

Lecture 5+6

Assemblers

CS 241: Foundations of Sequential Programs
Winter 2018

Troy Vasiga et al
University of Waterloo

Overview

- ▶ Input to an assembler

- ▶ Output from an assembler

Stages to an assembler

A look inside the picture

Why two passes?

Defining assembly language

- ▶ informal description: see website

- ▶ formal description: later

Basic assembly language syntax and semantics

- ▶ syntax

- ▶ semantics

- ▶ location and labels

Analysis (First pass)

- ▶ must generate an intermediate representation
- ▶ must generate a symbol table
- ▶ tokenization

Intermediate Representation

A continuum

Symbol Table

- ▶ stores the address and name of each label definition

Synthesis (Second pass)

Encoding a beq instruction

Encoding a `.word <label>` instruction

Encoding `lis` or `jalr` or ...

Efficiency of the symbol table

Think of (key, value) pairs

Error checking: Pass 1

Error checking: Pass 2 (or earlier)

Error checking: Philosophy

- ▶ this will be true for the rest of this course

A little bit about bits

You have to output bits, not ASCII 0 and 1.

See the example.

A gift

`asm.rkt` or `asm.c` or `asm.cc`

How to write an assembler

Slowly.

Carefully Test. Test. Test. Test. Test. Test. Test. Test. Test. Test.

Test.

Correctness

- ▶ **Read the spec carefully!**
- ▶ think reasonably and unreasonably
- ▶ remember memory: $O(n)$
- ▶ running times do matter: $O(n)$
- ▶ **Don't use Marmoset as your only testing tool!**