Computer Architecture

Homework 1

Student: Adil Hydari, adil.hydari@rutgers.edu
Professor: Maria Striki

Problem 1

1. Suppose that your processor has 4MB data cache and its block size is 256Bytes. Physical address to access the memory is 54-bit wide (addr[53:0]). For each of the following cache structures, calculate TAG size and index size.

   (a) A direct-mapped cache implementation
   The offset bits for a 256 byte implementation should be 8 bits to calculate this we can use log_2 of 256 to find that our offset for the block address is 8 bits. The same logic can be followed for our index bits, if we consider the log_2 of the number of blocks ($4,194,304 \div 256 = 16,384$), we end up with $log_2(16,384) = 14bits$. This only works because our block size and block number can both be found by computing some x for 2^x. Finally putting this all together, $54 - 14 - 8 = 32$ TAG bits.

   (b) An 8-Way set associative cache implementation
   In the case of a set associative cache, the amount of offset bits would remain the same however the formula for calculating the number of index bits changes (source): $Index = log_2(Blocks \div Associativity)$ and this comes out to be $log_2(2048) = 11$ bits for our index. Then, calculating for the TAG, $54 - 11 - 8 = 35$ bits.

   (c) A fully associative cache implementation
   For a fully associative cache, the offset would remain the same, however, in an associative cache structure any block is able to be placed anywhere in the cache, this means that we would not have any index bits and would instead require a comparator. $54 - 8 = 46$ TAG bits.

Problem 2

1. The following memory addresses are used consecutively by a running program (from left to right, shown in decimal). Note that the followings are memory address not block number:
   520, 408, 380, 560, 816, 1648, 204, 1348, 284, 440, 140, 1064, 44, 392, 404, 180
   In each of the following cache structures, compute the number of hits, misses and the final values stored in each cache location (show finally which block of memory is in each cache block). Each word is 4-bytes and the memory size is 64 Kbyte.

   (a) Direct-mapped cache with blocks containing 16 words and a total size of cache of 2048 words of data
   The first thing that has to be done in this case is to first convert all

the decimal values into words by dividing by 4. Then we can find the number of cache blocks, 2,048 words ÷ 16 words per block = 128 blocks, and the number memory blocks 16,384 words ÷ 16 words per block = 1,024 blocks (65,536 bytes ÷ 4 bytes per word = 16,384 words). With this information we can now layout a table that can simulate the hit and misses for this running program. Table 1 is the simulation. For calculating the block number, I divided by 16 then rounded to the nearest whole number and for the cache line, I did the block number % 128

(b) 4-way set associative cache with blocks containing 32 words each and a total size of cache 2048 words of data. (LRU replacement)
The same thing can be done for B. # of cache blocks: 2,048 words ÷ 32 words per block = 64 blocks. # of sets: 64 blocks ÷ 4 way = 16 sets. Finally, we will have 512 blocks for memory in the case that the cache cannot hold all the lines. Table 2 shows this simulation. In the end we did not exceed the block & set limit set by the cache.

Access	Address (Bytes)	Word Address	Block Number	Cache Line	Hit/Miss	Action
1	520	130	8	8	Miss	Load block 8 into line 8
2	408	102	6	6	Miss	Load block 6 into line 6
3	380	95	5	5	Miss	Load block 5 into line 5
4	560	140	8	8	Hit	–
5	816	204	12	12	Miss	Load block 12 into line 12
6	1648	412	25	25	Miss	Load block 25 into line 25
7	204	51	3	3	Miss	Load block 3 into line 3
8	1348	337	21	21	Miss	Load block 21 into line 21
9	284	71	4	4	Miss	Load block 4 into line 4
10	440	110	6	6	Hit	–
11	140	35	2	2	Miss	Load block 2 into line 2
12	1064	266	16	16	Miss	Load block 16 into line 16
13	44	11	0	0	Miss	Load block 0 into line 0
14	392	98	6	6	Hit	–
15	404	101	6	6	Hit	–
16	180	45	2	2	Hit	–

Table 1: Simulation of Direct-Mapped Cache Accesses

Problem 3

In order to find the cache sizes needed to reflect what is given in problem 2, we can perform a calculation very similar to problem 1.

