ECE3700J RC 10
Cache

Presenter: Ruan Renjian 阮仁剑

Locality

Temporal locality: Items that are accessed recently are likely to be accessed again soon

Example: constant, variable, and counter in LOOP

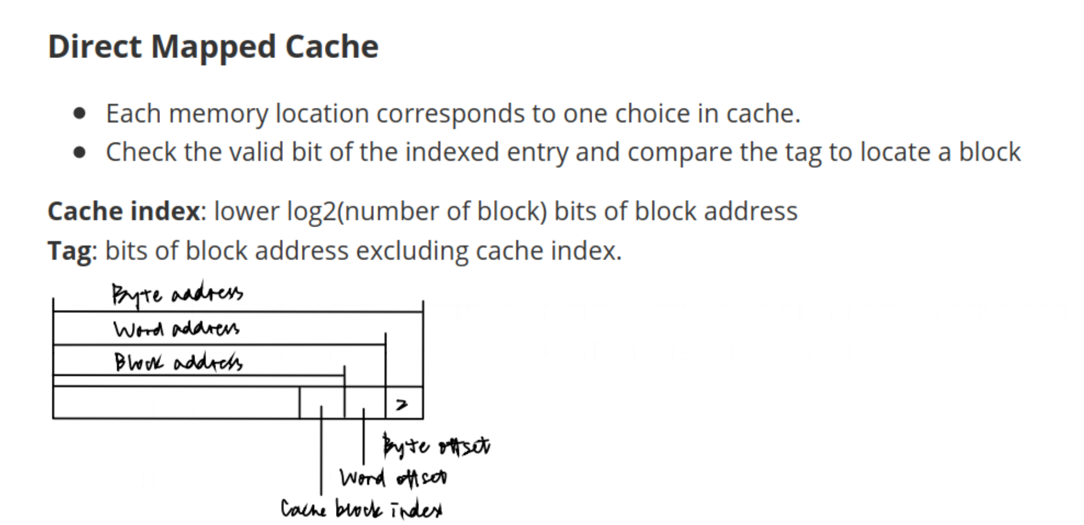
Spatial locality: Items **near those that** are accessed recently are like to be accessed soon

Example: array, memory, stack

Ex1. ([✓ Spatial Locality]/Temporal Locality) is the concept that the likelihood of referencing a piece of memory is higher if an access has occurred near it

Ex2. (True/[✓ False])We have two caches with the same block size, number of sets and associativity. When comparing the one with a LRU policy versus the one with a MRU policy (the most recently inserted block is the first to be evicted), the LRU cache will have better spatial locality

