Virtual Memory

1. What are 3 specific benefits of using virtual memory?
   
   1. Bridges memory and disk in memory hierarchy.
   2. Simulates full address space for each process.
   3. Enforces protection between processes.

2. What should happen to the TLB when a new value is loaded into the page table address register?
   
   The valid bits of the TLB should all be set to 0. The page table entries in the TLB correspond to the old page table, so none of them are valid once the page table address register points to a different page table.

3. x86 has an "accessed” bit in each page table entry, which is like the dirty bit but set whenever a page is used (load or store). Why is this helpful when using memory as a cache for disk?
   
   It allows smarter replacements. We naturally want fewer misses (page faults), so if possible, we would want to replace a page table entry that hasn't been used. The "accessed” bit is one way of giving us enough information to implement this.

4. Fill out the table!

<table>
<thead>
<tr>
<th>Virtual Address Bits</th>
<th>Physical Address Bits</th>
<th>Page Size</th>
<th>VPN Bits</th>
<th>PPN Bits</th>
<th>Bits per row of PT</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>32</td>
<td>16 KiB</td>
<td>18</td>
<td>18</td>
<td>22</td>
</tr>
<tr>
<td>32</td>
<td>26</td>
<td>8 KiB</td>
<td>19</td>
<td>13</td>
<td>17</td>
</tr>
<tr>
<td>36</td>
<td>32</td>
<td>32 KiB</td>
<td>21</td>
<td>17</td>
<td>21</td>
</tr>
<tr>
<td>40</td>
<td>36</td>
<td>32 KiB</td>
<td>25</td>
<td>21</td>
<td>25</td>
</tr>
<tr>
<td>64</td>
<td>40</td>
<td>64 KiB</td>
<td>48</td>
<td>24</td>
<td>28</td>
</tr>
</tbody>
</table>

Hamming ECC

Recall the basic structure of a Hamming code. Given bits $1, \ldots, m$, the bit at position $2^n$ is parity for all the bits with a 1 in position $n$. For example, the first bit is chosen such that the sum of all odd-numbered bits is even.

<table>
<thead>
<tr>
<th>Bit</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>P1</td>
<td>P2</td>
<td>D1</td>
<td>P4</td>
<td>D2</td>
<td>D3</td>
<td>D4</td>
<td>P8</td>
<td>D5</td>
<td>D6</td>
<td>D7</td>
<td>D8</td>
<td>D9</td>
<td>D10</td>
<td>D11</td>
</tr>
<tr>
<td>P1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>P2</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>P4</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>P8</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

1. How many bits do we need to add to $0011_2$ to allow single error correction?
   
   Parity Bits: 3

2. Which locations in $0011_2$ would parity bits be included?
   
   Using P for parity bits: PP0P011

3. Which bits does each parity bit cover in $0011_2$?
   
   Parity bit #1: 1, 3, 5, 7
   Parity bit #2: 2, 3, 6, 7
   Parity bit #3: 4, 5, 6, 7

4. Write the completed coded representation for $0011_2$ to enable single error correction.
5. How can we enable an additional double error detection on top of this?

Add an additional parity bit over the entire sequence.

6. Find the original bits given the following SEC Hamming Code: \(0110111_2\)

Parity group 1: error
Parity group 2: okay
Parity group 4: error
Incorrect bit: \(1 + 4 = 5\), change bit 5 from 1 to 0: \(0110011_2\)

7. Find the original bits given the following SEC Hamming Code: \(1001000_2\)

Parity group 1: error
Parity group 2: okay
Parity group 4: error
Incorrect bit: \(1 + 4 = 5\), change bit 5 from 1 to 0: \(1001100_2\)

8. Find the original bits given the following SEC Hamming Code: \(010011010000110_2\)

Parity group 1: okay
Parity group 2: error
Parity group 4: okay
Parity group 8: error
Incorrect bit: \(2 + 8 = 10\), change bit 10 from 0 to 1: \(01001101000110_2\)
\(0100110100110_2 \rightarrow 01100100110_2\)