Pipeline Hazards

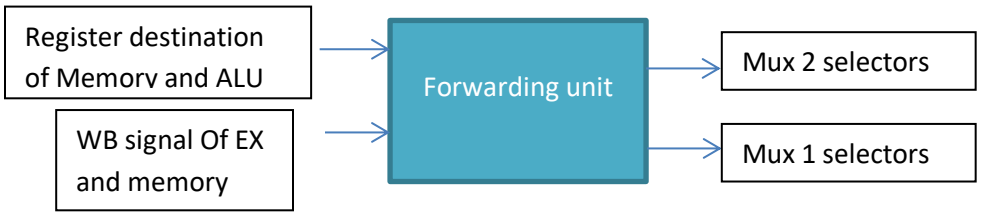
Pipeline registers details:

1- Fetch/Decode register:
   - ?? bits for instruction that will be executed.
   -Bit indicate if branch is taken or not
   - ?? bits for pc value
   - 1 bit for if branch signal
2- Decode/Execute register:
   - 4 control bits(WB,M,EX)
   - 32*2 bits for the values of the source registers
   - 3 bits destination register code if exist
   - 32 bits sign extend if exist
   - ?? bits for data value in load/execute instructions
   - ?? opcode of I instruction
   -1 bit indicate if this is ret or reti instruction
3- Execute/memory register:
   - 2 control bits(WB,M)
   - 32 bit the output of ALU
   - 3 bits destination register code if exist
   - ?? bits for data value in load/execute instructions
   -1 bit indicate if this is ret or reti instruction
4- Memory/Write Back register:
   - 1 control bits(WB)
   - 3 bits destination register code if exist
   - ??bits for (Memory output /Alu output)
   -1 bit indicate if this is ret or reti instruction

Pipeline hazards:

1- Data Hazard

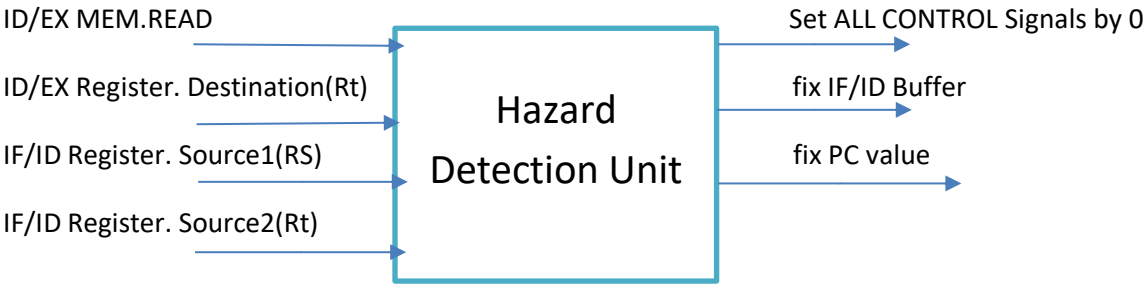
solution is full forwarding unit:



Where the unit will check the write back signals and if the signal is set it will compare the destination register of this stage with the source registers in execution stage to detect the values taken by the mux(from register file OR from memory buffer OR from execution buffer)

2- load-use data hazard:

- Solution is **Hazard Detection Unit** in Decode stage. It detects if the instruction is load and there is dependency in the next instruction so it will insert bubble.



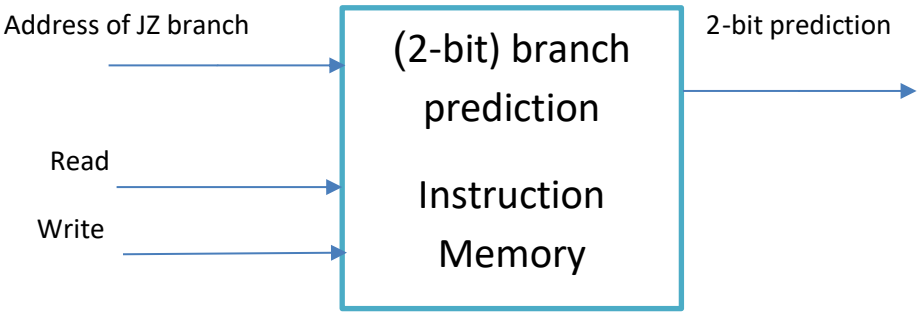
3- Control Hazard:

- JMP and CALL instructions will execute at fetch stage. In case of CALL instruction store old value of PC in IF/ID buffer to store in memory.

- Check condition of JZ instruction, if it was taken and condition is false it will flush the instruction and return the old PC value. If it was untaken and condition is true so it will flush instruction, calculate the address, and write it in PC.

