

North South University

*Project title: Design a 12-bit Custom RISC-V Microprocessor*

Assembler Design

Course: CSE 332 - Computer Organization & Design

Section – 06

Submitted to:

Submitted by: Group Members

|  |  |
| --- | --- |
| Name | ID |
| Kazi Aniya Ahmed | 2011775642 |
| Sheikh Rizwan Ahmed | 2012174642 |
| Faisal Ahammed Tareq | 1931529642 |
| Asif Mahmud | 1912947642 |

**Introduction:**

Our task is to design an assembler which will convert the assembly code to the machine language. As computer only understands machine code (0 and 1).

**Objective:**

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

**How to use:**

The user will assign some instructions to convert assembly language into machine language. The instructions will be written into a .txt file. From there we will create a java project to write some codes which will read the .txt file where all the assembly instruction are written and will generate a output into binary code.

**Input file:**

The input file is located in a folder named “File”. User will write down the assembly code in this file.

**Register list:**

As we have allocated 3 bits for each registers so total number of register will be 2^3=8. And all of them will be store type. We will use register R0-R7.

Register table:

|  |  |  |
| --- | --- | --- |
| Name of the registers | Register number | Value assigned(3 bits) |
| R0 | 0 | 000 |
| R1 | 1 | 001 |
| R2 | 2 | 010 |
| R3 | 3 | 011 |
| R4 | 4 | 100 |
| R5 | 5 | 101 |
| R6 | 6 | 110 |
| R7 | 7 | 111 |

OP-code List:

|  |  |  |
| --- | --- | --- |
| Op-code name | Type | Op-code binary |
| sub | R-type | 000 |
| add | R-type | 001 |
| addi | I-type | 010 |
| beq | I-type | 011 |
| sll | R-type | 100 |
| sw | I-type | 101 |
| jump | J-type | 110 |
| lw | I-type | 111 |

**Instruction description:**

**Add**: It adds two registers and stores the result in destination register.

• Operation: $rd = $rs+$rt

• Syntax: add $rs, $rt

**Sub**: It subtracts two registers and stores the result in destination register.

• Operation: $rd = $rs-$rt

• Syntax: sub $rs, $rt

**addi**: It adds a value from register with an integer value and stores the result in destination register.

• Operation: $rd = $rs+constant

• Syntax: addi $rs, constant

**lw:** It loads required value from the memory and write it back into the register.

• Operation: d = M [s + offset]

• Syntax: lw $rs, offset

**sw:** It stores specific value from register to memory.

# • Operation: M[d + offset] = s

• Syntax: sw $rs, offset

**Sll:** : It shifts bits to the left and fill the empty bits with zeros. The shift amount is depended on the offset value.

• Operation: $rd= $rs << offset

• Syntax: sll $rs, offset

**beq:** It checks whether the values of two registers are same or not. If it’s same it performs the operation located in the address at offset value.

• Operation: if ($rs==R0)

jump to offset

else

goto next line

• Syntax: beq $rs, offset

**Jump:** Jumps to the calculated address.

• Operation: PC = nPC

• Syntax: jmp target

**Limitation:**

This is a project for CSE332: Computer Organization and Architecture. The goal of this project is to design a custom RISC-V Microprocessor. For this we are going to need and assembler which will convert all the assembly language to machine language. We designed an application for assembler in Java. This is a console based application with no GUI. The whole program has been written in Java and no external library or frameworks were used. The entire program is written from scratch.