I. Pipelining

Note: 1 Hz = 1 cycle / second, MHz = 10^6 cycles / second, GHz = 10^9 cycles / second

A. Policies
1. Data hazards (instr. dependent on a previous instr. that hasn't finished)
    a) Detect and stall
        (1) Add, nor, lw, with dependency right after: 2 stall
        (2) Add, nor, lw with dependency 2 after: 1 stall
    b) Detect and forward
        (1) lw with dependency right after: 1 stall
2. Control hazards (branching)
    a) Detect and stall
        (1) beq: 3 stalls
    b) Predict not taken (speculate and squash)
        (1) beq: 3 stalls only if branch taken

B. Calculate program performance (CPI)
1. for a detect and forward, speculate and squash pipeline:
   CPI = 1 + %lw * %dep * 1+ %beq * %taken * 3
2. Note: only lw/sw instructions access cache

II. Caches

A. Determine number of bits for tag, set index, and block offset in an address

A memory address is formatted as:

Tag	Set Index	Block Offset

#set index bits = log_2(number of sets)
#block offset bits = log_2(block size)
#tag bits = remaining bits or log_2(number of blocks per set)

B. Calculate overhead- number of bits needed for a cache line (dirty, valid, tag)- not set index or block offset

C. Calculate miss rate

D. Identify type of miss
1. Miss in infinite cache- compulsory
2. Miss only in set associative cache- conflict
3. Otherwise (Miss fully associative cache and set associative cache but not miss in infinite)- capacity

E. Reverse engineer a cache's specifications
1. First find block size range by writing addresses in binary and checking the maximum number of first bits that match but there is still a miss
2. Once you know the blocksize, calculate the number of cache lines
3. Test if the cache is direct mapped by allowing enough set index bits so that there are as many sets as cache lines, and see if there's any inconsistencies
4. Keep going until you find one that works

III. Virtual Memory

Note: 1 KB = 2^10 bytes, 1 MB = 2^20 bytes, 1 GB = 2^30 bytes

A. Calculate number of bits for each component of a virtual and physical address

Virtual page number/ Physical page number	Page offset (same for virtual and physical)

VPN bits = log_2(# virtual pages); #PPN bits = log_2(# physical pages)

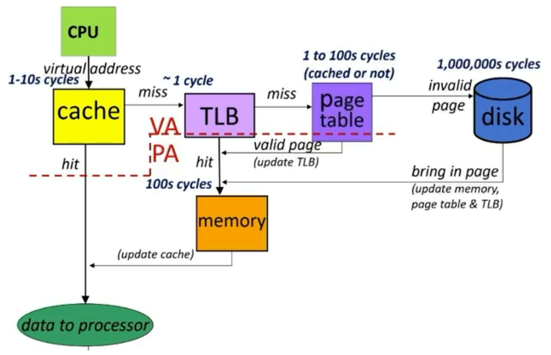
# Page offset bits = log_2(page size)
Number of physical pages = size of memory / page size
Length of virtual address is the number of bits of the system

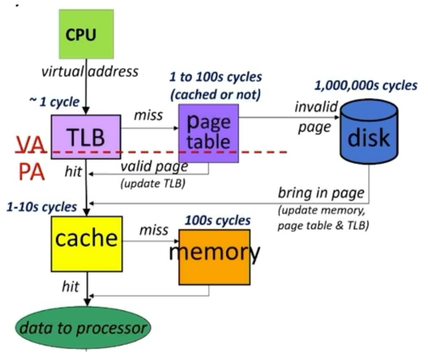
B. Multilevel page table

1st level offset	2nd level offset	…	Page offset (same for virtual and physical)

# bits for level N offset = log_2(number of pages per page table in level N)

C. Determine TLB hits/ Page Faults
1. For each reference:
    a) check if it's in TLB
        (1) If it is (hit), access the corresponding physical address
    b) else read the reference in the page table.
        (1) If the page table has the corresponding memory address, access it and add to it the TLB
    c) else (fault) retrieve the data from the disk and allocate it in memory, then update the page table and TLB. Finally re-access the TLB to get the physical address.





D.