Cache

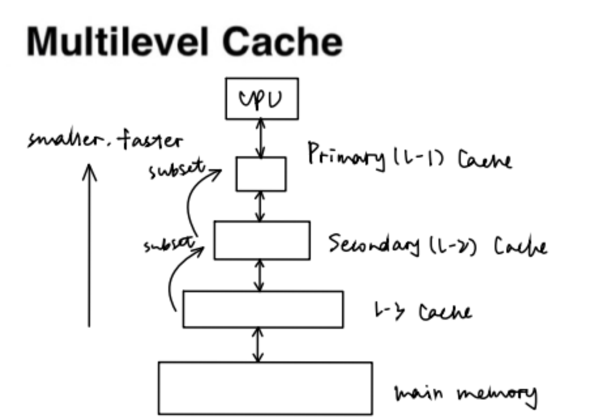


Direct Mapped Cache

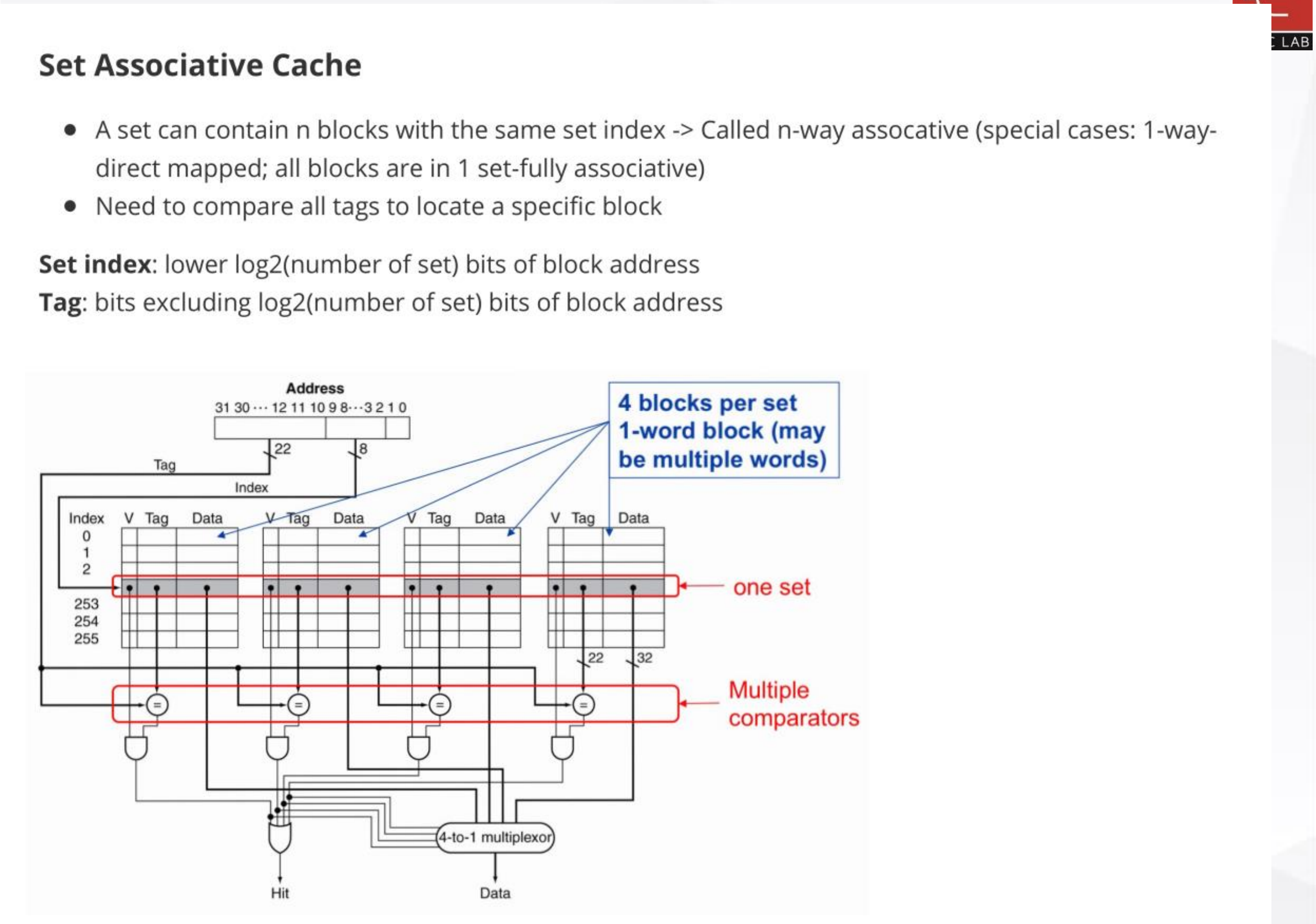
- Each memory location corresponds to one choice in cache.
- Check the valid bit of the indexed entry and compare the tag to locate a block

Cache index: lower log2(number of block) bits of block address

Tag: bits of block address excluding cache index.



Multilevel Cache



Set Associative Cache

- A set can contain n blocks with the same set index -> Called n-way assocative (special cases: 1-way-direct mapped; all blocks are in 1 set-fully associative)
- Need to compare all tags to locate a specific block

