

UNIVERSITY OF WATERLOO  
FACULTY OF ENGINEERING  
Department of Electrical & Computer Engineering

ECE 150 *Fundamentals of Programming*

# Main memory

Prof. Hiren Patel, Ph.D.  
Douglas Wilhelm Harder, M.Math. LEL  
hdpatel@uwaterloo.ca dwharder@uwaterloo.ca

© 2018 by Douglas Wilhelm Harder and Hiren Patel.  
Some rights reserved.

ECE150

CC BY NC SA

UNIVERSITY OF WATERLOO  
FACULTY OF ENGINEERING  
Department of Electrical & Computer Engineering

Main memory 2

## Outline

- In this lesson, we will:
  - Describe main memory
  - Define bytes and byte-addressable memory
  - Describe how addresses are stored
  - Describe how bytes are given addresses on various processors
  - Look at some images of memory

ECE150

UNIVERSITY OF WATERLOO  
FACULTY OF ENGINEERING  
Department of Electrical & Computer Engineering

Main memory 3

## Main memory

- Main memory stores data to be used by the processor
  - It also stores instructions for programs
  - It is volatile
    - When you shut down your computer, memory is lost
- Your executing programs must load instructions and data from memory to the processor and store data back in main memory
  - This is all done for you behind the scenes
  - You must never-the-less understand this


ECE150

UNIVERSITY OF WATERLOO  
FACULTY OF ENGINEERING  
Department of Electrical & Computer Engineering

Main memory 4

## Main memory

- How would you organize memory?
  - A book has pages, and each page has a unique page number
  - You could reference a specific word by saying:
    - "The 4<sup>th</sup> word on the 7<sup>th</sup> line of the 372<sup>nd</sup> page..."



Alex Proimos from Sydney, Australia

- We need something similar, but simpler for a computer
  - Recall we have years, months, days, hours, minutes and seconds
  - Unix counts time by seconds relative to midnight January 1, 1970
  - Every piece of data will have a unique integer address

ECE150



## Main memory

- Previously we described the computer only thinks in binary
  - Main memory is nothing more than billions or trillions of 0s and 1s
  - You could give each bit a separate address
    - You could then ask for the 32280514913<sup>rd</sup> bit
  - Problem: that's a lot of addresses
- Instead, hardware developers decided to give every 8 bits their own address
  - 8 bits is one byte
  - This is called *byte-addressable* memory



## Byte-addressable memory

- In byte-addressable memory, each byte has its own unique address
  - The address is a binary number
    - On a 32-bit computer, the addresses are 32 bits long
    - On a 64-bit computer, the addresses are 64 bits long
  - A 32-bit computer can address  $2^{32}$  bytes: 4 GiB
  - A 64-bit computer can address  $2^{64}$  bytes: 16 EiB or ~17 million TiB
  - Not every address may correspond to an actual physical byte
    - Most computers today do not have 16 EiB of main memory
- To contrast, a hard drive is block-addressable
  - Every 4 kilobyte block of bytes has its own address



## The byte

- A byte is 8 bits:
  - All possible bytes are shown here:

```
00000000 00000001 00000010 00000011 00000100 00000101 00000110 00000111 00001000 00001001 00001010 00001011 00001100 00001101 00001110 00001111
00010000 00010001 00010010 00010011 00010100 00010101 00010110 00010111 00011000 00011001 00011010 00011011 00011100 00011101 00011110 00011111
00100000 00100001 00100010 00100011 00100100 00100101 00100110 00100111 00101000 00101001 00101010 00101011 00101100 00101101 00101110 00101111
00110000 00110001 00110010 00110011 00110100 00110101 00110110 00110111 00111000 00111001 00111010 00111011 00111100 00111101 00111110 00111111
01000000 01000001 01000010 01000011 01000100 01000101 01000110 01000111 01001000 01001001 01001010 01001011 01001100 01001101 01001110 01001111
01010000 01010001 01010010 01010011 01010100 01010101 01010110 01010111 01011000 01011001 01011010 01011011 01011100 01011101 01011110 01011111
01100000 01100001 01100010 01100011 01100100 01100101 01100110 01100111 01101000 01101001 01101010 01101011 01101100 01101101 01101110 01101111
10000000 10000001 10000010 10000011 10000100 10000101 10000110 10000111 10001000 10001001 10001010 10001011 10001100 10001101 10001110 10001111
10010000 10010001 10010010 10010011 10010100 10010101 10010110 10010111 10011000 10011001 10011010 10011011 10011100 10011101 10011110 10011111
10100000 10100001 10100010 10100011 10100100 10100101 10100110 10100111 10101000 10101001 10101010 10101011 10101100 10101101 10101110 10101111
10110000 10110001 10110010 10110011 10110100 10110101 10110110 10110111 10111000 10111001 10111010 10111011 10111100 10111101 10111110 10111111
11000000 11000001 11000010 11000011 11000100 11000101 11000110 11000111 11001000 11001001 11001010 11001011 11001100 11001101 11001110 11001111
11010000 11010001 11010010 11010011 11010100 11010101 11010110 11010111 11011000 11011001 11011010 11011011 11011100 11011101 11011110 11011111
11100000 11100001 11100010 11100011 11100100 11100101 11100110 11100111 11101000 11101001 11101010 11101011 11101100 11101101 11101110 11101111
11110000 11110001 11110010 11110011 11110100 11110101 11110110 11110111 11111000 11111001 11111010 11111011 11111100 11111101 11111110 11111111
```

