Lecture 6
Assemblers take 2

CS 241: Foundations of Sequential Programs
Spring 2017

Troy Vasiga et al. (where Dan Holtby ∈ al)
University of Waterloo
Analysis (First pass)

Output:

- an intermediate representation
- a symbol table
Intermediate Representation

A continuum

- Could be very close to assembly: Assembly text split by instruction instead of by line
- Could be very close to machine code: Binary with label placeholders.
- Could be anywhere in between.
Symbol Table

- stores the address and name of each label definition (dictionary mapping identifiers to addresses)
Synthesis (Second pass)

- Intermediate Representation → Machine Code
- Just like A1 (only now you’re telling a computer how to do A1)
Encoding a `.word <integer>` instruction

There's nothing to do, just output the bits (make sure you get them in the right order!)
Encoding a `.word <label>` instruction

Just look the label up in the symbol table! Now it's an integer (see previous slide)
Encoding a beq instruction

See examples on website if the offset is an integer.

If branching to a label, need to figure out number of inst. between here and label
Efficiency of the symbol table

Think of (key, value) pairs. Hash Maps! Racket has them, C++ has them.

Constant Amortized Time

Racket: (make-hash) or (make-hasheq) or (make-hasheqv)
C++14: unordered_map < string, uint32_t >

See documentation! (Racket Help Desk, cplusplus.com)
Error checking: Pass 1

- Syntax errors (forgetting commas, etc.)
- Duplicate labels
Error checking: Pass 2 (or earlier)

Semantics errors (can catch some in pass 1, but some only after pass 1 is finished)
Error checking: Philosophy

- Think positive
- Consider all the ways to be right (usually only a few) not all the ways to be wrong (there are many)
A little bit about bits

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

Oops, you can’t, you have to output bytes. That’s OK, 32 is a multiple of 8.

MIPS expects “Big Endian” so you should output the most significant byte first.

E.g. if outputting the word 42 you should output bytes 0, 0, 0, then 42. Just like A1!

// C++
cout << byte; // but be careful of types!
// or
putchar(byte); // from <cstdio>
;; Racket
(write-byte byte)
A byte about bits

How do you split a word into bytes? Remainders and quotients. Instead of modulo 256, you can just use bitwise and

// C++
word & 0xFF; // least significant byte
;; Racket
(bitwise-and word #xFF) ; ditto

Instead of dividing by 256, you can use a right shift

// C++
word >> 8; // (word / 256)
;; Racket
(bitwise-arithmetic-shift-right word 8) ; ditto
How do you “fill in the blanks” for the “add” instruction? Start with the pattern on the reference sheet.

If $s$ is $3 (0 0011)_b$ then you need to move it 21 digits over. Multiply by $2^{21}$. Or, shift left by 21.

\[
\text{add : 0000 00ss ssst tttt dddd d000 0010 0000}
\]
\[
\text{s : 0000 0000 0000 0000 0000 0000 0000 0011}
\]
\[
\text{s \ll 21 : 0000 0000 0110 0000 0000 0000 0000 0000}
\]

To combine with the “add” pattern, can use $+$ (but bitwise or is faster)

// C++
pattern | (s \ll 21)

;; Racket
(bitwise-ior pattern (bitwise-arithmetic-shift-left s 21))
A gift

asm.rkt or asm.zip

It’s a scanner, and a sample program to demonstrate how to use it.
How to write an assembler

Slowly.


Test.
Correctness

- Read the spec carefully!

- think reasonably and unreasonably (error checking should only consider how to be right, but your tests should consider all the ways to be wrong!)

- remember memory: $O(n)$

- running times do matter: $O(n)$

- Don’t use Marmoset as your only testing tool!