Set index: lower log2(number of set) bits of block address

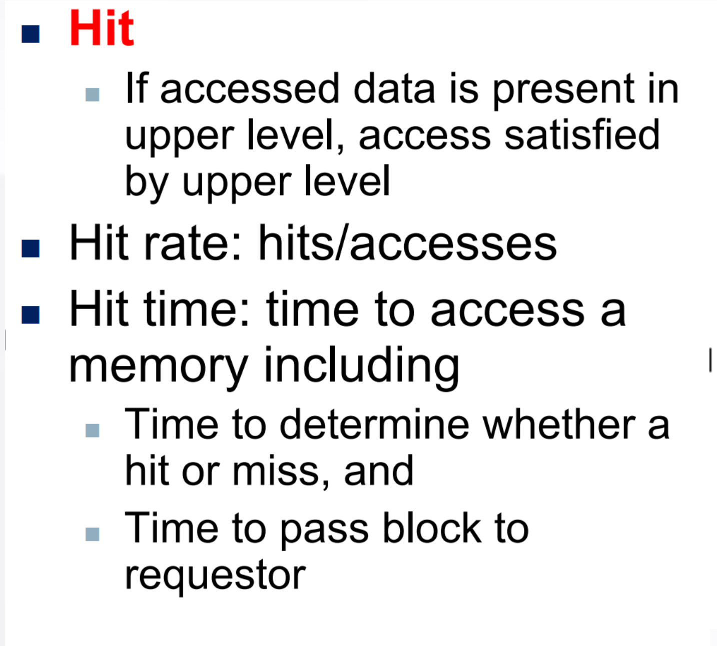
Tag: bits excluding log2(number of set) bits of block address

4 blocks per set 1-word block (may be multiple words)

