Welcome to
“Computer Organization and Design Logic”

Winter 2017

CS 64: Computer Organization and Design Logic
Lecture #1

Ziad Matni
Dept. of Computer Science, UCSB
Your Instructor

Your instructor: Ziad Matni (zee-ahd mat-knee)
Email: zmatni@cs.ucsb.edu
Happy to receive your feedback on this class!
--- Please put “CS64” at the start of the subject header ---
Office: SSMS 4409
Office hours: Wed. 1:00 – 3:00 PM
Your TAs

Your TAs & their office hours:

**Meteahan Ozten**  mozten@umail...  Tu 1:00 – 3:00 PM
**Zhujun Xiao**  zhujunxiao@umail...  Th 10:30 – 11:30 AM

*All TA office hours will be in *Trailer 936*
YOUR LAB IS HERE

All TA office hours will be in Trailer 936
You!

With a show of hands, tell me... how many of you...

A. Are Freshmen? Sophomores? Juniors? Seniors?
B. Are CS majors? Other?
C. Know: C, C++, Java, Python, JavaScript, PERL, Bash programming?
D. Have NOT used a Linux or UNIX system before?
E. Have seen actual “assembly code” before?
F. Programmed in assembly before?
G. Written/seen code for firmware?
H. Understand basic binary logic (i.e. OR, AND, NOT)?
I. Designed a digital circuit before?
This Class

- This is an introductory course in low-level programming and computer hardware.
  - Two separate but intertwined areas

- What happens between your C/C++/Java/Python command:
  "int a = 3, b =4, c = a+b;"
  and the actual "digital mechanisms" in the CPU that process this “simple” command?

- This class will move fast – so please prepare accordingly.
Lecture Etiquette!

• I need you INVOLVED and ACTIVE!

• **Phones OFF!**
  and laptops/tablets are for **NOTES** only
  – No tweeting, texting, FB-ing, surfing, gaming, Snapchatting, etc.!
  – I will ask you to leave class if you do not follow this policy. Especially if you are disrupting others.

• To succeed in this class, you need to take **thorough** notes
  – I’ll provide my slides, but not class notes
  – Studies show that written notes are superior to typing them into a laptop!
Logged In and Zoned Out: How Laptop Internet Use Relates to Classroom Learning

Susan M. Ravizza, Mitchell G. Uitvlugt, and Kimberly M. Fenn
Department of Psychology, Michigan State University, East Lansing

Abstract
Laptop computers are widely prevalent in university classrooms. Although laptops are a valuable tool, they offer access to a distracting temptation: the Internet. In the study reported here, we assessed the relationship between classroom performance and actual Internet usage for academic and nonacademic purposes. Students who were enrolled in an introductory psychology course logged into a proxy server that monitored their online activity during class. Past research relied on self-report, but the current methodology objectively measured time, frequency, and browsing history of participants' Internet usage. In addition, we assessed whether intelligence, motivation, and interest in course material could account for the relationship between Internet use and performance. Our results showed that nonacademic Internet use was common among students who brought laptops to class and was inversely related to class performance. This relationship was upheld after we accounted for motivation, interest, and intelligence. Class-related Internet use was not associated with a benefit to classroom performance.
Computer Organization and Design Logic (CS64)

Instructor: Ziad Matni, Winter 2017

Resources

• Syllabus
  ALWAYS CHECK THE SYLLABUS FIRST FOR ANY QUESTIONS YOU HAVE BEFORE CONTACTING US!!!

• Lab Assignments (and how to Connect to CSIL Remotely)
• Grading Policies Regarding MIPS Assembly Instructions
• MIPS Calling Convention

• Piazza, for all non-personal communication

Contacting Us

• Ziad Matni (Instructor): zmatni at cs dot ucsb dot edu
• Zhujun Xiao (TA): zhujunxiao at cs dot ucsb dot edu
• Metehan Ozten (TA): mozten at cs dot ucsb dot edu

Office Hours

• Ziad: Wednesday, 1:00 PM to 3:00 PM in SSMS 4409
• Zhujun: Thursday, 9:30 AM to 11:30 AM in Trailer 936
• Metehan: Monday, 1:00 PM to 3:00 PM in Trailer 936
• Additional office hours are also available by appointment

Weekly Course Materials
Just in Case...

