UMEÅ UNIVERSITY **December 5, 2013**
Department of computer science
Lab report

Lab 1

Computer Organization and Architecture 7.5hp

Mipps32 disassembler

Name	Robin Lundberg
E-mail	`ens10rlg@cs.umu.se`

Supervisor
Stephen J. Hegner, Thomas Johansson

1 Problem Specification

In the MIPS32 architecture, all machine code instructions are represented as 32-bit instructions. The goal of this lab is to write a *disassembler* in *C* that turns these machine code instructions into MIPS32 assembly language. The program should read a file as input that contains the instructions in decimal or hexadecimal—one in each row—for the MIPS32-program. For every line in the machine code file; the disassembler should output a line that contains the following:

- the original instruction.
- the type of instruction.
- the values for relevant fields for that type of instruction in both decimal and hexadecimal.
- and the mnemonic representation.

The disassembler should handle all instructions, except floating point instructions. If the instruction is not valid the program should output relevant information for that line.

2 Usermanual

To compile the disassembler, run `make dismips`. Run the disassembler by executing the command `dismips file`. Where `file` is the path to the file containing the MIPS32 machine code to be disassembled. The disassembler supports all instructions except those with op-code 16, 17 and 18 except for the commands `bclf`, `bclt`, `mfc1` and `mtc1` which are included.

The disassembler outputs to *standard out* the following for each line of machine code:

1. the original instruction.
2. the type of instruction.
3. the values for relevant fields for that type of instruction in decimal.
4. the values for relevant fields for that type of instruction in hexadecimal.
5. and the mnemonic representation.

in the format `1; 2; 3; 4; 5;` where the numbers corresponds to the information in the list above. The disassembler will output error messages that corresponds to whatever type of error is in the machine code e.g. if you try to use a floating point instruction it will tell you it is not allowed or if you use an invalid op- or function-code it will also tell you that—and then continue with the next line of machine code.