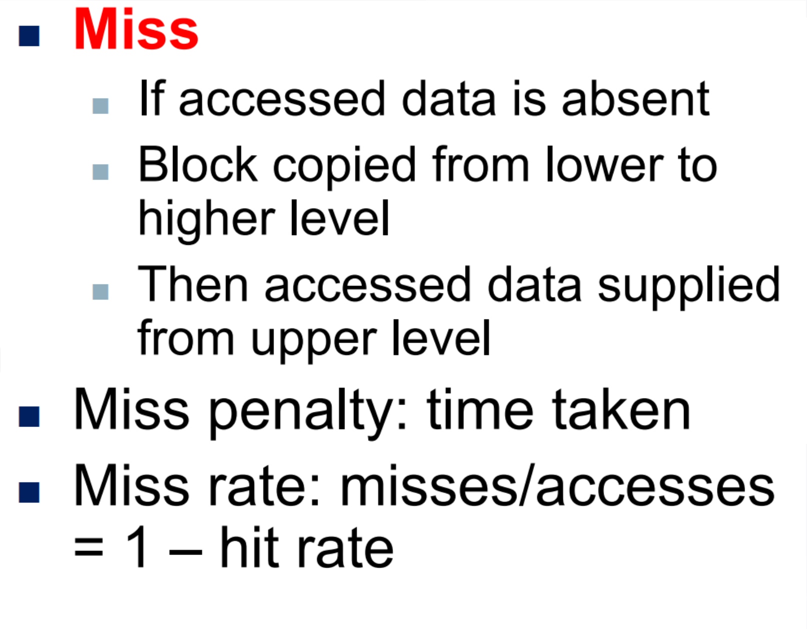
one set

Multiple comparators

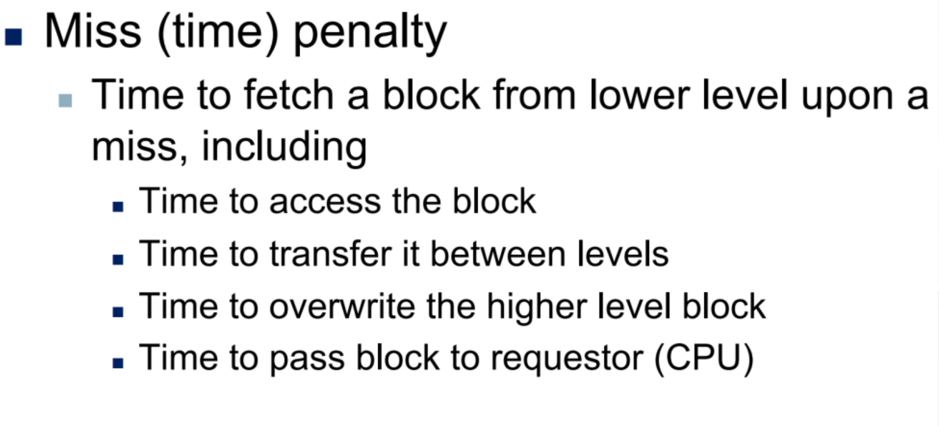
Hit, Miss, CPU Time



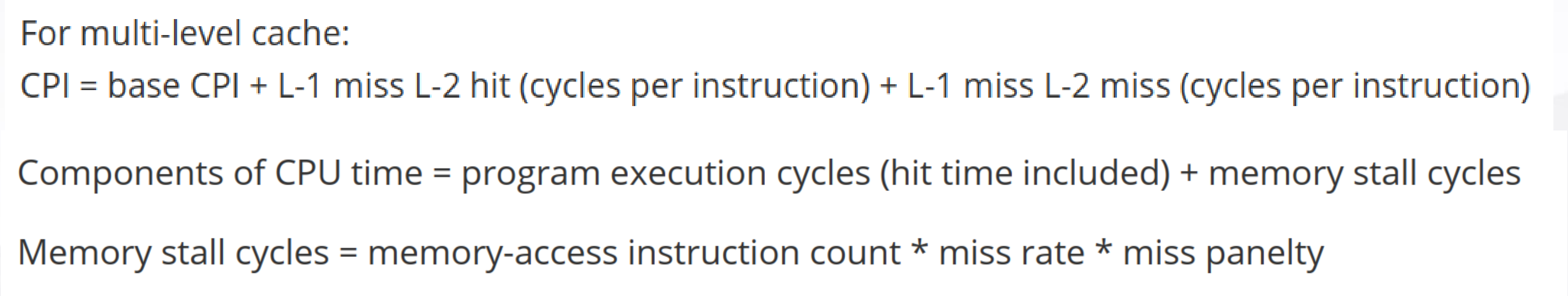
- **Hit**
  - If accessed data is present in upper level, access satisfied by upper level
- Hit rate: hits/accesses
- Hit time: time to access a memory including
  - Time to determine whether a hit or miss, and
  - Time to pass block to requestor



- **Miss**
  - If accessed data is absent
