

### Announcements for This Lecture

| This Week   | Announcements   |
|---|---|
| <ul style="list-style-type: none"> <li>• Today is an <b>Interlude</b> <ul style="list-style-type: none"> <li>▪ <b>Nothing today is on exam</b></li> <li>▪ "Big Picture" lecture; may still be very helpful to you</li> </ul> </li> <li>• Thursday is <b>Recursion</b> <ul style="list-style-type: none"> <li>▪ Hardest topic in course</li> <li>▪ But will try to make it easy</li> <li>▪ Be there or be LOST</li> </ul> </li> <li>• Then the class gets easier...</li> </ul> | <ul style="list-style-type: none"> <li>• Assignment 1 Resubmissions                     <ul style="list-style-type: none"> <li>▪ 181 out of 195 have a 10</li> <li>▪ Today is absolute last day</li> </ul> </li> <li>• Assignment 2 is graded                     <ul style="list-style-type: none"> <li>▪ Solution posted in CMS</li> <li>▪ Mean 12.8/15; Median 13</li> </ul> </li> <li>• Assignment 3 due today                     <ul style="list-style-type: none"> <li>▪ Turn in by Midnight</li> <li>▪ Will be graded by weekend</li> </ul> </li> </ul> |

2/28/12 Object Oriented 1

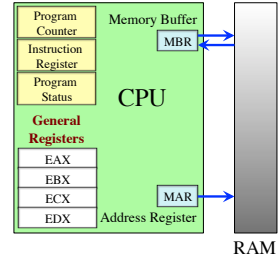
### A Short History of Programming: Assembly Language

|   |  |
|---|--|
| <pre>.data # start of data x: .long 1 .long 5 .long 2 .long 18  sum: .long 0  .text # start of code .globl _start</pre> | <pre>_start: movl \$4, %eax # EAX is counter                # for number of                # words to sum movl \$0, %ebx # EBX stores sum movl \$x, %ecx # ECX points to                # next item to sum  top: addl (%ecx), %ebx addl \$4, %ecx # move to next decl %eax    # decr. counter jnz top     # if counter not             # 0, then loop  done: movl %ebx, sum # done, store                # result in "sum"</pre> |
|---|--|

2/28/12 Object Oriented 2

### A Short History of Programming: Assembly Language

- Commands correspond to machine instructions
  - Close to machine language (no need to compile)
  - Programming "on the iron"
- Lot of load/store commands
  - movl from previous slide
  - **Registers** hold the data (like a variable)
  - Moving between RAM and registers is tedious
- Hard to write large programs



2/28/12 Object Oriented 3

### "High Level" Languages

- Language that is abstracted from the computer
  - Working with a **metaphor**, not **on the iron**
  - "Folders" are actually a **metaphor of a metaphor**
- Compiles (or translates) to assembly
  - And then down to pure machine language
  - One high-level line may be many lines of assembly
- Today there are many languages to choose from
  - We cannot agree on which metaphors are best
  - A lot of CS is **coming up with these metaphors**

2/28/12 Object Oriented 4

### A Short History of Programming: BASIC

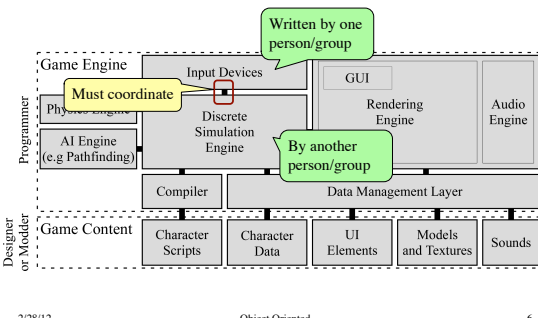
- **The Metaphor**
  - Commands have line numbers
  - Process the lines in order
  - GOTO, NEXT statements can jump forward or back
- Made programming easy!
  - On every computer in 80s
  - Primary hobbyist language
- But code was **monolithic**
  - Use a single number ordering
  - Code all fits in a single "file"
  - Large programs still hard

```

10 REM Sample BASIC Program
20 REM Counts To Ten
30 REM
40 PRINT "I am a BASIC program"
50 PRINT "that counts to ten."
60 PRINT
70 FOR I=1 TO 10
80 PRINT I
90 NEXT I
100 PRINT
110 PRINT "Thanks for running me."
120 END
```

