* 1. You might build an SoC for mining cryptocurrency. Some required components might include a general-purpose processor, peripherals for input and output, a network adaptor, and some specialised cores for computing cryptographic hash functions.
     1. 50% / 1.25 + 30% + 20% = 90% = 1.11 x speedup
     2. 50% + 30% + 20% / 2.5 = 88% = 1.14 x speedup
     3. 50% + 30% / 1.2 + 20% / 1.33 = 90% = 1.11 x speedup

Option b is the best

* + 1. 3x
    2. 2x

When run in parallel, the maximum speedup is 4x. When running sequentially, the maximum speedup is 5x

* 1. Diagram

     Description automatically generated(Copied from the lecture notes)

It needs to refresh because capacitors are not perfect and will eventually leak charge

* + 1. Since each bank operated independently, a request to one bank will not need to wait for the request to another bank to complete entirely before it can start. This allows for pipelining of bank requests, effectively increasing the bandwidth.
    2. Continuously accessing the same bank is beneficial when there is sequential access to different columns, and the memory controller has an open-page policy.
  1. A row access moves the data to the sense amps. The column read moves the data from the sense amps to the data bus. Both are required because it allows the sense amps to act like a cache for sequential column reads.
  2. This is a useful abstraction for programmers because all cores use the same address space and see the same values in memory. The programmer does not have to worry about creating robust communication protocols for data to travel between cores.
  3. // TODO
  4. Inclusive caches are ones in which the data in the higher-level caches is also stored in the lower-level caches. Exclusive caches are ones in which the data in the higher-level caches are not stored in the lower-level caches. Non-inclusive non-exclusive (NINE) caches are those in which some of the data in the higher-level caches may be stored in the lower-level caches as well, but it is not guaranteed either way.
  5. You would use a BusRdX instruction if you want to invalidate the block in the caches of the other processors.
     1. Return the data at address x
     2. Return the data at address x
     3. Initiate a bus read to bring the data into the cache and then return it
  6. A naïve spin lock or a locally caches spin lock
  7. It alters the source register to contain a flag which represents whether the write was successful or not.

lock: mov r3, #1

xchg r3, 0(r1) ; xchg swaps a register with a value in memory

; Assumes that the lock address is stored in r1

bneqz r3, lock

unlock: mov r3, #0

st r3, 0(r1)

* 1. // TODO
  2. A memory barrier guarantees the order of execution of memory operations within a core. All memory operations before a memory bar must complete before the bar finished executing.
  3. This is because once the lock has been taken, no other cores can take it and so the resources protected by the lock are only used by the one core. If a memory bar on a different core was placed before acquiring the lock, it might result in changes to these resources, which could cause conflicts with the first core.

Similarly, if one core executes a memory bar after releasing the lock, another core could acquire the lock in between the first core releasing it and the first core executing its memory bar. This could cause changes to the resources which could conflict with the second core.