

ASSEMBLER DOCUMENTATION

Group members

|  |  |
| --- | --- |
| name | ID |
| Mahbuba Tasmin | 1610064042 |
| Fatma harun | 1610592042 |
| umme habiba Maliha | 1612150042 |
| Rubaith Adnan | 1510538642 |

Submitted to –

Dr.Tanzilur Rahman

CSE 332 Section 1

Spring 2018

## Introduction

To design an assembler which will convert the assembly code (MIPS ) to machine language.

## objective

To generate machine language code from a text file containing assembly language instructions. This program reads a file written in assembly language, converts it to binary code and then produces the output in hexadecimal format.

## HOW TO USE

There has to be a file where the input instructions are stored and this will be the file the assembler will work on. There has to be set of text files of instructions of various formats depending on the ISA design.

All of these files to be used in the program has to be stored in the same directory as the program is in.

## LIST OF TABLES

* R-TYPE OP CODE LIST
* I-TYPE OP CODE LIST
* REGISTER TABLE

R type format

As per the ISA design we prepared, we have assigned 3 bit for each op code and for R-type format, we have assigned functionality values (1 bit). The following op codes were selected for R type format.

|  |  |  |
| --- | --- | --- |
| OPERATION NAME | OP CODE | Functionality |
| ADD | 000 | 0 |
| SUB | 000 | 1 |

I type format

The following op codes have been selected for I type formats. In addition to the standard op codes, we have included a new instruction named “disp” to facilitate the display functionality of the processor.

|  |  |
| --- | --- |
| OPERATION NAME | OP CODE |
| SLL | 001 |
| LW | 010 |
| SW | 011 |
| ADDi | 100 |
| SLT | 101 |
| J | 110 |
| Disp | 111 |

Register design

We have assigned 2 bits for the register, hence we have 4 registers as listed below.

|  |  |  |
| --- | --- | --- |
| Name of the Register | Register Number | Value Assigned(2 bits) |
| $ACC | 0 | 00 |
| $s1 | 1 | 01 |
| $s2 | 2 | 10 |
| $t | 3 | 11 |

## lIST OF INSTRUCTION

1.add: It adds two registers and stores the result in destination register.

Operation: $rd = $rs + $rd

Syntax: add $rd, $rs

2.sub: It subtracts two registers and stores the result in destination register.

Operation: $rd = $rd - $rs

Syntax: sub $rd, $rs

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

Operation: $rd 🡨 Mem [[$ACC] + offset]

Syntax: lw $rd, offset

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

Operation: $ACC = $ACC -$ACC

$ACC = Base + $ACC

$rd -- Mem [[$ACC] + offset]

Syntax: sw $rs, offset

5. ADdi: It adds an immediate value to the destination register.

Operation: $rd = offset

Syntax: addi $rd,offset

6. SLL: Shifts a register value left by the shift amount listed in the instruction and places the result in the destination register. Zeroes are shifted in.

Operation: $rd = $rd << immediate ;

Syntax: sll $rd, immediate

7.SLt: If $rd is less than immediate value, $ACC is set to one. It gets zero otherwise.

Operation : if $rd < immediate $ACC = 1;  else $ACC = 0;

Syntax: slt $rd, immediate

8. J: Jumps to the calculated address

Syntax: j immediate

Disp

Displays the calculated result through the CPU

Syntax: disp $rd , immediate [immediate assigned 0]

## manual

To run the program user needs to open the application file called “Assembler.cpp” which is provided in the folder by the same name. It is absolutely necessary that the folder which is containing the program, has a text file (.txt) called “input”. This is the file from where the assembler reads the instructions. After reading the “input.txt” file, the assembler writes the corresponding hexadecimal output code to “output.txt”file. If the user wants to try his own code for assembling, then just needs to rewrite the input file. One important thing to notice is that each line of the input file can only contain one instruction.

**Warning:** Under no circumstances the “R-type.txt”, “I-type.txt”,and “RegisterValue.txt” are to be edited

## LIMITATION

The user has to give spaces between instruction words in the input file. If user don’t follow this format the system will show a valid code as invalid.