CSCI-UA 201 Recitation 9

Ishan Pranav

March 31, 2023

1 The average access time

Let p represent the probability of a cache hit, m represent the cache access time, and M represent the main memory access time.

$$p = 95\%.$$

$$m = 2.$$

$$M = 150.$$

The average memory access time per reference is A_{MAT}.

$$\begin{aligned} A_{\text{MAT}} &= mp + (M + m)(1 - p) \\ &= (2)(95\%) + (150 + 2)(5\%) \\ &= 9.5. \end{aligned}$$

The average memory access time for 100 references is $100A_{\text{MAT}}$, or 950 cycles.

$$100A_{\text{MAT}} = 100 \times 9.5 = 950.$$

2 A medical application

Let p_t represent the probability of a memory access operation, p represent the probability of a cache hit, t represent the non-memory-access time, m represent the cache access time, and M represent the main memory access time.

$$p_t = 30\%.$$

$$t = 1.$$

$$p = 80\%.$$

$$m = 1.$$

$$M = 20.$$

The average operations time is A, or 5.3 cycles per operation.

$$\begin{aligned} A &= p_t t + mp + (M + m)(1 - p) \\ &= (30\%)(1) + (1)(80\%) + (20 + 1)(1 - 80\%) \\ &= 5.3. \end{aligned}$$

3 The breakdown of the direct-mapped cache address

The address size is 64 bits, the cache size is 32 kB, the block size is 64 bytes, and the cache is direct-mapped.

The offset size is 6 bits, required to address the 64 bytes per block.

$$\log_2(64) = 6.$$

The number of blocks is 512.

$$\frac{32 \text{ kB}}{64 \text{ B}} \times \frac{1024 \text{ B}}{1 \text{ kB}} = \frac{32768 \text{ B}}{64 \text{ B}} = 512.$$

The index size is 9 bits, required to address the 512 blocks.

$$\log_2(512) = 9.$$

The address size is 64 bits. The least significant 6 bits represent the offset, the next 9 bits represent the index, and the remaining 49 bits represent the tag.

4 The breakdown of the four-way set-associative cache address

The address size is 24 bits, the cache size is 64 kB, the block size is 16 bytes, and the cache is four-way set-associative.

The number of blocks is 4096.

$$\frac{64 \text{ kB}}{16 \text{ B}} \times \frac{1024 \text{ B}}{1 \text{ kB}} = \frac{65536 \text{ B}}{16 \text{ B}} = 4096.$$

The number of sets is 1024.

$$4096 \text{ blocks} \times \frac{1 \text{ set}}{4 \text{ blocks}} = 1024 \text{ sets}.$$

The offset size is 4 bits, required to address the 16 bytes per block.

$$\log_2(16) = 4.$$

The index size is 10 bits, required to address the 1024 sets.

$$\log_2(1024) = 10.$$

The address size is 24 bits. The least significant 4 bits represent the offset, the next 10 bits represent the index, and the remaining 10 bits represent the tag.

5 A small C program

```c
#include <stdio.h>

/**
 * Computes the floor of the integer binary logarithm of the given number.
 *
 * @param n the number
 * @return The base-2 logarithm, rounded down to the nearest integer.
 */

static int floorLog2(unsigned int n)
{
   return 31 - __builtin_clz(n);
}

/**
 * The main entry point for the application.
 *
 * @return An exit code.
 */

int main()
{
   int cacheSize, addressLength, blockSize, associativity;

   printf("Cache size\t(kB):\t");
   scanf("%d", &cacheSize);
   printf("Address length\t(bits):\t");
   scanf("%d", &addressLength);
   printf("Block size\t(B):\t");
   scanf("%d", &blockSize);
   printf("Associativity\t(-way):\t");
   scanf("%d", &associativity);

   int index = floorLog2((cacheSize * 1024 / blockSize) / associativity),
       offset = floorLog2(blockSize),
       tag = addressLength - (offset + index);

   printf("\nTag:\t%d\tbits\nIndex:\t%d\tbits\nOffset:\t%d\tbits\n",
      tag, index, offset);
}
```