Class Announcements

- Third homework is due today. Will post sample solution tomorrow morning
- Midterm 1: Friday, February 24, in class, closed book and notes
  - Please be early so we can start on time. Exam is 70 minutes.
  - No bathroom breaks
  - No smart phones or smart watches
  - Bring your RU ID - you will need to sign out when turning in your exam
  - If you have a Letter of Accommodations, arrange exam location with Disability Services
- Office hours in CoRE 305 before exam:
  - Wednesday, 10:00am - noon, Zining Fan
  - Wednesday, 1:45pm - 3:15pm, Uli Kremer (no office hours on Friday)
  - Thursday, 1:00pm - 3:00pm, Chun Lau

Project1: tinyL Language

<program> ::= <stmtlist> .

<stmtlist> ::= <stmt> <morestmts>

<morestmts> ::= ; <stmtlist> |ϵ

<stmt> ::= <assign> |<print>

<assign> ::= <variable> = <expr>

<print> ::= # <variable>

<expr> ::= + <expr> <expr> |
- <expr> <expr> |
* <expr> <expr> |
/ <expr> <expr> |
<variable> |
<digit>

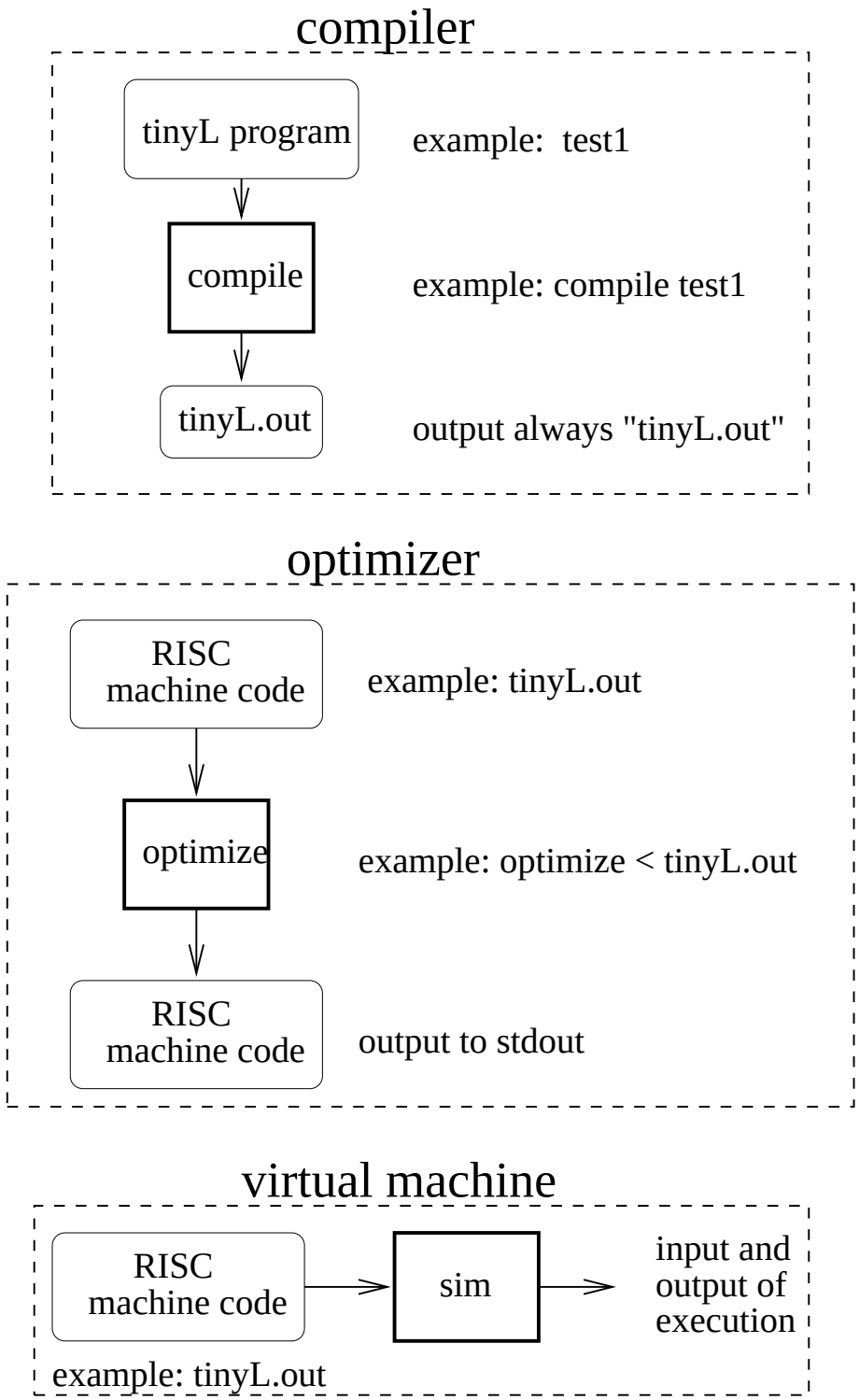
<variable> :: = a | b | c | d | e | f | g | h | i | j | k | s | t | u