It's in the syllabus.

This message brought to you by every instructor that ever lived.

WWW.PHDCOMICS.COM
So... let’s take a look at that syllabus...

Electronic version found at:
http://cs.ucsb.edu/~zmatni/syllabi/CS64W17_syllabus.pdf
<table>
<thead>
<tr>
<th>Wk 1</th>
<th>Mon 01/09</th>
<th>Tue 01/10</th>
<th>Wed 01/11</th>
<th>Thu 01/12</th>
<th>Fri 01/13</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lab 1</td>
<td>Lecture 1</td>
<td></td>
<td>Lecture 2</td>
<td></td>
</tr>
<tr>
<td>Wk 2</td>
<td>Mon 01/16</td>
<td>Tue 01/17</td>
<td>Wed 01/18</td>
<td>Thu 01/19</td>
<td>Fri 01/20</td>
</tr>
<tr>
<td></td>
<td>UCSB</td>
<td>Lecture 3</td>
<td></td>
<td>Lecture 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HOLIDAY</td>
<td>Lab 1 DUE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wk 3</td>
<td>Mon 01/23</td>
<td>Tue 01/24</td>
<td>Wed 01/25</td>
<td>Thu 01/26</td>
<td>Fri 01/27</td>
</tr>
<tr>
<td></td>
<td>Lab 2</td>
<td>Lecture 5</td>
<td></td>
<td>Lecture 6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lab 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wk 4</td>
<td>Mon 01/30</td>
<td>Tue 01/31</td>
<td>Wed 02/01</td>
<td>Thu 02/02</td>
<td>Fri 02/03</td>
</tr>
<tr>
<td></td>
<td>Lab 4</td>
<td>Lecture 7</td>
<td></td>
<td>MIDTERM 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lab 2 DUE</td>
<td></td>
<td></td>
<td>Lab 3 DUE</td>
<td></td>
</tr>
<tr>
<td>Wk 5</td>
<td>Mon 02/06</td>
<td>Tue 02/07</td>
<td>Wed 02/08</td>
<td>Thu 02/09</td>
<td>Fri 02/10</td>
</tr>
<tr>
<td></td>
<td>Lab 5</td>
<td>Lecture 8</td>
<td></td>
<td>Lecture 9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lab 4 DUE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wk 6</td>
<td>Mon 02/13</td>
<td>Tue 02/14</td>
<td>Wed 02/15</td>
<td>Thu 02/16</td>
<td>Fri 02/17</td>
</tr>
<tr>
<td></td>
<td>Lab 6</td>
<td>Lecture 10</td>
<td></td>
<td>Lecture 11</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lab 5 DUE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wk 7</td>
<td>Mon 02/20</td>
<td>Tue 02/21</td>
<td>Wed 02/22</td>
<td>Thu 02/23</td>
<td>Fri 02/24</td>
</tr>
<tr>
<td></td>
<td>UCSB</td>
<td>Lecture 12</td>
<td></td>
<td>Lecture 13</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HOLIDAY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wk 8</td>
<td>Mon 02/27</td>
<td>Tue 02/28</td>
<td>Wed 03/01</td>
<td>Thu 03/02</td>
<td>Fri 03/03</td>
</tr>
<tr>
<td></td>
<td>Lab 7</td>
<td>Lecture 14</td>
<td></td>
<td>MIDTERM 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lab 8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lab 6 DUE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wk 9</td>
<td>Mon 03/06</td>
<td>Tue 03/07</td>
<td>Wed 03/08</td>
<td>Thu 03/09</td>
<td>Fri 03/10</td>
</tr>
<tr>
<td></td>
<td>Lab 9</td>
<td>Lecture 15</td>
<td></td>
<td>Lecture 16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lab 7 DUE</td>
<td></td>
<td></td>
<td>Lab 8 DUE</td>
<td></td>
</tr>
<tr>
<td>Wk 10</td>
<td>Mon 03/13</td>
<td>Tue 03/14</td>
<td>Wed 03/15</td>
<td>Thu 03/16</td>
<td>Fri 03/17</td>
</tr>
<tr>
<td></td>
<td>Lab 10</td>
<td>Lecture 17</td>
<td></td>
<td>Lecture 18</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lab 9 DUE</td>
<td></td>
<td></td>
<td>Lab 10 DUE</td>
<td></td>
</tr>
</tbody>
</table>
A Simplified View of Modern Computer Architecture