  - Block copied from lower to higher level
  - Then accessed data supplied from upper level
- Miss penalty: time taken
- Miss rate: misses/accesses = 1 – hit rate



- Miss (time) penalty
  - Time to fetch a block from lower level upon a miss, including
    - Time to access the block
    - Time to transfer it between levels
    - Time to overwrite the higher level block
    - Time to pass block to requestor (CPU)



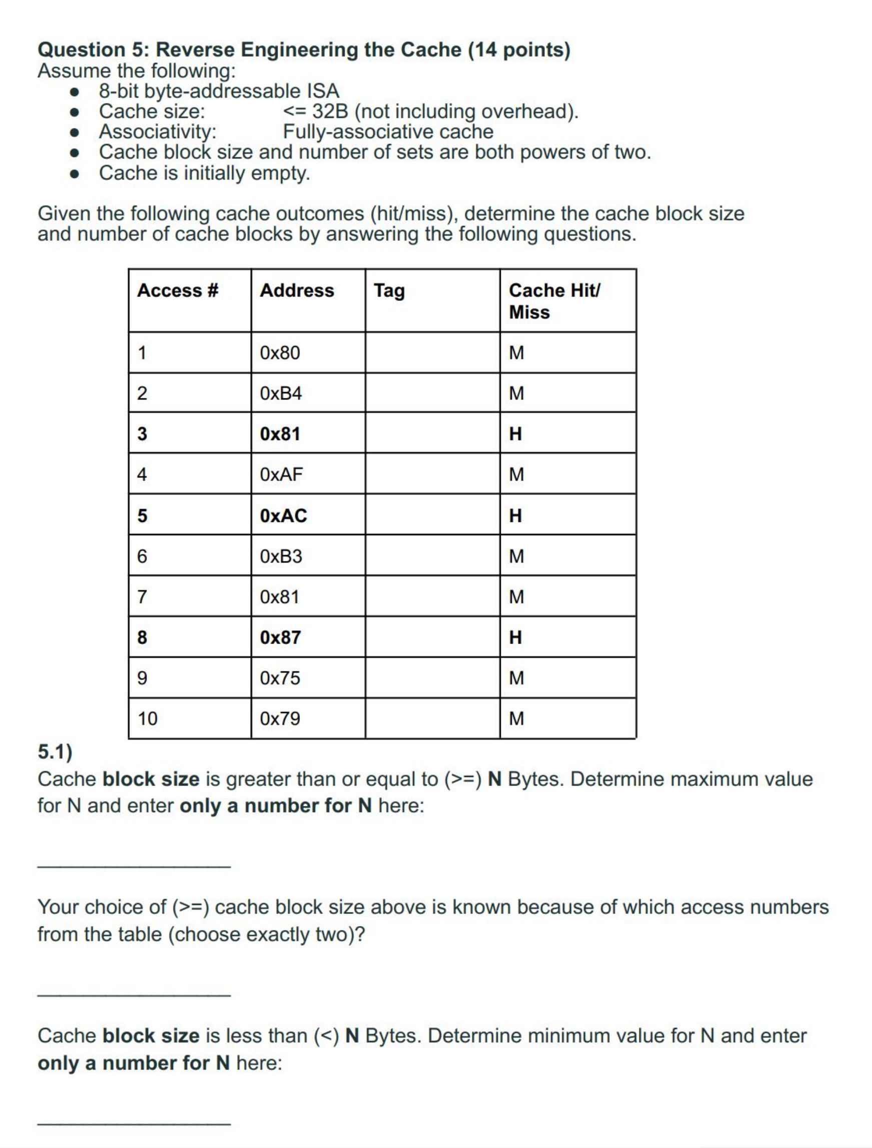
For multi-level cache:

