Cache Part II Practice Questions

Using the "fully associative" approach, what's the size of the 'tag' field, 'block offset' field, and 'set index' field, provided we have the cache below, and assuming that our address bus is 32-bit long.

Tag	DATA							
	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte
	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte
	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte
	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte

Tag is 32 (address length) - $\log_2(8)$ (block offset) = 29 bits

Set index will always be 0 in the "fully associative" approach

Block offset is 3 bits ($\log_2$(8 — num of bytes in one row/block))

Using the "direct mapped" approach, what's the size of the 'tag' field, 'block offset' field, and 'set index' field, provided we have the same cache above

Tag is 32 (address length) - $\log_2(8)$ (block offset) - $\log_2(4)$ (set index)= 27 bits

Set index is $\log_2$(num of rows/blocks) = $\log_2(4) = 2$

Block offset is 3 bits ($\log_2(8)$)

Cache Part II Practice Questions

What's the advantages/disadvantages of the "fully associative" approach versus the "direct mapped" approach?

The "fully associative" approach:

Main disadvantage is that it requires more hardware, one comparator for every row/block

Main advantage is that the content of one address can live in any block

The "direct-mapped" approach:

Main disadvantage is that the content of one address can live in only one block

Main advantage is that the hardware is much simpler (only one comparator) regardless of how many rows/blocks we have

Why do we need a valid bit?

Answer is in the slides.

Given the table that represents how the cache looks like, the address of the instruction/data-point the CPU is requesting, and where the set index field is (which bits in the address); you should be able to identify the row/block that will contain the tag that will be compared with the tag from the address requested by the CPU. You should also be able to know what will happen depending on the value of the valid bit and the value of the tag at that row/block. There are two scenarios:

1) the data at that row/block will be sent from the cache to the CPU (this scenario will happen if the valid bit is 1, and if the tag portion of the address requested by the CPU is equal to the tag at the row/block activated by the set index decoder)

2) the cache will need to go to the RAM (or next level cache) to bring the data-point/instruction stored at that address and 1) load it in the cache 2) send it to the CPU. This scenario will happen if either the valid bit is 0, or if the tag from the CPU does not equal to the tag stored at the row/block activated by the set index decoder)