Lec 10 (Post Midterm): Control Hazards Cont.

▼ Control Hazards

Speculation

Branch Target Buffer (BTB): table that stores branch targets, indexable using current PC
- useful for
taken branch prediction
— use PC to lookup the BTB, if match found → nextPC = BTB entry; o.w. PC = PC + 4

`flush`: boolean “valid” bit, stage will do nothing if invalid

Instruction memory/fetch is expensive
- check opcode for branch pattern

Every time we fetch an instruction, we need to query whether it is a branch. Increased cost!
BTB:
- ex) 32 entries, we can use a 32-bit register where each bit tells us whether an index is a valid entry

When do we update the BTB?

- when compute the branch target (execute stage)
- ID stage:
— need custom adder for computing branch target

BTB Hardware Implementation

implementing hash table in H/W is

expensive

- lookup area cost
- lookup latency
- table size must be small

PC address is 4-byte aligned (lower 2 bits are zero)

Formulas:

index_bits = log2(btb_size)

pc_index = (pc>>2) & index_bits

pc_tag = (pc >>2) >> index_bits

Initialization:

- clear valid bit at reset

BTB lookup:

- btb < = BTB[pc_index]
- check valid bit, and tag btb_tag == pc_tag, then use BTB entry; o.w. use PC+4

Code reordering optimization:

- if HW implements static prediction (either taken or not taken)

— what if we reorder code to prioritize predicted path

pro: improved accuracy

con: architecture specific, burden is on the programmer to understand the hardware, and the contract breaks if ??? is ever changed

Backward Taken, Forward Not Taken (BTFNT)

predicting that loops are taken and therefore going backwards in PC is taken!

con: for loops only

Branch Predictor

1 bit prediction FSM, high misprediction rate

→ 2 bit prediction FSM, essentially looks for 2 same transitions before changing predictions. accounts for case where switching not taken and taken