Universidad Carlos III de Madrid

Department of informatics
Computer Science degree
Computer Architecture

ARCOS

Cache memory exercises

B.1 You are trying to appreciate how important the principle of locality is in justifying the use of a cache memory, so you experiment with a computer having an L1 data cache and a main memory (you exclusively focus on data accesses). The latencies (in CPU cycles) of the different kinds of accesses are as follows: cache hit, 1 cycle; cache miss, 105 cycles; main memory access with cache disabled, 100 cycles.

a) When you run a program with an overall miss rate of 5%, what will the average memory access time (in CPU cycles) be'?
b) Next, you run a program specifically designed to produce completely random data addresses with no locality. Toward that end, you use an array of size 256 MB (all of it fits in the main memory). Accesses to random elements of this array are continuously made (using a uniform random number generator to generate the elements indices). If your data cache size is 64 KB, what will the average memory access time be?
c) If you compare the result obtained in part (b) with the main memory access time when the cache is disabled, what can you conclude about the role of the principle of locality in justifying the use of cache memory?
d) You observed that acache hit produces a gain of 99 cycles (1 cycle vs. 100). but it produces a loss of 5 cycles in the case of a miss (105 cycles vs.100). In the general case, we can express these two quantities as G (gain) andL (loss). Using these two quantities (G and L), identify the highest miss rate after which the cache use would be disadvantageous.

B.2 For the purpose of this exercise, we assume that we have 512-byte cache with &-byte blocks. We will also assume that the main memory is 2 KB large. We can regard the memory as an array of 64-byte blocks: MO, M1, …, M31. Figure B.30 sketches the memory blocks that can reside in different cache blocks if the cache was fully associative.

a) Show the contents of the table if cache is organized as a direct mapped cache.
b) Repeat part (a) with the cache organized as a four-way set associative cache.

B.3 Cache organization is often influenced by the desire to reduce the cache's power consumption. For that purpose we assume that the cache is physically distributed into a data array (holding the data), tag array (holding the tags), and replacement array (holding information needed by replacement policy). Furthermore, every one of these arrays is physically distributed into multiple sub-arrays (one per way) that can be individually accessed; for example, a four-way set associative least recently used (LRU) four data sub-arrays, four tag sub-arrays, and four replacement sub-arrays. We assume that the replacement sub-arrays are accessed once per access when the LRU replacement policy is used, and once per miss if the first-in, first-out (FIFO) replacement policy is used. It is not needed when a random replacement policy is used. For a specific cache, it was determined that the accesses to the different arrays have the following power consumption weights:

Array	**Power consumption weight (per way accessed)**
Data array	20 units
Tag Array	5 units
Miscellaneous array	1 unit

Estimate the cache power usage (in power units) for the following configurations. We assume the cache is four-way set associative. Main memory access power -albeit important-is not considered here. Provide answers for the LRU, FIFO, and random replacement policies.

a) A cache read hit. All arrays are read simultaneously.
b) Repeat part (a) for a cache read miss.
c) Repeat part (a) assuming that the cache access is split across two cycles. In the first cycle, all the tag sub-arrays are accessed. In the second cycle, only the sub-array whose tag matched will be accessed.
d) Repeat part (c) for a cache read miss (no data array accesses in the second cycle).
e) Repeat part (c) assuming that logic is added to predict the cache way to be accessed. Only the tag sub-array for the predicted way is accessed in cycle one. A way hit (address match in predicted way) implies a cache hit. A way miss dictates examining all the tag sub-arrays in the second cycle. In case of a way hit, only one data sub-array (the one whose tag matched) is accessed in cycle two. Assume there is way hit.
f) Repeat part (e) assuming that the way predictor missed (the way it chose is wrong). When it fails, the way predictor adds an extra cycle in which it accesses all the tag sub-arrays. Assume a cache read hit.
g) Repeat part (f) assuming a cache read miss.
h) Use parts (e), (f), and (g) for the general case where the workload has the following statistics: way-predictor miss rate = 5% and cache miss rate = 3%. (Consider different replacement policies.)

B.4 We compare the write bandwidth requirements of write-through versus write-back caches using a concrete example. Let us assume that we have a 64 KB cache with a line size of 32 bytes. The cache will allocate a line on a write miss. If configured as a write-back cache, it will write back the whole dirty line if it needs to be replaced. We will also assume that the cache is connected to the lower level in the hierarchy through a 64-bit-wide (8-bytewide) bus. The number of CPU cycles for a B-bytes write access on this bus is: 10 + 5*([B/8]-1).

Answer the following questions while referring to the C code snippet below:

```
#define PORTION 1
for (j=base; j < base+PORTION; j++) //assume j is stored in a register
      data[j]=j;
```

a) For a write-through cache, how many CPU cycles are spent on write transfers to the memory for the all the combined iterations of the j loop?
b) If the cache is configured as a write-back cache, how many CPU cycles are spent on writing back a cache line?
c) Change PORTION to 8 and repeat part (a).
d) What is the minimum number of array updates to the same cache line (before replacing it) that would render the write-back cache superior?