UMEÅ UNIVERSITET **Wednesday 14th March, 2018**

Assignments

Computer Architecture, 5DV193

Assignment 1

Name	**Username**	**E-mail**
Kristen Viguier	ens17kvr	`ens17kvr@cs.umu.se`

Graders
Thomas Johansson

1 Introduction

An assembler is a program that converts instructions in mnemonic form (e.g. add $1, $2, $3) into encoded processor instructions. An assembler also usually produces an assembly listing with information about the placement of instructions in memory. The Task of this assignment was to write an assembler for a subset of the MIPS instruction set. The program is written in Java, and is able to read one source code file with mnemonic instructions and produce two files, one with encoded instructions in hexadecimal format, and one list file

Contents

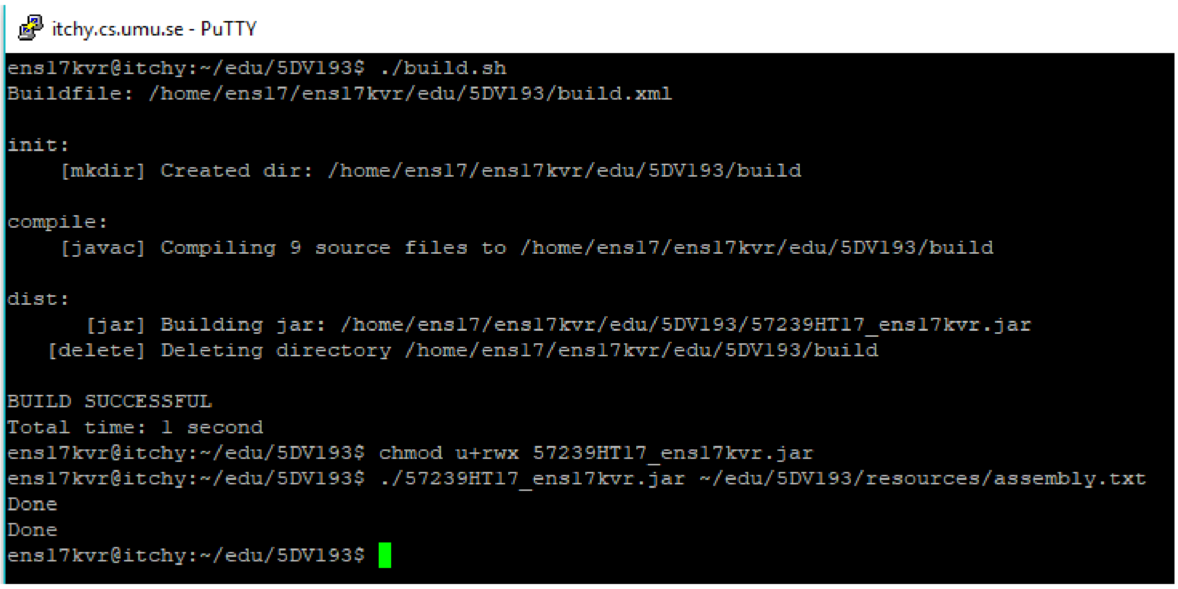
2 Compiling

As my application was written in Java you need to have the Java plateform installed to compile/start my application. I used ant to build the project. I create a build.sh that start the ant building process. When it is done you will find the $57239HT17\_ens17kvr.jar$ file. You need to start the file jar and give the file argument to process. The file to test is present inside the resources folder called assembly.txt. When the program is done you will see appear two new files inside the resources folder called outputFull and outputSmall.

Here is an example :

- cd ComputerArchitectureAssignment1/
- ls -l
  - $57239HT17\_ens17kvr.jar$
  - bin/
  - *build.sh*
  - *build.xml*
  - *lib/*
  - *resources*
  - *src*
- ./build.sh or type ant
- the jar is generated
- start the jar with : **java -jar** $57239HT17\_ens17kvr.jar$ **resources/assembly.txt**

You will see appear two Done in the console.



```
ensl7kvr@itchy:~/edu/5DV193$ ./build.sh
Buildfile: /home/ensl7/ensl7kvr/edu/5DV193/build.xml

init:
    [mkdir] Created dir: /home/ensl7/ensl7kvr/edu/5DV193/build

compile:
    [javac] Compiling 9 source files to /home/ensl7/ensl7kvr/edu/5DV193/build

dist:
      [jar] Building jar: /home/ensl7/ensl7kvr/edu/5DV193/57239HT17_ensl7kvr.jar
   [delete] Deleting directory /home/ensl7/ensl7kvr/edu/5DV193/build

BUILD SUCCESSFUL
Total time: 1 second
ensl7kvr@itchy:~/edu/5DV193$ chmod u+rwx 57239HT17_ensl7kvr.jar
ensl7kvr@itchy:~/edu/5DV193$ ./57239HT17_ensl7kvr.jar ~/edu/5DV193/resources/assembly.txt
Done
Done
ensl7kvr@itchy:~/edu/5DV193$
```

Figure 1: Screen shot on itchy.cs.umu.server with putty SSH connection compiling and testing

3 Running

This assembler program supports a subset of the MIPS instruction set *add*, *sub*, *and*, *or*, *nor*, *slt*, *lw*, *sw*, *beq*, *addi*, *sll*, *j*, *jr* and *nop*.

The program assume you entered a correct file path as argument. This file path is the input file that contains several line of MIPS instruction. Every line in the input can be :

- An empty line
- A comment line, which should begin with a number sign (#) in the first position of the comment
- A line with only a label on it - the label must begin in position 1 and end with a colon (:).
- An instruction statement which has several parts, all separated by one ore more white-space characters (spaces, tabs):
  - A MIPS instruction, that must be preceded by white-space, regardless of whether there is a label or not before it.
  - Zero or more operands to the instruction, separated by commas and possibly also white-space.
  - An optional comment that should begin with a number sign

The program handles errors. If there is an unknown instruction or something wrong or unexpected there is a console line written and/or a message write inside the output file.

4 User guide

The project is organized like this :

- Model
- Controller
- View

Inside the view, you will find the class Main.java. This is the main class that start the assembler program. This class needs as parameter the file path to test. You can notice that I created a resources directory. Inside there are the assembly.txt, that contains the assembly lines code that you used for the assignment 1, and also the two produced file. One contains only encoded memory addresses and instructions (called ouputSmall) and the other is the full one with every data needed (outputFull).