COMP 7300: Lab Assignment-3

Assembly Language to Machine Language

Samir Hasan (szh0064@auburn.edu)
Steffi Mariya Gnanaprakasa Paul Arasu
(smg0033@tigermail.auburn.edu)

Group ID: 25

November 26, 2013

Introduction

In this project, we wrote a Java program to translate MIPS assembly instructions to machine code. The input to our program is a text file with assembly code (.asm file), which are translated into their corresponding 32-bit instruction format and written out in a binary file. We covered a subset of the instruction set: lw, sw, add, sub, slt, and, or, beq and j.

Instruction formats

The encodings of the three instruction types are shown below.

R-format

opcode (6 bits)	rs (5 bits)	rt (5 bits)	rd (5 bits)	shamt (5 bits)	function (6 bits)
31-26	25-21	20-16	15-11	10-6	5-0

The R format instructions are:

- `add $rd,$rs,$rt`
- `sub $rd,$rs,$rt`
- `slt $rd,$rs,$rt`
- `and $rd,$rs,$rt`
- `or  $rd,$rs,$rt`

I-format

opcode (6 bits)	rs (5 bits)	rt (5 bits)	constant or address (16 bits)
31-26	25-21	20-16	15-0

The I format instructions we support are:

- `lw  $rt,offset($rs)`
- `sw  $rt,offset($rs)`
- `beq $rt,$rs, offset`

J-format

opcode (6 bits)	address (26 bits)
31-26	25-0

We support a single kind of jump instruction.

- `j address`

Running the Assembler

The assembler is a Java application. It runs on JDK 1.7 or 1.6. All the files are in a single folder named Assembler. The main class is in **Assembler.java**. Assembler.java expects the input .asm file name to be provided as a command line parameter. The output is always a set of two files: **out.bin**, the binary object file, and **out.hex**, a textfile containing one-instruction-per-line hexadecimal representation of the binary instructions.

Running the code

1. In the terminal (command prompt) window, type: **cd Assembler/src**
2. Compile the java files: **javac *.java**
3. Run the Program class: **java Assembler input.asm**
4. Two output files will be produced: *out.bin* and *out.hex*. The out.hex file can be opened using any text editor.

Inside the code

Assembler The main entry point for the assembler. It handles the file input, uses internal assembling modules to generate the binary instructions code, and writes the output to a binary and hex file.

Instruction An abstract class that defines the interface and basic functions for the instructions.

RTypeInstruction Extends Instruction, and implements the abstract method encode and decode, and defines instruction-specific properties for an R-type instruction.

ITypeInstruction Extends Instruction, and implements the abstract method encode and decode, and defines instruction-specific properties for an I-type instruction.

JTypeInstruction Extends Instruction, and implements the required methods for handling a J-type instruction.

Mappings Stores string-number mappings that instruction name has with opcode and function code.