Lecture: Review Session

• Final exam details:
  – Monday 12/13, 1pm – 3pm
  – 80%+ on post-midterm material
  – A couple unseen problems, a few “short-response” questions
  – Questions ordered easy to difficult
  – 3+3 reference sheets (double sided)
  – Show steps; calculators allowed
# OoO Timeline

- **InQ**: Cycle at which the instruction arrived into the Issue Queue
- **Issued**: Cycle at which the instruction is issued (leaves Issue Queue)
- **Complete**: Cycle at which the instruction completes
- **Commit**: Cycle at which the instruction gets committed

*1: Fetch width full. Fetched in next cycle.*
*2: Issue width full. Issued in next cycle.*
*3: Commit width full. Committed in next cycle.*
*4: Commit delayed in order to commit in order.*
*5: No free register in Free Register List. Must wait until a physical register frees up.*

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LD LR1, 0(LR2)</td>
<td>LD PR33, 0(PR2)</td>
<td>LR1-&gt;PR33</td>
<td>i+1</td>
<td>i+7</td>
<td>i+7</td>
<td>LR1-&gt;PR33</td>
<td>PR1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>DADD LR1, LR1, LR3</td>
<td>DADD PR34, PR33, PR3</td>
<td>LR1-&gt;PR34</td>
<td>i+3</td>
<td>i+8</td>
<td>i+8</td>
<td>LR1-&gt;PR34</td>
<td>PR33</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ST.D LR1, 0(LR5)</td>
<td>ST.D PR34, 0(PR5)</td>
<td>-</td>
<td>i+4</td>
<td>i+10</td>
<td>i+10</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>DADD LR2, LR2, 8</td>
<td>DADD PR35, PR2, 8</td>
<td>LR2-&gt;PR35</td>
<td>i+2</td>
<td>i+7</td>
<td>i+10</td>
<td>LR2-&gt;PR35</td>
<td>PR2</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>DADD LR5, LR5, 8</td>
<td>DADD PR36, PR5, 8</td>
<td>LR5-&gt;PR36</td>
<td>i+2</td>
<td>i+7</td>
<td>i+10</td>
<td>LR5-&gt;PR36</td>
<td>PR5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>BNE LR2, LR4, line1</td>
<td>BNE PR35, PR4, line1</td>
<td>-</td>
<td>i+3</td>
<td>i+8</td>
<td>i+11</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>LD LR1, 0(LR2)</td>
<td>LD PR37, 0(PR35)</td>
<td>LR1-&gt;PR37</td>
<td>i+2</td>
<td>i+11</td>
<td>i+11</td>
<td>LR1-&gt;PR37</td>
<td>PR34</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>DADD LR1, LR1, LR3</td>
<td>DADD PR38, PR37, PR3</td>
<td>LR1-&gt;PR38</td>
<td>i+5</td>
<td>i+10</td>
<td>i+11</td>
<td>LR1-&gt;PR38</td>
<td>PR37</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>ST.D LR1, 0(LR5)</td>
<td>ST.D PR38, 0(PR36)</td>
<td>-</td>
<td>i+6</td>
<td>i+12</td>
<td>i+12</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>DADD LR2, LR2, LR8</td>
<td>DADD PR1, PR35, 8</td>
<td>LR2-&gt;PR1</td>
<td>i+8</td>
<td>i+9</td>
<td>i+14</td>
<td>LR2-&gt;PR1</td>
<td>PR35</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>DADD LR5, LR5, LR8</td>
<td>DADD PR33, PR36, 8</td>
<td>LR5-&gt;PR33</td>
<td>i+10</td>
<td>i+15</td>
<td>i+15</td>
<td>LR5-&gt;PR33</td>
<td>PR36</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>BNE LR2, LR4, line1</td>
<td>BNE PR1, PR4, line1</td>
<td>-</td>
<td>i+10</td>
<td>i+15</td>
<td>i+15</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>
Problem 4

