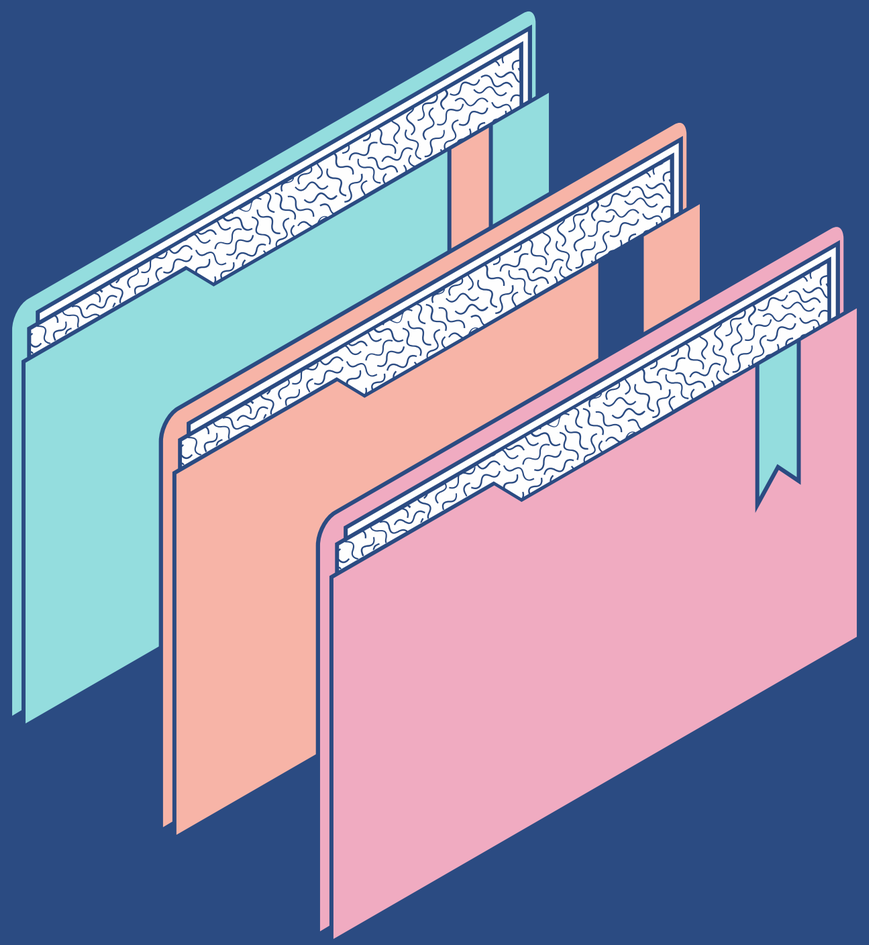
CS 215 GROUP PROJECT

MIPS ASSEMBLER AND DISASSEMBLER

By Harshvardhan Vala (20110075)
Inderjeet Singh Bhullar (20110080)
Kalash Kankaria (20110088)
Kanishk Singhal (20110091)



Goal

THE GOAL OF THIS PROJECT WAS TO:

- Make an assembler and disassembler for MIPS 32 architecture using Python
- Implement a GUI interface using Tkinter library

Importance of the Project

This project is the culmination of our coursework and knowledge earned during the semester. We have tried to implement all the concepts of MIPS 32 architecture, learned until now to make a platform, that is easy to access and use.

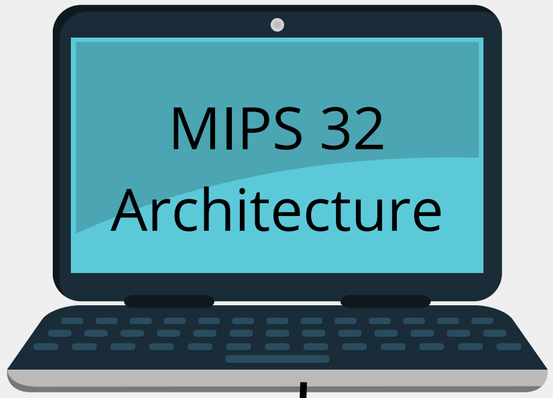
It enables one to assemble an assembly-level code and disassemble this assembled code (machine code) back to the assembly-level code.

The Python Code

We implemented this assembler and disassembler by writing basic Python codes that cater to various instruction formats available in the MIPS-32 Architecture and further coded the different cases that may occur for assembling a particular function and similarly the different cases that may occur on the input of a machine code.

The Interface

We designed and developed a complete GUI application that provides an easy-to-use interface for this assembling-disassembling. We also provide the user with a choice to deal in binary numbers or hexadecimal numbers. While the computer hardware only understands binary, we added this extra feature to enable the user to read even the machine code easily.