The 5 Main Components of a Computer:
1. Processor
2. Memory
3. Input
4. Output
5. Secondary Data Storage

Input
Keyboard
Mouse
Microphone
Scanner
--or--
From a Program

Output

von Neumann Architecture

CPU
Processing
for calculations, etc...

RAM and ROM
Memory
for instructions, etc...

Secondary Data Storage
HDD and SSD
Mini Flash Drive
CD-ROM
Tape Drive

--or--

Display screen
Speakers
Printer

To a Program

1/10/17
Computer Memory

• Usually organized in two parts:
  – Address: Where can I find my data?
  – Data (payload): What is my data?

• The smallest representation of the data
  – A binary *bit* (“0”s and “1”s)
  – A common collection of bits is a *byte*
    • 8 bits = 1 byte
  – What is a *nibble*?
    • 4 bits = 1 nibble – not used as often...
  – What is the minimum number of bits needed to convey an alphanumeric character? And WHY?
What is the Most Basic Form of Computer Language?

• Binary *a.k.a* Base-2

• Expressing data AND instructions in either “1” or “0”
  – So,
    “01010101 01000011 01010011 01000010 00100001 00100001”

  could mean an *instruction* to “calculate 2 + 3”
  Or it could mean a *number* (856783663333)
  Or it could mean a *string of 6 characters* (“UCSB!!”)
So... like... what process stuff in a computer?

• The Central Processing Unit (CPU)
  – Executes program instructions

• Typical capabilities of CPU include:
  – Add
  – Subtract
  – Multiply
  – Divide
  – Move data from location to location

You can do just about anything with a computer with just these simple instructions
Parts of the CPU

• The CPU is made up of 2 main parts:
  – The Arithmetic Logic Unit (ALU)
  – The Control Unit (CU)

• The ALU does the calculations in binary using “registers” (small RAM) and logic circuits

• The CU handles breaking down instructions into control codes for the ALU and memory
The CPU’s Fetch-Execute Cycle

- **Fetch** the next instruction
- **Decode** the instruction
- **Get data** if needed
- **Execute** the instruction

*Why is it a cycle??*

This is what happens inside a computer interacting with a program at the “lowest” level
Computer Languages and the F-E Cycle

- Instructions get executed in the CPU in machine language (i.e. all in “1”s and “0”s)
  - Even the *smallest* of instructions, like “add 2 to 3 then multiply by 4”, need *multiple* cycles of the CPU to get executed fully
  - But THAT’S OK! Because, typically, CPUs can run *many millions* of instructions per second

- In *low-level languages*, you need to spell those cycles out
- In *high-level languages*, you don’t
  - 1 HLL statement, like “\(x = c*(a + b)\)” is enough to get the job done
  - This would translate into multiple statements in LLLs
“high level” vs. “low level” Programming

• High Level computer languages, like C++ or Java, are A LOT simpler to use!
• Uses syntax that “resembles” human language
• Easy to read and understand:
  \[ x = c \times (a + b) \quad \text{vs.} \quad 101000111010111 \]

• But, still... the CPU *NEEDS* machine language to do what it’s supposed to do!

• So *SOMETHING* has to “translate” high level code into machine language...
Compilers

- *SOMETHING* has to “translate” high level code into machine language...
  - A program called a Compiler
  - Compilers are “language-specific”
Machine vs. Assembly Language

• Machine language is the actual 1s and 0s
  Example:
  101111011101110000101010101000

• Assembly language is one step above (towards a high-level) where instructions are given mnemonic codes but still displayed one step at a time
  – Assembly code has some other advantages of HLL re: human readability
  Example:
  lw $t0, 4($gp)  # fetch N
  mult $t0, $t0, $t0  # multiply N by itself
  # and store the result in N
int main(int argc, char** argv) {
  ...
}

3.14956
int main(int argc, char** argv) {
  ...
}

In reality... →

3.14956
int main(int argc, char** argv) {
    ...
}

With a more efficient algorithm

You Mad Bro?

