Speculation

Exploiting the runtime patterns of software

Motivating speculation

Branch penalty

- Avoid stalls with branch prediction
- Static branch prediction is inaccurate
- Can we dynamically predict branches based on *control flow patterns*?

Memory latency

- Avoid long latency with cache hierarchy
- A cache miss is still very expensive
- Can we anticipate misses based on *memory reference patterns*?

A branch example

```
sum = 0;

for(i = 0; i < n; i++)
    sum = sum + i;

return sum
```

- The branch for entering the for loop is evaluated n times
  - It may be *taken* (T)
  - It may be *not taken* (N)
- Assume n = 5, the pattern is:
  - N, N, N, N, N, T
- The branch's target is the same each time (`return sum`)

Speculative execution in hardware

1. Guess the branch target in the fetch stage

2. Verify the guess by executing the branch
   - Mispredictions must be flushed from the pipeline

- *Do not* store values to registers or memory until guess is verified
  - Not too bad for an in-order pipeline
  - Out-of-order pipeline?

Branch prediction data structures

Predicting targets

- Use a dictionary (cache) called `btb`
  - Key: Instruction address
  - Value: Target address
- Look-up cache using PC as key (`btb[pc]`)
- Cache hit: this is a control instruction
  - `target = btb[pc]`
- Cache miss: may not be a control instruciton

Predicting direction (taken vs. not taken)

- Use a dictionary
  - Key: Branch instruction address
  - Value: data to help with prediction
- What should data be?
  - What happened last time? (1 bit)
  - What happened the last n times? (n bits)
  - Other options

A prefetch example

```
sum = 0;

for(i = 0; i < n; i++)
    sum += a[i] * b[i]

return sum
```

- n memory references are made sequentially to arrays a and b
  - A cache's block size can help reduce misses (spatial locality)
- The reference pattern is:
  - 0, 1, 2, 3, 4, 5, 6, 7, 8, ….
- The *stride* between references is:
  - 1, 1, 1, 1, 1, 1, 1, 1, …

Evaluating prefetchers

- Timeliness
  - Did the prefetched data make it back before the next memory reference?
- Coverage
  - How many cache misses were mitigated with the prefetcher?
- Accuracy
  - How many prefetches were made to unnecessary blocks?

Examples of hardware prefetchers

Name	Prefetch	Notes
Next-line	X + 1	• Simple implementation • Useful for sequential access
Next-N-line	X + 1, X + 2, …, X + N	• N is the prefetching depth • Choosing N can be tricky
Stride	X + S, X + 2S, …, X + NS	• S is the detected stride • Can be combined with N • Useful for looping through array of structs

Conclusion

Dynamic branch prediction

- Minimize performance hit from control hazards
  - Branch behaviour is *highly* predictable
- Need to predict both target and direction
- Need to recover from mispredictions

Prefetching

- Minimize performance hit from memory accesses
- Complementary to caches
- Overly aggressive techniques are wasteful