Lecture 11 (Alternative Approaches to Issue and Commit)

1 Address Speculation

1. Predict the memory address for a load or store (using n LSB of PC)
2. Predict the stride
3. Store last address, stride, pattern
4. Pattern bit will be set if strided access (array access)

2 Load Store Dependence Speculation

1. Predict if there is collision
2. If collision is not predicted, directly send to cache
3. Else forward values
4. Collision History Table (CHT) is used - 1 bit entry
5. This can be augmented with store load distance and load waits till there are less than D entries in LSQ

M1 has more of these tricks hence it is fast; small savings here and there leads to a lot of savings; being kanjoos :)

2.1 Store Sets

1. n LSBs map to a SSIT (Store Set Id Table)
2. This entry maps to LFST (Last Fetched Store Table)
3. Store updates LSFT (update happens during rename/decode)
4. On load store dependence, SSIT is updated

sad reality: Indians are good in theory, we don't make anything practical. "Chashma bhi imported hai, at least mera to hai; I specifically paid for it to be imported"

We made good TVs, each state had its own TV manufacturing unit

3 Load Latency Speculation

Same idea as branch predictor - hit-miss predictor

4 Value Prediction

4.1 Why are values Predictable?

1. Data redundancy
2. Bit masking
3. Constants
4. Error checking code
5. Register spill code
6. Virtual function code

4.2 Predictor

1. Last value
2. Stride based
3. Based on profiling result

We have a confidence predictor since values aren't always same.

4.3 Virtual Function Code

Each object has a virtual function table where mapping of function to PC is stored