## Homework #1

## (Due date: 12 November 2021)

1. Matrix multiplication is a popular operation in scientific computing. In this question, you need to analyze the cache miss situation during the matrix multiplication. Now, we have two matrices $A^{4\times4}$ and $B^{4\times4}$ and each of the two matrices contains 16-words data. Besides, we access each element of these matrices from the main memory by using row-major order policy (i.e., the row index changes fastest.) At last, we assume each block in the L1 cache of our machine includes 4 words.

   a. Please write a simple C code to perform this matrix multiplication.
   b. If the L1 cache only has two entries with 1-way associative, please analyze how many cache-misses we have before counting the result.
   c. If we hold the 1-way associative design strategy but increase the entries, please analyze the cache-miss again. Is the total cache-misses saturated eventually? Note that the total cache size is not limited.
   d. If we still use two entries but increase the way associative, please analyze the cache-miss again. Is the total cache-misses saturated eventually? Note that the total cache size is not limited.
   e. Now, let's go back to the originally simple cache structure (i.e., 1-way associative and 2 entries.) By following Problem 1.a, could we use the loop interchange method to reduce the total cache misses? Please explain your reasons.
   f. By following Problem 1.e, we don't change the order of the loops but divide each matrix into four 2×2 blocks. Could the total cache-missed be reduced? Please explain your reasons.
   g. Please write a simple C code to perform this block matrix in Problem 1.f.

2. TLB is usually much smaller than the page table. Hence, fully associative is a good choice to design the TLB. If we enlarge the TLB size, is it still a good choice to use fully associative? Please explain your reasons.

3. There are two kinds of design methods to design the cache of the FastMATH processor, which are shown in Fig. 1 and Fig.2. Please analyze the pros and cons of the two structures.

Note: We encourage the discussion but prohibit plagiarism seriously. Both two students will get zero grades if we find a cheating situation.
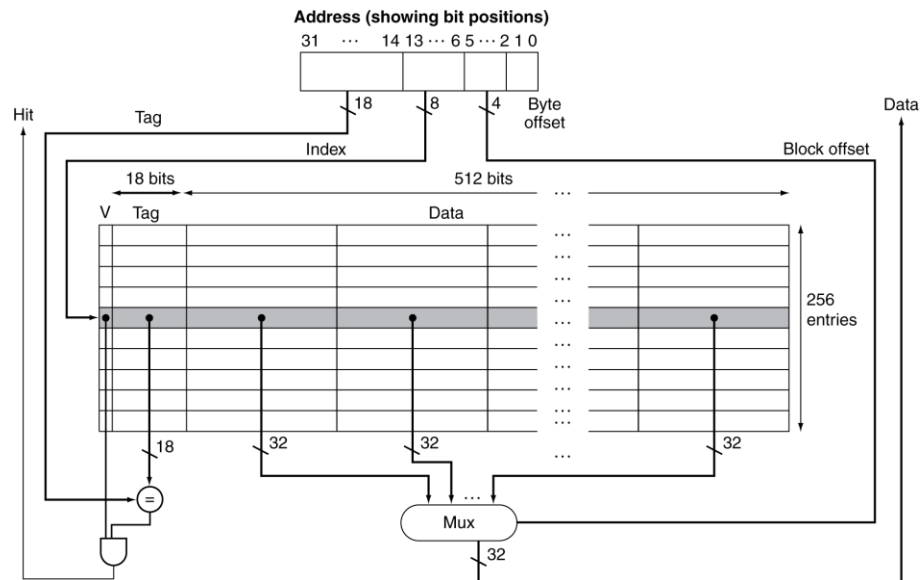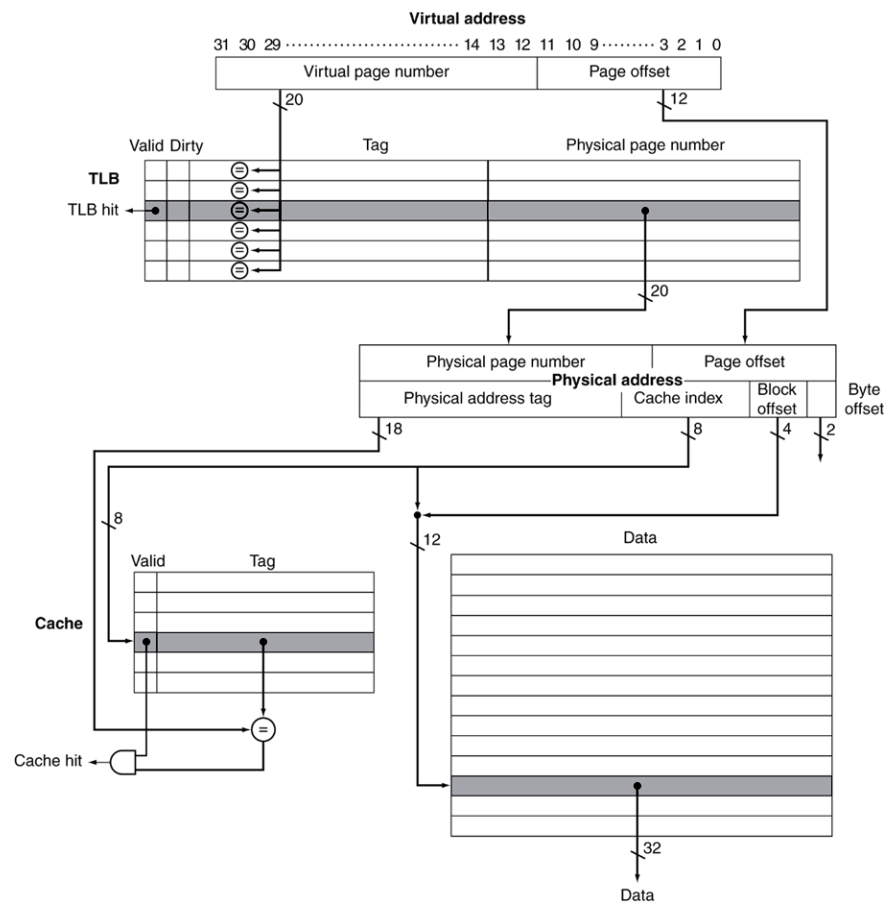
**Address (showing bit positions)**

31 ⋯ 14 13 ⋯ 6 5 ⋯ 2 1 0

Hit    Tag                18    8    4    Byte offset    Data

Index    Block offset

18 bits ⟶    512 bits

V    Tag    Data

256 entries

18    32    32    32

=

Mux

32

Fig. 1

**Virtual address**

31 30 29 ···························· 14 13 12 11 10 9 ········· 3 2 1 0

Virtual page number    Page offset

20    12

Valid Dirty    Tag    Physical page number

**TLB**

TLB hit

=
=
=
=
=
=

20

Physical page number    Page offset

**Physical address**

Physical address tag    Cache index    Block offset    Byte offset

18    8    4    2

8

12    Data

Valid    Tag

**Cache**

=

Cache hit

32

Data

Fig.2