• Consider the following LSQ and when operands are available. Estimate when the address calculation and memory accesses happen for each ld/st. Assume memory dependence prediction.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LD</td>
<td>R1 ← [R2]</td>
<td>3</td>
<td>abcd</td>
<td></td>
</tr>
<tr>
<td>LD</td>
<td>R3 ← [R4]</td>
<td>6</td>
<td>adde</td>
<td></td>
</tr>
<tr>
<td>ST</td>
<td>R5 → [R6]</td>
<td>4 7</td>
<td>abba</td>
<td></td>
</tr>
<tr>
<td>LD</td>
<td>R7 ← [R8]</td>
<td>2</td>
<td>abce</td>
<td></td>
</tr>
<tr>
<td>ST</td>
<td>R9 → [R10]</td>
<td>8 3</td>
<td>abba</td>
<td></td>
</tr>
<tr>
<td>LD</td>
<td>R11 ← [R12]</td>
<td>1</td>
<td>abba</td>
<td></td>
</tr>
</tbody>
</table>
Problem 1

• Memory access time: Assume a program that has cache access times of 1-cyc (L1), 10-cyc (L2), 30-cyc (L3), and 300-cyc (memory), and MPKIs of 20 (L1), 10 (L2), and 5 (L3). Should you get rid of the L3?

With L3: $1000 + 10 \times 20 + 30 \times 10 + 300 \times 5 = 3000$
Without L3: $1000 + 10 \times 20 + 10 \times 300 = 4200$
Problem 3

• Assume a 2-way set-associative cache with just 2 sets. Assume that block A maps to set 0, B to 1, C to 0, D to 1, E to 0, and so on. For the following access pattern, estimate the hits and misses:

A B B E C C A D B F A E G C G A
M MH M MH MM HM HMM M H M
Problem 5

• 8 KB fully-associative data cache array with 64 byte line sizes, assume a 40-bit address
• How many sets (1) ? How many ways (128) ?
• How many index bits (0), offset bits (6), tag bits (34) ?
• How large is the tag array (544 bytes) ?

Equations:
Data array size (cache size) = #sets x #ways x blocksize
Tag array size = #sets x #ways x tagsize
Index bits = log₂ (#sets)
Offset bits = log₂ (blocksize)
Tag bits + index bits + offset bits = address width
Problem 3

- Assume that page size is 16KB and cache block size is 32 B. If I want to implement a virtually indexed physically tagged L1 cache, what is the largest direct-mapped L1 that I can implement? What is the largest 2-way cache that I can implement?
HW 7, Q1

• Assume a large shared LLC that is tiled and distributed on the chip. Assume that the OS page size is 16KB. The entire LLC has a size of 32 MB, uses 64-byte blocks, and is 32-way set-associative. What is the maximum number of tiles such that the OS has full flexibility in placing a page in a tile of its choosing?
Problem 1

• What is the maximum memory capacity supported by the following server: 2 processor sockets, each socket has 4 memory channels, each channel supports 2 dual-ranked DIMMs, and x4 4Gb DRAM chips?

2 sockets x 4 channels x 2 DIMMs x 2 ranks x 16 chips x 4Gb capacity = 256 GB

What is the memory bandwidth available to the server if each memory channel runs at 800 MHz?
2 sockets x 4 channels x 800M (cycles per second) x 2 (DDR, hence 2 transfers per cycle) x 64 (bits per transfer) = 102.4 GB/s
Problem 4

For the following access stream, estimate the finish times for each access with the following scheduling policies:

<table>
<thead>
<tr>
<th>Req</th>
<th>Time of arrival</th>
<th>Open</th>
<th>Closed</th>
<th>Oracular</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>10 ns</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>X+1</td>
<td>15 ns</td>
<td>70</td>
<td>70</td>
<td>70</td>
</tr>
<tr>
<td>Y</td>
<td>100 ns</td>
<td>160</td>
<td>140</td>
<td>140</td>
</tr>
<tr>
<td>Y+1</td>
<td>180 ns</td>
<td>200</td>
<td>220</td>
<td>200</td>
</tr>
<tr>
<td>X+2</td>
<td>190 ns</td>
<td>260</td>
<td>300</td>
<td>260</td>
</tr>
<tr>
<td>Y+2</td>
<td>205 ns</td>
<td>320</td>
<td>240</td>
<td>320</td>
</tr>
</tbody>
</table>

Note that X, X+1, X+2, X+3 map to the same row and Y, Y+1 map to a different row in the same bank. Ignore bus and queuing latencies. The bank is precharged at the start.

