COMPUTER ARCHITECTURE & ASSEMBLY LANGUAGE
14:332:331
Rutgers University
Fall 2016

Homework 5 (Solution)

Note: Question 6 is for your practice. You do not need to answer question 6.

1. Suppose that your processor has 8MB data cache and its block size is 64B. 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
   (b) A 4-Way set associative cache implementation
   (c) A fully associative cache implementation

Answer:

a) # of blocks = 8MB/64B=$2^{23}/2^{6}=2^{17}$→index:17 bits
Tag size= 54- 6 (for block size) – 17 (for index) =31

b) #of set= $2^{17}/4=2^{15}$→index:15 bits
Tag size= 54- 6 (for block size) – 15 (for index) =33

c) Tag size= 54- 6 (for block size) = 48 ; Index Size =0

2. 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, 400, 380, 540, 816, 204, 1348, 200, 440, 140, 1064, 44, 196, 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 8Kbyte

   (a) Direct-mapped cache with 32-word blocks and a total size of cache is 128 words of data
   (b) 2-way set associative cache with 32-word blocks and a total size of cache is 128 words of data. (LRU replacement)

Answer:

a) # of cache block= 128/32 = 4
Size of each block=32 (since one block is 32 words)*4 (since one word is 4 bytes)=128 byte

Memory Address	Block # in Main Memory	Block in Cache	Word address included	Hit/Miss
520	4 (=520/128)	0 (=4%4)	512, 516,……636	Miss
400	3	3	384, 386,……508	Miss

380	2	2	256,260……..380	Miss
540	4	0	512, 516,……636	Hit
816	6	2	768,772,….892	Miss
204	1	1	128,132,…….252	Miss
1348	10	2	1280, 1284,…..1404	Miss
200	1	1	128,132,…….252	Hit
440	3	3	384, 386,……508	Hit
140	1	1	128,132,…….252	Hit
1064	8	0	1024, 1028,….1148	Miss
44	0	0	0, 4, 8,…..124	Miss
196	1	1	128,132,…….252	Hit
404	3	3	384, 386,……508	Hit
180	1	1	128,132,…….252	Hit

➔ Hit rate=7/15=46.7%, Miss_rate=53.3%. The final values stored in cache are as below:

Block 0 of cache	0, 4, 8,…..124
Block 1 of cache	128,132,…….252
Block 2 of cache	1280, 1284,…..1404
Block 3 of cache	384, 386,……508

b) # of cache block= 128/32=4
# of sets = 4/2 =2
Size of each block=32*4=128 byte

Memory Address	Block # in Main Memory	Set in Cache	Word address included In block 0 of set 0	Word address included In block 1 of set 0	Word address included In block 0 of set 1	Word address included In block 1 of set 1	Hit/ Miss
520	4(=520/128)	0	512,516, ……636				Miss
400	3	1	512,516, ……636		384,386,……508		Miss
380	2	0	512,516, ……636	256,260 ……380	384,386,……508		Miss
540	4	0	512,516, ……636	256,260 ……380	384,386,……508		Hit
816	6	0	768,772, ….892	256,260 ……380	384,386,……508		Miss
204	1	1	768,772, ….892	256,260 ……380	384,386,……508	128,132,…. ….252	Miss
1348	10	0	768,772, ….892	1280,….. 1404	384,386,……508	128,132,…. ….252	Miss
200	1	1	768,772, ….892	1280,….. 1404	384,386,……508	128,132,…. ….252	Hit
440	3	1	768,772, ….892	1280,….. 1404	384,386,……508	128,132,…. ….252	Hit
140	1	1	768,772, ….892	1280,….. 1404	384,386,……508	128,132,…. ….252	Hit

1064	8	0	1024,…. 1148	1280,….. 1404	384,386,… …508	128,132,… ….252	Miss
44	0	0	1024,…. 1148	0,4,8,….. 124	384,386,… …508	128,132,… ….252	Miss
196	1	1	1024,…. 1148	0,4,8,….. 124	384,386,… …508	128,132,… ….252	Hit
404	3	1	1024,…. 1148	0,4,8,….. 124	384,386,… …508	128,132,… ….252	Hit
180	1	1	1024,…. 1148	0,4,8,….. 124	384,386,… …508	128,132,… ….252	Hit

Hit rate=7/15=46.7%, Miss_rate=53.3%.

