**Wk5D**

**Andrew Rohn**

**Cache and Memory Mapping**

Part I

Consider a computer that you have used or now use. (You may make an assumption on word size if needed.)

* How large is the memory of the computer?

16 GB of main memory

* How long is a word in the computer above?

64 bits

* How many bits are required to address the bytes in that memory?

16 \* Gigabytes = 24 \* 230 = 234 = 34-bit addresses

* How many data lines are required to read data from the computer memory?

64 data lines

Part II

* Does your computer have cache?

Yes

* If so how big is it?

L1 Data: 4 x 32 Kbytes

L1 Inst.: 4 x 32 Kbytes

Level 2: 4 x 256 Kbytes

Level 3: 6 Mbytes

* How does your cache size compare with the size of main memory?

The cache size on my processor is only 6 MB and the size of the main memory is 16GB. So, the size of the main memory is 2,667 times larger than size of the cache.

* Do you know what kind of cache it is?

My processor has 4 kinds of cache:

Level 1 Data Cache

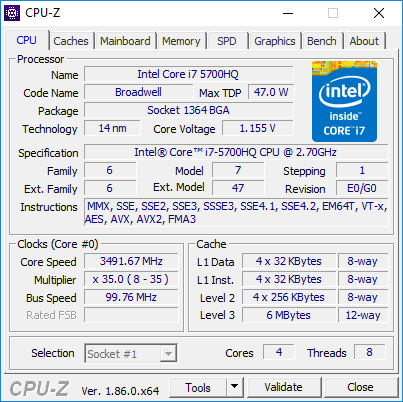
Level 1 Instructions Cache

Level 2 Cache

Level 3 Cache

* How did you find out about your computer's cache?

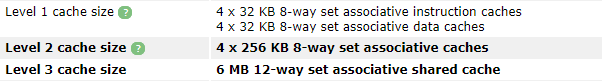
I downloaded free program called CPU-Z that gives you detailed information about your processor. Here is a screenshot of the program:



 Make an assumption that your cache is either:

* Fully associative
* Direct mapped
* Two-way set-associative
* Four-way set-associative

My processor uses 8-way and 12-way associative caches according to CPU-World’s specification listing for my processor. Here is a screenshot of the cache specification:



 Using the relationships in Part I above, determine:

 the size of the Tag and Word for Associative cache;

 OR

 the size of the Tag, Line, and Word for Direct-Mapped Cache ;

 Or

 the size of Tag, Set, and Word for K-Way Set-Associative Cache.

You may make any assumptions necessary including the number of Words in each block (recommend 2 or 4 or 8)

For 8-Way Associative Cache:

For 12-Way Associative Cache:

Part III

Given the following:

* Logical Memory size of 1000
* Physical Memory size of 2000
* Page (and frame) size of 100
* Block A contains data for a program

Select Block A’s size and its starting point in both memories. Then write the page table for Block A based on your selections.

See below for the layout of both memories and an example of Block A of size 200.

Logical Memory                               Physical Memory

location/ **page**                                    location/**frame**

|  |  |  |
| --- | --- | --- |
| 0     to     99/ **0** Block A |  | 0     to     99/ **0** |
| 100 to 199 /**1**     Block A | 100 to 199/ **1** |
| 200 to 299/ **2**     Block A | 200 to 299/ 2 |
| 300 to 399/ **3**  Block A | 300 to 399/ 3 |
| 400 to 499/ **4** Block A | 400 to 499/ 4 |
| 500 to 599/ **5** | 500 to 599/ 5 |
| 600 to 699/ **6** | 600 to 699/ 6 |
| 700 to 799/ **7** | 700 to 799/ 7 |
| 800 to 899/ **8** | 800 to 899/ 8 |
| 900 to 999/ **9** | 900 to 999/ 9 |
|  | 1000 to 1099/ 10 |
| 1100 to 1199/ 11 |
| 1200 to 1299/ **12** Block A |
| 1300 to 1399/ **13**         Block A |
| 1400 to 1499/ **14**        Block A |
| 1500 to 1599/ **15** Block A |
| 1600 to 1699/ **16** Block A |
| 1700 to 1799/ **17** |
| 1800 to 1899/ **18** |
| 1900 to 1999/ **19** |

|  |  |
| --- | --- |
| Page | Frame |
| **0** | **12** |
| **1** | **13** |
| **2** | **14** |
| **3** | **15** |
| **4** | **16** |

Part IV

Discuss the pros and cons of paging.

Paging is where the computer stores and retrieves data in same-size blocks (pages) from the secondary storage for use in main memory.

**Pros:**

* can run programs that are larger than the physical memory
* fixed partition size
* no compaction needed
* simple partition management because of non-contiguous loading
* only portions that are called are loaded

**Cons:**

* need compatible hardware to translate addresses which lengthens memory cycle times
* not all of the page size is used, the excess memory is wasted
* the Job Size must be less than or equal to the memory size
* each time a page is replaced, it needs to be reloaded, increasing the program turnaround time