North South University

Department of Electrical & Computer Engineering

Course Code: CSE332

Course Title: Computer Organization and Architecture

Submitted to: Dr. Mainul Hossain (MHo1)

Design a 10-bit Custom RISC-V Microprocessor

ASSEMBLER DESIGN

Section: 10

Group Number: 4

Submitted by:

Serial No	Student Name	ID
1	Foyez Ahmed	2013122042
2	Nusrat Zahan	2013942642
3	Reem Gazi Mahmud Hossain	2014234042
4	Syed Niamul Kazbe Rayian	2021931642

Introduction:

Our task was to design an assembler which will convert the assembly code to machine language.

Objective:

Our objective was to generate a machine code from a file (.txt) containing assembly language. The assembler reads a program written in an assembly language, then translate it into binary code and generate output file (.txt) containing machine code.

Instruction to use:

In the input file (input.txt) the user has to give some instructions to convert into machine codes. The system will convert valid instructions into machine language and generate those codes into output file (output.txt)

List of Registers:

We allocated 2 bits for $rt registers, 1 bit for $rs registers and 4 bits for $rd registers.

Register Table

Register Name	Type	Register Number	Binary Value
$t0	Saves values	0	0000
$t1	Saves values	1	0001
$t2	Saves values	2	0010
$t3	Saves values	3	0011
$t4	Saves values	4	0100

$t5	Saves values	5	0101
$t6	Saves values	6	0110
$t7	Saves values	7	0111
$t8	Saves values	8	1000
$t9	Saves values	9	1001

List of Op-Code:

We selected the following op-codes and assigned functionality values (3-bits) for each op-code.

Name	Opcode binary	Type
add	000	R
sub	001	R
lw	010	I
sw	011	I
sll	100	I

Instruction Description:

Addition: It adds values of two temporary registers and stores the result in a destination register.

ADD

- Operation: d = s + t
- Syntax: add $rd, $rs, $rt

SUB

- Operation: d = s - t

- Syntax: sub $rd, $rs, $rt

Load Word: It loads required value from the memory to the register for calculation.

lw

- Operation: d = M[s + t]
- Syntax: lw $rd, $rs, imm

Store Word: It stores specific value from register to memory.

sw

- Operation: M[s + t] = d
- Syntax: sw $rd, $rs, imm

Manual:

In order to perform operations in the Logisim, the hexadecimal value for the instruction fetch unit needs to be converted from the input file. Inside the input file, there are instructions written such as, add $rd, $rs, $rt, this instruction is then converted into binary and then to hexadecimal by using C++ assembler language. The input file is formatted as a text file (input.txt), and output file is formatted as a text file (output.txt). Hence, both the input and output file has been included in the project folder. If one wants to try their own assembly code then, that person needs to write the codes to the application through the input file. There can be more than one instruction in the input file.