Note: Do not refer to the processor configuration in the case study notes for this tutorial. A smaller system will be used instead.

8.1 Cache

1. Given a processor system with the following characteristics
   - Processor has a direct-mapped cache with 32 cache blocks and a cache size of 512 bytes.
   - Cache Memory Access time = 5ns.
   - Cache Hit rate = 0.9
   - 64Kbyte DRAM used as the main memory.
   - DRAM Memory access time = 200ns

   a. In doing cache mapping analysis, how many **blocks** would the main memory be partitioned to?

   [Suggested Solution]

   Cache block size = 512/32 = 16 Bytes

   Number of main memory blocks = 64KByte/16Byte = (2^16)/(2^4) = 2^12 = 4096

   b. What is the format of a memory address as seen by the cache (i.e. determine the sizes of the tag, block and offset fields)?

   [Suggested Solution]

   Cache Block Size = 16 => Offset Field = 4 bits

   Number of blocks in the cache = 32 => Block Field = 5 bits

   Main Memory Size = 64KByte => 16 address bits

   Tag Field bits = Main Memory Address bits – Offset – Block = 7

   ⇨ TAG:BLOCK:OFFSET = 7:5:4

   c. CPU needs to read a byte from main memory address 0x0DB63.

      i. Which cache block would CPU looked at to search for the required data?
      ii. How many main memory blocks could potentially be mapped to the same cache block as that of 0x0DB63?
      iii. How does the CPU knows if the cache block identified in (i) above contains the data that it needs?
      iv. What is the purpose of the 'offset' field in the cache mapping?

[Suggested Solution]

i. 0x0DB63 = **1101 1011 0110 0011**

⇨ Block 10110b = Block 22

ii. Number of MM blocks = 4096

Number of cache blocks = 32

=> each cache block is a potential destination to 4096/32 = 128 MM blocks.

iii. By looking at the TAG value entry of the corresponding cache block

For 0xDB63, the TAG value for the cache block should be equal to

1101 101b

iv. Offset refers to the offset of the byte of interest from the cache block boundary. For 0xDB63, the byte is the byte 3 of block 22. First byte of the block is byte 0.

d. What is the effective access time of the memory in this system?

[Suggested Solution]

Assume cache memory and main memory access do not overlap.

EAT = $H*Cache_{Access} + (1-H)*(Cache_{Access} + Mem_{Access}) = 0.9*5ns + (1-0.9)*(5ns+200ns) = 25ns$.

8.2 Virtual Memory

2. In a processor system with the following characteristics,
   - 1 MByte Virtual memory space
   - 64 Kbyte DRAM as main memory
   - Paging scheme used for virtual memory management, Page Table as shown in Table 8.2
   - Virtual Page size = 1 KByte
   - TLB with 4 entries

Table 8.2 – Page Table

Virtual Page Number	Valid Bit	Page Frame Number
0	1	1
1	1	2
2	0	-
3	1	16
4	1	9

a. How many bits are required for each virtual address?

[Suggested Solution]

1Mbyte = 2^20 byte => 20 bits

b. How many bits are required for each physical address?

[Suggested Solution]

64 KByte = 2^6 * 2^10 = 2^16 => 16 bits

c. What is the maximum number of entries in the page table in Table 8.2?

[Suggested Solution]

Number of virtual pages = 2^20 / 2^10 = 2^10 = 1024

⇨ Max number of entries = 1024

d. What is the maximum number of valid entries in the page table in Table 8.2?

[Suggested Solution]

Number of Physical Frames = 2^16 / 2^10 = 64

⇨ Max number of valid entries= 64

e. With reference to Table 8.2, answer the following. Indicate when a page fault occurs.

(i) The compiler mapped the UART routine to virtual address 0x005F0 – 0x006FF, where in the DRAM would you be able to find the UART routine?

[Suggested Solution]

0x005F0 ➔ virtual page number 1 ➔ mapped to page frame number 2
0x006FF ➔ virtual page number 1 ➔ mapped to page frame number 2

Virtual Address (20 bits)	Physical Address (16 bits)
0000 0000 0101 1111 0000	0000 1001 1111 0000
0000 0000 0110 1111 FFFF	0000 1010 1111 FFFF

Physical Address = 0x09F0 – 0x0AFF

(ii) The compiler mapped the I2C routine to virtual address 0x009C0 - 0x009DF, where in the DRAM would you be able to find the I2C routine?

[Suggested solution]

0x009C0 ➔ virtual page number 2 ➔ page fault occurs (valid bit =0).

0x009DF ➔ virtual page number 2 ➔ page fault occurs (valid bit =0).

(iii) What happens when there is a page fault?

[Suggested Solution]

Page fault => the required data/code is not in the main memory

=> OS needs to retrieve the required information from the storage memory and update the paging table accordingly.

f. What memory are the Page Table and TLB resided?

[Suggested Solution]

Main Page Table => Main Memory

TLB => Internal Fast Memory close to CPU

g. What is the function and effect of a TLB?

[Suggested Solution]

- ⇨ Cache the frequently used Page table information
- ⇨ Leads to shorter access time for paging information => increase in system performance.

(Not necessary to be covered during tutorial)

[Optional, but students are encouraged to attempt these questions]

3. Consider a system with the following characteristics.
   - Direct mapped cache of 32 cache blocks and cache block size of 32 bytes
   - Cache uses <u>Physical Address</u> for address mapping
   - Virtual Memory page size 2048 bytes
   - Virtual Memory size is 1Mbyte. Physical Memory size is 64KByte
   - Extracts of Page Table (valid entries)
     - Virtual Page 0 → Physical Frame 9
     - Virtual Page 1 → Physical Frame 3
     - Virtual Page 2 → Physical Frame 5
     - Virtual Page 3 → Physical Frame 2
     - Virtual Page 4 → Physical Frame 7
   - The main program is 5KByte in size and starts at virtual address 0x01006

   (i) Assuming that the compiler allocates the program sequentially in the virtual memory, what is the physical address of the start and end of the main program?

[Suggested Solution]

Virtual Address (Start) = 0x01006, end = 0x02405. (5KByte size)
0x1006 => 0001 0000 0000 0110 => Page 2 => Physical Frame 5 (101b)
Physical address = 0010 1000 0000 0110 = **<u>0x2806</u>**
0x2405 => 0010 0100 0000 0101 => Page 4 => Physical Frame 7 (111b)
Physical Address = 0011 1100 0000 0101 => **<u>0x3C05</u>**

   (ii) Which cache block should the CPU check in the cache for the start of the main program? What is the corresponding TAG value used to check for cache hit/miss?

[Suggested Solution]

Physical Address used for tagging. Hence, TAG:BLK:OFFSET = 6:5:5
Start address = 0x2806 = 0010 1000 0000 0110
TAG value = 001010 = **<u>0xA</u>**, BLK = 000000 = **<u>0</u>**