(a) The offset bits are $log_2(64) = 6$ bits. The index bits are $log_2(128) = 7$ bits. To find the physical addressable size of memory we can do $log_2(65,536) = 16$ bits. So then our TAG is $16 - 7 - 6 = 3$ bits. We know we have 128 cache lines so $128 * 3 = 384$ bits for the TAG, 128 bits for the valid bit for each line, then adding these two we end up with 64 bytes. Then we can finally add this to the cache size ($2048 * 4 = 8,192$ bytes), $8,192 + 64 = 8,256$ bytes for our cache.

(b) For a set associative cache the calculations are a bit different. Our offset bits remain the same at 7, however to calculate the index bits, we must take the log_2 of the number of sets (16) which is 4 bits. We can then figure out that our TAG is $16 - 4 - 7 = 5$ bits. To find the amount of storage needed for TAG $64 * 5$ bits = 320 bits (64 being the amount of blocks). The same can be done for valid where only 64 bits are needed. Thus the total amount needed is $8,192 + 54 = 8,246$ bytes.

Access	Address (Bytes)	Word Address	Block Number	Cache Set	Hit/Miss	Action
1	520	130	4	4	Miss	Load block 4 into set 4
2	408	102	3	3	Miss	Load block 3 into set 3
3	380	95	2	2	Miss	Load block 2 into set 2
4	560	140	4	4	Hit	Update LRU in set 4
5	816	204	6	6	Miss	Load block 6 into set 6
6	1648	412	12	12	Miss	Load block 12 into set 12
7	204	51	1	1	Miss	Load block 1 into set 1
8	1348	337	10	10	Miss	Load block 10 into set 10
9	284	71	2	2	Hit	Update LRU in set 2
10	440	110	3	3	Hit	Update LRU in set 3
11	140	35	1	1	Hit	Update LRU in set 1
12	1064	266	8	8	Miss	Load block 8 into set 8
13	44	11	0	0	Miss	Load block 0 into set 0
14	392	98	3	3	Hit	Update LRU in set 3
15	404	101	3	3	Hit	Update LRU in set 3
16	180	45	1	1	Hit	Update LRU in set 1

Table 2: Simulation of 4-Way Set Associative Cache Accesses (LRU Replacement)

Problem 4

In each of the following three cases, calculate the CPI (Cycles Per Instruction) for a processor with these specifications:
Base CPI=4
Processor speed=3 GHz $\rightarrow$ 1/3 GHz = 0.333 ns
Main memory access time=200ns
First-level cache miss rate per instruction=15%

(a) We only have a first level (L1) cache
First we can calculate the miss penalty for the L1 cache:

$$\frac{200ns}{0.333ns/cycle} = 600cycles$$

Then, the effective CPI will be

$$4 + (0.15 * 600) = 94CPI$$

(b) Along with L1 cache, we also have a second level direct-mapped cache in which:
Second-level cache direct-mapped speed=15 cycle
Global miss rate with second-level cache direct-mapped=6%
Our global miss rate is 0.06, with an access time of 15 cycles. This means that our penalty is $\frac{15}{0.333} = 45.05$ cycles and from the L1 calculation we know the primary miss penalty is 600 cycles. From this we can calculate: $4+(0.15*45.05)+(.06*600) = 46.7575$ **CPI**

(c) Along with L1 cache, we also have a second level 4-way set associative cache in which:
Second-level cache 4-way set-associative speed=25 cycles.
Global miss rate with second-level cache 4-way set-associative=3%
It is a similar calculation for the set associative when compared to the base implementation.

$$\frac{25}{0.333} = 75.08$$

$$4 + (0.15 * 75.08) + (.03 * 600) = 33.262\textbf{CPI}$$

Problem 5

(a) What is the average memory access time (AMAT) if a cache uses write-back strategy and 25% of the data blocks to be swapped out are dirty. Assume that the miss rate is 12%, the hit time of the cache is 1 cycle, and the miss penalty is 7 cycles for the data blocks that are not dirty and 24 cycles for those blocks that are dirty.
(b) What is the speedup if we add a "write-buffer" that eliminates 35% of the stall cycles to write back the dirty blocks?
(a) To calculate the Average Memory Access Time (AMAT), we use the formula:

$$\text{AMAT} = \text{Hit Time} + (\text{Miss Rate} \times \text{Miss Penalty})$$