CPI = base CPI + L-1 miss L-2 hit (cycles per instruction) + L-1 miss L-2 miss (cycles per instruction)

Components of CPU time = program execution cycles (hit time included) + memory stall cycles

Memory stall cycles = memory-access instruction count * miss rate * miss panelty

Exercise



Question 5: Reverse Engineering the Cache (14 points)
Assume the following:

- 8-bit byte-addressable ISA
- Cache size: <= 32B (not including overhead).
- Associativity: Fully-associative cache
- Cache block size and number of sets are both powers of two.
- Cache is initially empty.

Given the following cache outcomes (hit/miss), determine the cache block size and number of cache blocks by answering the following questions.

Access #	Address	Tag	Cache Hit/ Miss
1	0x80		M
2	0xB4		M
3	**0x81**		**H**
4	0xAF		M
5	**0xAC**		**H**
6	0xB3		M
7	0x81		M
8	**0x87**		**H**
9	0x75		M
10	0x79		M

5.1)
Cache **block size** is greater than or equal to (>=) **N** Bytes. Determine maximum value for N and enter **only a number for N** here:

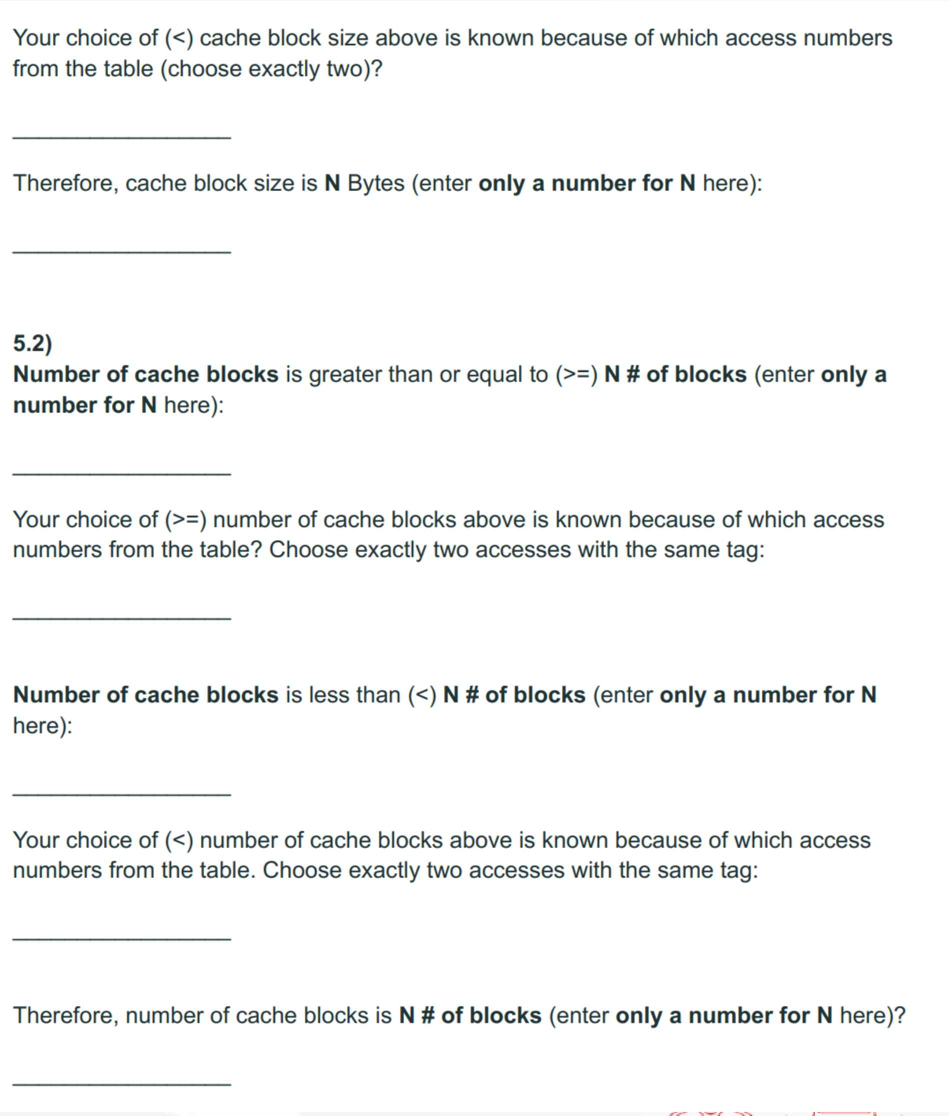
____

Your choice of (>=) cache block size above is known because of which access numbers from the table (choose exactly two)?

____

Cache **block size** is less than (<) **N** Bytes. Determine minimum value for N and enter **only a number for N** here:

____



Your choice of (<) cache block size above is known because of which access numbers from the table (choose exactly two)?

____

Therefore, cache block size is **N** Bytes (enter **only a number for N** here):

____

5.2)
Number of cache blocks is greater than or equal to (>=) **N # of blocks** (enter **only a number for N** here):

____

Your choice of (>=) number of cache blocks above is known because of which access numbers from the table? Choose exactly two accesses with the same tag:

____

Number of cache blocks is less than (<) **N # of blocks** (enter **only a number for N** here):

