Exercise 1

Below, you are provided with a snapshot of a 4 entry, fully associative TLB and a page table. The TLB uses a “least recently used” replacement policy and the entry that has not been used for the longest time will be evicted if a new entry is to be added
(The page size is 4 KB)

Initial TLB State:
(Note that ‘1’ = “Most Recently Used and ‘4’ = “Least Recently Used”)
(Note that ‘1’ in the Valid column indicates that the entry is valid)
(Note that ‘0’ in the Dirty column of the TLB indicates that data has NOT been modified)

| Valid | Dirty | LRU | Tag | Physical Page # |
|---|---|---|---|---|
| 1 | 0 | 3 | 0110 | 1000 |
| 1 | 0 | 4 | 0011 | 1101 |
| 1 | 0 | 2 | 1000 | 0110 |
| 1 | 0 | 1 | 0100 | 1010 |

Page Table

| Address | Valid | Physical Page # |
|---|---|---|
| 0000 | 1 | 1110 |
| 0001 | 0 | Disk |
| 0010 | 1 | 1100 |
| 0011 | 1 | 1101 |
| 0100 | 1 | 1010 |
| 0101 | 1 | 1011 |
| 0110 | 1 | 1000 |
| 0111 | 1 | 1001 |
| 1000 | 1 | 0110 |
| 1001 | 1 | 0111 |
| 1010 | 1 | 0100 |
| 1011 | 0 | Disk |
| 1100 | 0 | Disk |
| 1101 | 1 | 0011 |
| … | … | … |

Show the final state of the TLB after processing the virtual address sequence (2 items) given below
(If there is a page fault, the physical page number to be entered into the page table would be 0000; if there is another page fault, use physical page number 0001

1011000000010010 (this is a load instruction)
1011110111101111 (this is a store instruction)

Exercise 2

Assume that you have 4 Gbytes of main memory at your disposal

1 Gbyte of the 4 Gbytes has been reserved for process page table storage. Each page table entry consists of:

1) A physical frame number, 1 valid bit, 1 dirty bit, 1 LRU status bit
2) Virtual addresses are 32 bits
3) Physical addresses are 26 bits
4) The page size is 8 Kbytes

How many process page tables can fit in the 1 Gbyte space?

Exercise 3

Describe in detail how a full link of the QPI is made of (all signals, number of wires etc.). What is a quadrant?

Exercise 1

The final TLB state is as shown below:
The entry with tag 0011 has been evicted.
Other LRU bits have been updated accordingly
The fact that the last entry involves a write *will cause the dirty bit to be set to 1*

| Valid | Dirty | LRU | Tag | Physical Page # |
|---|---|---|---|---|
| 1 | 0 | 4 | 0110 | 1000 |
| **1** | **1** | **1** | **1011** | **0000** |
| 1 | 0 | 3 | 1000 | 0110 |
| 1 | 0 | 2 | 0100 | 1010 |

Exercise 2
We can first calculate the number of page table entries associated with each process (one page table per process int the virtual environment).

If the virtual address is 32 bit, 8 KB pages implies that the offset is 13 bits ($2^{13}$= 8KB). Thus, the remaining 19 bits are used to represent the virtual page numbers and serve as indices to the page table. Therefore each page table has $2^{19}$ entries. Each physical address (PA) is 26 bits: 13 bits of the PA are the offset (which in the physical and virtual addresses are the same), the other 13 come from the table page lookup
Given that each PT entry consists of a page physical number (13), a valid bit (1), a dirty bit (1), and a LRU bit (1), each page table entry is 2 bytes

1 Gbyte = $2^{30}$ bytes

Therefore one page table (page table of a process) occupies $2^{19}$ (number of PT entries per process) x $2^{1}$ bytes (bytes per entry) = $2^{20}$ bytes

Number of page tables (number of processes) = $2^{30}$ / $2^{20}$ = $2^{10}$ = 1024