# 3810 Review Session

Spring 2020

## Pipeline Performance

### Unpipelined processor

CPI:

Clock speed:

Throughput:

### Pipelined processor

CPI:

Clock speed:

Throughput:

### Circuit Assumptions

Length of full circuit:

Length of each stage:

No hazards

## Data Hazards

### No Bypassing

Point of production: always RW middle
Point of consumption: always D/R middle

```
                                   * PoP
I1  add:   IF   DR    AL    DM    RW
I2  add:        IF    DR    DR    DR    AL  DM  RW
                                   * PoC
```

### Bypassing

Point of production:
- add, sub, etc.: end of ALU
- lw: end of DM

Point of consumption:
- add, sub, lw: start of ALU
- sw  $1, 8($2): start of ALU for $2,
  start of DM for $1

```
                              * PoP
I1  add:   IF   DR    AL    DM    RW
I2  add:        IF    DR    AL    DM  RW
                                * PoC
```

### Assumptions

100 instructions
20 branches
14 Not-Taken, 6 Taken
Branch resolved in 6th cycle (penalty of 5)

### Approach 1: Panic and wait

### Approach 2: Fetch-next-instr

### Approach 3: Branch Delay Slot

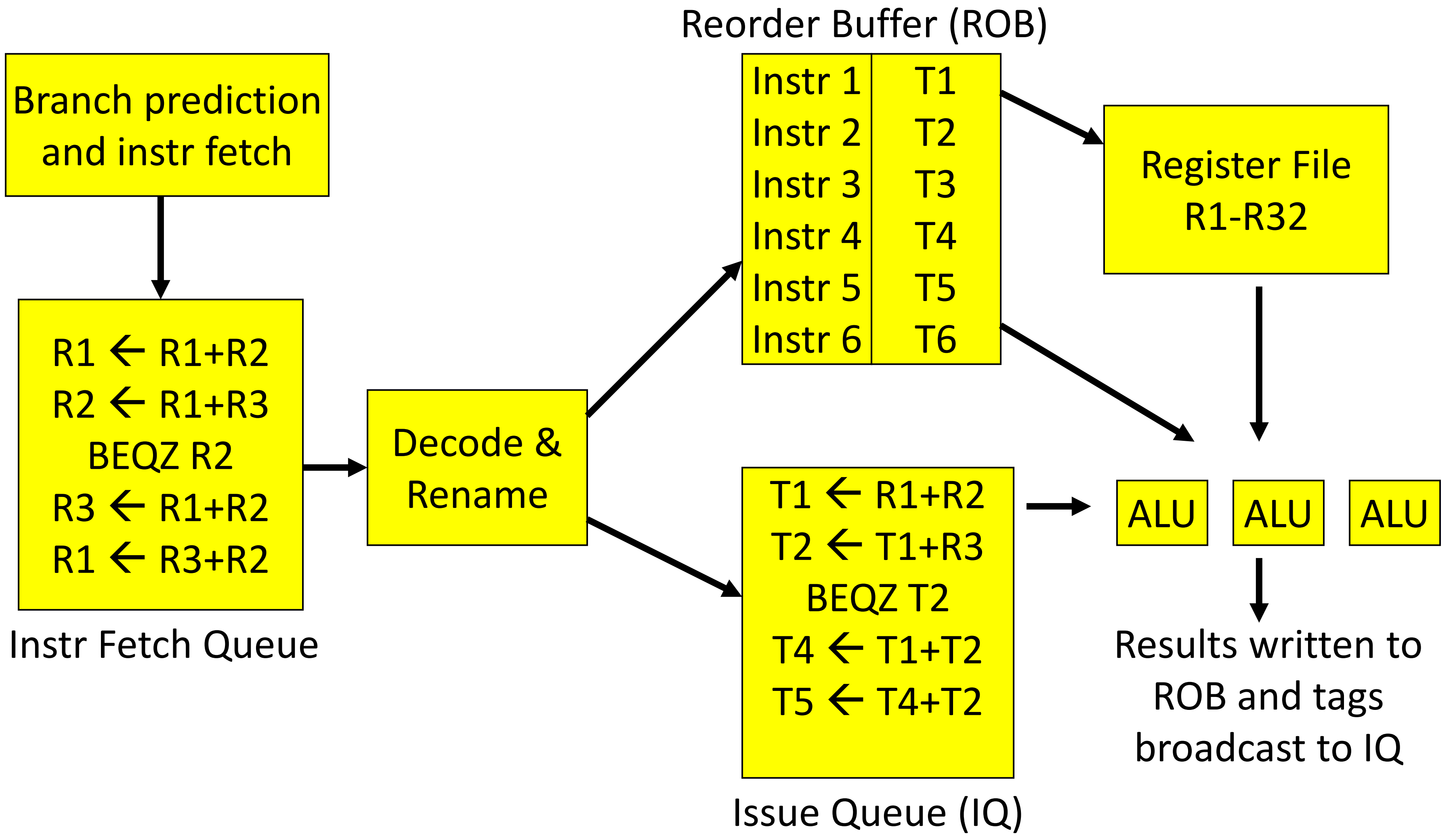
Option A: always useful
Option B: useful when the branch goes along common fork
Option C: useful when the branch goes along uncommon fork
Option D: no-op, always non-useful

| Option A | |
|---|---|
| Branch | |
| Slot | |
| NTaken | Taken |
| Option B | Option C |

### Approach 4: Branch predictor

Accuracy of 90%

## Control Hazards



## Out of Order Processor

Assumptions

1000 instructions, 1000 cycles, no stalls with L1 hits
# loads/stores:
% of loads/stores that show up at L2:
% of loads/stores that show up at L3:
% of loads/stores that show up at mem:
L2 acc = 10 cyc,   L3 acc = 25 cyc,   mem acc = 200 cyc

## Cache Latency

Assumptions

512KB cache, 8-way set-associative, 64-byte blocks, 32-bit addresses

Data array size = #sets x #ways x blocksize
Tag array size = #sets x #ways x tagsize
Offset bits = log(blocksize)
Index bits = log(#sets)
Tag bits + index bits + offset bits = addresswidth

## Cache Size

## Cache Hits/Misses

Assumptions

16 sets, 1 way, 32-byte blocks

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Access pattern: | 4 | 40 | 400 | 480 | 512 | 520 | 1032 | 1540 |
| Set #: | | | | | | | | |
| Block: | | | | | | | | |

6. Consider a 4-processor multiprocessor connected with a shared bus that has the following properties: (i) centralized shared memory accessible with the bus, (ii) snooping-based MSI cache coherence protocol, (iii) write-invalidate policy. Also assume that the caches have a writeback policy. Initially, the caches all have invalid data. The processors issue the following three requests, one after the other. Similar to slide 4 of lecture 25, fill in the following table to indicate what happens for every request. Also indicate if/when memory writeback is performed. **(12 points)**

(a) P3: Read X

(b) P3: Write X

(c) P2: Write X

| Request | Cache Hit/Miss | Request on bus | Who responds | State Cache 1 | State Cache 2 | State Cache 3 | State Cache 4 |
|---|---|---|---|---|---|---|---|
| | | | | Inv | Inv | Inv | Inv |
| P3: Rd X | | | | | | | |
| P3: Wr X | | | | | | | |
| P2: Wr X | | | | | | | |

Security

Virtual Memory

Synchronization, GPUs