MIPS_Simulator

- A List named **reg** of size 32 is defined for representing 32 registers.
- A dictionary named **memory_dictionary(**in which KEY is memory address and VALUE is value stored in that address**)** of size 1024 is defined for representing 4kb of memory.
- The given assembly file is read line by line and each line is stored a sub_list of **list S[ ]**.
- All the spaces and empty lines are removed.
- All the data elements from the given assembly file is stored in a dictionary named **data_elements.**
- Index of **".main"** is found using a while loop and it is stored in variable **p.**
- All the labels after main are stored in dictionary named **labels**.
- All the instructions present after the **".main"** are executed using a while loop.All these instructions are accessed from the **list S[ ].**

Instructions that can be executed:-

- **li** (load immediate)
- **add**(addition)
- **sub**(subtraction)
- **bne**(branch on not equal)
- **beq**(branch on equal)
- **addi**(add immediate)
- **slt**(set on less than)
- **slti**(set on less than)
- **sll**(shift left logical)
- **la**(load address)
- **lw**(load word)

- **sw**(store word)
- **j**(jump)
- **move**
- **syscall**
- **jr**(jump register)

- All the above instructions represents the standard instructions of MIPS(32-bit) assembly language

Phase 2:

- Implemented pipelining in the simulator.
- We used 5 variables namely ins_fetch, ins_decode, execute, memory_stage, writeback for simulating pipeline.
- We start a while loop and activate or deactivate stages in pipeline using the 5 variables according to the flow of the pipeline and stalls detected.
- We incorporated latches for the pipeline stages namely insf_insd, insd_ex, ex_mem, mem_wb.
- We have analysed different cases for getting a stall and implemented them in our simulator.
- If a stall is detected in any stage by the variables for stall detection then we increment the stall variable and stall the pipeline and wait for the next cycle.
- After each cycle we update all the latches.
- In the end we print Register and Memory contents, number of stalls, cycles, instructions and IPC.

Note:-

- A sample input file**(bubble_sort.asm)** is attached along with the code.In case of changing the input file,update the **input_file** present in **MIPS_sim.py** at **line no.7.**
- In case of updating the data of given **bubble_sort.txt** file,update the values in $s3,$s4 as they represent N,N-2 respectively.N represents number of elements to be sorted