2st Homework for Computer Architecture

Submission deadline: Oct. 27 , 11：55pm

Read Chapter 2 Appendix B then do the following problems.

(Total 152 points)

In 6th Edition

2.1 2.24 B.5 B.8

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

$$\begin{bmatrix} A11 & A12 & A13 & A14 \\ A21 & A22 & A23 & A24 \\ A31 & A32 & A33 & A34 \\ A41 & A42 & A43 & A44 \end{bmatrix} \Rightarrow \begin{bmatrix} A11 & A21 & A31 & A41 \\ A12 & A22 & A32 & A42 \\ A13 & A23 & A33 & A43 \\ A14 & A24 & A34 & A44 \end{bmatrix}$$

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 the row major order (*row major order* means that the row index changes fastest). Assume that you are executing a 256·256 double-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 2 processor cycles. Assume that each iteration of the preceding inner loop requires 4 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.

2.1 [10/15/15/12/20] <2.3> For the preceding simple implementation, 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.

a. [10] <2.3> What should be the minimum size of the cache to take advantage of blocked execution?

b. [15] <2.3> How do the relative number of misses in the blocked and unblocked versions compare in the preceding minimum-sized cache?

c. [15] <2.3> Write code to perform a transpose with a block size parameter B that uses $B \cdot B$ blocks.

d. [12] <2.3> What is the minimum associativity required of the L1 cache for consistent performance independent of both arrays' position in memory?

e. [20] <2.3> Try out blocked and nonblocked 256·256 matrix transpositions on a computer. How closely do the results match your expectations based on what you know about the computer's memory system? Explain any discrepancies if possible.

Answer	
a	Double precision: 8B. 64B has 8 elements. Each matrix has 8*8=64 elements. Minimum cache size: 128*8=1024B=1KB.
b	For the blocked version, input and output element will be fetched for only once. For the unblocked version, every 8 row elements will cause a miss, and column elements will be replaced before reused. So in this version the miss number will be 9, which contains 1 row miss and 8 column misses.

c	for (i=0; i<256; i=i+B){
	for (j=0; j<256; j=j+B){
	for (x=0; x<B; x++){
	for (y=0; y<B; y++){
	output[j+n][i+m] = input[i+m][j+n];
	}
	}
	}
	}

d	At least 2-way set associativity is required for consistent performance.
e	The reasons for discrepancies: 1. For write misses, the cache in this question uses a write-allocate fetch-on-write policy. But in real case cache may use write-through policy instead. 2. In this question, writing back dirty cache blocks requires 0 cycles, but this is impossible for real case.

C 题想不清楚可以全部算出来取比值，blocked 版本是 256/8 * 256/8 * 16(每个块 8 个 input 8 个 output)，unblocked 版本是 256*256/8（input 8 个 1 次 miss）+256*256（output 全 miss），比值是 4.5

2.24 [20] <2.1, 2.6> You are designing a PMD and optimizing it for low energy. The core, including an 8 KB L1 data cache, consumes 1 W whenever it is not in hibernation. If the core has a perfect L1 cache hit rate, it achieves an average CPI of 1 for a given task, that is, 1000 cycles to execute 1000 instructions. Each additional cycle accessing the L2 and beyond adds a stall cycle for the core. Based on the following specifications, what is the size of L2 cache that achieves the lowest energy for the PMD (core, L1, L2, memory) for that given task?

a. The core frequency is 1 GHz, and the L1 has an MPKI of 100.

b. A 256 KB L2 has a latency of 10 cycles, an MPKI of 20, a background power of 0.2 W, and each L2 access consumes 0.5 nJ.

c. A 1 MB L2 has a latency of 20 cycles, an MPKI of 10, a background power of 0.8 W, and each L2 access consumes 0.7 nJ.

d. The memory system has an average latency of 100 cycles, a background power of 0.5 W, and each memory access consumes 35 nJ.

Ans：

这个性能是要求功耗(1000inst)的并不是功率

纯 memory：(1000ns(hit time) + 100(miss) * 100ns (miss penalty)) * (1w(CPU power) + 0.5w(memory power)) + 100(memory miss) * 35nJ(miss energy) = 20000nJ

L2 256KB: (1000ns(hit time) + 100(L1 miss) * 10ns (L1 miss penalty) + 20(L2 miss rate)*100ns(L2 miss penalty)) * (1w(CPU power) + 0.2w(L2 power) + 0.5w(memory power)) + 100(L1 miss) * 0.5nJ(L1 miss energy) +20(L2 miss) * 35nJ(L1 miss energy)+ = 7550nJ

L2 1M: (1000ns(hit time) + 100(L1 miss) * 20ns (L1 miss penalty) + 10(L2 miss rate)*100ns(L2 miss penalty)) * (1w(CPU power) + 0.8w(L2 power) + 0.5w(memory power)) + 100(L1 miss) * 0.7nJ(L1 miss energy) +10(L2 miss) * 35nJ(L1 miss energy)+ = 9620nJ

所以选择 L2 256KB

B.5 [10/10/10/10/] <B.2> You are building a system around a processor with in-order execution that runs at 1.1 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 95% of all writes. The 512 KB write-back, unified L2 cache has 64-byte blocks and an access time of 15 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, 80% 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 60 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.

a. [10] <B.2> What is the average memory access time for instruction accesses?

