Part I

How large is the memory of the computer?

16 GiB = 2^34

How long is a word in the computer above?

64 bits

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

34 bits

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

64

Part II

Does your computer have cache?

Yes

If so how big is it?

L2: 1 MiB

L3: 8 MiB

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

(2^34) / ((2^23) + (2^20)) is roughly 1820

Main memory is roughly 1820 times the size of the L2 and L3 cache combined.

Do you know what kind of cache it is?

L2 and L3

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

`wmic cpu get L2CacheSize, L3CacheSize` in windows 10 command prompt

Running command for L1CacheSize gave an error for an invalid query.

Make an assumption that your cache is:

Two-way set-associative

Determine the size of Tag, Set, and Word for 2-Way Set-Associative Cache.

Assuming 4 words per block

Address = 34 bits

(2^34) / (2^2) = 2^32 blocks

L2 Cache = 2^20 bits

(2^20) / (2^2) = 2^18 blocks / 2 = 2^17 sets in L2 Cache

L3 Cache = 2^23 bits

(2^23) / (2^2) = 2^21 blocks / 2 = 2^20 sets in L3 Cache

2^21 > (2^20 + 2^17)

So 21 bits for set

SET = 21 bits

Since there are 4 (2^2) words per block, 2 bits for Word

WORD = 2 bits

2 Blocks per set (2^1 = 2), 1 bit for set

Tag = 34 - 21 - 2 = 11

TAG = 11 | SET = 21 | WORD = 2

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.

Block A size = 400

Logical Memory                               Physical Memory

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

|  |  |  |
| --- | --- | --- |
| 0     to   99/ **0** |  | 0     to     99/ **0** |
| 100 to 199 /**1** | 100 to 199/ **1** |
| 200 to 299/ **2** | 200 to 299/ 2 |
| 300 to 399/ **3** | 300 to 399/ 3 |
| 400 to 499/ **4** | 400 to 499/ 4 |
| 500 to 599/ **5 Block A** | 500 to 599/ 5 |
| 600 to 699/ **6 Block A** | 600 to 699/ 6 |
| 700 to 799/ **7 Block A** | 700 to 799/ 7 |
| 800 to 899/ **8 Block A** | 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** |
| 1700 to 1799/ **17** |
| 1800 to 1899/ **18** |
| 1900 to 1999/ **19** |

|  |  |
| --- | --- |
| Page | Frame |
| **5** | **12** |
| **6** | **13** |
| **7** | **14** |
| **8** | **15** |

Part IV

Pros and cons of paging

The reason we use paging is that it allows for virtual memory. This means that the system is not limited by its physical memory. This is a huge benefit allowing users to run programs without getting errors because of running out of memory. The downside to paging is that when virtual memory has to be accessed, it can be slower, especially if there is a page fault. Additionally, since paging uses fixed size blocks, not all the space inside of the blocks may be used, which is known as fragmentation. In short, we get to extend our physical memory limits but the big downside is potential fragmentation.