R Type

add
sub
and
or
nor
slt,sll,sll
jr

I Type

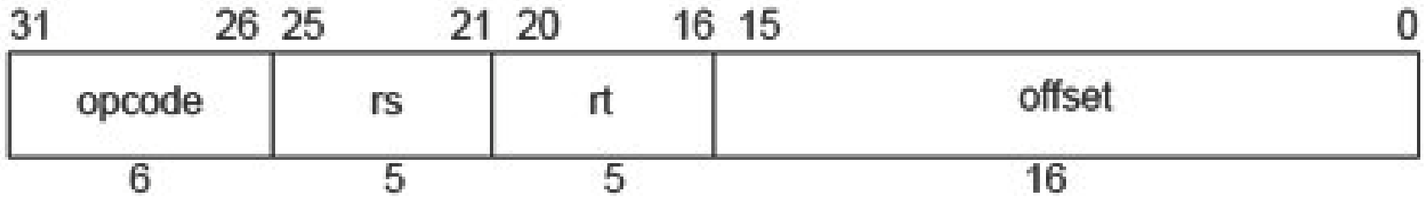
lw
sw
addi
andi
slti, sltiu
or
beq, bne

J Type

j
jal

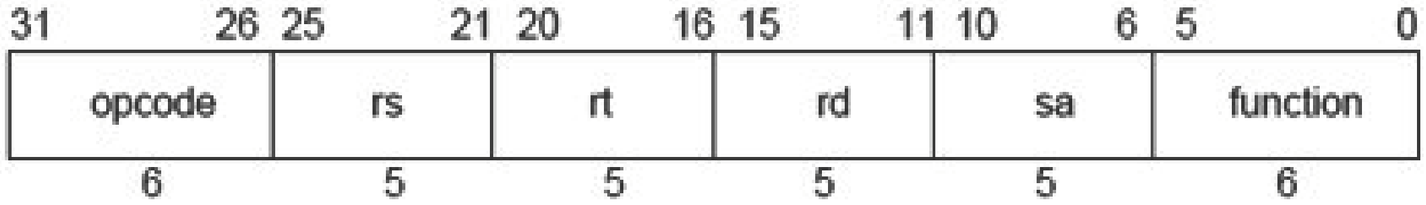
MIPS 32 Instruction Formats

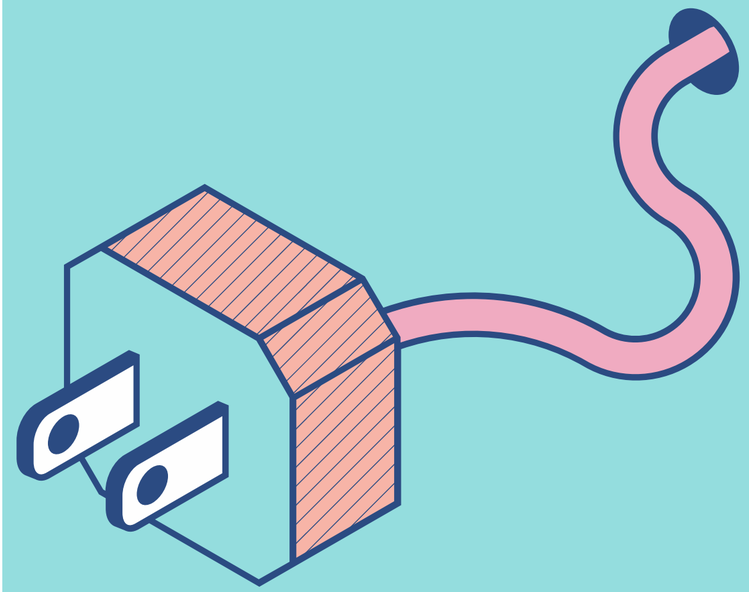
I-Type (Immediate).



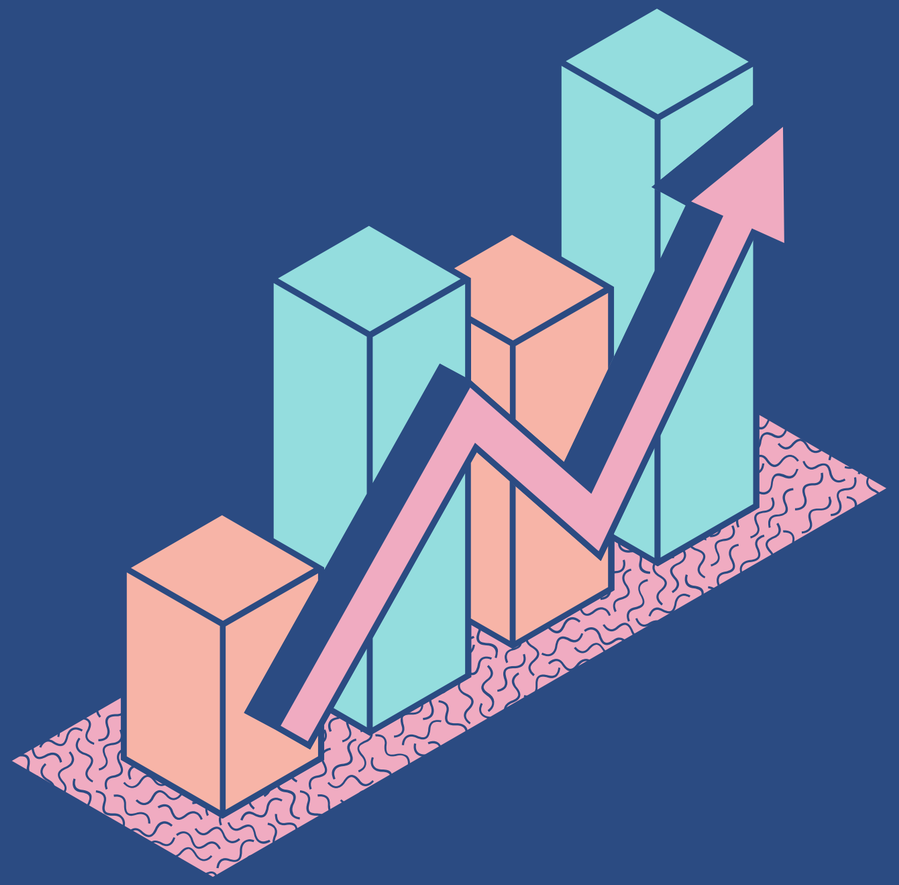
J-Type (Jump).