____

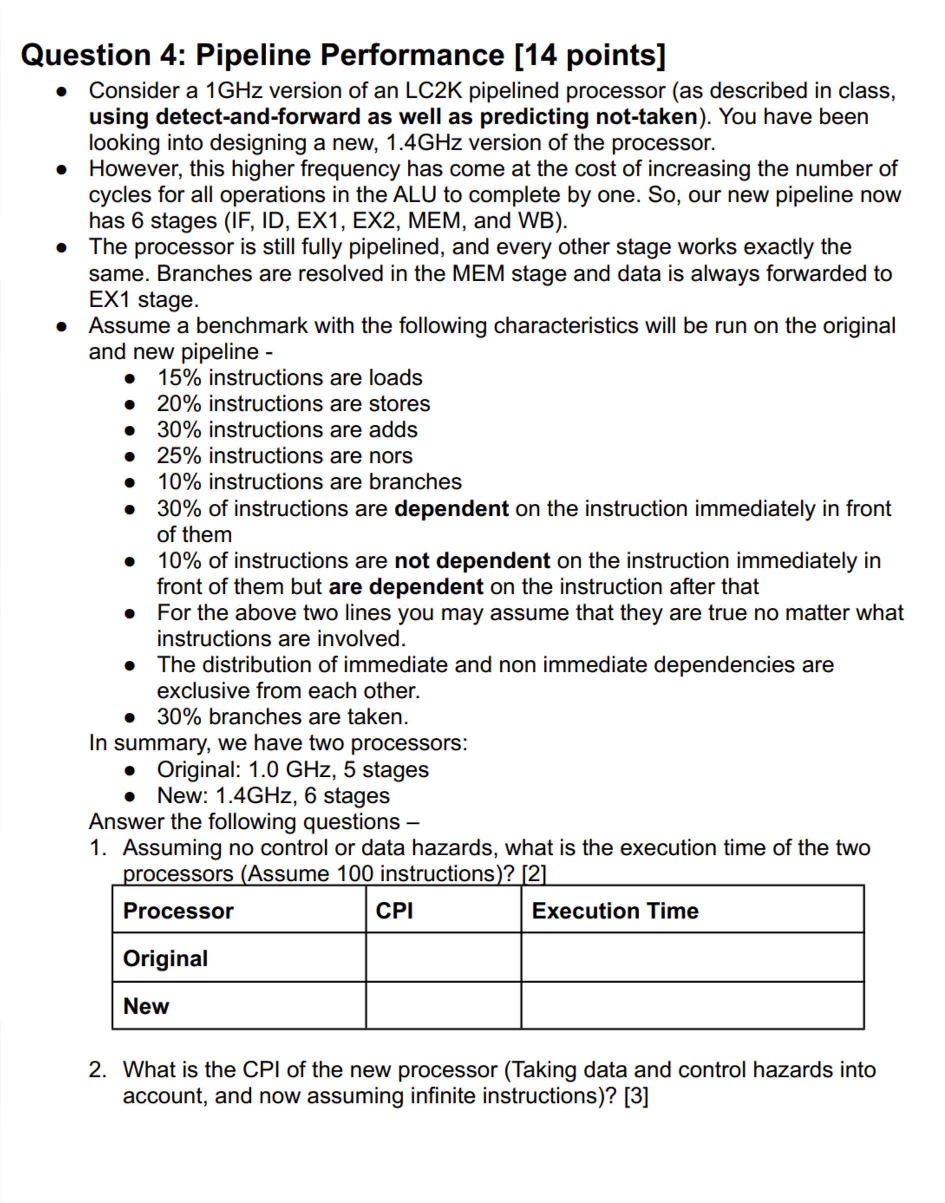
Your choice of (<) number of cache blocks above is known because of which access numbers from the table. Choose exactly two accesses with the same tag:

____

Therefore, number of cache blocks is **N # of blocks** (enter **only a number for N** here)?

____

Exercise



Question 4: Pipeline Performance [14 points]

- Consider a 1GHz version of an LC2K pipelined processor (as described in class, **using detect-and-forward as well as predicting not-taken**). You have been looking into designing a new, 1.4GHz version of the processor.
- However, this higher frequency has come at the cost of increasing the number of cycles for all operations in the ALU to complete by one. So, our new pipeline now has 6 stages (IF, ID, EX1, EX2, MEM, and WB).
- The processor is still fully pipelined, and every other stage works exactly the same. Branches are resolved in the MEM stage and data is always forwarded to EX1 stage.
- Assume a benchmark with the following characteristics will be run on the original and new pipeline -
    - 15% instructions are loads
    - 20% instructions are stores
    - 30% instructions are adds
    - 25% instructions are nors
    - 10% instructions are branches
    - 30% of instructions are **dependent** on the instruction immediately in front of them
    - 10% of instructions are **not dependent** on the instruction immediately in front of them but **are dependent** on the instruction after that
    - For the above two lines you may assume that they are true no matter what instructions are involved.
    - The distribution of immediate and non immediate dependencies are exclusive from each other.
    - 30% branches are taken.

In summary, we have two processors:

