Question 7: Virtual Memory

Our system has the following setup:

- 24-bit virtual addresses and 512 KiB of RAM with 4 KiB pages
- A 4-entry TLB that is fully associative with LRU replacement
- A page table entry contains a valid bit and protection bits for read (R), write (W), execute (X)

(A) Compute the following values: [2 pt]

<table>
<thead>
<tr>
<th>Page offset width</th>
<th>PPN width</th>
<th>Entries in a page table</th>
<th>TLBT width</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(B) Briefly explain why we make the page size so much larger than a cache block size.

(C) Fill in the following blanks with “A” for always, “S” for sometimes, and “N” for never if the following get updated during a page fault.

<table>
<thead>
<tr>
<th>Page table</th>
<th>Swap space</th>
<th>TLB</th>
<th>Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(D) The TLB is in the state shown when the following code is executed. Which iteration (value of i) will cause the protection fault (segfault)? Assume sum is stored in a register.

Recall: the hex representations for TLBT/PPN are padded as necessary.

```
long *p = 0x7F0000, sum = 0;
for (int i = 0; 1; i++) {
    if (i%2)
        *p = 0;
    else
        sum += *p;
    p++;
}
```