Lecture 13 (Replay and Simpler OOO)

1 Token Based Selective Replay

1. 90% of the misses are accounted by 10% of the instructions
2. Each instruction that is predicted to have a miss gets a **free token**
3. The id of the token is stored in the instruction packet
4. Token is a n bit vector if there are n tokens
5. Tokens are propagated similar to poison bits

1.1 After Execution

1. If the token head completed execution in expected number of cycles, broadcast the token id and the operands can turn the bit off
2. Else, token id is broadcasted to signal a replay

1.2 Misprediction

If an instruction that is predicted to not miss has a miss, we flush the pipeline when the instruction reaches head of ROB

2 Simpler OOO Design

1. Physical Register File (PRF) -> ARF
   - have a dedicated ARF to store the committed state
   - enhance the ROB to store uncommitted values
   - rename stage points to either ARF or ROB depending on where the latest value is stored

2.1 Pros

1. Recovery from misspeculation is easy
2. We do not need a free list

2.2 Cons

Values are stored at multiple places

Hardware development has 60% of people who test and verify since correctness is a hard requirement, 25% backend team which handles hardware constraints of the design, 10% of the designers of the chipset (frontend), remaining 5% are the architects

Sir is 16 years post PhD; his friends are now becoming architects

3 Compiler Based Optimizations

1. Constant folding - storing constants into variables instead of computing them
2. Strength reduction - convert multiplication and division to shifts and adds
3. Common subexpression elimination
4. Dead code elimination
5. Silent stores - repititive stores (which are not needed)

3.1 Loop Based Optimizations

1. Loop invariant based code motion - invariant moved outside the loop
2. Induction variable based optimization - multiplication changed to addition with some initialization
3. Loop fusion
4. Loop unrolling