DOCUMENTATION (PYTHON)

1) Following classes have been implemented for the project :
    a) Fetch, Decode, Xecute, Memory, WriteBack.
    b) Clock : contains a clock object (integer variable).
    c) CPU : contains the register file, program counter and a global clock.
    d) InstructionMemory: contains the instruction cache.
    e) DataMemory: contains the data cache.
2) Following methods have been defined for making the program modular and readable :
    a) DTB : converts a base 10 number into a 32 bit binary number (String). This was used since our machine was assumed to be capable of handling 32 bits instructions and data at one go.
    b) BTD : converts a 32 bits binary to a base 10 number.
    c) Reverse : helper method to reverse a binary string.
    d) PartialCPUState(): This method was implemented in mid-sem evaluation to write the partial CPU state to the logfile (text file). Now, we have extended the same method to write the complete CPU state which includes info. about instruction in each pipeline stage and whether the CPU is stalled or not in addition to the partial CPU state already written.
    e) getTotalInstructionsOfGivenType() is a method to find the total number of instructions of memory and register type in the given test binary and store the number in a list and return to the caller method.
    f) instGraph() : plots a graph using the list returned from the getTotalInstructionsOfGivenType() method .
    g) DatamemAccessGraph() : used to plot the graph of data memory access(address) against the cycle numbers.
    h) InstmemAccessGraph() : used to plot the graph of instruction memory access(address) against the cycle numbers.
    i) stallGraph() : used to plot the graph of stalls against each of the cycle numbers. If there is a stall, we represent it as 1 , otherwise, we represent it as 0.
    j) Apart from the above methods, there is a main() method which performs the following :
        i) Instantiates the objects of the classes defined.
        ii) Initializes the variables and loads the data cache and the instruction cache.
        iii) Declares the lists for storing the information about the stalls, memory accesses, types of instruction, etc.

iv) Using various helper variables(integer and boolean) and while loop, we have simulated the input binary using a 5-stages pipelined CPU.

v) Following features have been implemented and tested against various test cases :
   (1) Stall Logic.
   (2) Data Hazards :
      (a) Control Hazards
      (b) Data Hazards
      (c) Structural Hazards
   (3) Kill Logic has been implemented along with the control hazards checking.
   (4) Full by-passing.

GROUP-3 MEMBERS

1) *Rohan Gupta : 2020113*
2) *Pathik Sharma : 2020095*
3) *Prasun Pratik : 2020101*
4) *Chirag Bansal : 2020047*