# To Develop an Assembler for SDLX Processor

**By Group 20**
Abdul Qadir Ronak: 21110003
Darshi Gaurang Doshi: 21110050
Sharika S: 21110194

# What is an Assembler?

- A software which takes inputs from the user in the assembly language and then converts it into binary (machine language).
- This is then used as the input to a processor of the computer to perform basic operations.
- Usually the instructions given in Assembly language in the format specific to the processor is converted into the binary 32 bits Instruction.
- This goes to the instruction register in case of an SDLX microprocessor.

# Assembly Language

- Low level language and is specific for the processors.
- It is easier for the user to provide inputs in Assembly language as dealing with long binary instructions to carry out operations with the processor is pretty tedious.

# Input/Output Format

On giving n lines of assembly code as input, we will get n lines of 32 bits binary instructions which can be used to implement on the FPGA board using the switches. Initially we printed only the 32 bits instruction line by line. Then, we printed the names of the opcodes alongside the 32 bits instruction to confirm whether the line corresponds to the correct opcode or not.

We have also taken care of the cases of the assembly code, i.e., the assembler will give the output independent of the case (uppercase or lowercase) in which the assembly code is written.

# Convention Used for the Assembly Code

Example 1: ADD R1 R2 R3

- The above line implies that R3 is the destination register where the value of R1+R2 is stored.

Example 2: ADDI R1 $4 R1

- We have also used ‘$’ sign before immediate and offset values.

# Result of our Assembler Code

| | |
|---|---|
| addi R0 $100 R1 | 10000000000000010000000001100100 |
| addi R0 $200 R2 | 10000000000000010000000011001000 |
| addi R0 $1 R3 | 10000000000000011000000000000001 |
| addi R0 $7 R4 | 10000000000001000000000000000111 |
| Loop: addi R4 $1 R4 | 10000000100001000000000000000001 |
| lw 0(R1) R5 | 11011100001001010000000000000000 |
| lw 0(R2) R6 | 11011100010001100000000000000000 |
| addi R1 $4 R1 | 10000000001000010000000000000100 |
| addi R2 $4 R2 | 10000000010000100000000000000100 |
| add R3 R9 R3 | 00000000011010010001100000000000 |
| bne R4 R0 Loop | 11110000100000001111111111111001 |
| add R0 R0 R0 | 00000000000000000000000000000000 |
| addi R0 $300 R7 | 10000000000001110000000100101100 |
| sw 0(R7) R3 | 11101000111000110000000000000000 |

# Thank You !!