CS110 sp25 HW5

Due: TBD

Complete this homework either by writing neatly by hand or using **typst**. You can find the `.typ` file on Piazza.

1 True or False

Fill in your answer (T or F) in the table below.

1. When the same address is accessed multiple times using a cache, it primarily benefits from spatial locality.
2. If a cache system changes from direct-mapped to N-way set-associative ($N > 1$), the number of index bits in the address breakdown increases.
3. Higher cache associativity decreases hit time, miss rate, and miss penalty simultaneously.
4. Assume a direct-mapped cache with 8-byte blocks, 2 sets, using LRU replacement, and initially empty. Given 8-bit addresses, accessing addresses from `0x00` to `0xFF` sequentially will result in no cache hits.

1	2	3	4
F	F	F	F

2 Set-Associative Caches

Assume a 12-bit address space. All cache lines are shown in the table below.

Set index	Tag 1	Tag 2
0		
1		
2		
3		

1. Assume loading a 16-byte struct at address `0x00F` requires only 1 cache lookup, but loading one at `0x011` requires 2 lookups. Fill in the cache system parameters in the table below.

Cache block size	16
Total Capacity	128
Level of set associativity	2
# of cache blocks	8
# of sets	4
Address Breakdown (Tag | Index | Offset)	6|2|4

2. For the sequence of addresses below, indicate whether each access results in a hit, miss, or replacement. Assume the cache is initially empty.

Address	Cache Access Result (Hit/Miss/Replacement)
0x3A8	Miss
0x1A6	Miss
0x04C	Miss
0x5AD	Replacement
0x3B9	Miss
0x44F	Miss
0x1B3	Miss
0x055	Miss
0x241	Replacement
0x45E	Miss
0x1A1	Hit
0x1A5	Hit
0x642	Replacement
0x444	Replacement

3. Calculate the cache hit rate for the sequence of memory accesses above. How can you improve the cache hit rate without increasing the cache **capacity**?

Solution: Hit rate $= \frac{2}{14} = \frac{1}{7}$. By changing it into a fully associative cache.

3 AMAT

Consider a 3-level cache system with the following parameters, where the CPU runs at 4GHz:

Cache Level	Hit Time	Local Miss Rate
L1	2 cycles	8%
L2	12 cycles	35%
L3	44 cycles	10%
Memory	80ns	-

1. Calculate the global miss rate for the 3-level cache.

   Solution: Global Miss Rate $= 0.08 * 0.35 * 0.1 = 0.0028$.

2. Calculate the average memory access time (AMAT) for the CPU.

   Solution: Memory Hit time $= \frac{80}{0.25} = 320$ cycles.

   AMAT $= 2 + 0.08 * (12 + 0.35 * (44 + 0.1 * 320)) = 5.088$ cycles.

3. Now, suppose the L1 cache is changed from 2-way set-associative to fully associative. This change reduces the L1 local miss rate to **6%** but increases the L1 hit time by **60%**. Calculate the new AMAT.

   Solution: L1 hit time $= 2 * 1.6 = 3.2$ cycles. L1 local miss rate $= 0.06$.

   AMAT $= 3.2 + 0.06 * (12 + 0.35 * (44 + 0.1 * 320)) = 5.456$ cycles.

4 Code analysis

Consider the following code:

```
struct body {
   float x, y, z, r;
};

struct body bodies[64];

// check whether two physics bodies overlap in 3D space
bool is_collide(struct body a, struct body b);

int check_collision() {
   int count = 0;
   for (int i = 0; i < 64; i++) {
      for (int j = i + 1; j < 64; j++) {
         if (is_collide(bodies[i], bodies[j])) {
            count++;
         }
      }
   }
   return count;
}
```

Note:

- Assume the cache parameters are: 128 bytes capacity, 32 bytes per block, 2-way set associative.
- Elements in the bodies array are aligned to the cache lines.
- sizeof(float) == 4
- You can ignore what is_collide exactly does and assume each body structure is loaded only **once** within the inner loop iteration (i.e., bodies[i] and bodies[j] are loaded into registers).
- The variables i, j and count are stored in registers.
- Instruction cache is not considered.
- Assume the cache is initially empty.

1. Calculate the hit rate for the above code.

Solution: Each block can store bodies[i] and bodies[i + 1]. 4 blocks, 2 sets in cache.

Beacause of the cache associativity, we can have 2 bodies in the cache at a time. And bodies[i] is accessed each loop, so it will always be in the cache.

For each i, accesses $= 64 - i$, hit $= \lfloor \frac{64-i}{2} \rfloor$. For $i \geq 57$, all kept in cache, so hit $=$ accesses.

In all, hit rate $= \frac{32+2*\sum_{i=4}^{31} + \sum_{i=1}^{7}}{\sum_{i=1}^{64}} = 0.5$

2. How can the hit rate be improved by modifying only the code (without changing the cache configuration)? Briefly explain your solution.

Solution: We can improve the hit rate by using cache blocking.

```
#define N 64
// Choose a block size B.
#define B 4

int check_collision_blocked() {
   int count = 0;

   for (int ii = 0; ii < N; ii += B) {
      for (int jj = ii; jj < N; jj += B) {
         int i_limit = min(ii + B, N);
         int j_limit = min(jj + B, N);

         for (int i = ii; i < i_limit; i++) {
            int j_start = (ii == jj) ? i + 1 : jj; // ensure j > i

            if (j_start < j_limit) {
               for (int j = j_start; j < j_limit; j++) {
                  if (is_collide(bodies[i], bodies[j])) {
                     count++;
                  }
               }
            }
         }
      }
   }
   return count;
}
```

Cache blocking divides the N x N iteration space into smaller B x B blocks. The outer loops (ii, jj) iterate over these blocks. The inner loops (i, j) iterate within a block. By processing pairs (bodies[i], bodies[j]) within a block (ii, jj), we increase the likelihood that both bodies[i] and bodies[j] remain in the cache for subsequent accesses within that block.