Homework 4

Hamza Kamal

February 28, 2025

Question:

Q1. Cache and Memory mapping (6 points)
Suppose a byte-addressable memory has a total memory capacity of 2M bytes and the cache consists of 64 blocks, where each block contains 32 bytes.
1. Direct Mapping
a. Divide the bits into tag, block and offset bits.

Answer:

$\log_2(2M) = \log_2(2 * 2^{20})$
$\log_2(2 * 2^{20}) = 21$ bits
Every memory address is represented by 21 bits.
Each cache block contains 32 bytes so we need:
$\log_2(32) = 5$ bits = Offset.
The cache size is 64 blocks, so to address 64 rows we need:
$\log_2(64)$ bits.
$\log_2(64) = 6$ bits = Row Index.
The remaining bits are tag bits.
We have 6 Offset bits and 6 Row Index bits, leaving us with $21 - 6 - 5$.
$= 10$ tag bits.
Tag: 10
Row Index: 6
Offset: 5

Question:

b. What is the tag, line and offset for the address \$123A63, in hexadecimal?

Answer:

Convert 123A63 into binary format.
= 000100100011101001100011
Tag: 1001000111
Row Index: 010011
Offset: 00011
Convert back to hexadecimal.
Tag: $0x247$
Row Index: $0x13$
Offset: $0x3$

Question:

2. Fully Associative Mapping
a. Divide the bits into tag and offset bits.

Answer:

Offset bit is determined by the byte within a block.
$\log_2(32) = 5$ bits for offset.
Tag bits is determined by the following formula:
$\log_2(L/K)$ where L is memory size and K is block size
$\log_2((2 * 2^{20})/(32))$
= 16 tag bits.
Tag: 16
Offset: 5

Question:

b. What is the tag and offset for the address $\$123A63$, in hexadecimal?

Answer:

Convert $123A63$ into binary:
000100100011101001100011
Tag: 1001000111010011
Offset: 00011
Convert back to hexadecimal:
Tag: $0x91D3$
Offset: $0x3$

Question:

3. 4-way set associative mapping
a. Divide the bits into tag, set and offset bits

Answer:

The formula for calculating the offset bits is $\log_2(K)$ where K is the cache block size.
$\log_2(32) = 5$ bits for offset.
The formula for calculating the sets bits is $\log_2((M/K)/N)$ where M is the size of the cache, K is the cache block size and N is 4.
$M = 64 * 32 = 2048$
$\log_2((2048/32)/4)$
$= 4$ bits for the set.
The formula for calculating the tag bits is $\log_2(L/(M/N))$ where L is memory

size, M is the size of the cache, and N is 4.
$\log_2(2M/(2048/4))$ $\log_2((2*2^{20})/(2048/4)) = 12$ bits for tag.
Tag: 12
Set: 4
Offset 5

Question:

b. What is the tag, set and offset for the address \$$123A63$, in hexadecimal?

Answer:

Convert $123A63$ into binary.
000100100011101001100011
Tag: 100100011101
Set: 0011
Offset: 00011
Convert back to hexadecimal:
Tag: $0x91D$
Set: $0x3$
Offset $0x3$

Question:

Q2. Cache hit and miss (3 points)
Suppose we have a computer that uses a memory with a total memory capacity

of 256 bytes. The computer has a 16-byte direct-mapped cache with 4 bytes per block. The computer accesses a number of memory locations throughout the course of running a program. Here is the memory addresses in this exact order: 0x91, 0xA8, 0xA9, 0xAB, 0xAD, 0x93, 0x6E, 0xB9, 0x17, 0xE2, 0x4E, 0x4F, 0x50, and 0xA4. The cache Tag and Block information has been filled out as shown below.

