1ai) -219

ii) 10 1101 1011

iii) 00 0101 0001 (can someone confirm this is correct because it seems wrong?)

iv) 1445

v) 325

b) Binary Coded Decimal; 1101 0010 0001 1001

ci) ceil(m/p \* n/q)

ii) ceil(m/p)

iii) lg(m\*n/8)

d) Low order interleaved memory can improve performance if the CPU is ‘advanced’ and can access adjacent memory locations in parallel.

High order interleaved memory can improve performance if the memory locations can be accessed independently by different units. If the units are using different modules then we will have parallel operations which increases performance.

Old answer:

Byte Addressable:

High Order = (100/4) div 16M (*length of module*) = 0 -> Module 0

Low Order = (100/4) mod 4 (*number of modules*) = 1 -> Module 1

Word Addressable (assuming 32-bit word):

High Order = 100 div 16M = Module 0

Low Order = 100 mod 4 = Module 0

From Piazza:

As shown in the discussion below, there should be 4 memory modules (0-3), so there should be 2 bits to determine which module a piece of data is stored on.

Using low-order interleave, we use the 2 **least** significant bits to determine which module the piece of data is stored on. So, using Byte Address 10010 = 11001002 the two least significant bits are 00 therefore the data will be stored on Module 0.

Using high-order interleave, we use the 2 **most** significant bits. So, to word address all 64M words we need 26 bits (64 \* 220 = 226) to address all the words, then since each 32-bit word contains 4 bytes, we need will 28 bits (226 \* 22 = 228) to address each byte. Extending 11001002 to 28-bits gives 0000 0000 0000 0000 0000 0110 01002 so the two most significant bits are 00 again so the data will still be stored in module 0.