1. **Paging Scheme with 4MB pages**

   The Pentium 4 processor offers both a mode for using 4KB page sizes (used by default) and a newer mode for using a page size of 4MB. How would our basic paging diagram, reproduced here from the lecture slides, have to be modified for the large page sizes?

   What would the size of a process page table be?
2. Worked Paging Example with TLB
Below the general scheme for paging with TLB is reproduced. Work through the same example as done in the lecture notes (for decimal 12291), indicating the correct entries for address, page table, TLB and memory.

<table>
<thead>
<tr>
<th>Page Table</th>
<th>Main Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pr MP</td>
<td></td>
</tr>
<tr>
<td>Physical Page Number</td>
<td></td>
</tr>
<tr>
<td>Index gives Line Number</td>
<td></td>
</tr>
<tr>
<td>Index plus Tag gives Line Number</td>
<td></td>
</tr>
</tbody>
</table>

TLB Tag | Index | Page Offset |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>18 17 12 11 0</td>
<td></td>
</tr>
</tbody>
</table>

3. Suppose we now next access decimal address 274435 (= 12291 + 4096*64)? What will happen? Indicate on the diagram the steps (in order) that the processor will take.
4. TLB Missrates

2-dimensional arrays in C are typically laid out in memory in row-major order. This means that for an NxN array, element A[i][j] is stored at location A_base + i*N+j. This is illustrated in the following drawing:

What is the expected TLB miss rate on a Pentium 4 when traversing this array in the following two loops?

**Loop 1**
```c
int I, J, a[1024][1024]
for( I = 0; I < 1024; ++I ) {
    for( J = 0; J < 1024; ++J ) {
        a[I][J] = 0;
    }
}
```

**Loop 2**
```c
int I, J, a[1024][1024]
for( I = 0; I < 1024; ++I ) {
    for( J = 0; J < 1024; ++J ) {
        a[J][I] = 0;
    }
}
```