1. Consider the demand paging system with 1024-byte pages.

<table>
<thead>
<tr>
<th>Running Process B</th>
<th>Page Table for B</th>
<th>Valid</th>
<th>Frame Number</th>
<th>Physical Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Frame#</td>
<td>Bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td>page 2 of B</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td>page 3 of A</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td>page 5 of A</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td>page 5 of B</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
<td>page 0 of A</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td></td>
<td></td>
<td>page 2 of A</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>page 4 of B</td>
</tr>
</tbody>
</table>

**Logical Addr.**

**Physical Addr.**

a) Complete the above page table for **Process B**.

b) If process B is currently running and the CPU generates a logical/virtual address of \(2060_{10}\), then what would be the corresponding physical address?

2. For a 32-bit address machine with 4 KB \((2^{12}\) bytes) pages and 4 byte page-table entries, how big would the page table be?

3. How does a TLB (translation-lookaside buffer) speed the process of address translation?
Design issues for Paging Systems

Conflicting Goals:

- Want as many (partial) processes in memory (high degree of multiprogramming) as possible so we have better CPU & I/O utilization ⇒ allocate as few page frames as possible to each process
- Want as low of page-fault rate as possible ⇒ allocate enough page frames to hold all of a process’ current working set (which is dynamic as a process changes locality)

4. Explain the shape of each section indicated on the above curve:
   a) (rising part of the curve)
   b) (falling part of the curve)

5. There are many similarities between the cache-memory level and memory-disk level of the memory hierarchy, but there are also important differences. For example, a cache miss stalls the running program temporarily, but a page fault causes the running program to turnover the CPU to another program. Why are these cases treated differently by the computer system?