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? 512 ÷32 = 16 Bytes
   $64 \times 2^{10} \div 16 = 4096$ blocks

   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)? $\log_2(4096) = 12$ $\log_2(32) = 5$ $\log_2(16) =$
   7 : 5 : 4

   c. CPU needs to read a byte from main memory address 0xDB63. 1101 1011 0110 0011

      i. Which cache block would CPU looked at to search for the required data? 0x16
      ii. How many main memory blocks could potentially be mapped to the same cache block as that of 0xDB63? $(64 \times 2^{10}) \div 512 = 128$ / or, $2^7 = 128$
      iii. How does the CPU knows if the cache block identified in (i) above contains the data that it needs? compare tag
      iv. What is the purpose of the 'offset' field in the cache mapping? Know which byte

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

$5 \times 0.9 + (200+5) \times 0.1 = 25$ ns

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? $2^{20} \times 1 \rightsquigarrow 20$ bits

b. How many bits are required for each physical address? $64 \times 2^{10} = 65536$
$\log_2 65536 = 16$ bits

c. What is the maximum number of entries in the page table in Table 7.2?
$2^{10} \div 2^{10} = 1024$,

Offset = 10 bits

d. What is the maximum number of valid entries in the page table in Table 7.2?
$64 \times 2^{10} \div 2^{10} = 64$,

e. With reference to Table 7.2, answer the following. Indicate when a page fault occurs.

5F0
0000 0000 01/01 1111 0000
6FF
0000 0000 01/10 1111 1111

(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?
→ Found! → Convert to 1001 1111 0000 → 0x9F0 ; 1010 1111 0000 → 0xAF0

(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?

9C0
10/01 1100 0000
9DF
10/01 1101 1111
→ Page fault

(iii)What happens when there is a page fault?
Go to storage, fetch, update page table

f. What memory are the Page Table and TLB resided?
↳ main mem → internal fast mem

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

(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 Physical Address 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?

   (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?

32 blocks 512 pages

Cache: 5/5 VM: 9/11 MM: 3/11

address: 0x01006 → 0001 0000 0000 0110 Start

→ Hit to physical 5 → 0010 1000 0000 0110

↳ 0x 2806

End: 0x01006 + 5 KB = 0x01006 + 0x1400 = 0x 2406

0010 0100 0000 0110

→ Hit to 7 →

0011 1100 0000 0110

↳ 0x 3C06

Answer say 0x3C05 ?

Physical Address: 0x 2806: 0010 1000 0000 0110

Cache: 11 bit → 6/5/5 tag: 0xA

Tag / block / offset block: 0 offset: 6

(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 Physical Address 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?

    (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?