2/28/12 Object Oriented 5

### Challenge: Breaking Up Software



2/28/12 Object Oriented 6

### A Short History of Programming: C (and its descendants)

```

int main() {
    int var1 = 1;
    int var2 = 2;
    printf("max(%d,%d)=%d",
        var1, var2,
        max(var1,var2));

    exit 0;
}

```

```

int max(int a, int b) {
    if (a > b) {
        return a;
    }
    return b;
}

```

2/28/12
Object Oriented
7

### A Short History of Programming: C (and its descendants)

- The Metaphor**
  - Organized via **functions** (with side-effects)
  - Procedures**: functions that return a value of type void
  - Function declarations can spread over multiple files
- Supports **modularity**
  - Each programmer works to define a function (or many)
  - Can *call* a function made by another programmer

C is like programming in Java, but only using static methods

```

int max(int a, int b) {
    if (a > b) {
        return a;
    }
    return b;
}

```

2/28/12
Object Oriented
8

### Encapsulation: Reducing Dependencies

- Development is iterative
  - You are always making changes (to improve your software)
- Coordination hurts iteration
  - Others are calling your functions
  - If you change your functions work, their code may no longer work
    - Example**: Our test code in A1
- Encapsulation**: limit what the other programmers can access in your code
  - If cannot access, changes are okay

2/28/12
Object Oriented
9

### Encapsulation is the Primary Purpose of Object Oriented Programming

- Encapsulation applies to both code and data!
  - Data in JFrame is hidden
  - Could you tell if it was changed in later versions?
- Encapsulation in Java
  - Make all data private
  - Force data access through the methods (getters/setters)
- Public methods: **the interface**
  - Not allowed to change the interface without permission

```

@105dc
fahrenheit 32.0 double
celsius 0.0 double
-----
Temperature(double)
setFahrenheit(double) getFahrenheit()
setCelsius(double) get Celsius()

@3cf92
fahrenheit 32.0 double
-----
Temperature(double)
setFahrenheit(double) getFahrenheit()
setCelsius(double) get Celsius()

```

2/28/12
Object Oriented
10

### Object Oriented Descendants of C

| Direct Descendants  | "Cousin" Languages  |
|---|---|
| <ul style="list-style-type: none"> <li><b>C++</b> <ul style="list-style-type: none"> <li>Optional OO features for C; a superset of C language</li> <li>High performance language</li> <li>But kludgy and messy</li> </ul> </li> <li><b>Objective-C</b> <ul style="list-style-type: none"> <li>Also superset of C language</li> <li>Trades performance for collaboration features</li> <li>Used in OS X, iPhone</li> </ul> </li> </ul> | <ul style="list-style-type: none"> <li><b>Java</b> <ul style="list-style-type: none"> <li>"Reimagining" of C++                             <ul style="list-style-type: none"> <li>Fixes a lot of the problems</li> <li>A little easier for beginners</li> </ul> </li> <li>Compile one, run anywhere                             <ul style="list-style-type: none"> <li>Ideal language for the web</li> <li>Others are platform specific</li> </ul> </li> </ul> </li> <li><b>C#</b> <ul style="list-style-type: none"> <li>"Reimagining" of Java</li> <li>Microsoft specific language</li> </ul> </li> </ul> |

2/28/12
Object Oriented
11

### A Short History of Programming: Is Python Best for Beginners?

- The Metaphor**
  - Mainly functions (like C)
  - Classes and objects exist, but are all optional
  - No distinction between the "interactions pane" and file
- An **untyped** language
  - Language checks the types for you automatically
  - Both a plus and a minus
- CS 1110 starting this Fall

```

def greet(name):
    print 'Hello', name

greet('Jack')
greet('Jill')
greet('Bob')

```

2/28/12
Object Oriented
12