HW2

1.

Consider the following description of a memory hierarchy.**

- Virtual address width = 45 bits
- Physical memory address width = 38 bits
- Page size = 4 KB
- Cache capacity = 8 KB
- Block size = 32 Bytes
- It is a write-back 2-way associative cache.

(a) How many bits are there in the fields of tag, index, and block offset of the physical memory address?

(b) Draw a graph to show a cache line (including tag, data, and some other control bits) in the cache.

(c) Draw a graph to show if it is implemented in the way of virtually indexed and physically tagged cache. Draw both cache and TLB.

(d) Please describe the access procedure to the memory hierarchy in (c) when a CPU address (virtual address) is given to access the cache.

2.

Assume that we have two machines A and B. The only difference between A and B lies in their cache hierarchies:

- **Machine A:** 64 KB level-one data cache with an 8 ns access time and a miss rate of 8%.
- **Machine B:** 8 KB level-one data cache with a 2 ns access time and a miss rate of 15%, and a 1 MB level-two cache with a 20 ns access time and a miss rate of 10%.

Assume that both machines have an I-cache miss rate of 0%, a main memory access time of 50 ns, and all the bus transfer time could be ignored.

Which machine will have better performance in memory access (AMAT)? Why?

3.

You are building a system around a processor with in-order execution that runs at 1.0 GHz and has a CPI of 1.35 excluding memory accesses. The only instructions that read or write data from memory are loads (20% of all instructions) and stores (10% of all instructions). The memory system for this computer is composed of a split L1 cache that imposes no penalty on hits. Both the I-cache and D-cache are direct-mapped and hold 32 KB each. The I-cache has a 2% miss rate and 32-byte blocks, and the D-cache is write-through with a 5% miss rate and 16-byte blocks. There is a write buffer on the D-cache that eliminates stalls for 90% of all writes. The 512 KB write-back, unified L2 cache has 64-byte blocks and an access time of 12 ns. It is connected to the L1 cache by a 128-bit data bus that runs at 266 MHz and can transfer one 128-bit word per bus cycle. Of all memory references sent to the L2 cache in this system, 85% are satisfied without going to main memory. Also, 50% of all blocks replaced are dirty. The 128-bit-wide main memory has an access latency of 80 ns, after which any number of bus words may be transferred at the rate of one per cycle on the 128-bit-wide 133 MHz main memory bus.

Questions:

a. What is the average memory access time for instruction accesses?
b. What is the average memory access time for data reads?
c. What is the average memory access time for data writes?
d. What is the overall CPI, including memory accesses?

提示: L1 cache 的 miss penalty 认为是替换数据从 L2 到 L1 的传输时间，忽略响应时间。

4.（此题在期中考中被改编过，注意要完整理解其概念）

The transpose of a matrix interchanges its rows and columns; this is illustrated below:

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix} \Rightarrow \begin{bmatrix} A_{11} & A_{21} & A_{31} & A_{41} \\ A_{12} & A_{22} & A_{32} & A_{42} \\ A_{13} & A_{23} & A_{33} & A_{43} \\ A_{14} & A_{24} & A_{34} & A_{44} \end{bmatrix}$$

Here is a simple C loop to show the transpose:

```
for (i = 0; i < 3; i++) {
    for (j = 0; j < 3; j++) {
        output[j][i] = input[i][j];
    }
}
```

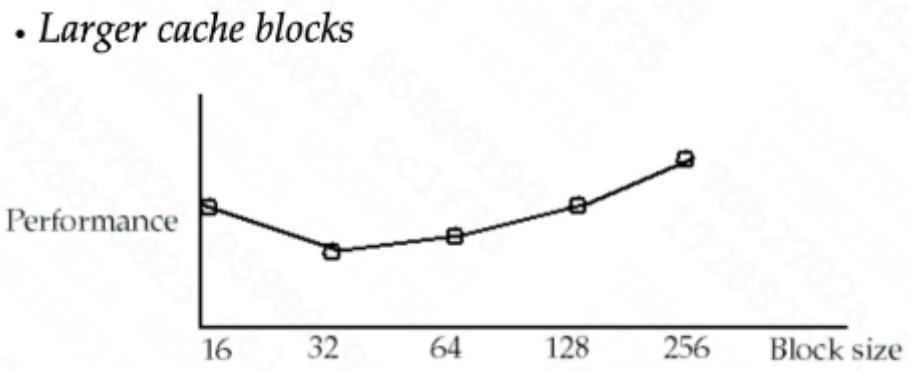
Assume that both the input and output matrices are stored in row-major order (row-major order means that the row index changes fastest). Assume that you are executing a 256×256 **single-precision** transpose on a processor with a 16 KB fully associative (don't worry about cache conflicts) least recently used (LRU) replacement L1 data cache with 64-byte blocks. Assume that the L1 cache misses or prefetches require 16 cycles and always hit in the L2 cache, and that the L2 cache can process a request every two processor cycles. Assume that each iteration of the inner loop above requires four cycles if the data are present in the L1 cache. Assume that the cache has a write-allocate fetch-on-write policy for write misses. Unrealistically, assume that writing back dirty cache blocks requires 0 cycles.

For the simple implementation given above, this execution order would be nonideal for the input matrix; however, applying a loop interchange optimization would create a nonideal order for the output matrix. Because loop interchange is not sufficient to improve its performance, it must be blocked instead.

Questions:

a. What should be the minimum size of the cache to take advantage of blocked execution?
b. How do the relative number of misses in the blocked and unblocked versions compare in the minimum sized cache above?
c. Write code to perform a transpose with a block size parameter B which uses $B \times B$ blocks.
d. What is the minimum associativity required of the L1 cache for consistent performance independent of both arrays' position in memory?

5.



解释上图为什么会呈U型

6.

Below are various cache optimization techniques. Please match each technique with the corresponding optimization goal by choosing from the following options:

1. **Reduce the miss penalty**
2. **Reduce the miss rate**
3. **Reduce the miss penalty and miss rate via parallelism**
4. **Reduce the time to hit in the cache**

Techniques:

1. Multilevel caches: ( )
2. Critical word first: ( )
3. Larger block size: ( )
4. Non-blocking caches: ( )
5. Hardware prefetching: ( )
6. Small and simple caches: ( )
7. Higher associativity: ( )
8. Avoiding address translation: ( )

9. Victim caches: ( )
10. Compiler optimizations: ( )
11. Way prediction and pseudo-associativity: ( )
12. Pipelined cache access: ( )