CSE443
Compilers

Dr. Carl Alphonce
alphonce@buffalo.edu
343 Davis Hall
OH: TW 1:15-2:45

www.cse.buffalo. edu/faculty/alphonce/SP24/CSE443

Roadmap

- Syllabus: posted on website
- Course overview
- Course structure and assessment
- Capstone status of course

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

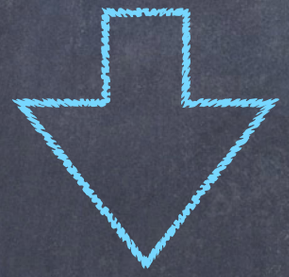
- Project (50%)
  - design and build a compiler
  - team-based
- Final Exam (20%)
  - during final exam period
  - sample questions give out the last week of classes
- Teamwork (20%)
  - four sprints
  - each team will have a project manager (PM)
- Presentation (10%)
  - each team will present/demo their compiler

Teams and PM Meetings

- Form teams as soon as possible, preferably no later than Tuesday next week (after add/drop)
- Teams must be of size 3 or 4, with all members in the same recitation (these will be the PM meeting times).
  - A1 has 12 students: three teams of 4
  - A2 has 9 students: three teams of 3
- 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 set these up via GitHub Classroom; don't set repos up on your own before then.

Goal: build a compiler

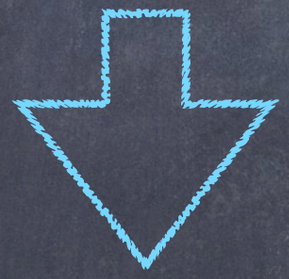
source program



executable

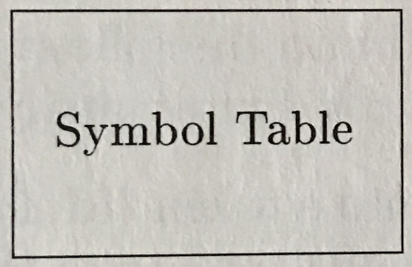
Phases of a compiler

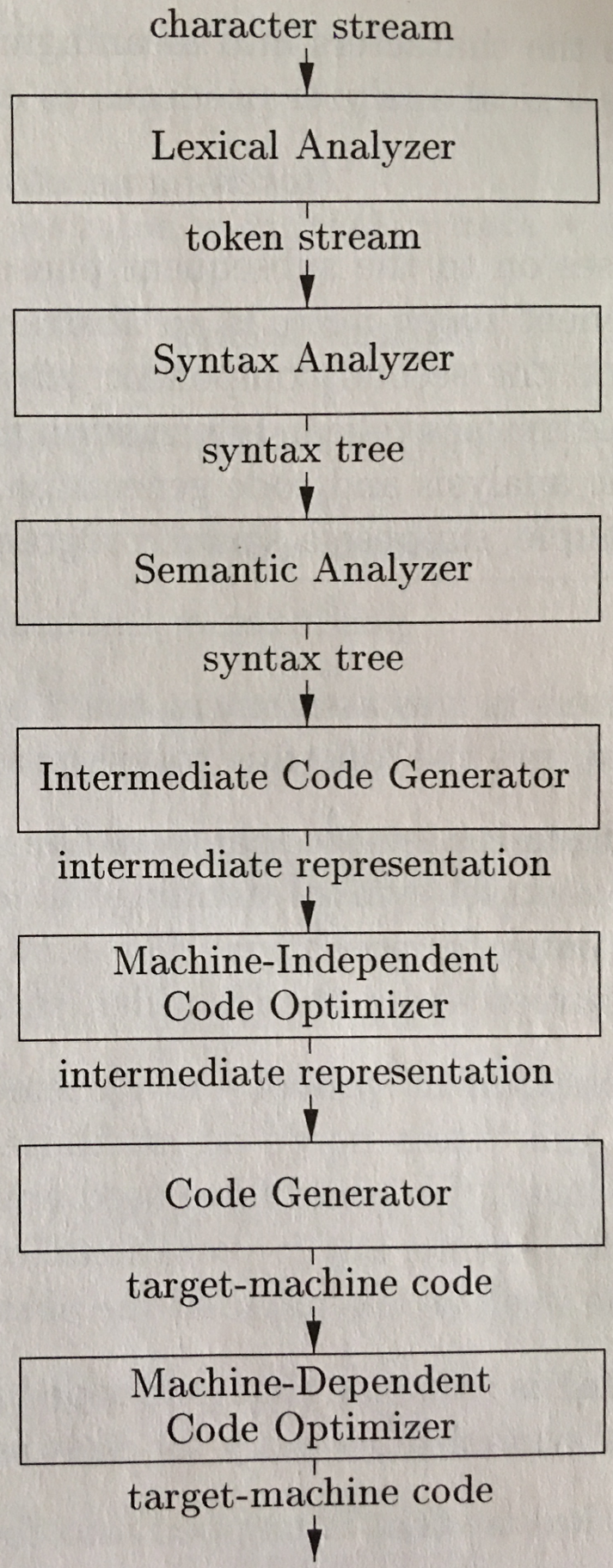
source program



executable

Figure 1.6, page 5 of text





Setting the stage

Deep understanding - ex 1

identifier

vs

name

vs

variable

name

y.x

identifier

x

refers to

variable

location

in

memory

static
(in program text)

dynamic
(at runtime)