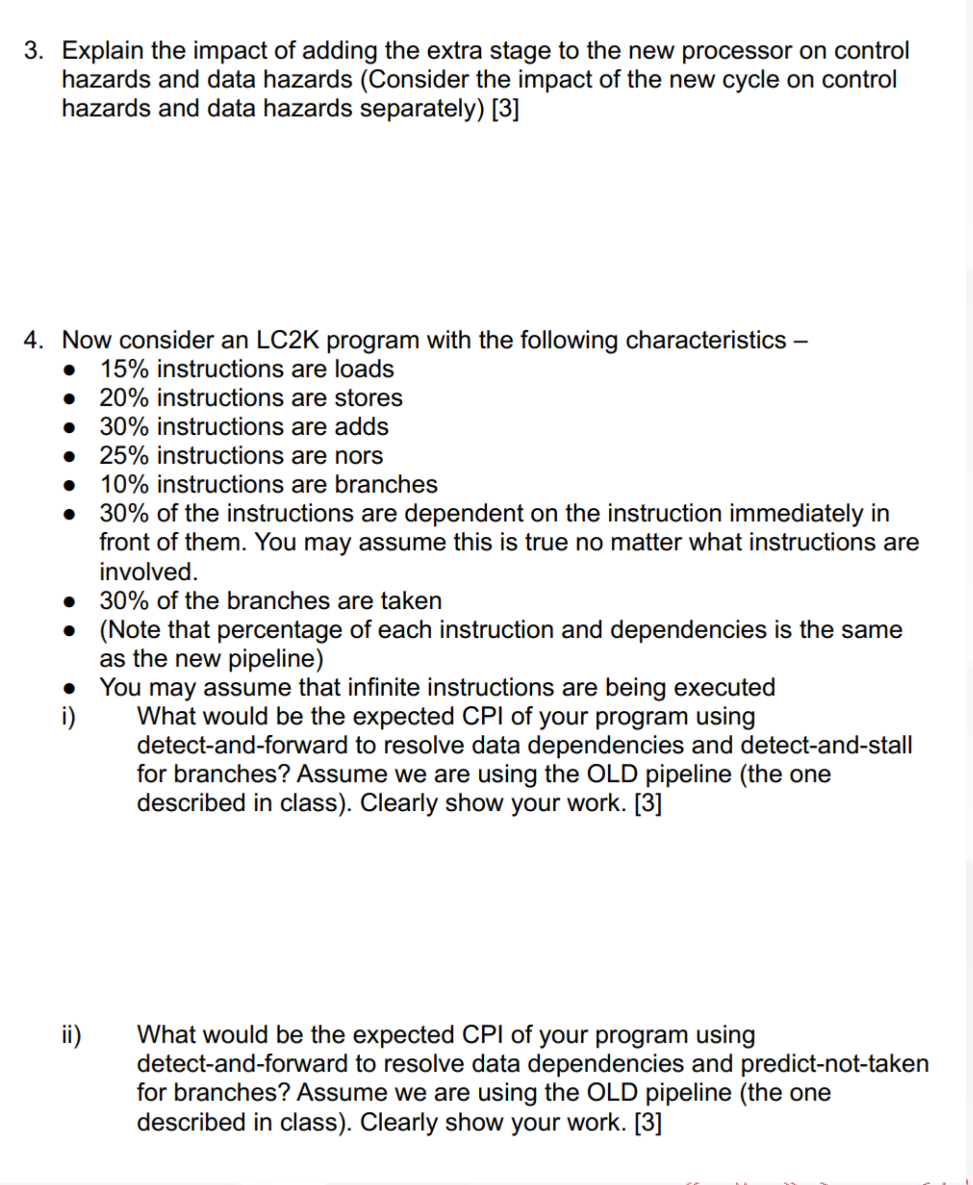
- Original: 1.0 GHz, 5 stages
- New: 1.4GHz, 6 stages

Answer the following questions –

1. Assuming no control or data hazards, what is the execution time of the two processors (Assume 100 instructions)? [2]

Processor	CPI	Execution Time
Original		
New		

2. What is the CPI of the new processor (Taking data and control hazards into account, and now assuming infinite instructions)? [3]



3. Explain the impact of adding the extra stage to the new processor on control hazards and data hazards (Consider the impact of the new cycle on control hazards and data hazards separately) [3]

4. Now consider an LC2K program with the following characteristics –
    - 15% instructions are loads
    - 20% instructions are stores
    - 30% instructions are adds
    - 25% instructions are nors
    - 10% instructions are branches
    - 30% of the instructions are dependent on the instruction immediately in front of them. You may assume this is true no matter what instructions are involved.
    - 30% of the branches are taken
    - (Note that percentage of each instruction and dependencies is the same as the new pipeline)
    - You may assume that infinite instructions are being executed

    i) What would be the expected CPI of your program using detect-and-forward to resolve data dependencies and detect-and-stall for branches? Assume we are using the OLD pipeline (the one described in class). Clearly show your work. [3]

    ii) What would be the expected CPI of your program using detect-and-forward to resolve data dependencies and predict-not-taken for branches? Assume we are using the OLD pipeline (the one described in class). Clearly show your work. [3]