

**PROJECT (Part 2)**

**CSE 332**

Project Name: Assembler Design

Section: 03

Group Members:

|  |  |
| --- | --- |
| Name | ID |
| Tamanna Islam Sushoma | 1520412042 |
| Raktim Rayhan Prova | 1530600642 |
| A S M Rayhan | 1530080042 |
| Syeda Sumia Jabin Nowrin | 1511392042 |

Submission date: 30/03/2018

**INTRODUCTION:**

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

**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 then convert it into hexa-decimal code and generates output file containing machine code. Source code is written in Java.

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

**INPUT:**

The input file is located in the main folder and the source code is in “src” folder. User has to write down a MIPS code in this file.

**REGISTERS:**

We have selected registers $t0-$t3 and assigned 2 bits for each of the register as in our instruction field in MIPS containing the register rs, rt and contains 2 bits.

Apart from those we have two special register $zero and $dis

|  |  |  |
| --- | --- | --- |
| Name of the Registers | Register Number | Value Assigned ( 2 Bits) |
| $t0 | 1 | 00 |
| $t1 | 2 | 01 |
| $t2 | 3 | 10 |
| $t3 | 4 | 11 |

\*\* user can’t use $t3 in any of his instructions directly, $t3 is reserved for lw, sw, slti, beq and disp operations in which

**R-TYPE LIST:**

We have selected following op codes and assigned functionality values (1 bits) for each of the op codes. For R-type operation the opcode is always 000

|  |  |  |
| --- | --- | --- |
| Operation | Opcode | Function code |
| ADD | 000 | 0 |
| SUB | 000 | 1 |

**I -TYPE LIST:**

We have selected following op codes for each of the operations as we know in the instruction field in MIPS, the op-code contains 3 bits.

|  |  |
| --- | --- |
| Operation | Op-code |
| ADDi | 001 |
| LW | 010 |
| SW | 011 |
| BEQ | 100 |
| SLTi | 101 |

**J-TYPE LIST:**

We have selected following op codes for each of the operation as we know in the instruction field in MIPS, the op-code contains 3 bits.

|  |  |
| --- | --- |
| **Operation** | Op-code |
| J | 110 |
| Disp | 111 |

**INSTRUCTIOS DESCRIPTION:**

add: It adds two registers and stores the result in one of the two register.

* Operation: $t0=$t0+$t2
* Syntax: add $t0, $t2

sub: It subtracts two registers and stores the result in one of the register.

* Operation: $t0=$t0-$t2
* Syntax: sub $t0, $t2

addi: It adds a value from register with an integer value and stores the result in destination register. Offset value for this type can hold (-4<value<4)

Operation: $t0 = $t0 + offset

Syntax: addi $t0, offset

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

Offset value for this type can hold (0<=value<1)

Operation: $t3=Mem[4+$t0]

Syntax: Lw 1($t0)

sw: It stores specific value from register to memory.

Offset value for this type can hold (0<=value<1)

Operation: Mem[4+$t0] = $t3

Syntax: sw 1($t0)

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.

Offset value for this type can hold (value<8)

Operation: If($t3==$zero) goto address

Syntax: Beq $t3, address

Slti: If $t0 is less than any offset value,

Offset value for this type can hold (value<8)

Operation: If($t0<3) then $t3=1

Else $t3=0

Syntax: Slti $t0,3

J: Jumps to the target address. Offset value for this type can hold (0 <=value <32)

Operation: PC = nPC

Syntax: j target address

Disp: Display the content of $dis.

Syntax: Disp

Content of $t3 will be displayed

**LIMITATION:**

The user has to write one input at a time and has to give single spaces between instruction words in the input file. If user doesn’t follow this format the system will show a valid code as invalid, and hence will not provide correct output.