b. [10] <B.2> What is the average memory access time for data reads?

c. [10] <B.2> What is the average memory access time for data writes?

d. [10] <B.2> What is the overall CPI, including memory accesses?

$$\text{AMAT} = \text{hit time} + \text{miss rate}_{L1} \times \text{miss penalty}_{L1} + \text{miss rate}_{L2} \times \text{miss penalty}_{L2}$$

Answer

a

$$CPU\ clock\ cycle\ time = \left(\frac{1}{1.1G}\right)s = 0.91ns$$

$$\text{hit time} = 0 \text{ ns}$$

$$\text{miss penalty}_{L1}(inst) = 15ns + \left(\frac{32}{16}\right) \times \left(\frac{1}{266M}\right)s = 22.5ns$$

$$\text{miss penalty}_{L2}(inst) = \left(60ns + \left(\frac{64}{16}\right) \times \left(\frac{1}{133M}\right)s\right) \times (1 + 50\%) = 135ns$$

$$\text{AMAT(inst)} = 2\% \times 22.5ns + 2\% \times (1 - 80\%) \times 135ns = 0.99ns$$

b

$$\text{miss penalty}_{L1}(data_{read}) = 15ns + \left(\frac{16}{16}\right) \times \left(\frac{1}{266M}\right)s = 18.75ns$$

$$\text{miss penalty}_{L2}(data_{read}) = (60ns + \left(\frac{64}{16}\right) \times \left(\frac{1}{133M}\right)s) \times (1 + 50\%) = 135ns$$

$$\text{AMAT}(data_{read}) = 5\% \times 18.75ns + 5\% \times (1 - 80\%) \times 135ns = 2.29ns$$

c

$$\text{miss penalty}_{L1}(data_{write}) = 15ns + \left(\frac{16}{16}\right) \times \left(\frac{1}{266M}\right)s = 18.75ns$$

$$\text{miss penalty}_{L2}(data_{write}) = (60ns + \left(\frac{64}{16}\right) \times \left(\frac{1}{133M}\right)s) \times (1 + 50\%) = 135ns$$

$$\text{AMAT}(data_{write}) = (1 - 95\%) \times 18.75ns + 5\% \times (1 - 80\%) \times 135ns = 2.29ns$$

d

$$\text{CPI} = 1.35 + 100\% \times \left(\frac{0.99}{0.91}\right) + 20\% \times \left(\frac{2.29}{0.91}\right) + 10\% \times \left(\frac{2.29}{0.91}\right) = 3.19$$

AMAT 是平均存储访问时间，是一个度量，用于最终求整体 CPI，这里认为 hit time 是 0 或者 hit time 是 CPI*CC 或者 CC 都可以算对，最终算 CPI(d)减回去就好。(b)里面 L1 read miss rate 说了是 5%。(c)里面 write 是 write through 的，通常也是认为 no allocate，那么意味着每一次写操作都会涉及一次 L2 访存，write buffer 减掉了 95%的 L2 访存，所以得到上述结果。

B.8 [5/5/5] <B.3> We want to observe the following calculation

$$d_i = a_i + b_i * c_i, \qquad i : (0:511)$$

Arrays a, b, c, and d memory layout is displayed below (each has 512 4-byte-wide integer elements).

The above calculation employs a for loop that runs through 512 iterations.

Assume a 32 Kbyte 4-way set associative cache with a single cycle access time. The miss penalty is 100 CPU cycles/access, and so is the cost of a write-back. The cache is a write-back on hits write-allocate on misses cache (Figure B.32).

a. [5] <B3> How many cycles will an iteration take if all three loads and single store miss in the data cache?

b. [5] <B3> If the cache line size is 16 bytes, what is the average number of cycles an average iteration will take? (Hint: Spatial locality!)

c. [5] <B3> If the cache line size is 64 bytes, what is the average number of cycles an average iteration will take?

d. If the cache is direct-mapped and its size is reduced to 2048 bytes, what is the average number of cycles an average iteration will take?

Mem. address in bytes	Contents
0–2047	Array a
2048–4095	Array b
4096–6143	Array c
6144–8191	Array d

Figure B.32 Arrays layout in memory.

Answer:

B.8 a. Assume the number of cycles to execute the loop with all hits is c. Assuming the misses are not overlapped in memory, then their effects will accumulate. So, the iteration would take

$$t = c + 4 \times 100 \text{ cycles}$$

b. If the cache line size, then every fourth iteration will miss elements of a, b, c, and d. The rest of iterations will find the data in the cache. So, on the average, an iteration will cost

$$t = (c + 4 \times 100 + c + c + c)/4 = c + 100 \text{ cycles}$$

c. Similar to the answer in part (b), every 16th iteration will miss elements of a, b, c, and d.

$$t = (c + 4 \times 100 + 15 \times c)/16 = c + 25 \text{ cycles}$$

d. If the cache is direct-mapped and is of same size as the arrays a, b, c, and d, then the layout of the arrays will cause every array access to be a miss! That is because a_i, b_i, c_i, and d_i will map to the same cache line. Hence, every iteration will have 4 misses (3 read misses and a write miss). In addition, there is cost of a write-back for d_i, which will take place in iterations 1 through 511.
Therefore, the average number of cycles is

$$t = c + 400 + 511/512 \times 100$$