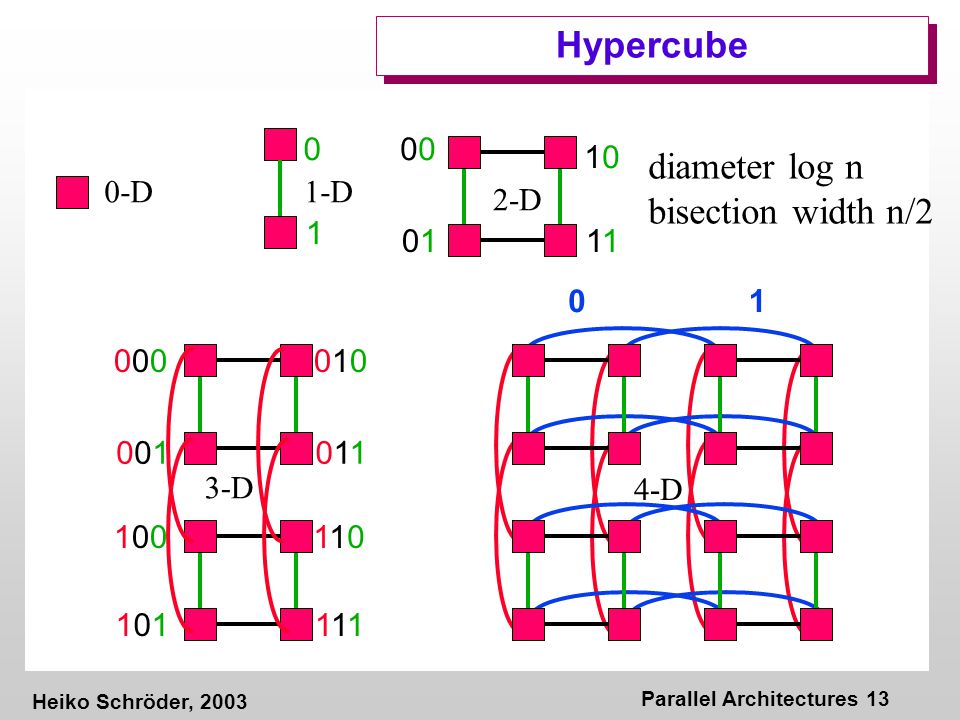
1. Suppose that fetch and store each take 2 nanoseconds and the remaining operations each take 1 nanosecond
2. 1 floating point addition = 7 operations (including 1 fetch and 1 store) = 5\*1 + 2\*2 = 9 nanoseconds
3. 1000 floating point additions = 1000 \* 9 = 9000 nanoseconds
4. 1000 pipelined floating point addition = 9 + 2 \* 999 = 2007  
   the first addition takes 9 nanoseconds. The rest of the additions are completed every 2 nanoseconds.
5. Suppose that a fetch from a level 1 cache takes two nanoseconds, while a fetch from a level 2 cache takes five nanoseconds, and a fetch from main memory takes fifty nanoseconds  
   level 1 cache miss: 2007 + (2 + 5) nanoseconds = 2014 // pipeline is delayed 7 nanoseconds: 2 for searching level 1 and 5 for searching level 2  
   level 2 cache miss: 2007 + (2 + 5 + 50) nanoseconds = 2064 // pipeline is delayed 57 nanoseconds: first we check level 1 then level 2 cache then fetch from main memory
6. In write-through every time the program updates a block in the cache, the cache writes the block to main memory immediately.   
   Implementing a queue that temporarily stores the modified blocks (dirty bits) where they will be written to main memory later, will help in reducing bus traffic therefore improving its utilization.  
   (must use protocols to target out-of-date blocks in memory o other caches)
7. larger matrix:   
    - more data to be read from memory  
    - more cache misses  
   larger cache:  
    - less cache misses  
    - more cache search time  
   if MAX = 8 & 4 cache lines each row in the array will need 2 cache lines to fit, therefore each row will result in a 2 cache misses. Therefore, the total number of misses will be 8 \* 2 = 16.   
   in the second pair of loops, each element in the array will cause a miss, the total number of misses would be 8 \* 8 = 64.