3.14956
Why Can Programs be Slow?

• After all, isn’t just as simple as getting an instruction, finding the value in memory, and doing stuff to it???

• Yes... except for the “simple” part...

• Ordering the instructions matters
  Where in memory the value is matters
  How instructions get “broken down” matters
  What order these get “pipelined” matters
The Point...

• If you really want performance, you need to know how the “magic” works

• If you want to write a naive compiler (CS160), you need to know some low-level details of how the CPU does stuff

• If you want to write a fast compiler, you need to know tons of low-level details
So Why Digital Design?

• Because that’s where the “magic” happens

• Logical decisions are made with 1s and 0s that come from electrical currents switching one way or the other that modify semiconducting material that obeys the laws of electromagnetism that is... physics...
So Why Digital Design?
Digital Design in this Course

- We will not go into “deep” dives with digital design in this course
  – For that, check out courses in ECE

- We will, however, delve deep enough to understand the fundamental workings of digital circuits and how they are used for computing purposes.
COMPUTERS ARE DIGITAL MACHINES
THEY ARE DESIGNED TO COUNT IN...
Counting Numbers in Different Bases

• We “normally” count in 10s
  – Base 10: decimal numbers
  – We use 10 numerical symbols in Base 10: “0” thru “9”

• Computers count in 2s
  – Base 2: binary numbers
  – We use 2 numerical symbols in Base 2: “0” and “1”

• Represented with 1 bit \(2^1 = 2\)
Other convenient bases in computer architecture:

- Base 8: octal numbers
  - Number symbols are 0 thru 7
  - Represented with 3 bits \((2^3 = 8)\)

- Base 16: hexadecimal numbers
  - Number symbols are 0 thru F:
    \[
    A = 10, \ B = 11, \ C = 12, \ D = 13, \ E = 14, \ F = 15
    \]
  - Represented with 4 bits \((2^4 = 16)\)

- Why are 4 bit representations convenient???
What’s in a Number?

642

What is that???

Well, what NUMERICAL BASE are you expressing it in?
Decimal Numbers

Counting 642 as 600 + 40 + 2
is counting in TENS (aka BASE 10) --- what we’re used to

There are 6 HUNDREDS 6 x 100
There are 4 TENS 4 x 10
There are 2 ONES 2 x 1

\[
\begin{array}{ccc}
6 & 4 & 2 \\
100 & 10 & 1
\end{array}
\]

\[642 = 600 + 40 + 2\]
Positional Notation in Decimal

Continuing with our example…

642 in base 10 positional notation is:

\[ 6 \times 10^2 = 6 \times 100 = 600 \]
\[ + 4 \times 10^1 = 4 \times 10 = 40 \]
\[ + 2 \times 10^0 = 2 \times 1 = 2 \]
\[ = 642 \text{ in base 10} \]

\[ \begin{array}{ccc}
6 & 4 & 2 \\
100 & 10 & 1 \\
\end{array} \]

\[ 642 \text{ (base 10)} = 600 + 40 + 2 \]
What if “642” is expressed in the base of 13?

\[
\begin{align*}
6 \times 13^2 &= 6 \times 169 &= 1014 \\
+ 4 \times 13^1 &= 4 \times 13 &= 52 \\
+ 2 \times 13^0 &= 2 \times 1 &= 2
\end{align*}
\]

\[
642_{(base\ 13)} = 1014 + 52 + 2 = 1068_{(base\ 10)}
\]
Positional Notation in Binary

11101 in base 2 *position notation* is:

\[
\begin{align*}
1 \times 2^4 &= 1 \times 16 = 16 \\
+ 1 \times 2^3 &= 1 \times 8 = 8 \\
+ 1 \times 2^2 &= 1 \times 4 = 4 \\
+ 0 \times 2^1 &= 1 \times 2 = 0 \\
+ 1 \times 2^0 &= 1 \times 1 = 1
\end{align*}
\]

So, 11101 in base 2 is 16 + 8 + 4 + 0 + 1 = **29** in base 10
In-Class Exercise: Converting Binary to Decimal

Q: What is the decimal equivalent of the binary number 1101110?