R-Type (Register).





Implementation



MIPS ASSEMBLER

Assembly Language

Assembler is a program that is used to convert the assembly language code into machine code. Assembly language is human readable and easily convertible into machine code. It takes in input the assembly language codes like, MIPS assembly code, and converts it into binary that the computer understands. Each operation has a binary code associated with it that uniquely identifies the operation

Implementation

Our implementation is based on parsing through the instruction, identifying the type of instruction, as well as the format in which it occurs, and then converting it into the binary code.

How the Code Works

1 STEP

Parsing

Parse throught the assembly code and identify the type of instruction

2 STEP

Format

Identify and narrow down the format of the Instruction

3 STEP

Important Information

Identify the operation, registers, immediate values, addresses associated with the instruction

4 STEP

Final Conversion

Convert the assembly code into binary after making necessary computations

MIPS DISASSEMBLER

Disassembler

The disassembler is the program that can convert Machine Code back into MIPS Assembly Code. The disassembler we have programmed can take machine code in the format of binary or hexadecimal and it will output the assembly code. The format of the input should be such that each machine code instruction is in a different line.

Implementation

In order to disassemble the code, the program first figures out the type of Instruction using the opcode.
We will be using a lot of cases and conditions in our code in order to cover various formats of all the 3 types of instructions.

How the Code Works

1 STEP

Parsing and Conversion

Parse and convert input string to binary

2 STEP

OPCODE

Read the first 6 bits to obtain the opcode

3 STEP

Decide Instruction

Decides the instruction according to the opcode

4 STEP

Format

Decides the instruction according to the opcode

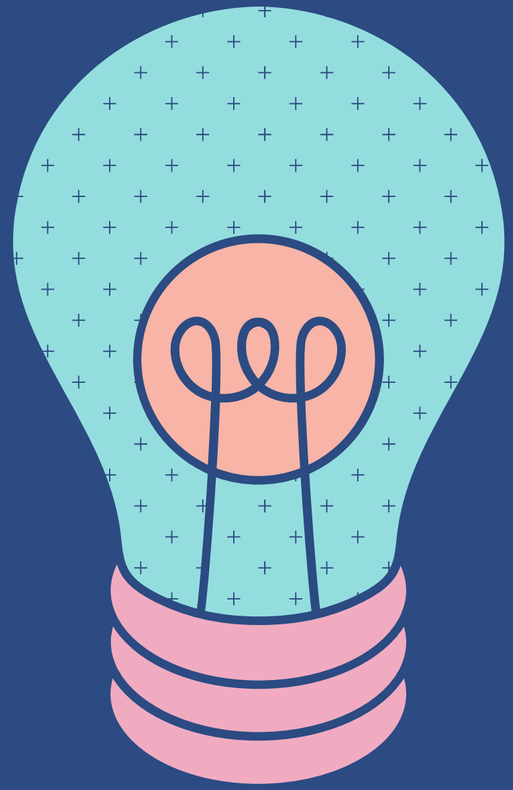
5 STEP

Concatenation

Concatenate the instruction based on the value of registers and what operation was used

Conclusion

- The Assembler-Disassembler developed under this project is a GUI Application that takes a MIPS assembly code and assembles it into a Binary Machine Code and vice-versa.
- Human-readable instructions <---> Hardware compatible instructions
- Implemented for most of the MIPS 32 instructions.
- May be used to implement a compiler for conversion of Assembly Language to the Binary Machine Code and vice versa.
- Our project is a simple and free application that can be used to efficiently reduce the labor-intensive work of assembling and disassembling and can be used for educational purposes.



Future Scope?

- Much larger set of instructions. (Even Pseudo Instructions)
- Better Error Handling
- Use a faster language like C/C++
- Directory in the GUI Application that keeps a track of the addresses of the instructions in the memory to optimize Memory Organization. Applications in handling Control Hazards in pipelining
    - Filling of the Delay Slot in Delayed Branching
    - Branch Prediction improvement by efficient stacking of code.

Thank You