** A more sophisticated oracle can do even better.
Problem 5

- Consider a single 4 GB memory rank that has 8 banks. Each row in a bank has a capacity of 8KB. On average, it takes 40ns to refresh one row. Assume that all 8 banks can be refreshed in parallel. For what fraction of time will this rank be unavailable? How many rows are refreshed with every refresh command?

The memory has $\frac{4\text{GB}}{8\text{KB}} = 512\text{K}$ rows
There are 8K refresh operations in one 64ms interval. Each refresh operation must handle $\frac{512\text{K}}{8\text{K}} = 64$ rows
Each bank must handle 8 rows
One refresh operation is issued every 7.8us and the memory is unavailable for 320ns, i.e., for 4% of time.
Meltdown

**Attacker code**
Fill the cache with your own data $X$

```
lw  R1  \leftarrow [\text{illegal address}] 
lw  \ldots \leftarrow [\text{R1}] 
```

Scan through $X$ and record time per access
Spectre: Variant 1

if (x < array1_size)
    y = array2[array1[x]];

x is controlled by attacker

Victim Code

Access pattern of array2[] betrays the secret

Thanks to bpred, x can be anything

array1[] is the secret
Spectre: Variant 2

Attacker code

Label0: if (1)

Label1: ...

Victim code

R1 ← (from attacker)
R2 ← some secret
Label0: if (...) ...

Victim code

Label1:

lw [R2]

...
## Snooping Example

<table>
<thead>
<tr>
<th>Request</th>
<th>Cache Hit/Miss</th>
<th>Request on the bus</th>
<th>Who responds</th>
<th>State in Cache 1</th>
<th>State in Cache 2</th>
<th>State in Cache 3</th>
<th>State in Cache 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1: Rd X</td>
<td>Miss</td>
<td>Rd X</td>
<td>Memory</td>
<td>Inv</td>
<td>Inv</td>
<td>Inv</td>
<td>Inv</td>
</tr>
<tr>
<td>P2: Rd X</td>
<td>Miss</td>
<td>Rd X</td>
<td>Memory</td>
<td>S</td>
<td>Inv</td>
<td>Inv</td>
<td>Inv</td>
</tr>
<tr>
<td>P2: Wr X</td>
<td>Perms Miss</td>
<td>Upgrade X</td>
<td>No response. Other caches invalidate.</td>
<td>Inv</td>
<td>M</td>
<td>Inv</td>
<td>Inv</td>
</tr>
<tr>
<td>P3: Wr X</td>
<td>Write Miss</td>
<td>Wr X</td>
<td>P2 responds</td>
<td>Inv</td>
<td>Inv</td>
<td>M</td>
<td>Inv</td>
</tr>
<tr>
<td>P3: Rd X</td>
<td>Read Hit</td>
<td>-</td>
<td>-</td>
<td>Inv</td>
<td>Inv</td>
<td>M</td>
<td>Inv</td>
</tr>
<tr>
<td>P4: Rd X</td>
<td>Read Miss</td>
<td>Rd X</td>
<td>P3 responds. Mem wrtbk</td>
<td>Inv</td>
<td>Inv</td>
<td>S</td>
<td>S</td>
</tr>
</tbody>
</table>
# Directory Example