Tag (binary)	**Block #**	**Offset 0**	**Offset 1**	**Offset 2**	**Offset 3**
1110	0				
0001	1				
1011	2				
0110	3				

Table 1: Tag and Block Mapping Table

1. What is the hit ratio for the entire memory reference sequence (given in bold)?

Answer:

The total number of bits for each memory address is determined by $\log_2(L)$ where L is the memory capacity.
$\log_2(256) = 8$ bits. The formula for calculating the offset is $\log_2(K)$ where K is the cache block size.
$\log_2(4) = 2$ bits for the offset. The formula for the row index is $\log_2(M/K)$ where M is the cache size and K is the cache block size.
$\log_2(16/4) = 2$ bits for the row index.
The rest of the bits are the tag bits.
$8 - 2 - 2 = 4$ bits for tag.
Now we have to convert each address into binary and check if its a hit or miss.
$0x91 = 10010001$
Tag: 1001
Row Index: 00
Offset: 01
Cache is empty at 0 so this is a miss.
Update data.

$0xA8 = 10101000$
Tag: 1010
Row Index: 10
Offset: 00
Cache is empty at 2 so this is a miss.
Update data.

$0xA9 = 10101001$
Tag: 1010
Row Index: 10
Offset: 01
Cache has tag 1010 at 2 so this is a hit.

$0xAB = 10101011$
Tag: 1010
Row Index: 10
Offset: 11
Cache has tag 1010 at 2 so this is a hit.

$0xAD = 10101101$
Tag: 1010
Row Index: 11
Offset: 01
Cache is empty at 3 so this is a miss.
Update data.

$0x93 = 10010011$
Tag: 1001
Row Index: 00
Offset: 11
Cache has tag 1001 at 0 so this is a hit.

$0x6E = 01101110$
Tag: 0110
Row Index: 11
Offset: 10
Cache has tag 1010 at 3 so this is a miss.
Update data.

$0xB9 = 10111001$
Tag: 1011
Row Index: 10
Offset: 01
Cache has tag 1010 at 2 so this is a miss.
Update data.

$0x17 = 00010111$
Tag: 0001
Row Index: 01
Offset: 11
Cache is empty at 1 so this is a miss.
Update data.

$0xE2 = 11100010$
Tag: 1110
Row Index: 00
Offset: 10
Cache has tag 1001 at 0 so this is a miss.
Update data.

$0x4E = 01001110$
Tag: 0100
Row Index: 11
Offset: 10
Cache has tag 0110 at 3 so this is a miss. Update data.

$0x4F = 01001111$
Tag: 0100
Row Index: 11
Offset: 11
Cache has tag 0100 at 3 so this is a hit.

$0x50 = 01010000$
Tag: 0101
Row Index: 00
Offset: 00
Cache has tag 1110 at 0 so this is a miss. Update data.

$0xA4 = 10100100$
Tag: 1010
Row Index: 01
Offset: 00
Cache has tag 0001 at 1 so this is a miss. Update data.

Total hits: 4.
Total Addresses: 14
Hit Ratio $= 4/14 \approx 0.2857$
$= 28.57\%$

Question:

2. What memory blocks will be in the cache after the last address has been assessed? Please fill in the Tag and Block first. Then, fill the actual address value for each offset location in the corresponding cell.

Answer:

Cache Line	**Tag**	**Offset 0**	**Offset 1**	**Offset 2**	**Offset 3**
0	0100	0x50	0x51	0x52	0x53
1	1010	0xA4	0xA5	0xA6	0xA7
2	1011	0xB8	0xB9	0xBA	0xBB
3	0110	0x6C	0x6D	0x6E	0x6F

Table 2: Cache State After Last Access

Question:

Q3. Virtual memory and cache (6 points)
Consider a processor with the following memory hierarchy:
256K virtual address space (byte addressable)
128K physical address space, each page (frame) has 32K bytes (byte addressable)
2Kbyte direct-mapped cache, a block (refill line) has 256 bytes

The machine uses a two entry TLB.
All replacement policies are LRU. There are two LRU stack.
The entry of these stacks are the page number of a virtual memory.
Note that all the values are represented as hexadecimal.
TLB:

