IAS PROCESSOR DESIGN

* SAHAS SANGAL IMT2023556
* SHIVAM PANDYA IMT2023097
* PARITOSH TIWARI IMT2023123

In this project we have taken a C++ program and wrote it’s IAS code, made an Assembler that generates the Binary Code of the given C++program and a Processor which takes the Binary Code and evaluates the output.

Test Program

* We tested the program that evaluates the sum of the digits of a number.
* Eg. INPUT- 123 OUTPUT-6

Architecture of The Processor

* Our processor is capable of working with only positive **24bit** integer.
* Our Memory has addresses from **0 to 127**.
* Each of these addresses are capable of storing **24 bits** of data.
* Our instructions are split into left and right instruction, which are further split into **5bit Opcode** and **7bit address**.
* Our instruction set contains **18 instructions** each with an unique Opcode.
* We built our processor upon Von Neumann architecture and further added instructions- **AND, OR, NOT, COMPARE\_1, COMPARE\_2.**
* The control flow of our processor **Fetch->Decode->Execute->Decode->Execute** except in case of jump statements.
* Our memory is divided into- **Program instruction** 64-127, **Program Data** 32-63 and **Buffer** from 16-31.

IMPLEMENTATION OF THE PROCESSOR

* We used **Python3** to construct our processor.
* Our processor has implemented all the components of the Von Neumann architecture with **ALU, MBR, IBR and Memory as Classes and AC, MQ, IR and MAR**
* To showcase the Van Neuman Architecture In a loop we did the following function calls in order **fetch->decode->execute->decode->execute**
* All the program instructions and program data is already loaded inside the memory before the running of python3 file
* Inside the fetch, decode, execute functions are all the cycle specific operations.
* The control signals are represented by flag attributes of various classes.

INSTRUCTIONS AND ITS USES

ASSEMBLER

We implemented assembler using Python3 and it converts our specified format assembly language code into Machine codes of Data and instructions of 24 bits.

OBJECTIVES

VARIABLES AND THEIR USES

RESULT