<digit> :: = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Example program:

a=3;b=5;c=/3*ab;d=+c1;#d.

Project 1: Structure Overview



Project 1: Peephole Optimizations and Dead Code Elimination

Goal: To replace a sequence of instructions by another, more efficient sequence.

Peephole Optimizations - window size is 3 instructions or less.

Constant folding example:

```
loadI 5 => r4
loadI 7 => r5          can be replaced by    loadI 12 => r6
add r4, r5 => r6
```

Partial evaluation example:

```
loadI 1 => r7
mult r6, r7 => r8      can be replaced by    i2i r6 => r8
```

Strength reduction example:

```
loadI 4 => r7
mult r6, r7 => r8      can be replaced by    lshiftI r6, 2 => r8
```

Dead Code Elimination

```
loadI 5  => r4
loadI 7  => r5          can be replaced by    loadI 12 => r6
loadI 12 => r6
```

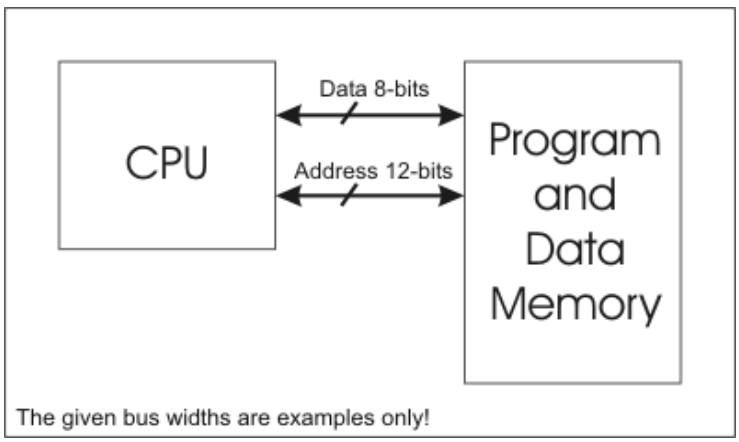
Replacement only valid if r4 and r5 are no longer used in the program. Note: Static single assignment code shape

Imperative Programming Languages

Imperative:

Sequence of state-changing actions.

- Manipulate an abstract machine with:
  1. Variables naming memory locations
  2. Arithmetic and logical operations
  3. Reference, evaluate, assign operations
  4. Explicit control flow statements
- Key operations: *Assignment* and "*Goto*"
- Fits the von Neumann architecture closely



Von Neumann Architecture

Next Lecture

Things to do:

First project will be posted after the first midterm exam.

Read Scott: Chap. 3.1 - 3.4; 8.3 ; ALSU Chap. 7.1 - 7.3

Next time:

- pointers and dynamic memory allocation
- Dynamic vs. static scoping
- Runtime environment
- access links and control links management