Computer Organization and Architecture

Project 1

Prashant Kumar

Algorithm:

- Read the MIPS code
- Arrange the MIPS code in a vector of vector of instructions
- Record labels and label line numbers
- Re read the MIPS code, now in the vector. When bne or beq statements come, calculate the relative distance of labels using the recorded label line numbers.
- Implement a table to get instruction type (R or I) using the opcode.
- Read MIPS instruction to rs, rt, rd, shamt and funct according to R or I type instruction. Implement special handling of sll and srl for R type and beq and bne for I type.
- Based on R and I type, handle the read information in separate handling functions
- Implement a table to get decimal values of registers
- Implement a function to convert decimal values to binary numbers
- Implement a table to get funct value based on opcode for R type instruction
- Implement functions to convert hex to binary and binary to hex value
- Handle negative immediate values by implementing 2's complement and sign extension of binary number
- Since switch statements do not take strings as inputs, implement an enum for opcode strings

Note: I haven't used any libraries for handling binary or hex values. I am using double numbers for binary values and handling their computations manually.

Error handling: I have built a try and catch statement for exception handling. If I notice an error in MIPS code, I throw an exception when then outputs the current line number and MIPS code.

Whenever required, I have put comments in the program. Variable names have been chosen to self-explain themselves.

Code has been implemented on Visual Studio Community 2017.

Way to run program: In command prompt go to folder in which "myAssembler.exe" file is and write:
myAssembler filename
for a MIPS code file with "filename.s" name. Output will be created in the same folder with name "filename.obj".