Deep understanding - ex 1

```
void foo(void) {
    int x = 0;
    printf(x);
}
```

```
void bar(void) {
    double x = 3.8;
    printf(x);
}
```

Deep understanding - ex 1

```
struct Pair {
    int x;
    int y;
};

void bar(void) {
    struct Pair r, s;
    /* ... */
}
```

Deep understanding - ex 1

```
int f(int x) {
    if (x == 0) { return 1; }
    else { return x * f(x-1); }
}
```

CODE

RUNTIME

identifier

x

variable

variable

variable

variable

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

```
a + b * c
```

What is the order of evaluation?

Deep understanding - ex 2

```
a + b * c
```

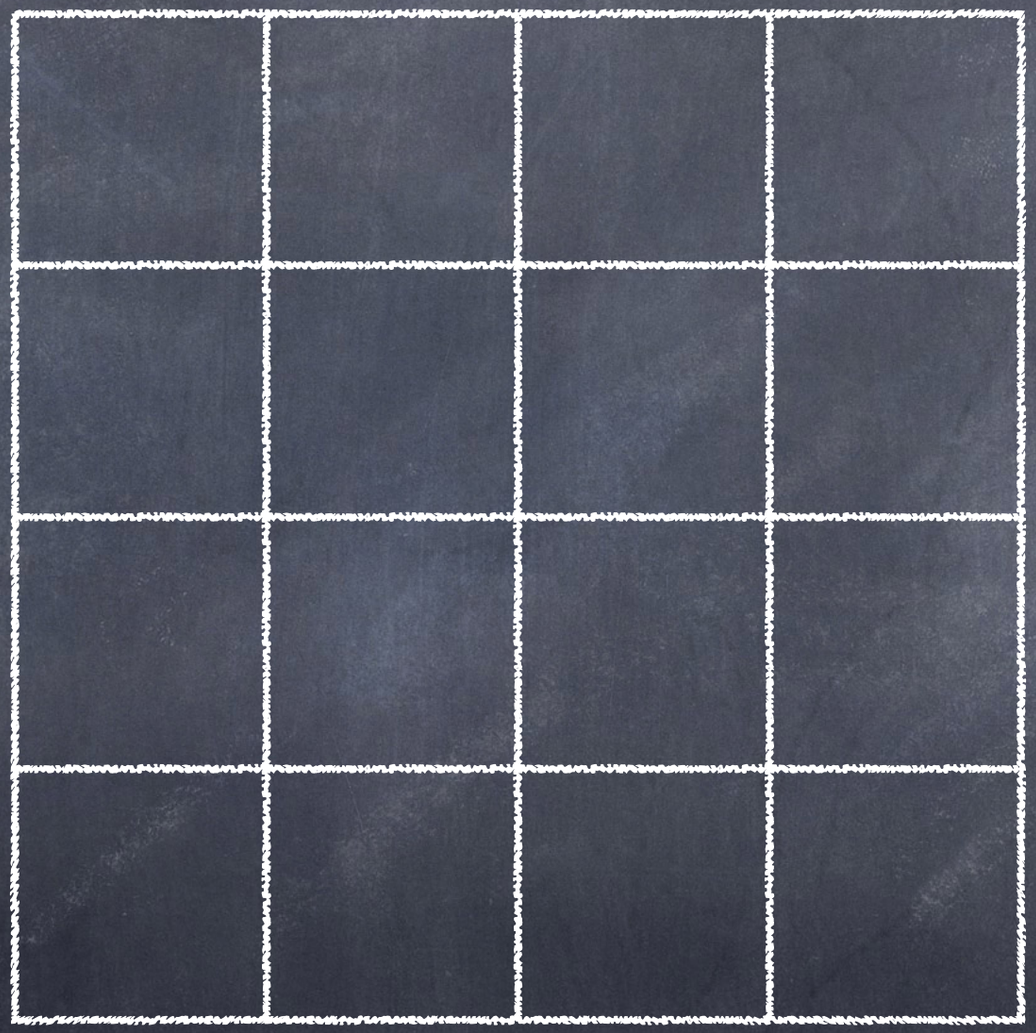
What is the order of
evaluation of the expressions?

Deep understanding - ex 2

```
a + b * c
```

How many expressions are there?

Deep understanding - ex 2



How many squares are there?

Deep understanding - ex 2

```
f() + g() * h()
```

What is the order of evaluation?