CS 152: Discussion Section 3
PS1 Review and Memory Hierarchy

Yue Dai, Albert Ou
02/07/2020
Administrivia

- Lab 1 due on **Wednesday, Feb 19th**. Submit direct portion and open portion separately
- PS2 released and due on Feb 26th. Review lecture slides and handout 2 first!
- No lecture on Monday. Enjoy!
Temporary OH Rescheduling

Changes apply only to next week due to ISSCC

- Albert:
  - 3:00-4:30pm (529 Cory) on Tues, Feb 18
  - 4:30-6:00pm (347 Soda) on Tues, Feb 18
- Yue: 11am-12pm (258 Cory) on Thu, Feb 20

(The processors session looks more interesting this year - we’ll do a fun recap next section)
Agenda

- PS1 Review
- Memory Hierarchy
DRAM Organization

Channel → DIMM → Chip → Bank → Subarray → Cell
Cache Organization

Consider a small 4-way set associative cache with cache size 1KiB and 32-byte cache lines. The output data is a 32-bit word (4 bytes per word). The address is 12 bits wide. How are the address bits partitioned?

Tag:

Index:

Offset
Replacement Policy

- Least recently used (LRU) (exercise in PS2)
- FIFO (exercise in PS2)
- Not most recently used (NMRU)
- Pseudo least recently used (PLRU)
Replacement Policy - Exercise

Fill out the form on worksheet with the address bit partition you got from Q1.1.

Q: What’s the last state of PLRU tree?
<table>
<thead>
<tr>
<th>Cache Optimization</th>
<th>Hit Time</th>
<th>Miss Penalty</th>
<th>Miss Rate</th>
<th>Hardware Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small and simple caches</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-level caches</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Smart replacement policy</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pipelined Write</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write buffer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sub-blocks</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code optimization</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Cache Optimization

<table>
<thead>
<tr>
<th></th>
<th>Hit Time</th>
<th>Miss Penalty</th>
<th>Miss Rate</th>
<th>Hardware Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compiler prefetching</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hardware prefetching (Stream buffer)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Victim cache</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Cache Tiling - MatMul without tiling

Give a worst-case expression for the number of cache misses, assuming that the matrix sizes (N x N) are several orders of magnitude larger than the cache and one cache line holds one word:

```c
for (i=0; i < N; i++) {
    for (j=0; j < N; j++) {
        r = 0;
        for (k=0; k < N; k++)
            r = r + y[i][k] * z[k][j];
        x[i][j] = r;
    }
}
```
Cache Tiling - MatMul without tiling

\[
x \times y = z
\]
Cache Tiling - MatMul without tiling

\( x \times y = z \)
Cache Tiling - MatMul without tiling

\[
x \times y = z
\]
Cache Tiling - MatMul without tiling

\[ x \times y = z \]
Cache Tiling - MatMul without tiling

\[ x \times y = z \]
Cache Tiling - MatMul without tiling

\[ x \times y \times z \]
Cache Tiling - MatMul without tiling

\[
x \times y = z
\]
Cache Tiling - MatMul without tiling

\[ x \times y = z \]
Cache Tiling - MatMul without tiling

Because the matrix size is way larger than cache size, then for the worst case, each word (element) in matrix Y and Z will be accessed N times. But each word in X will be only accessed once. So for X, the cache misses should be $N^2$, and for both Y and Z, the worst case misses should be $(N^2)*N$.
Cache Tiling - MatMul with tiling

With cache tiling, give an expression for the number of cache misses, assuming the tiling size (B x B) is on the same order of magnitude as the cache capacity and there are no conflict misses:

```c
for(jj=0; jj < N; jj=jj+B) {
    for(kk=0; kk < N; kk=kk+B) {
        for(i=0; i < N; i++) {
            for(j=jj; j < min(jj+B,N); j++) {
                r = 0;
                for(k=kk; k < min(kk+B,N); k++)
                    r = r + y[i][k] * z[k][j];
                x[i][j] = x[i][j] + r;
            }
        }
    }
}
```
Cache Tiling - MatMul with tiling

With cache tiling, give an expression for the number of cache misses, assuming the tiling size (BxB) is on the same order of magnitude as the cache capacity and there are no conflict misses:

```c
for(jj=0; jj < N; jj=jj+B) {
    for(kk=0; kk < N; kk=kk+B) {
        for(i=0; i < N; i++){
            for(j=jj; j < min(jj+B,N); j++) {
                r = 0;
                for(k=kk; k < min(kk+B,N); k++)
                    r = r + y[i][k] * z[k][j];
                x[i][j] = x[i][j] + r;
            }
        }
    }
}
```
Cache Tiling - MatMul with tiling

\[
x \times y = z
\]
Cache Tiling - MatMul without tiling

\[ x = y \times z \]
Cache Tiling - MatMul without tiling

\[
\begin{array}{c}
x \\
\end{array}
\begin{array}{c}
= \\
\end{array}
\begin{array}{c}
y \\
\end{array}
\begin{array}{c}
* \\
\end{array}
\begin{array}{c}
z \\
\end{array}
\]
Cache Tiling - MatMul with tiling

\[ x \times y = z \]
Cache Tiling - MatMul with tiling

\[
x \quad = \quad y \quad * \quad z
\]
Cache Tiling - MatMul without tiling

\[ x \times y = z \]
Cache Tiling - MatMul without tiling

\[
x \times y = z
\]
Cache Tiling - MatMul with tiling

\[ x \times y = z \]
Cache Tiling - MatMul with tiling

From the above animation we can see that the BxB block in Z matrix can be reused in the most inner loop. Also, the sub-block of Y matrix (the blue blocks) can also be reused in the most inner loop. And same for X.

As a result, the number of misses for X is \((N/B)^2 \times N \times B = \frac{(N^2)}{B}\). The number of misses for Y is \((N/B)^2 \times N \times B = \frac{(N^2)}{B}\). The number of misses for Z is \(N^2\) (only compulsory misses).