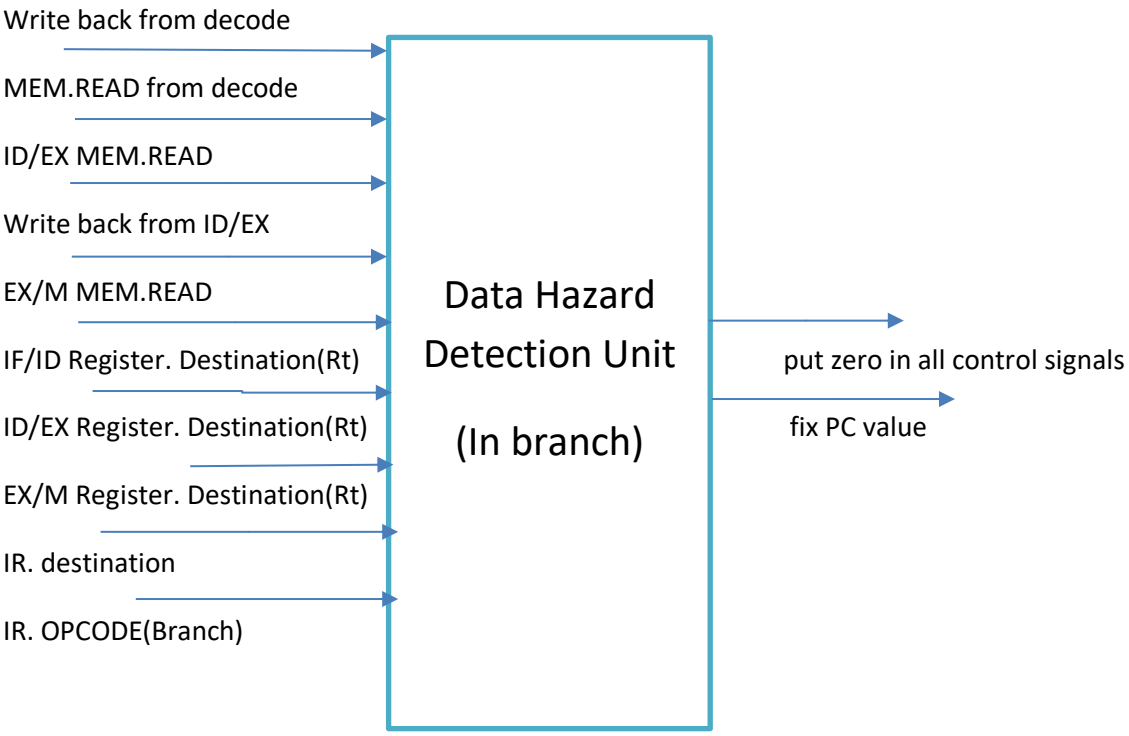
4- (2-bit) branch prediction :

- In fetch stage for JZ branch: Having Prediction bits in **The instruction Memory**



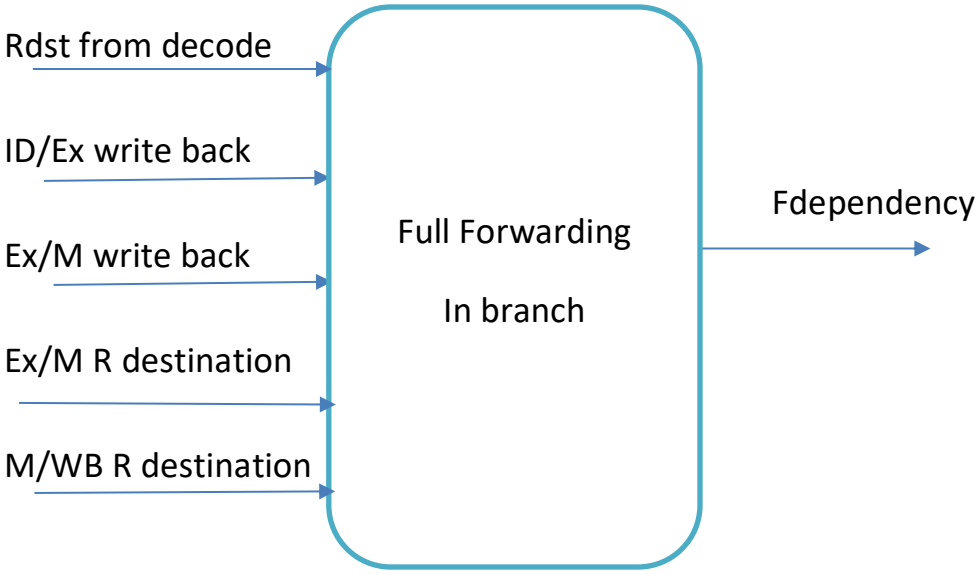
- To check dependencies :

- In fetch stage, we extract OP-code and addresses Destination Register from IR instruction.
- If MEM.READ equal to one in stage (Decode, Execute, Memory), it is a load instruction or IN instruction, and if there is dependencies, so it will insert bubble.
- If MEM.READ equal to zero and Write back(Decode) equal one, it is an ALU instruction, and if there is dependencies, so it will insert bubble.



