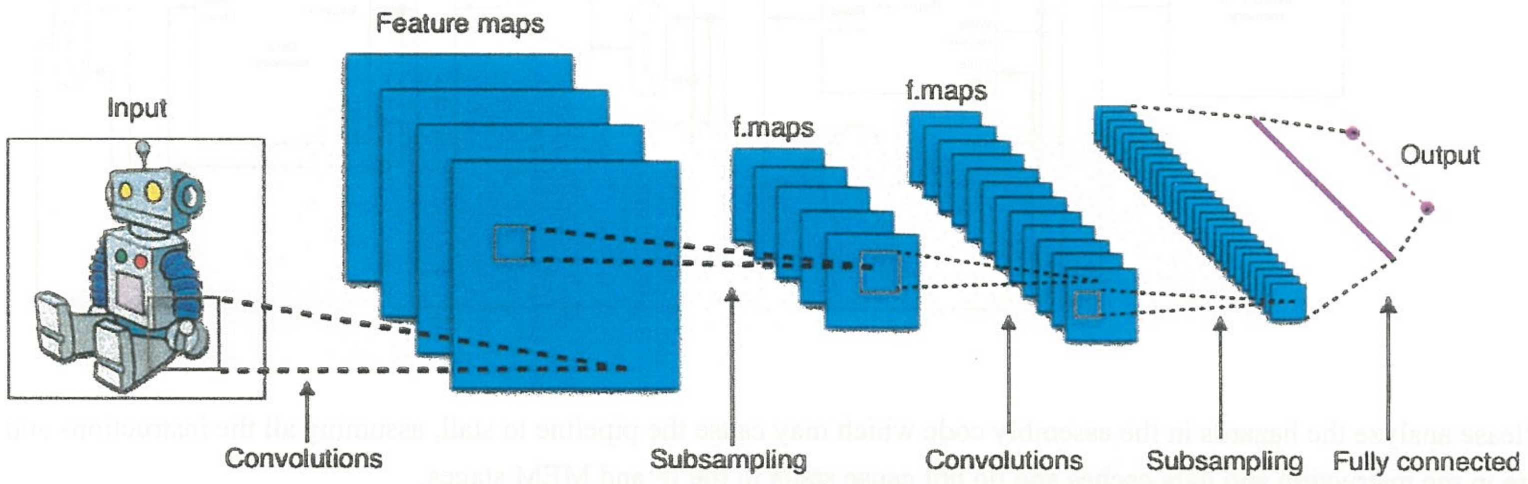
Part I: Computer Architecture

1. (50 points) The convolutional layer is the core building block of a *Convolutional Neural Network* (CNN), which is widely used in today's machine learning methods to classify images and recognize speeches. For example, the figure below shows a CNN which contains two convolutional layers. The first convolutional layer applies 4 learnable filters (or kernels) to the input image and produces 4 feature maps.



A typical CNN architecture (from Wikipedia)

Although the above example uses 2-dimensional convolutions, let us start with 1-dimensional convolution here for simplicity. The convolution of two 1-dimensional finite sequences is defined by as the following:

$$(f * g)[n] = \sum_{m=-M}^{M} f[n-m]g[m]$$

where the symbol * denotes a convolution operator, *f* is the input sequence, and *g* is the convolution kernel of size (*2M+1*).

Suppose the input sequence *f* in our case study contains *N* data points, and each data point is represented by a 32-bit floating-point number. The convolution kernel *g* is also composed by 32-bit floating-point numbers.

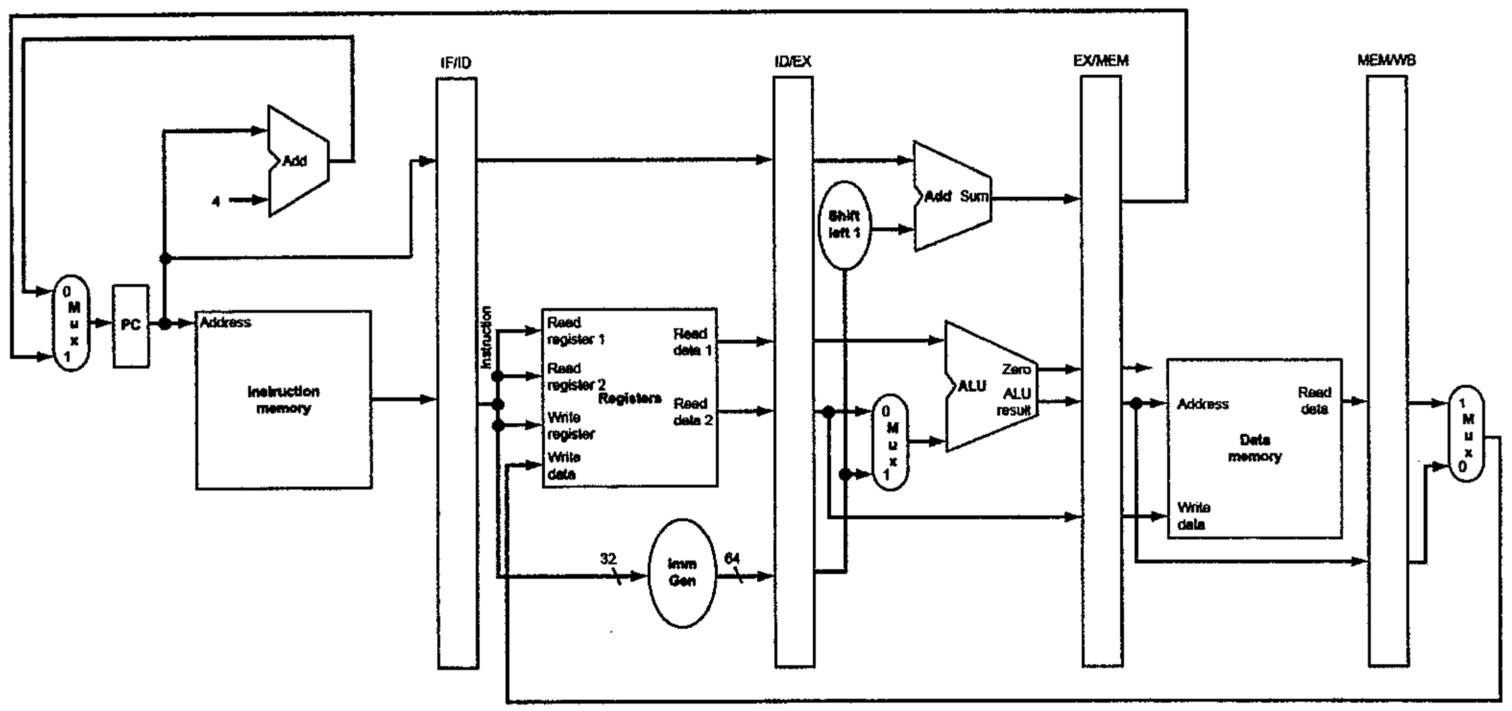
Please answer the following questions:

