**Design Documentation**

Programming Project 2: Simulator for APEX with in-order issue and Out-Of-Order WriteBack with Multiple FU’s

**Programming Language :**

* Programming Language used for this Project is JAVA.

**Pipeline Structure:**

* This APEX pipeline has multiple FU with DIV Fu (4 cycle Latency), MUL FU (2 cycle Latency) and INT FU(1 Cycle Latency).
* DIV and Halt Instruction flows from DIV FU. HALT takes the Longest Path since it cannot have instruction behind it, So there is condition that Instruction Stalls if it takes the longest Path.
* MUL instruction Flows through MUL FU.
* Rest ALL the instruction will flow from INT FU.
* In this Project, Data Forwarding Mechanism is used. So the If the Instruction is in the last Stage Of FU(INT FU, DIV4,MUL2) and there is a dependent Instruction in D/RF it won’t stall due to flow dependency and Data will be Forwarded.
* Since Out-of-Order is Implemented , OUTPUT Dependency is taken care of. So, there is no new instance of Destination Register will be created.

**File Structure and Representation :**

* The Program Consists of Three Package :
  + General (consists of file which gives state of each Stages in a particular cycle)
  + Pipeline (Main Function)
  + Stages (Consists of all the stages)
* General Package consists of following files :-
  + Instruction.java :- File consists of getter and setter which set and gets the required elements of the Instruction.
  + ProcessingUnit.java :- It consists of the execute function which does the execution in bottom to top manner, but the instruction flow in the regular manner. Also it Consists of the Display Function which displays the register value and memory addresses.
  + ProcessingUnitInner.java :- It Does the Initialization and consists of all the getter and setters of all the stages and the flags value which are used extensively in the code to retrieve some value or set some value.
  + ProcessingUnitComputation.java :- It consists of some general method which are required like getting the register index, getting that index value, checking if the register is valid or not.
  + Register.java :- It consists of all the elements a register has like sources, destination, valid flag, etc and stores the values. It is Initialized from the init Function and all values and the flags are set. Getter and setter are defined here for setting or getting the required elements.
  + VirtualRegister.java :- Used for storing the Values and Forwarding the values if the dependent Register enters the FU. IT is Exactly same as Register.java
* Pipeline Package Consists of the Main Function which flows through the code by executing and initializing the code. Basically it consists of the user menu which does the following function.
* Stages Package consists of :
  + Fetch.java
  + Deode.java
  + IntFu.java
  + MUL1.java
  + MUL2.java
  + DIV1.java
  + DIV2.java
  + DIV3.java
  + DIV4.java
  + Memory.java
  + WriteBack.java

It does all the necessary computation required.

* For Reseting the register flags there is Lock method in the Writeback.
* ForwardingLogic function is defined in the Decode which heart of forwarding and does all the forwarding.
* There is Control function in Last Stages of FU to handle the BZ and BNZ during Forwarding.
* There is ReleaseForwarding Function in the first Stage of Fu to release the forwarding if any it was done.

**Flow Of The Program :**

* Initialize the R[] to 0
* Adds the Instruction and pc\_address line by line with pc\_address increment of 4.
* Display all the register value correctly.
* Display the Memory location value whose value is set

1.Initialize

I

3.Display

User Console Menu

Cycle runs in bottom Up fashion. That is, WB is first called then it checks for instruction in MEM. Mem asks for Instruction from(IV$ or MUL2, INTFu.Instruction flows the respective EX stages and then DIV!, MUL1 and INTFu ask for instruction in D/RF stage. D/RF asks finally from Fetch Stage.

Loop ends when the cycle specified by the user is reached or the HALT is Encountered in WriteBack.

2. There is important variable that keeps the track of instruction count i.e instCount

1. I have define a for loop and it’s the main loop that will run up to that cycle and displays the cycle count.

* Takes the User Input about the no. of cycle to be simulated.

2.Simulate