A: Look for the position of the digits in the number. This one has 7 digits, therefore positions 0 thru 6

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>2^6</td>
<td>64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2^0</td>
</tr>
<tr>
<td>2^5</td>
<td></td>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2^1</td>
</tr>
<tr>
<td>2^4</td>
<td></td>
<td></td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td>2^2</td>
</tr>
<tr>
<td>2^3</td>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td>2^3</td>
</tr>
<tr>
<td>2^2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td>2^4</td>
</tr>
<tr>
<td>2^1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>2^5</td>
</tr>
<tr>
<td>2^0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2^6</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
1 \times 2^6 &= 1 \times 64 = 64 \\
+ 1 \times 2^5 &= 1 \times 32 = 32 \\
+ 0 \times 2^4 &= 0 \times 16 = 0 \\
+ 1 \times 2^3 &= 1 \times 8 = 8 \\
+ 1 \times 2^2 &= 1 \times 4 = 4 \\
+ 1 \times 2^1 &= 1 \times 2 = 2 \\
+ 0 \times 2^0 &= 0 \times 1 = 0 \\
\end{align*}
\]

= 110 in base 10
Converting Binary to Octal and Hexadecimal
(or any base that’s a power of 2)

NOTE THE FOLLOWING:
• Binary is 1 bit
• Octal is 3 bits
• Hexadecimal is 4 bits

• Use the “group the bits” technique
  – Always start from the least significant digit
  – Group every 3 bits together for bin → oct
  – Group every 4 bits together for bin → hex
### Convenient Table...

<table>
<thead>
<tr>
<th>HEXADECIMAL</th>
<th>BINARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HEXADECIMAL (Decimal)</th>
<th>BINARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>A (10)</td>
<td>1010</td>
</tr>
<tr>
<td>B (11)</td>
<td>1011</td>
</tr>
<tr>
<td>C (12)</td>
<td>1100</td>
</tr>
<tr>
<td>D (13)</td>
<td>1101</td>
</tr>
<tr>
<td>E (14)</td>
<td>1110</td>
</tr>
<tr>
<td>F (15)</td>
<td>1111</td>
</tr>
</tbody>
</table>
Converting Binary to Octal and Hexadecimal

• Take the example: \textbf{10100110}
  
  \textit{...to octal:}

\[ \begin{array}{c|c|c|c}
1 & 0 & 1 & 00110 \\
2 & 4 & 6 & \text{246 in octal} \\
\end{array} \]

\textit{...to hexadecimal:}

\[ \begin{array}{c|c|c|c}
1 & 0 & 1 & 00110 \\
10 & 6 & & \text{A6 in hexadecimal} \\
\end{array} \]
Converting Decimal to Other Bases

Algorithm for converting number in base 10 to other bases
While (the quotient is not zero)
  1. Divide the decimal number by the new base
  2. Make the remainder the next digit to the left in the answer
  3. Replace the original decimal number with the quotient
  4. Repeat until your quotient is zero

Example: What is 98 (base 10) in base 8?

\[
\begin{align*}
98 / 8 &= 12 \text{ R } 2 \\
12 / 8 &= 1 \text{ R } 4 \\
1 / 8 &= 0 \text{ R } 1
\end{align*}
\]

1 4 2
In-Class Exercise:
Converting Decimal into Binary & Hex

Convert 54 (base 10) into binary and hex:

- 54 / 2 = 27 R 0
- 27 / 2 = 13 R 1
- 13 / 2 = 6 R 1
- 6 / 2 = 3 R 0
- 3 / 2 = 1 R 1
- 1 / 2 = 0 R 1

54 (decimal) = 110110 (binary)
= 36 (hex)

Sanity check:

110110
= 2 + 4 + 16 + 32
= 54
YOUR TO-DOs

• Assignment #1
  – Due next Tuesday, 1/17

• Remember: NO LAB NEXT MONDAY!
  – Start looking at Lab Assignment #2
  – We’ll discuss it in class next week and you’ll have time to work on it
</LECTURE>