<table>
<thead>
<tr>
<th>Request</th>
<th>Cache Hit/Miss</th>
<th>Messages</th>
<th>Dir State</th>
<th>State in C1</th>
<th>State in C2</th>
<th>State in C3</th>
<th>State in C4</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1: Rd X</td>
<td>Miss</td>
<td>Rd-req to Dir. Dir responds.</td>
<td>X: S: 1</td>
<td>S</td>
<td>Inv</td>
<td>Inv</td>
<td>Inv</td>
</tr>
<tr>
<td>P2: Rd X</td>
<td>Miss</td>
<td>Rd-req to Dir. Dir responds.</td>
<td>X: S: 1, 2</td>
<td>S</td>
<td>S</td>
<td>Inv</td>
<td>Inv</td>
</tr>
<tr>
<td>P2: Wr X</td>
<td>Perms Miss</td>
<td>Upgr-req to Dir. Dir sends INV to P1. P1 sends ACK to Dir. Dir grants perms to P2.</td>
<td>X: M: 2</td>
<td>Inv</td>
<td>M</td>
<td>Inv</td>
<td>Inv</td>
</tr>
<tr>
<td>P3: Wr X</td>
<td>Write Miss</td>
<td>Wr-req to Dir. Dir fwds request to P2. P2 sends data to Dir. Dir sends data to P3.</td>
<td>X: M: 3</td>
<td>Inv</td>
<td>Inv</td>
<td>M</td>
<td>Inv</td>
</tr>
<tr>
<td>P3: Rd X</td>
<td>Read Hit</td>
<td>-</td>
<td>-</td>
<td>Inv</td>
<td>Inv</td>
<td>M</td>
<td>Inv</td>
</tr>
<tr>
<td>P4: Rd X</td>
<td>Read Miss</td>
<td>Rd-req to Dir. Dir fwds request to P3. P3 sends data to Dir. Memory wrtbk. Dir sends data to P4.</td>
<td>X: S: 3, 4</td>
<td>Inv</td>
<td>Inv</td>
<td>S</td>
<td>S</td>
</tr>
</tbody>
</table>
Test-and-Test-and-Set

- lock: test register, location
  bnz register, lock
  t&s register, location
  bnz register, lock
  CS
  st location, #0
Spin Lock with Low Coherence Traffic

lockit:    LL    R2, 0(R1) ; load linked, generates no coherence traffic
          BNEZ   R2, lockit ; not available, keep spinning
          DADDUI R2, R0, #1 ; put value 1 in R2
          SC     R2, 0(R1) ; store-conditional succeeds if no one
                 ; updated the lock since the last LL
          BEQZ   R2, lockit ; confirm that SC succeeded, else keep trying

• If there are \( i \) processes waiting for the lock, how many bus transactions happen?
  1 write by the releaser + \( i \) (or 1) read-miss requests +
  \( i \) (or 1) responses + 1 write by acquirer + 0 (i-1 failed SCs) +
  i-1 (or 1) read-miss requests + i-1 (or 1) responses

(The \( i/i-1 \) read misses can be reduced to 1)
Example Programs

Initially, A = B = 0

P1
A = 1
if (B == 0)
critical section

P2
B = 1
if (A == 0)
critical section

Initially, Head = Data = 0

P1
Data = 2000
while (Head == 0)
Head = 1
{ }

P2
... = Data

Initially, A = B = 0

P1
A = 1
if (A == 1)
B = 1
if (B == 1)
register = A

P2

P3

P1

P2

P3
Problem 1

• What are possible outputs for the program below?

Assume x=y=0 at the start of the program

Thread 1                                      Thread 2
A      x = 10                                    a      y=20
B      y = x+y  b      x = y+x
C      Print y

Possible scenarios:  5 choose 2 = 10
  ABCab  ABaCb  ABabC  AaBCb  AaBbC
  10    20    20    30    30
  AabBC  aABCb  aABbC  aAbBC  abABC
  50    30    30    50    30
Fences

P1
{
    Region of code
    with no races
}
Fence
Acquire_lock
Fence
{
    Racy code
}
Fence
Release_lock
Fence

P2
{
    Region of code
    with no races
}
Fence
Acquire_lock
Fence
{
    Racy code
}
Fence
Release_lock
Fence
Deadlock

- Deadlock happens when there is a cycle of resource dependencies – a process holds on to a resource (A) and attempts to acquire another resource (B) – A is not relinquished until B is acquired.
# Topology Examples

