Lecture 2

CIS 341: COMPILERS
**HW1: Hellocaml**

- Homework 1 is available on the course web site.
  - Individual project – no groups
  - **Due:** *Wednesday, 24 Jan. 2018 at 11:59pm*
  - **Topic:** OCaml programming, an introduction

- OCaml head start on eniac:
  - Run “ocaml” from the command line to invoke the top-level loop
  - Run “ocamlbuild main.native” to run the compiler

- We recommend using:
  - Emacs/Vim + merlin
  - (less recommended: Eclipse with the OcaIDE plugin)
  - See the course web pages about the CIS341 tool chain to get started
How to represent programs as data structures.
How to write programs that process programs.
Consider this implementation of factorial in a hypothetical programming language:

```plaintext
x = 6;
ANS = 1;
whileNZ (x) {
    ANS = ANS * x;
    x = x + -1;
}
```

We need to describe the constructs of this hypothetical language

- **Syntax**: which sequences of characters count as a legal “program”?
- **Semantics**: what is the meaning (behavior) of a legal “program”?
Grammar for a Simple Language

• Concrete syntax (grammar) for a simple imperative language
  – Written in “Backus-Naur form”
  – <exp> and <cmd> are nonterminals
  – ‘::=’, ‘|’, and <…> symbols are part of the meta language
  – keywords, like ‘skip’ and ‘ifNZ’ and symbols, like ‘{’ and ‘+’ are part of the object language

• Need to represent the abstract syntax (i.e. hide the irrelevant of the concrete syntax)
• Implement the operational semantics (i.e. define the behavior, or meaning, of the program)
OCaml Demo

simple.ml
translate.ml