Problem 1: how many physical loads for each instruction?

Assume 256-byte pages.
Assume 16-bit addresses.
Assume ASID of current process is 211.
Assume this TLB state:

<table>
<thead>
<tr>
<th>ASID</th>
<th>VPN</th>
<th>PFN</th>
<th>valid</th>
<th>Prot</th>
</tr>
</thead>
<tbody>
<tr>
<td>211</td>
<td>0xBB</td>
<td>0x91</td>
<td>1</td>
<td>???</td>
</tr>
<tr>
<td>211</td>
<td>0xFF</td>
<td>0x23</td>
<td>1</td>
<td>???</td>
</tr>
<tr>
<td>112</td>
<td>0x05</td>
<td>0x91</td>
<td>1</td>
<td>???</td>
</tr>
<tr>
<td>211</td>
<td>0x05</td>
<td>0x12</td>
<td>0</td>
<td>???</td>
</tr>
</tbody>
</table>

How many physical accesses are needed for each instruction?
Consider each instruction independently (e.g., for “b”, don’t worry how “a” changes the TLB).

(a) 0xA10: movl 0x1111, %edi
(b) 0xBB13: addl $0x3, %edi
(c) 0x0519: movl %edi, 0xFF10

Other questions:
(d) what do the “prot” bits probably contain for the first TLB entry?
(e) why do two entries map to the same physical page, 0x91?
(f) which fields in the TLB would also appear in a PT? Do any have a different meaning?
(g) would accessing 0x0500 cause a segfault?

Problem 2: how large must PTE’s be?

(a) Phys addr space contains 1024 pages. 7 bits needed for extras (prot, valid, etc.)
(b) Phys addrs are 16-bits, pages are 32 bytes, 8 bits needed for extras
(c) Phys addr space is 4 GB, pages are 4 KB, 6 bits needed for extras

Problem 3 how large is the page table?

(a) PTE’s are 3 bytes, and there are 32 possible virtual page numbers
(b) PTE’s are 3 bytes, virtual addrs are 24 bits, and pages are 16 bytes
(c) PTE’s are 4 bytes, virtual addrs are 32 bits, and pages are 4 KB
(d) PTE’s are 4 bytes, virtual addrs are 64 bits, and pages are 4 KB
(e) assume each PTE is 10 bits. We cut the size of pages in half, keeping everything else fixed, including size of virt/phys addresses. By what factor does the PT increase in size?

Problem 4: how large should pages be?

(a) Goal PT size is 512 bytes. PTE’s are 4 bytes. Virtual addrs are 16 bits.
(b) Goal PT size is 1 KB. PTE’s are 4 bytes. Virtual addrs are 16 bits.
(c) Goal PT size is 4 KB. PTE’s are 4 bytes. Virtual addrs are 32 bits.
Problem 5: translate with segments over page tables

Translate the following (refer to slides):
(a) 0x12FFF =>
(b) 0x10FFF =>
(c) 0x01ABC =>
(d) 0x11111 =>

Problem 6: translate with page directory (of PT pieces)

(a) how many PT pages would have been needed if we instead had a simple, linear array?
(b) how many bits of a virt addr are used for the page-directory index?
(b) how many virtual pages are there?
(c) translate 0x01ABC
(d) translate 0x00000
(e) translate 0xFEED0
(f) does 0x10000 segfault?

Problem 7: how many levels are needed?

How large is the virtual addr space, assuming 4-KB pages and 4-byte entries with N levels? (PT can now no longer require more than 1 contiguous page for bookkeeping)
(a) N = 1
(b) N = 2
(c) N = 3

Problem 8: how many physical loads for each instruction (v2)?

Assume a 3-level page table.
Assume 256-byte pages.
Assume 16-bit addresses.
Assume ASID of current process is 211.
Assume this TLB state:

<table>
<thead>
<tr>
<th>ASID</th>
<th>VPN</th>
<th>PFN</th>
<th>valid</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>211</td>
<td>0xBB</td>
<td>0x91</td>
<td>1</td>
<td>???</td>
</tr>
<tr>
<td>211</td>
<td>0xFF</td>
<td>0x23</td>
<td>1</td>
<td>???</td>
</tr>
<tr>
<td>112</td>
<td>0x05</td>
<td>0x91</td>
<td>1</td>
<td>???</td>
</tr>
<tr>
<td>211</td>
<td>0x05</td>
<td>0x12</td>
<td>0</td>
<td>???</td>
</tr>
</tbody>
</table>

How many physical accesses are needed for each instruction? Consider each instruction independently (e.g., for “b”, don’t worry how “a” changes the TLB).
(a) 0xAA10: movl 0x1111, %edi
(b) 0xBB13: addl $0x3, %edi
(c) 0x0519: movl %edi, 0xFF10