CSE443 Compilers

Dr. Carl Alphonce
alphonce@buffalo.edu
343 Davis Hall
OH: T 10:00-11:00, W 2:00-3:30

www.cse.buffalo.edu/faculty/alphonce/SP23/CSE443
Roadmap

- Syllabus: posted on website
- Course overview
- Course structure
What?

BUILD A COMPILER!
Why?

- Deeper understanding of languages
- Become a better programmer
- Learn how to build tools
- Build special-purpose languages (DSLs)
- Theory meets practice
- High-level meets low-level
How?

- That's the rest of the course!
Assessment plan

- **Homework (20%) - individual**
  - four - due dates on web site schedule
  - mix of final exam prep and project support

- **Project (60%) - team**
  - four milestones before final submission
  - team-based

- **Final Exam (20%) - individual**
  - during final exam period
  - sample questions give out the last week of classes
<table>
<thead>
<tr>
<th>Learning outcome</th>
<th>Instructional methods</th>
<th>Assessment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify and describe the function of the major phases of a compiler.</td>
<td>Lecture-based instruction</td>
<td>HW, EX</td>
</tr>
<tr>
<td>Define formally the grammars used in the front end of a compiler, their application in the front end, and techniques for parsing such grammars.</td>
<td>Hands-on lecture activities</td>
<td></td>
</tr>
<tr>
<td>Evaluate (compare and contrast) different intermediate representations.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Explain the compiler’s role in creating and managing run-time environments.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Explain and evaluate (compare and contrast) different approaches to code generation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identify and explain the applicability and operation of code optimizations.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Build both the front and back ends of a compiler.</td>
<td>Lecture-based instruction</td>
<td>PROJ</td>
</tr>
<tr>
<td>Collaborate effectively as a member of a software development team.</td>
<td>Team project w/team-faculty meetings</td>
<td>PROJ</td>
</tr>
</tbody>
</table>
Teams & Meetings

- Form teams no later than Tuesday next week (after add/drop)

- Teams must be of size 3 or 4. With 30 students, we are aiming for 8 teams (six of 4, two of 3)

- Teams must set up mandatory weekly ~30 minute meetings

- One member of each team must make a private post in Piazza with the UBIT and GitHub username of each person on their team.

- All code must be maintained in private git repo hosted on GitHub - I will help teams set this up at first meeting via GitHub Classroom; don't set this up on your own before then.
Goal: build a compiler

source program

executable
Phases of a compiler

source program

executable

Figure 1.6, page 5 of text
Phases of a compiler

source program

executable

Figure 1.6, page 5 of text
Setting the stage
Deep understanding - ex 1

identifier vs name vs variable
A variable location in memory refers to a variable. The name of the variable is represented in the program text, while its location is determined at runtime.
void foo(void) {
    int x = 0;
    printf(x);
}

void bar(void) {
    double x = 3.8;
    printf(x);
}
struct Pair {
    int x;
    int y;
};

void bar(void) {
    struct Pair r, s;
    /* ... */
}
Deep understanding - ex 1

```c
int f(int x) {
    if (x == 0) { return 1; }
    else { return x * f(x-1); }
}
```
identifier in distinct scopes
identifier in distinct record instances
identifier in recursive function invocations
Deep understanding - ex 2

order of evaluation

Does source code completely determine order of evaluation/execution at machine language level?
Deep understanding - ex 2

What is the order of evaluation?

\[ a + b \times c \]
Deep understanding - ex 2

\[ a + b \times c \]

What is the order of evaluation of the expressions?
Deep understanding - ex 2

\[ a + b \times c \]

How many expressions are there?
Deep understanding - ex 2

How many squares are there?
Deep understanding - ex 2

What is the order of evaluation?
Deep understanding - ex 2

\[ f(0) + f(0) \times f(0) \]

What is the order of evaluation?
In most languages the result will be consistent with the evaluation of 
\( a + (b \times c) \)
Deep understanding - ex 2

\[ a + b \times c \]

Order of operations is important here, but order of evaluation of the variable expressions a, b, and c is not (as long as they are evaluated before they are needed).
Deep understanding - ex 2

\[ a++ + a++ \times a++ \]

Order of operations is important, as is order of evaluation of the variable expressions if there are side effects!