Lecture topics:
I. Syllabus
II. Language processors
III. Why learn compilers
IV. Compiler phases

I. Syllabus

II. Language processors
II.1 Compiled languages (e.g., C, C++, Java)
- source program -> Compiler -> error messages
- input -> target program -> output

II.2 Interpreted languages (e.g., Python, JavaScript)
- source program -> Interpreter -> error messages
- input -> virtual machine -> output

II.3 Managed languages (e.g., Java, C#)
- source program -> Compiler -> error messages
- input -> virtual machine -> output

II.4 Tool chain (e.g.,)
- source program (main.c)
- modified source program
- Compactor -> error messages
- assembly program (main.s)
- Assembler (main.s) x64
- relocaatable machine code (main.o)
- Linker/Loader (main.o) -> target machine code (a.out)

III. Why learn compilers?
- understand language better
- power expressions useful in many domains
- possess useful for data forced
- project makes you better programmer
- compiled technology helps verify software
- compilers make new systems usable

IV. Compiler phases
- position = initial + rate * 60

IV.1 Lexical analyzer
- character stream -> Lexical analyzer
- token stream -> Syntax analyzer

IV.2 Syntax analyzer

IV.3 Semantic analyzer

IV.4 IR generator
- IR generator
- IR generator
- IR generator
- IR generator
- IR generator
- IR generator
- IR generator
- IR generator
- IR generator
- IR generator
- IR generator
- IR generator
## IV.7 Passes vs. phases

- pass = look at entire representation
- phase = conceptual component of compile

<table>
<thead>
<tr>
<th>phase</th>
<th>pass</th>
</tr>
</thead>
<tbody>
<tr>
<td>lexical analyzer</td>
<td></td>
</tr>
<tr>
<td>syntax analyzer</td>
<td></td>
</tr>
<tr>
<td>semantic analyzer</td>
<td></td>
</tr>
<tr>
<td>IR generator</td>
<td></td>
</tr>
<tr>
<td>Optimizer</td>
<td></td>
</tr>
<tr>
<td>Code generator</td>
<td></td>
</tr>
</tbody>
</table>

### Reminders
- Sign up for class mailing list
- hw1 due Fri 3/16 at 4pm
- read 2.1-2.5 for Wed 3/14