Given:

- Miss rate (MR) = 12% = 0.12

- Hit time (HT) = 1 cycle
- Miss penalty for clean blocks (MPC) = 7 cycles
- Miss penalty for dirty blocks (MPD) = 24 cycles
- Probability that a block to be swapped out is dirty (P_{dirty}) = 25% = 0.25
- Probability that a block to be swapped out is clean (P_{clean}) = 75% = 0.75

Find the Miss rate

$$\begin{aligned} AMP &= (P_{\text{clean}} \times MPC) + (P_{\text{dirty}} \times MPD) \\ &= (0.75 \times 7\,\text{cycles}) + (0.25 \times 24\,\text{cycles}) \\ &= 5.25\,\text{cycles} + 6\,\text{cycles} \\ &= 11.25\,\text{cycles} \end{aligned}$$

Find the AMAT

$$\begin{aligned} AMAT &= HT + (MR \times AMP) \\ &= 1\,\text{cycle} + (0.12 \times 11.25\,\text{cycles}) \\ &= 1\,\text{cycle} + 1.35\,\text{cycles} \\ &= 2.35\,\text{cycles} \end{aligned}$$

(b) We need to calculate the speedup achieved by adding a write buffer that eliminates 35% of the stall cycles to write back dirty blocks.
Given:

- The write buffer reduces the miss penalty for dirty blocks by 35% of the difference between the dirty and clean miss penalties.

Find the reduction in miss penalty

$$\begin{aligned} \Delta MPD &= 0.35 \times (MPD - MPC) \\ &= 0.35 \times (24\,\text{cycles} - 7\,\text{cycles}) \\ &= 0.35 \times 17\,\text{cycles} \\ &= 5.95\,\text{cycles} \end{aligned}$$

Find the new Miss penalty(MPD')

$$MPD' = MPD - \Delta MPD = 24\,\text{cycles} - 5.95\,\text{cycles} = 18.05\,\text{cycles}$$

Find the average Miss penaltyAMP')

$$\begin{aligned} AMP' &= (P_{\text{clean}} \times MPC) + (P_{\text{dirty}} \times MPD') \\ &= (0.75 \times 7\,\text{cycles}) + (0.25 \times 18.05\,\text{cycles}) \\ &= 5.25\,\text{cycles} + 4.5125\,\text{cycles} \\ &= 9.7625\,\text{cycles} \end{aligned}$$

Find new memory access time ($AMAT'$)

$$\begin{aligned} AMAT' &= HT + (MR \times AMP') \\ &= 1\,\text{cycle} + (0.12 \times 9.7625\,\text{cycles}) \\ &= 1\,\text{cycle} + 1.1715\,\text{cycles} \\ &= 2.1715\,\text{cycles} \end{aligned}$$

Step 5: Calculate the Speedup

$$\begin{aligned} \text{Speedup} &= \frac{\text{Original } AMAT}{\text{New } AMAT} \\ &= \frac{2.35\,\text{cycles}}{2.1715\,\text{cycles}} \\ &\approx 1.082 \end{aligned}$$

Problem 6

Virtual memory uses a page table to track the mapping of virtual addresses to physical addresses. The following data constitutes a stream of virtual addresses as seen on a system. Assume 4 KB pages, a 4-entry fully associative TLB, and LRU replacement. If pages must be brought in from disk, increment the next largest page number. Virtual addresses: 4669, 2227, 13916, 34587, 48870, 12608, 49225.

Initial TLB:

Valid	Tag	Physical Page Number
1	11	12
1	7	4
1	3	6
0	4	9

Initial Page Table:

Valid	Physical Page or Disk
1	5
0	Disk
0	Disk
1	6
1	9
1	11
0	Disk
1	4
0	Disk
0	Disk

Given the address stream shown and the initial TLB and Page Table states provided above, determine for each reference whether it is a hit in the TLB, a hit in the page table, or a page fault. Continue in the same format as the examples provided.

