# Introduction (pg. 1)

Lecture topics

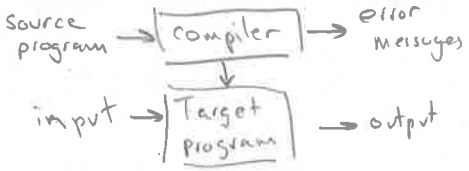
I. Syllabus and administrivia
II Language processors
III Why learn compilers
IV Compiler phases

## I. Syllabus

www.inf.usi.ch/faculty/soule/teaching/2015-fall/cc/index.html

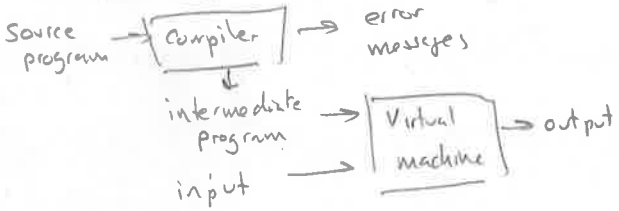
## II. Language processors

### II.1 Compiled languages (e.g. C, C++)

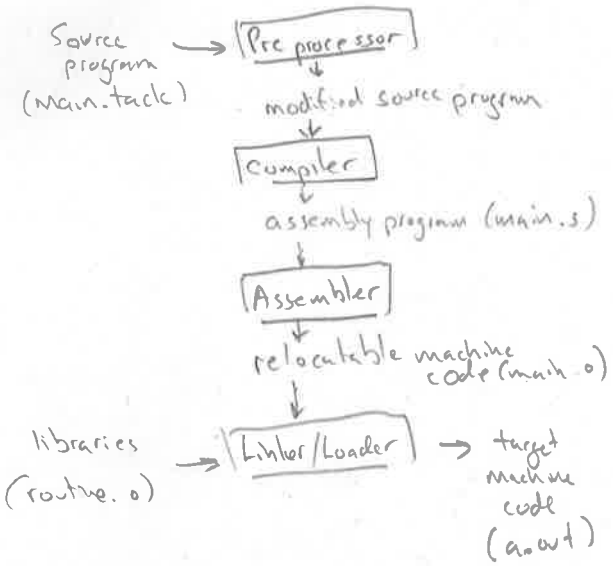



### II.2 Interpreted languages (eg. Python, Javascript)


### III.3 Managed languages (e.g. Java, C#)




### II.4 Tool chain (e.g., C)

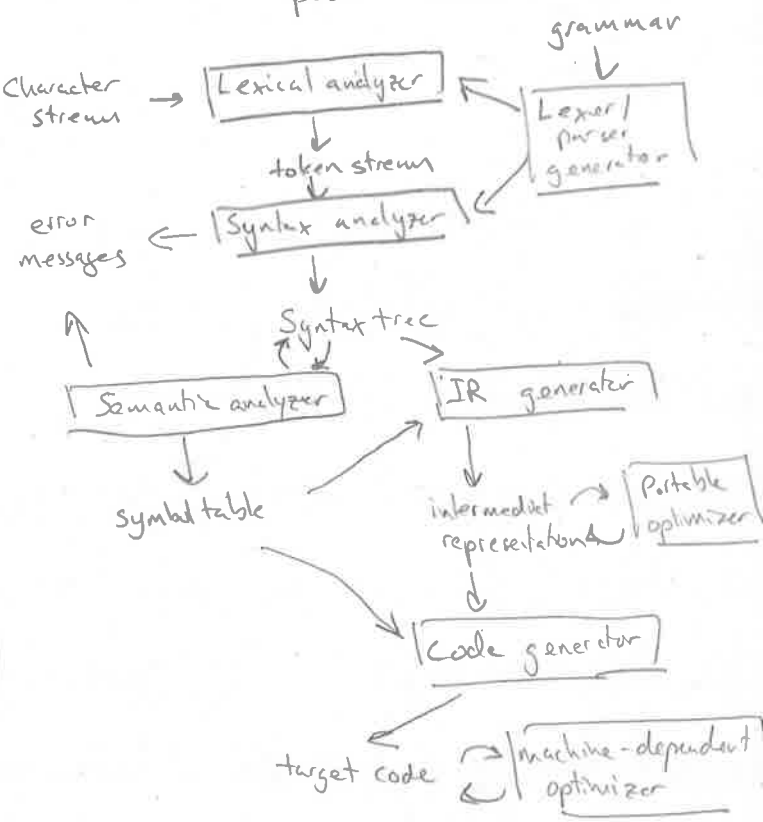



## III Why learn compilers?

- understand languages better
- regular expressions useful in many domains
- parsers useful for data formats
- project makes you better programmers
- compiler technology helps verify software
- compilers make new systems usable

## IV Compiler phases

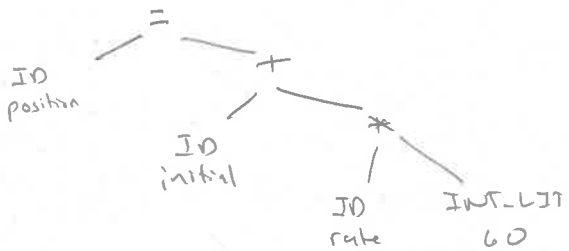
position = initial + rate * 60



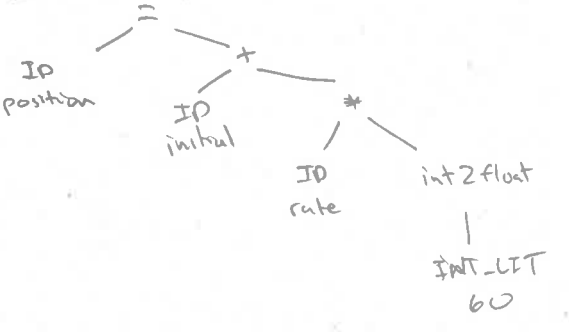

### IV.1 Lexical analyzer

ID(position), =, ID(initial), +, ID(rate), *, INT_LIT(60)

### IV.2 Syntax analyzer




### IV.3 Semantic analyzer



## IV.4 IR generator ("3-address code")

```
t0 = int2float(60)
t1 = rate * t0
t2 = initial + t1
position = t2
```

## IV.5 Optimizer

```
t1 = rate * 60.0
position = initial + t1
```

## IV.6 code generator

```
move  %rax, [%rbp +16]   /* offset(rate) == 16 */
mul   %rax, 60.0
move  %rcx, [%rbp +24]   /* offset(initial) == 24 */
add   %rcx, %rax
move  [%rbp+8], %rcx     /* offset(position) == 8 */
```

## IV.7 Passes vs. Phases

pass = in/out entire representation

phase = conceptual component of compiler

| Phase | Pass |
|---|---|
| lexical analyzer | Parser |
| syntax analyzer | Parser |
| | Normalizer |
| semantic analyzer | Scope analyzer |
| | type analyzer |
| IR generator | IR generator |
| Optimizer | ( / ) |
| code generator | code generator |