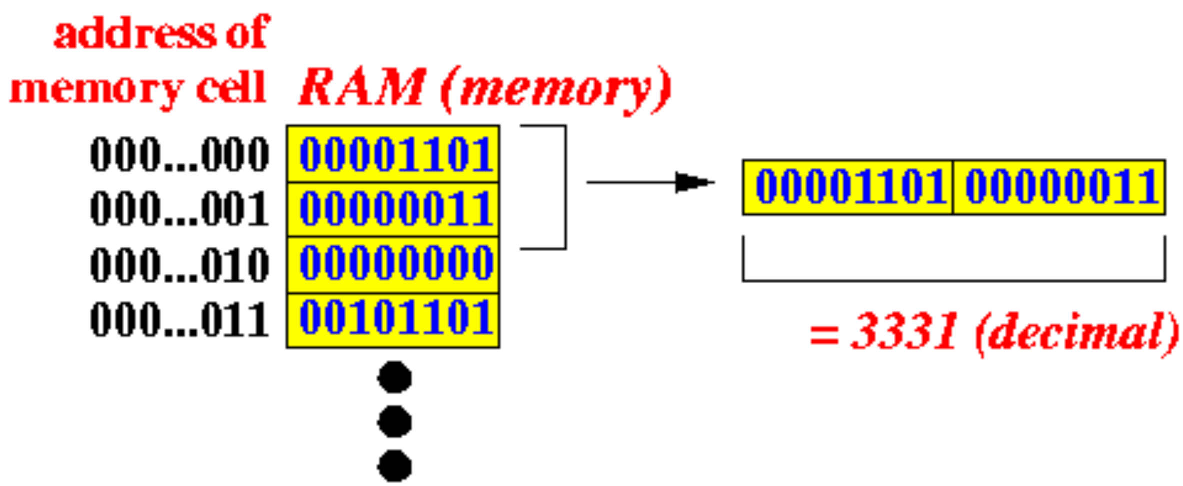


A 16 bits memory cell can store one of $2^{16} = 65536$ different patterns.

Therefore, it can represent (larger) numbers ranging from: 0 – 65535.

Combining Adjacent Memory Cells (cont.)

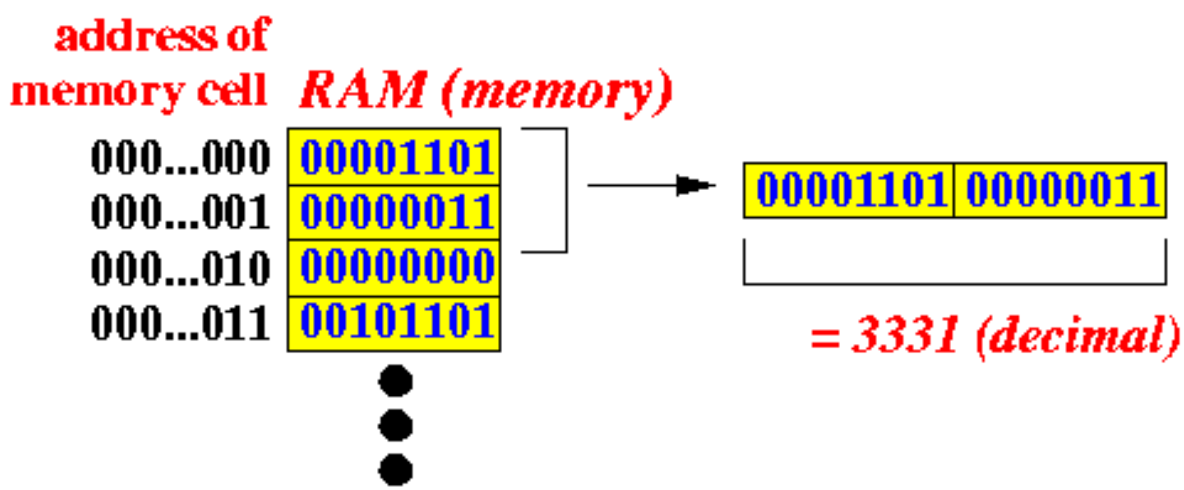
- Example: how a computer can use 2 *consecutive* bytes as a 16 bits memory cell:



- The bytes at address 0 and address 1 can be interpreted as a 16 bits memory cell (with address 0)

Combining Adjacent Memory Cells (cont.)

- When the computer accesses the memory, it specifies:
  - The memory location (address)
  - The number of bytes it needs
  - E.g. read from 000…000 for two bytes: It reads 3331 (decimal number)



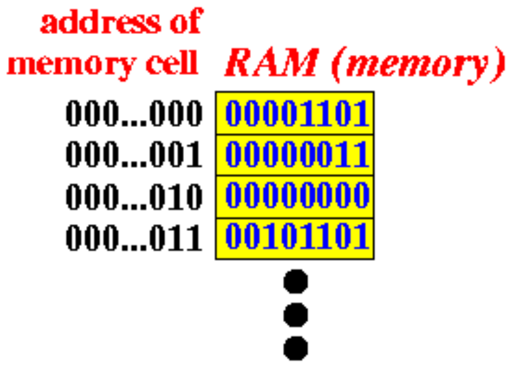
Combining Adjacent Memory Cells (cont.)

- Combine 4 *consecutive* bytes and use them as a 32 bits memory cell
  - To represent numbers ranging from: 0 – (2^{32}-1) or 0 – 4294967295

- combine 8 *consecutive* bytes and use them as a 64 bits memory cell
  - To represent numbers ranging from: 0 – (2^{64}-1) or 0 – 18446744073709551615

Data Store in Memory

- What information is stored in the RAM memory (what is the number represents) depends on:



  - The type of data (this is the context information)
  - Example of types: marital status, gender, age, salary, and so on.
  - This determines the encoding scheme used to interpret the number

Variables are Memory Locations

Compiler maps variable → memory location.

Declarations do not initialize!

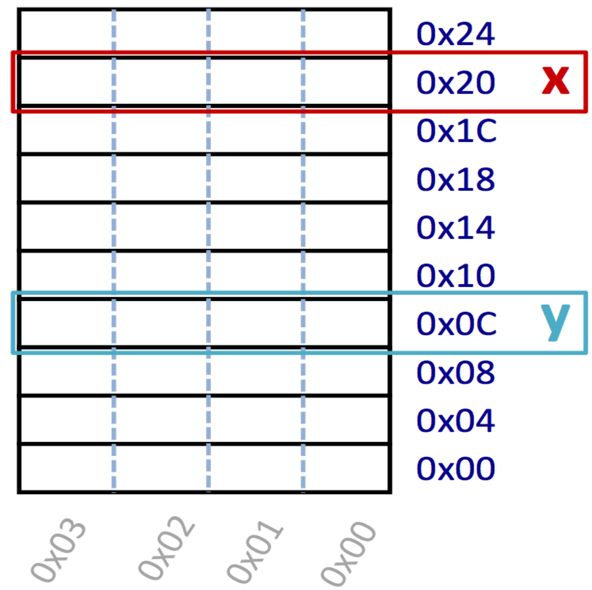
```
int x; // x at 0x20
int y; // y at 0x0C

x = 0; // store 0 at 0x20

// store 0x3CD02700 at 0x0C
y = 0x3CD02700;

// load the contents at 0x0C,
// add 3, and store sum at 0x20
x = y + 3;
```

int is a 4-byte data type.



- Variable (x) is symbolic representation of a memory location
  - = x: Right value, i.e. appears on the right side of =
    - read/load the content from the memory location
  - x =: Left value, i.e. appears on the left side of =
    - Write a value to the memory location