Explanation: the first pair of loops read the elements of each row in the array a00, a01, a02, …. While the second pair of for loops reads the elements of each column in the array a00, a10, a20, ….

1. 2^20 = 1048576 pages
2. no, cache is used to decrease the number of memory accesses, while virtual memory allows more processes to reside in memory. Therefore, they do not intervene with an SISD system.  
   similarly, the implementation of pipelining, multiple issue, and hardware multithreading, will not change the designation of an SISD system.
3. 10, fetching data from multiple memory banks reduces latency, since a second access to the same memory bank will be delayed.
4. a vector processor with n registers (size of the vector processor) this requires a single load, add and store. If the array size (n) is larger more cycles will be needed.  
   a GPU works in a similar way to a vector processor except that data will be sent to and received from the GPU.
5. larger cache = more search time overhead.  
   more threads than available cores will cause overhead (switching).  
   more threads also mean more swapping between memory and backing store.
6. a MISD will load multiple simultaneous instructions on single data streams (meaning that it can fetch or store one item of data at a time)  
   this can be thought of a single control unit and a single alu.  
   example: cinema reservation system.
7. a) 1 message = 10^-9 sec, p = 1000  
   10^12/1000 = 10^9 instructions / processor  
   10^12 instruction = 10^6 sec / processor 🡪 10^9 instruction = 1000 sec / processor  
   each processor sends 10^9(999) messages 🡪all messages = 10^-9 sec \* 10^9(999) message = 999 second  
   time per processor = 999 + 1000 = 1999 sec  
   b) all messages = 10^-3 sec \* 10^9(999) message = 999 \* 10^6 second  
   time per processor = 999 \* 10^6 + 1000 = 999001000
8. direct  
    ring: 2p  
    toroidal mesh: 3p (1 to connect the processor and 4 to connect to other switches divided by 2 since every link connects 2 switches)  
   ideal direct: p^2 / 2 + p / 2  
    d dimension hypercube: dp/2
9. a) the same as a mesh (removing the wrap around links does not affect the bisection width)  
   b) a 3d mesh with a q^3 processors has a bisection width of q^2
10. a) a 4d hypercube is called a tesseract   
    b) p / 2  
    [](https://www.google.com/url?sa=i&url=https%3A%2F%2Fslideplayer.com%2Fslide%2F4797258%2F&psig=AOvVaw1Qvlho9ARyL8iI10-SrXMt&ust=1625398156927000&source=images&cd=vfe&ved=0CAsQjhxqFwoTCLiQrpHmxvECFQAAAAAdAAAAABAJ)
11. if n is even:  
     an n \* n crossbar would yield after splitting it in half, n/2  
    if n is odd:  
     an n \* n crossbar would also yield a number that is <= to n/2 since we need to remove some link
12. a) since core 1 is attempting to access the value of x after core 0 has updated it, then core 1 will access the :  
     -old value of x and this is because in a write back cache the updated value will not be written immediately to memory  
     -new value of x if the memory was updated before core 1 requested the value of x  
    b) the old value of x will be assigned to y since core 1 does not have the value of x in its cache therefore it will not be broadcasted about the update of the variable x made by core 0  
    or the new value of x and that is if core 0 wrote the updated value to memory before core 1 trying to access it

c) we can use a write through cache which updates memory immediately

1. a) simple code

b) lim n 🡪0 (T overhead) => T parallel = Tserial/p

c) lim n 🡪 infinity (T overhead) => T parallel = infinity

1. increased cache size.  
   running a program on a small cache vs a big cache.
3. Speedup = T serial / T parallel = = 🡪n’ =   
   if k = 2 and p = 8 🡪 …
4. yes, since in a linear speedup the efficiency is always constant.