- Do you care? No



## Byte-addressable memory

- What byte addressable means is:
  - You cannot read or write a single bit from main memory
  - If you need just a single bit, you must never-the-less refer to an entry byte in memory
  - If you want to write a single bit to main memory, you must write the entire byte in which it is contained
  - It is very convenient therefore to use multiples of bytes





## The byte

- However, every byte can be represented by two hexadecimal digits:

0b00000000	0x00	0000	0
0b00101100	0x2c	0001	1
0b01000100	0x44	0010	2
0b10001010	0x8a	0011	3
0b10110101	0xb5	0100	4
0b11101000	0xe8	0101	5
0b11111111	0xff	0110	6
		0111	7
		1000	8
		1001	9
		1010	a
		1011	b
		1100	c
		1101	d
		1110	e
		1111	f

- This is a common means of examining memory



## Addresses

- Today:
  - Most processors have 64-bit addresses
  - Many microcontrollers have 32-bit addresses
  - Smaller microcontrollers may have 24-bit or 16-bit addresses
- Each of these are multiples of 8 bits
  - A 64-bit address occupies 8 bytes
  - A 32-bit address occupies 4 bytes
  - A 24-bit address occupies 3 bytes
- Thus, we may represent a
  - 64-bit address with 16 hexadecimal numbers
  - 32-bit address with 8 hexadecimal numbers
  - 24-bit address with 6 hexadecimal numbers



## Addresses

- Thus, an address on your 64-bit desk- or lap-top computer may look like:

0x0000839d542a8b38

- The address of the next byte would be

0x0000839d542a8b39

- An address on a 32-bit computer might look like:

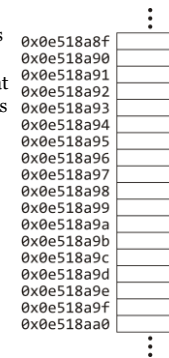
0x0e518a93



## Addresses

- If we are showing a block of bytes in memory, each byte will be listed with its address

- In general, the addresses are not significant
- What is important is what we store at those addresses



If you care: these are the addresses of bytes on a 32-bit processor





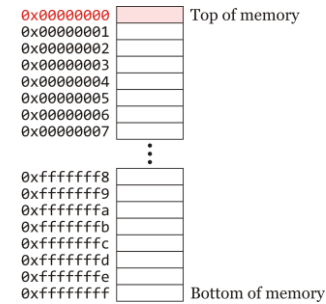
## Addresses

- You can always buy as much memory as the processor can address
  - If you buy 4 GiB of memory for a 32-bit computer, every address is associated with a byte



## Addresses

- The first address is always reserved
  - We will never be able to use or access the byte at 0x00000000
  - It is at the *top* of memory



## Addresses

- On occasion, addresses will be printed without leading zeros, but you should assume they are there:
  - On a 64-bit processor, the address 0x59243d48c is actually 0x000000059243d48c



## Naming conventions

- Main memory is immediately available to the processor
  - Other storage devices are referred to *auxiliary memory*
  - Data from auxiliary memory must be loaded into main memory prior to being used by the computer
- Main and auxiliary memory may also be referred to as
  - Internal* and *external* memory, respectively
  - Primary* and *secondary* memory, respectively





## Properties

- Main memory is almost universally volatile
  - In general, when you shut down a computer, main memory is erased
- Auxiliary memory is non-volatile
  - Removing a power source to auxiliary memory does not erase the contents
  - E.g., flash drives, magnetic disks, optical discs and magnetic tapes
- Main memory is often referred to as *random access memory* (RAM)
  - Any byte can be accessed in the same amount of time
  - Flash drives are also RAM



## References

- [1] No references?



## Summary

- Following this lesson, you now
  - Understand that memory is byte addressable
    - Each byte has its own unique integer address
  - Know that the byte is the smallest amount of memory that can be accessed
  - Know that addresses tend to be multiples of bytes in length
    - They are best described as hexadecimal numbers



## Colophon

These slides were prepared using the Georgia typeface. Mathematical equations use Times New Roman, and source code is presented using Consolas.

The photographs of lilacs in bloom appearing on the title slide and accenting the top of each other slide were taken at the Royal Botanical Gardens on May 27, 2018 by Douglas Wilhelm Harder. Please see

<https://www.rbg.ca/>

for more information.





## Disclaimer

These slides are provided for the ECE 150 *Fundamentals of Programming* course taught at the University of Waterloo. The material in it reflects the authors' best judgment in light of the information available to them at the time of preparation. Any reliance on these course slides by any party for any other purpose are the responsibility of such parties. The authors accept no responsibility for damages, if any, suffered by any party as a result of decisions made or actions based on these course slides for any other purpose than that for which it was intended.

