OO software systems are systems of interacting objects.

Objects have

properties:
these are things that objects know
e.g. what you had for breakfast

behaviors:
these are things objects do
e.g. being able to reply to the question “What did you have for breakfast?”
CSE115 / CSE503
Introduction to Computer Science I
Dr. Carl Alphonce
343 Davis Hall
alphonce@buffalo.edu

Office hours:
Tuesday 10:00 AM – 12:00 PM*
Wednesday 4:00 PM – 5:00 PM
Friday 11:00 AM – 12:00 PM

OR request appointment via e-mail

*Tuesday adjustments: 11:00 AM – 1:00 PM on 10/11, 11/1 and 12/6
Dr. Bina Ramamurthy
ANNOUNCEMENTS
Robotics Club

UB ACM
Association for Computing Machinery

&

UB Robotics

BBQ

Friday, September 9th, 2016
Starts at 6:00pm
Greiner Patio

OPEN TO EVERYONE!
JOIN US FOR...

Burgers  Veggie Burgers  Hot Dogs
Ice Cream  Giant Jenga  Frisbee
Remote Control Racecars
Class today

Announcements (Robotics BBQ, Sapphire project)
Instruction decoding
Fetch/Decode/Execute cycle
Low-level and high-level languages

Coming up
Expressions and objects
Please turn off and put away electronics:

cell phones
pagers
laptops
tablets
etc.
REVIEW
For which input values is output 1?

For which input values is output 0?
## Truth table for AND

<table>
<thead>
<tr>
<th>Input 1</th>
<th>Input 2</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
For which input values is output 1?

For which input values is output 0?
<table>
<thead>
<tr>
<th>Input 1</th>
<th>Input 2</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
For which input value is output 1?

For which input value is output 0?
<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
MEMORY
Flip-flop (a bit of memory!)

R (reset)

S (set)

remembered value
MOVING ON
INSTRUCTION

DECODING
Example:
Digital Equipment Corp PDP 11

photo credit:
https://commons.wikimedia.org/wiki/File:Pdp-11-70-panel.jpg
by Dave Fischer
0110000001000010
Encoding instructions (PDP 11)

0110000001000010  a sequence of 0s and 1s

0110  000001  000010  bit groupings
Encoding instructions (PDP 11)

0110000001000010  a sequence of 0s and 1s

0110  000001  000010  bit groupings

OpCode  Arg1  Arg2  general pattern
<table>
<thead>
<tr>
<th>OpCode</th>
<th>Arg1</th>
<th>Arg2</th>
<th>General Pattern</th>
<th>Specific Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>Reg1</td>
<td>Reg2</td>
<td>general pattern</td>
<td>specific instruction</td>
</tr>
</tbody>
</table>

0110000001000010 a sequence of 0s and 1s

0110 000001 000010 bit groupings
This wire will carry a 1 only if the op code of the instruction is 0110.

Circuitry to decode rest of instruction and carry it out ("execute" the instruction)
This wire will carry a 1 only if the op code of the instruction is 1100.

This wire will carry a 1 only if the op code of the instruction is 1101.

This wire will carry a 1 only if the op code of the instruction is 1110.
Controlling information flow

Data from a register

Data to ALU
FETCH
DECODE
EXECUTE

cycle
Fetch an instruction (& update PC)

Decode instruction

Execute instruction

Fetch
(load instruction into IR from location in PC)

Update PC

Execute

Decode
Language levels

HARDWARE

1940s
MACHINE LANGUAGE
(1101000001000010)

HARDWARE

1940s
Language levels

- HARDWARE
  - MACHINE LANGUAGE
    - (1101000001000010)
  - ASSEMBLY LANGUAGE
    - (ADD R1 R2)

1940s
Language levels

- **HARDWARE**
  - Machine Language
    - (1101000001000010)
- **ASSEMBLY LANGUAGE**
  - (ADD R1 R2)
- **ASSEMBLY**
- **HARDWARE**
Language levels

HARDWARE

ASSEMBLY LANGUAGE
(ADD R1 R2)

MACHINE LANGUAGE
(1101000001000010)

HIGH LEVEL LANGUAGE (e.g. Java)
\[ x + y \]

ASSEMBLY

1950s
**Language levels**

- **MACHINE LANGUAGE**
  - (1101000001000010)

- **ASSEMBLY LANGUAGE**
  - (ADD R1 R2)

- **HIGH LEVEL LANGUAGES**
  - \( x + y \)

- 1950s

**Compilation**

**Assembly**
Language levels

LOW LEVEL LANGUAGES

HIGH LEVEL LANGUAGES
\[ x + y \]

ASSEMBLY LANGUAGE
(ADD R1 R2)

MACHINE LANGUAGE
(1101000001000010)

HARDWARE
Java

We can write,

\[ z = x + y \]

instead of something like this,

MOV (R3) R1
MOV (R4) R2
ADD R1 R2
MOV R2 (R5)

Others: C#, Erlang, Python, ML, Prolog, Lisp, etc.
NOTE

Differences between book’s & lecture’s presentation of code
I pointed out that the text’s early code examples are quite involved.

We will discuss the details in lecture over the next several days. The following slides give a sense of the order in which we will have our discussion.

**DO NOT PANIC** if you do not yet understand the book’s code examples at this point – we don’t expect you to.
public class AttendanceExample {
    public static void main(String[] args) {
        PeopleCounter attendeeCounter = new PeopleCounter();

        attendeeCounter.incrementCount();
        attendeeCounter.incrementCount();
        attendeeCounter.incrementCount();

        System.out.print("Attendee count: ");
        attendeeCounter.printCount();

        return;
    }
}
We will start by discussing this bit of code: an expression.
public class AttendanceExample {
    public static void main(String[] args) {
        PeopleCounter attendeeCounter = new PeopleCounter();
        attendeeCounter.incrementCount();
        attendeeCounter.incrementCount();
        attendeeCounter.incrementCount();
        System.out.print("Attendee count: ");
        attendeeCounter.printCount();

        return;
    }
}
public class AttendanceExample {
    public static void main(String[] args) {
        PeopleCounter attendeeCounter = new PeopleCounter();

        attendeeCounter.incrementCount();
        attendeeCounter.incrementCount();
        attendeeCounter.incrementCount();

        System.out.print("Attendee count: ");
        attendeeCounter.printCount();

        return;
    }
}

And then assignment statements.
public class AttendanceExample {
    public static void main(String[] args) {
        PeopleCounter attendeeCounter = new PeopleCounter();

        attendeeCounter.incrementCount();
        attendeeCounter.incrementCount();
        attendeeCounter.incrementCount();

        System.out.print("Attendee count: ");
        attendeeCounter.printCount();

        return;
    }
}

Method calls will also be discussed, in their various forms.
public class AttendanceExample {
    public static void main(String[] args) {
        PeopleCounter attendeeCounter = new PeopleCounter();

        attendeeCounter.incrementCount();
        attendeeCounter.incrementCount();
        attendeeCounter.incrementCount();

        System.out.print("Attendee count: ");
        attendeeCounter.printCount();

        return;
    }
}
public class AttendanceExample {
    public static void main(String[] args) {
        PeopleCounter attendeeCounter = new PeopleCounter();

        attendeeCounter.incrementCount();
        attendeeCounter.incrementCount();
        attendeeCounter.incrementCount();

        System.out.print("Attendee count: ");
        attendeeCounter.printCount();

        return;
    }
}

And eventually class definitions.