Example Entries:

Address	VPN	TLB H/M	PT H/M	Page Fault	TLB Entries After Access
4669	1	Miss	Miss	Yes	Entry 3: Valid=1, Tag=1, PPN=13
2227	0	Miss	Hit	No	Entry 0: Valid=1, Tag=0, PPN=5

Now, let's proceed with the remaining addresses:

1. **Address: 13916**

   Calculation of Virtual Page Number (VPN):

   $$\text{VPN} = \left\lfloor \frac{13916}{4096} \right\rfloor = 3$$

   TLB Lookup:

   - VPN 3 is in TLB Entry 2 with Tag 3 and Valid bit is 1. - **TLB Hit**

   Update TLB Entry's Last Access Time:

   - Update Entry 2's last access time to the current time.

2. **Address: 34587**

   Calculation of Virtual Page Number (VPN):

   $$\text{VPN} = \left\lfloor \frac{34587}{4096} \right\rfloor = 8$$

   TLB Lookup:

   - VPN 8 is not in the TLB. - **TLB Miss**

   Page Table Lookup:

   - VPN 8 is invalid (Valid bit is 0). - **Page Table Miss**

   Page Fault:

   - **Yes**

Action:

- Assign next available Physical Page Number (PPN): PPN 14.
- Update Page Table:
- Set VPN 8 Valid bit to 1.
- Assign PPN 14 to VPN 8.
- Update TLB:
- Replace LRU Entry (Entry 1).
- Set Entry 1: Valid=1, Tag=8, PPN=14.

3. **Address: 48870**

   Calculation of Virtual Page Number (VPN):

$$\text{VPN} = \left\lfloor \frac{48870}{4096} \right\rfloor = 11$$

   TLB Lookup:

   - VPN 11 is not in the TLB. - **TLB Miss**

   Page Table Lookup:

   - VPN 11 is valid with PPN 12 (from initial TLB). - **Page Table Hit**

   Page Fault:

   - **No**

   Update TLB:

   - Replace LRU Entry (Entry 3). - Set Entry 3: Valid=1, Tag=11, PPN=12.

4. **Address: 12608**

   Calculation of Virtual Page Number (VPN):

$$\text{VPN} = \left\lfloor \frac{12608}{4096} \right\rfloor = 3$$

   TLB Lookup:

   - VPN 3 is in TLB Entry 2. - **TLB Hit**

   Update TLB Entry's Last Access Time:

   - Update Entry 2's last access time.

5. **Address: 49225**

   Calculation of Virtual Page Number (VPN):

$$\text{VPN} = \left\lfloor \frac{49225}{4096} \right\rfloor = 12$$

   TLB Lookup:

   - VPN 12 is not in the TLB. - **TLB Miss**

   Page Table Lookup:

   - VPN 12 is invalid. - **Page Table Miss**

   Page Fault:

   - **Yes**

Actions:

- Assign next available PPN: PPN 15.
- Update Page Table:
- Set VPN 12 Valid bit to 1.
- Assign PPN 15 to VPN 12.
- Update TLB:
- Replace LRU Entry (Entry 0).
- Set Entry 0: Valid=1, Tag=12, PPN=15.

Table summary on next page

Summary Table for Problem 6:

Address	VPN	TLB H/M	PT H/M	Page Fault	TLB Entries After Access
4669	1	Miss	Miss	Yes	Entry 3: Valid=1, Tag=1, PPN=13
2227	0	Miss	Hit	No	Entry 0: Valid=1, Tag=0, PPN=5
13916	3	Hit	-	No	Entry 2's Last Access Updated
34587	8	Miss	Miss	Yes	Entry 1: Valid=1, Tag=8, PPN=14
48870	11	Miss	Hit	No	Entry 3: Valid=1, Tag=11, PPN=12
12608	3	Hit	-	No	Entry 2's Last Access Updated
49225	12	Miss	Miss	Yes	Entry 0: Valid=1, Tag=12, PPN=15