Computer Organization and Assembly   
Language Programming

Project 1

Report

Presented to: Dr. Tamer ElBatt

TA: Suzanne Safwat

Made by:

Ali ElSaid  
Habiba Gamal

Objectives:

* Design an assembler to RISC-V RV32
* Output RISC-V RV32 machine code
* Execute instructions
* Bonus: output MIPS machine code given RISC-V RV32 machine code

Design:

* Assembler:
* Function: parse: This function takes the RISC-V RV32 assembly instruction as a string and divides it into its different fields, then accordingly concatenates the values to get the machine code
* Execute instructions:
* Function: instAssembleExec: This function checks the opcode to determine which type of instruction, R, I, UJ, J, SB, S, and accordingly call the equivalent function that executes the instruction.
* For the specific type functions, they check for function 3 and function 7 to recognize the particular instruction and execute it accordingly
* Output:
* Done in Files.
* Bonus:
* A new project was created.
* Program inputs a file which contains risc machine code which is inputted as a string and then converted to hexadecimal
* The hexadecimal machine code is parsed into its different fields in the struct for risk words
* A struct for mips words is created
* Function: mars: It recognizes the type of the instruction and accordingly call the respective function.
* The specific type functions recognize the specific instruction and accordingly convert it to MIPS by filling in the different fields and concatenating the machine code.
* The machine code is again outputted in a file.

How to Use:

* Program:
* Type the risc assembly instructions into a text file and let it be the input to the program
* The output will be in another file: the equivalent risc machine code
* The registers are outputted through regular output stream
* Bonus:
* Type the risc machine code into a text file and make it the input to the program
* The output will be in another file: the equivalent mips machine code

Difficulties Faced:

* Learning the differences between risc instructions and mips instructions
* Learning how to handle strings professionally to manipulate its contents
* Converting some of the risc true instructions to mips true instructions

Limitations:

* Interface
* Registers out of range lead to errors