3. For each part of Question 2, find the size of the cache required to hold the data (as mentioned in 2.a and 2.b above the cache should hold 128 words of data)?
Answer:

a) # of bits for memory addressing=$\log_2$ 8K=13
Index_size= $\log_2$ 4 (since we have 4 blocks in cache)=2, offset= $\log_2$ 128=7→ tag_size=13-2-7=4

For each block we need: 1 (for valid)+4(for tag)+128*8 (for data per block)=1029 bit
➔ Total cache size=1029*4(=number of blocks)=4116 bit

b) # of bits for memory addressing=$\log_2$ 8K=13
Index_size= $\log_2$ 2(since we have 2 sets in cache)=1, offset= $\log_2$ 128=7→ tag_size=13-1-7=5

For each block we need: 1 (for valid)+5(for tag)+128*8 (for data per block)=1030 bit
➔ Total cache size=1030*4(=number of blocks)=4120 bit=515 bytes

4. In each of the following three cases, calculate the CPI for a processor with these specifications:
   - Base CPI=3
   - Processor speed=2 GHz
   - Main memory access time=100ns
   - First-level cache miss rate per instruction=12%

(a) We only have a first level (L1) cache
(b) Along with L1 cache, we also have a second level direct-mapped cache in which
. Second-level cache direct-mapped speed=20 cycle
. Global miss rate with second-level cache direct-mapped=4%

(c) Along with L1 cache, we also have a second level 4-way set associative cache in which:
.Second-level cache 8-way set-associative speed=24 cycle
.Global miss rate with second-level cache 4-way set-associative=3%

Answer:

Main memory access time =100ns, clk_cycle_time=1/(2GHZ)=0.5ns ➔ Main memory access time=100/0.5=200 clock cycle

a) CPI= 3+12%*200=27

b) CPI= 3+12%*20+4%*200= 13.4

c) CPI= 3+12%*24+3%*200= 11.88

5.

(a) What is the average memory access time (AMAT) if a cache uses write-back strategy and 20% of the data blocks to be swapped out are dirty. Assume that the miss rate is 15%, the hit time of the cache is 1 cycle and the miss penalty is 8 cycles for the data blocks that are not dirty and 20 cycles for those blocks that are dirty.

(b) What is the speedup up if we add a "write-buffer" that eliminates 40% of the stall cycles to write back the dirty blocks?

Answer:

a) AMAT=1+15%(20*20%+8*(100%-20%))=1+15%(20*20%+8*(80%))=2.56

b) Buffer removes 40% of stall cycles for dirty blocks so in case of dirty blocks we spend only 60% of the time we spent in part "a", meaning that for dirty blocks we spend 0.60*20=12 cycles

AMT=1+15%(12*20%+8*80%)=

Speedup=2.56/2.26=1.1327

6. As discussed, virtual memory uses a page table to track the mapping of virtual addresses to physical addresses. This exercise shows how this table must be updated as addresses are accessed. 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

TLB

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

Page Table

Valid	Physical Page or in Disk
1	5

0	Disk
0	Disk
1	68
1	9
1	11
0	Disk
1	4
0	Disk
0	Disk
1	3
1	12

Given the address stream shown, and initial TLB and Page Table States provided above, show if each reference is hit in the TLB, a hit in the page table, or a page fault. (To make it clear, the first two accesses have been shown below. Please continue in the same way)

Address	Virtual Page	TLB H/M	TLB		
			Valid	Tag	Physical Page
4669	1	TLB miss PT hit PF	1	11	12
			1	7	4
			1	3	6
			1 (last access 0)	1	13
2227	0	TLB miss PT hit	1 (last access 1)	0	5
			1	7	4
			1	3	6
			1 (last access 0)	1	13

Answer: The following table shows the status of TLB after each access.

Address	Virtual Page	TLB H/M	TLB		
			Valid	Tag	Physical Page
4669	1	TLB miss PT hit PF	1	11	12
			1	7	4
			1	3	6
			1 (last access 0)	1	13
2227	0	TLB miss PT hit	1 (last access 1)	0	5
			1	7	4
			1	3	6
			1 (last access 0)	1	13
13916	3	TLB hit	1 (last access 1)	0	5
			1	7	4
			1 (last access 2)	3	6
			1 (last access 0)	1	13
34587	8	TLB miss PT hit PF	1 (last access 1)	0	5
			1 (last access 3)	8	14
			1 (last access 2)	3	6
			1 (last access 0)	1	13
48870	11	TLB miss PT hit	1 (last access 1)	0	5
			1 (last access 3)	8	14
			1 (last access 2)	3	6
			1 (last access 4)	11	12
12608	3	TLB hit	1 (last access 1)	0	5
			1 (last access 3)	8	14
			1 (last access 5)	3	6
			1 (last access 4)	11	12
49225	12	TLB miss PT miss	1 (last access 6)	12	15
			1 (last access 3)	8	14
			1 (last access 5)	3	6
			1 (last access 4)	11	12