Branch Predictor in Pipeline Implementation

The architecture contains a total of 9 modules:

IMEM: Accepts instruction address from CPU and sends instruction value to the CPU.

DMEM: Sends the value stored in memory to the CPU.

Register File: Sends values from registers addressed by rs1 and rs2 to the CPU.

CPU: This acts like the control unit and sends the required signals to the remaining modules. It detects the type of instruction from the opcode and selects the required output using a multiplexer from the outputs of the following 5 modules. Since JAL, JALR, LUI and AUIPC are instructions with unique opcodes, the CPU generates the required output within itself without depending on an external module.

R_type: This module is used for the R - type instructions. It accepts idata, rv1 and rv2 from the CPU and uses the value of funct3 and idata [30] as select lines to a multiplexer to choose the specific R_type instruction and assign the value of output (regdata_R) based on the correct ALU operation. This is sent to the CPU.

I_type: This module is used for the I - type instructions. It accepts idata, rv1 and imm from the CPU and uses the value of funct3 as select lines to a multiplexer to choose the specific I_type instruction and assign the value of output (regdata_I) based on the correct ALU operation. This is sent to the CPU.

L_type: This module is used for the Load type instructions. It accepts idata, daddr and drdata from the CPU and uses the value of funct3 as select lines to a multiplexer to choose the specific L_type instruction. Based on this, the 32 - bit value of the output (regdata_L) is assigned with or without sign extension based on the instruction. The output is sent to the CPU which sends it to the Register file as the data to be written.

S_type: This module is used for the Store type instructions. It accepts idata, daddr and we_S from the CPU and uses the value of funct3 as select lines to a multiplexer to choose the specific S_type instruction. Based on this, the 4 – bit value of the write enable signal (we_S) is assigned depending on whether a word, half word or a byte is to be written to DMEM. we_S is sent to the CPU.

B_type: This module is used for the Branch type instructions. It accepts idata, iaddr, imm, rv1 and rv2 from the CPU and uses the value of funct3 as select lines to a multiplexer to choose the specific B - type instruction. Based on this, the comparison is performed and the value of the output (iaddr_val) is assigned as pc+offset or pc+4 as required. The output is sent to the CPU which assigns the value of PC accordingly.

Procedure followed:

A table called branch prediction table is newly created. It is indexed by iaddr values and stores a Taken/Not Taken flag (T/NT flag) and the target address wherever applicable.

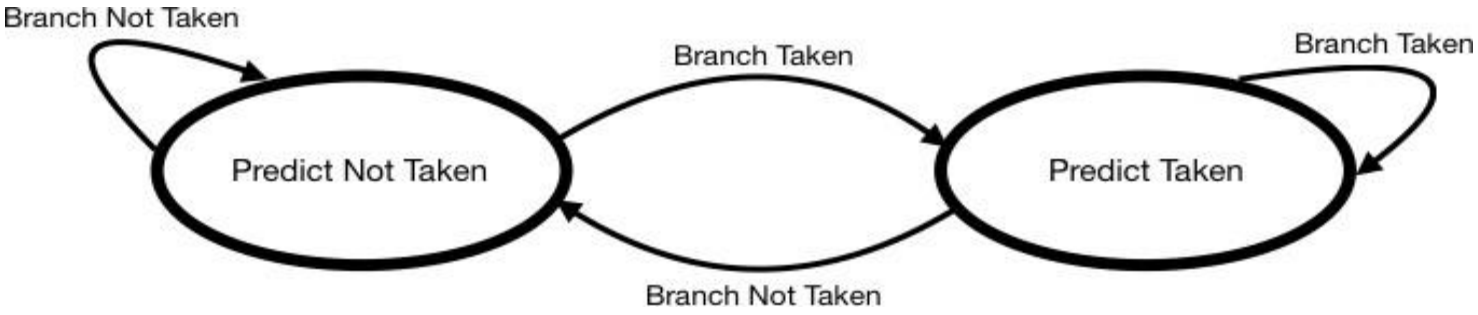
In the code, the T/NT flags are stored in a register called tnt_tab and target addresses are stored in a register called targ_tab. These 2 registers together form the branch prediction table.

For every new iaddr value, this table will be checked to see if the Taken flag is 1. If it is 1, a flag called 'pc_taken' is set and the corresponding target address is stored in a register called 'pc_pred' and loaded as the next iaddr value. A flag called 'branched' is set when the new iaddr value is loaded.

The decision of whether branching should take place or not is known only in the EX-stage. At this stage, there are 4 possibilities which are described in the table below:

Value of 'branched' flag	True Decision	Action
0	Branch	Branching takes place and the T/NT flag corresponding to the iaddr value is set
0	Don't Branch	No changes made. Program continues without branching
1	Branch	Since the branch was already taken, the program continues without branching
1	Don't Branch	Since the branch was wrongly taken, program branches to old iaddr+4 (which is stored as ignored_instr). T/NT flag corresponding to the iaddr value is reset to 0.

Branch prediction state diagram:



Pipelined CPU

Forwarding: The value of rd at EX/MEM register and MEM/WB register are compared with the rs1 and rs2 values at the ID/EX register. If the instruction requires the rs1 and rs2 values, then the MUX (within pipeline_CPU module) selects the required value and forwards it to the CPU.

Stalling: When an instruction intends to change the PC value, the pc_replace signal is set. Then the control signals of the instruction moving out of the ID/EX register are set to 0 and the idata corresponding to the instruction moving out of the IF/ID register is changed to the instruction ADDI R0, R0, 0. In this way, both the previously loaded instructions are invalidated. The instruction at the new PC value is loaded in the next clock cycle.

Hazard detection: If there is an ALU instruction immediately following a Load instruction which writes to the register read by the ALU instruction, the hazard is detected, and the flag signal is set to 0 and the iaddr_upd value is set to 4. This decreases the PC value by 4 so that the same instruction is loaded again. The act signal is also set to 0, which sets all control signals of the ALU instruction to 0, so that the wrong register values do not result in wrong outputs being written. The PC offset is given by the pc_new value. If the PC value is taken from a register (in case of JALR), the pc_JALR signal is set to 1.