## Grid

- **Criteria:** 64 nodes
- **Performance:**
  - Diameter: 1
  - Bisection BW: 1
- **Cost:**
  - Ports/switch: 3
  - Total links: 64

## Torus

## Hypercube

- **Criteria:** 64 nodes
- **Performance:**
  - Diameter: 8
  - Bisection BW: 16
- **Cost:**
  - Ports/switch: 5
  - Total links: 128

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Bus</th>
<th>Ring</th>
<th>2Dtorus</th>
<th>Hypercube</th>
<th>Fully connected</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>64 nodes</strong></td>
<td>1</td>
<td>32</td>
<td>8</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diameter</td>
<td>1</td>
<td>2</td>
<td>16</td>
<td>32</td>
<td>1024</td>
</tr>
<tr>
<td>Bisection BW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Cost</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ports/switch</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>64</td>
</tr>
<tr>
<td>Total links</td>
<td>1</td>
<td>64</td>
<td>128</td>
<td>192</td>
<td>2016</td>
</tr>
</tbody>
</table>
k-ary d-Cube

• Consider a k-ary d-cube: a d-dimension array with k elements in each dimension, there are links between elements that differ in one dimension by 1 (mod k)

• Number of nodes \( N = k^d \)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of switches</td>
<td>( N )</td>
<td>Avg. routing distance</td>
<td>( d(k-1)/4 )</td>
</tr>
<tr>
<td>Switch degree</td>
<td>( 2d + 1 )</td>
<td>Diameter</td>
<td>( d(k-1)/2 )</td>
</tr>
<tr>
<td>Number of links</td>
<td>( Nd )</td>
<td>Bisection bandwidth</td>
<td>( 2wk^{d-1} )</td>
</tr>
<tr>
<td>Pins per node</td>
<td>( 2wd )</td>
<td>Switch complexity</td>
<td>( (2d + 1)^2 )</td>
</tr>
</tbody>
</table>

The switch degree, num links, pins per node, bisection bw for a hypercube are half of what is listed above (diam and avg routing distance are twice, switch complexity is \( (d + 1)^2 \) ) because unlike the other cases, a hypercube does not have right and left neighbors.

Should we minimize or maximize dimension?
Problem 1

Assume that a server consumes 100W at peak utilization and 50W at zero utilization. Assume a linear relationship between utilization and power. The server is capable of executing many threads in parallel. Assume that a single thread utilizes 25% of all server resources (functional units, caches, memory capacity, memory bandwidth, etc.). What is the total power dissipation when executing 99 threads on a collection of these servers, such that performance and energy are close to optimal?

For near-optimal performance and energy, use 25 servers. 24 servers at 100% utilization, executing 96 threads, consuming 2400W. The 25th server will run the last 3 threads and consume 87.5~W.
RAID 4 and RAID 5

• Data is block interleaved – this allows us to get all our data from a single disk on a read – in case of a disk error, read all 9 disks

• Block interleaving reduces throughput for a single request (as only a single disk drive is servicing the request), but improves task-level parallelism as other disk drives are free to service other requests

• On a write, we access the disk that stores the data and the parity disk – parity information can be updated simply by checking if the new data differs from the old data
The GPU Architecture
Weights are pre-loaded during previous phase and inputs flow left to right.
Practice Questions

Q1. Describe how errors are detected, localized, and corrected in a RAID implementation. ("localized" refers to the identification of the location of the possibly erroneous values)

Q2. How does efficiency/performance vary as we move from RAID-3 to RAID-4 to RAID-5?

Q3. See the server consolidation example problem we solved last Wednesday.

Q4. In a 256-node system, what is the expected improvement in average routing distance if I go from a torus topology to a hypercube?

Q5. Describe a couple of hardware features that allow a GPU to achieve high compute density when executing a highly parallel workload.

Q6. What structure allows a Google TPU to efficiently execute several multiply-accumulate operations in parallel?

Q7. Mention a few architectural features seen in the Tesla FSD chip, but not in the Google TPU.