Module 10 Example Set 3

1. How does a regular or forward page map table differ from an inverse page map table?
A forward page map table (PMT) contains an entry (PTE) for each logical or virtual page while an inverted page map table contains an entry for each of the frames (i.e., physical pages). The number of frames = (physical memory size) / (frame size).

2. Given a specific physical address, how can the corresponding memory block number be computed?
For a given frame size N, the number of bits in the offset field is F = $\log_2 N$. The bits within the physical address to the left of this offset field contain the memory block number. This is equivalent to the integer quotient produced when the address is divided by the frame size.

3. What distinguishes a physical memory block from a logical memory page?
Logical pages are the same size as physical memory blocks, but the logical pages exist within the logical address space which may exceed the size of the physical address space. Physical memory blocks correspond to the actual hardware memory available in the system. A given logical page may be mapped into any available memory block. Adjacent logical pages do not have to reside within adjacent physical memory blocks. On a virtual memory system, logical pages are called virtual pages (or just pages) while physical memory blocks are called frames.

4. For a direct mapped cache, how would the cache line number be computed for a particular physical memory ?
Memory is thought of as being subdivided some number of blocks. Each block has the same size as a cache line and fits exactly into a single cache line. Given the memory block number B, the line to which the block maps = B % L (i.e., B modulo L) where L is the number of lines in the cache.

5. A logical address can be viewed as containing three fields for a direct mapped cache system. What determines the number of bits contained within each logical address?
The rightmost field is the offset within the selected line and requires a number of bits = $\log_2$(line size). The next field to the left contains the line number and requires a number of bits = $\log_2$(lines in cache). The next field to the left contains all of the remaining bits from the address and indicates the tag for the memory block that maps to the cache line.

6. What is meant by a “byte addressable” menory system?
This is one in which memory access instructions can reference items as small as individual bytes within the memory system. For a memory system of size 2^N, the address of the first byte is 0 and the address of the final byte is $2^N - 1$.

7. For a big-endian memory system that employs 4-byte words, to which byte within the word does the word address correspond?
For a big-endian system, the word address corresponds to the most significant (i.e., left-most) byte within the 4-byte word. On a little-endian system, the address would corrspond to the least significant (i.e., right-most) byte within the 4-byte word.

8. When a "load byte" or a "load half-word" instruction is executed on our MIPS system, the load byte places one byte into the low 8 bits of the destination register but the load half-word places two bytes into the low 16 bits within the destination register. What happens to the remaining bytes within the destination register?
The remaining bits within the register are filled with a copy of the MSB in the low byte if the instruction is either lb or lh, but if the instruction is either lbu (load byte unsigned) or lhu (load half-word unsigned) the remaining bits within the register are filled with 0s.

9. What is meant by an un-aligned memory reference?
This refers to an attempt to read or write a multi-byte item at a memory address that is not a multiple of the size of the multi-byte item. For example, an attempt to access a word at an address that is not a multiple of 4, or to access a half-word at an address that is not a multiple of 2.

10. a) A memory system contains 64 memory chips each of which has a width of 32 and a depth of 8388608. What is total storage capacity in bytes of this memory system?
Since each chip is 32 bits wide, each memory cell contains 4 bytes. The depth indicate how many cells are in each chip, hence each chip contains 4*8388608 = 33554432 bytes. The total number of bytes for the system = 64 * 33554432 = 2147483648 bytes (i.e., 2 GB).
b) By how much would the addresses of adjacent cells within each chip differ?
Since each cell is 4 bytes wide, the addresses of adjacent cells would differ by 4.

c) Assuming the system uses a 32-bit data bus, how many chips would have to be accessed to transfer one word?
If the address used is properly aligned (i.e., is a multiple of 4), then only one chip would have to be accessed. Using an address that is not a multiple of 4 would require accessing two chips since the bytes that comprise the word would span two chips.

d) Assuming the system uses 32-bit addresses and enforce memory alignment, how would the bits within each address be used to identify which cell to access?
Requiring that the address be a multiple of 4 means that the low 2 bits of the address would always be 00. Hence the low 2 bits need not be used. Since $\log_2(8388608) = 23$, the next 23 bits to the left would identify which cell in the selected chip to access. The next 6 bits to the left would select a particular chip (since there are $2^6 = 64$ chips). The 32-bit (i.e., 4-byte) cell contents would be transferred into the MDR (memory data register) and copied from the MDR into the destination register. Hence the address format would be:

Bit 31 = 0 since size is only 2 GB	Bits 30 to 25 = chip-select	Bits 24 to 2 = cell# within chip	Bits 1 to 0 = byte select within MDR

e) Why are unaligned memory references a problem or disadvantage for this system?
For a read the contents of the 32-bit cell would be read from the chip and transferred into the MDR (memory data register) and the low 2 bits of the address would be used to select which of the 4 bytes in the MDR to start transferring into the destination register.
If the starting byte is not on the proper boundary, the MDR may not all of the required bytes. For example, a properly aligned word address would correspond to byte offset 0 within the MDR and all 4 bytes would be copied from MDR into the desitination register; but if the word address is not aligned then at least one of the desired bytes would not be in the MDR. This is why an exception is triggerred to indicate the inability to successfully complete the operation.