EE126 Lab7, Fall17

Michelle Chan and Ashton Stephens, Tufts University

December 11, 2017

Introduction

Background

Specifications

This MIPS-32 assembler translates assembly code into machine code in binary format. The program reads from a file specified as a command line argument or from stdin if a file is absent.

This assembler recognizes all 32 MIPS-32 registers and labels. A list of supported instructions, including pseudo-instructions, are shown in Table X.

Implementation

On the surface, the assembler conducts two passes where the first pass handles labels and the second pass translates each assembly line of code.

Tables

There were three tables used to lookup information used in the implementation.

1. The mneumonic table mapped a mneumonic to a corresponding opcode.
2. The register table mapped the register name to its number in memory.
3. The pseudoinstruction table mapped a pseudoinstruction to the instructions it expanded into along with the order to pass in registers.

First Pass

Each line of assembly code in the program is read line-by-line and stored internally to be read again from the second pass. At each line, the first word is checked for a label and the word address is incremented the appropriate amount. Labels are stored as the key to its word address value. The word address that is tracked always increments by one on a MIPS instruction because there exists the invariant that each instruction is one word long. The first word is also checked against the internal pseudoinstruction map to increment more than one word forward.

Second Pass

In the second pass, the mneumonic and operands of each line are parsed into strings. Using the current word address and parsed strings, the assemble method is called to translate into machine code.

Internally, the translation looks up the mneumonic to get either an Instruction type or Pseudoinstruction type internally defined as structs containing all the necessary information for assembling. In "struct Mnemonic_func, " the information stored includes the opcode (or funct field), the instruction format (R-type, I-type, or J-type), and the syntax group. Each syntax group referred to a unique order in which the operands for an assembly instruction were translated and placed into the machine instruction.

In the case that the mneumonic maps to a pseudoinstruction, is returned. This type contains four vectors each with length equal to the total number of instructions after expanding the pseudoinstruction. One vector holds the mneumonics of the actual instructions while the other three hold the order

After the translation, the number of words to increment the program counter is also added into the word address.

Results

Conclusion