5- FULL Forwarding for Branch:

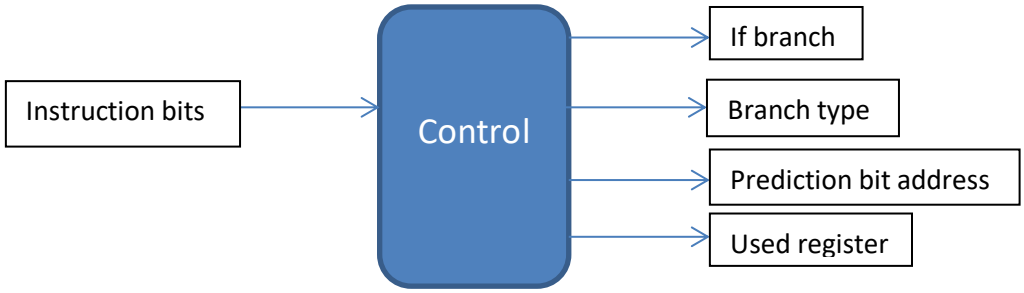
- In decode stage; it detects dependency in case of branch instruction with instruction of write back signal. If it detects dependency, it will pass data from E/M buffer or M/WB buffer to Rdst in fetch stage.



Some used components and their functionalities:

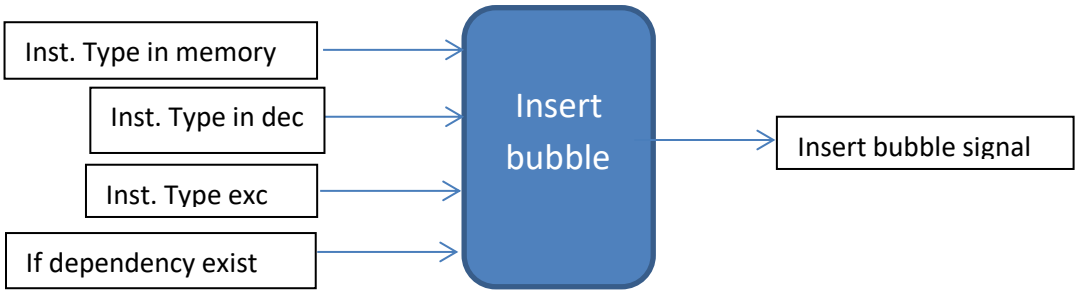
1- Control in fetch state:

This component find if this instruction is from branch type instruction and if so it out signal indicate that it is branch instruction and the type of branch ,the used register if exist and the address of prediction bits



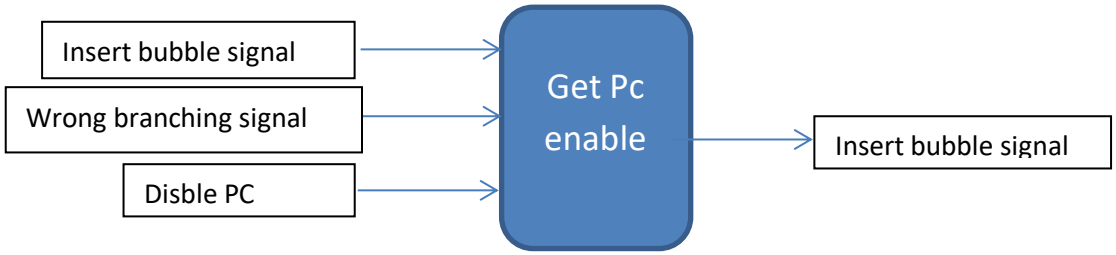
2- Insert bubble component:

This take the type of instruction in ex,dec,mem stages to check if its ret or reti and take the dependency signal to decide insertion of bubble or not



3- Get Pc enable:

This component to control pc enable to fix the PC value in case of bubble and in case of bubble and flushing conflict



4- Get PC selector:

This indicate weather pc value will be

1- PC+4

2- Register value

3- value come from memory

4-The right branch value from Decode stage

From different signals