Virtual Page #	**Physical Page #**	**Valid**
5	3	1
0	2	1

TLB LRU Stack:

0
5

Page Table:

Virtual Page #	**Physical Page #**	**Valid**
0	2	1
1	1	1
2	—	0
3	—	0
4	0	1
5	3	1
6	—	0
7	—	0

Mem LRU Stack:

0
5
4
1

Cache:

Line #	**Tag**	**Data block**
0	10	*
1	0A	*
2	3C	*
3	14	*
4	28	*
5	04	*
6	37	*
7	1D	*

1. Split the bits of virtual address and physical address.

Answer:

Virtual Address:
The bits required for the virtual address space is:
$\log_2(256Kb)$
$\log_2(256 * 2^{10})$
$\log_2(2^{18}) = 18$ bits.
The bits required for a page is:
$\log_2(32Kb)$ $\log_2(32 * 2^{10})$
$\log_2(2^{15})$
$= 15$ bits.
The remaining 3 bits are reserved for the virtual page number.
Virtual Address: 18 bits.
Page Bits: 15 bits.
Page Number: 3 bits.

Physical Address:
The physical address space is:
$\log_2(128Kb)$ $\log_2(128 * 2^{10})$
$\log_2(2^{17}) = 17$ bits. The bits required for a page is:
$\log_2(32Kb)$ $\log_2(32 * 2^{10})$
$\log_2(2^{15})$
$= 15$ bits.
The remaining 2 bits are reserved for the frame number.
Physical Address: 17 bits.
Page Bits: 15 bits.
Frame Number: 2 bits.

Question:

2. Split the bits in memory address based on the cache.

Answer:

Since each block is 256 bytes the block offset is:
$\log_2(256Kb) = 8$ bits.
The index identifies which cache line the block is mapped to. Since we have 8 cache lines:
$\log_2(8) = 3$ bits for the cache index.
The remaining bits in the memory address will be used as the tag.
The physical address space is:
$\log_2(128Kb)$ $\log_2(128 * 2^{10})$
$\log_2(2^{17})$
$= 17$ bits.
Block Offset: 8 bits.
Cache Index: 3 bits.
Tag: $17 - 8 - 3 = 6$ bits.

Question:

Suppose the processor has requested to access a memory in 0x32764 (which is virtual address)
a. Is it a page fault? Explain.
Page Table:

Virtual Page #	**Physical Page #**	**Valid**
0		
1		
2		
3		
4		
5		
6		
7		

Cache:

Line #	**Tag**	**Data**
0		
1		
2		
3		
4		
5		
6		
7		

Answer:

Yes because in the page table, the virtual addres 0x3 does not point to a physical address.

Question:

b. Show the changes of TLB, TLB LRU, page table and Mem LRU

Answer:

The computer first looks at the TLB for the page.
TLB:

Virtual Page #	**Physical Page #**	**Valid**
5	3	1
0	2	1

It does not find the page in the TLB so it looks in the page table.
Page Table:

Virtual Page #	**Physical Page #**	**Valid**
0	2	1
1	1	1
2	—	0
3	—	0
4	0	1
5	3	1
6	—	0
7	—	0

In the page table the virtual page 3 does not point to a physical page so it results in a page fault.
The MEM LRU Stack has virtual page 1 in the physical memory. So page 1 will be replaced by page 3
New Page Table:

Virtual Page #	**Physical Page #**	**Valid**
0	2	1
1	—	1
2	—	0
3	1	0
4	0	1
5	3	1
6	—	0
7	—	0

New MEM LRU:

3
0
5
4

New TLB LRU:

3
0

Question:

c. Show the changes in Cache.

Answer:

New Cache:

Line #	Tag	Data block
0	10	*
1	0A	*
2	3C	*
3	14	*
4	28	*
5	04	*
6	37	*
7	1D	*

The new value of the cache does not change as the requested virtual address

$0x32764$ results in a page fault and the cache is not accessed to retrieve or replace data.