(a) [5 points] Please write a C code to implement the convolution function in our case study. Please make sure that your code is comprehensive with sufficient comments.

(b) [5 points] Please convert your C code to an assembly code. No optimization is needed here. You can use any instruction set architecture, but please make sure that your code is comprehensive with sufficient comments.

(c) [5 points] Suppose that we have a simple 5-stage pipelined processor here. The pipeline stages are IF (instruction fetch), ID (instruction decode/register), EX (execution), MEM (memory), and WB (write-back), as shown in the figure below. The pipeline is in-order issue and in-order execution with ideal one cycle-per-instruction (CPI). It executes a branch instruction in the EX stage. It can detect hazards, but does not support forwarding between pipeline stages.

見背面



Please analyze the hazards in the assembly code which may cause the pipeline to stall, assuming all the instructions and data are in the instruction and data caches and do not cause stalls in the IF and MEM stages.

(d) [5 points] To improve the pipeline performance in the previous question, we would like to reduce the impact of control hazards by adding *branch prediction* to the processor pipeline. Please describe how this can be done and how it affects the performance.

(e) [5 points] To further improve the pipeline performance in the previous question, we would like to reduce the impact of data hazards by *unrolling the loop*. Please describe how this can be done and how it affects the performance.

(f) [5 points] Now, let us take the data memory into consideration. Assume each memory access takes 10 cycles, and there is a *processor data cache* to speed up the memory access for reused data. If an access hits the data cache, then the processor would not stall. Suppose the data cache is a direct-mapped cache which has 16 blocks and each block has 16 bytes. Estimate the data cache miss rates for $M=1$, 4, and 8.

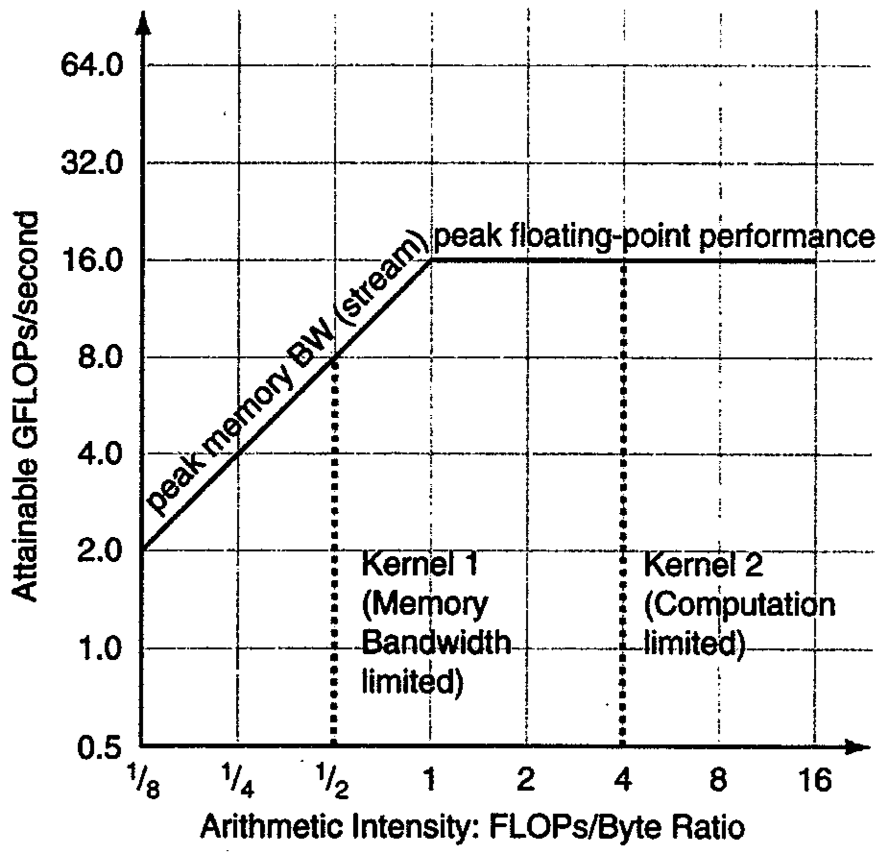
(g) [5 points] Can "*blocking*" be used to reduce the cache misses in the previous question? If yes, please rewrite the code with blocking and estimate the performance benefit of blocking.

(h) [5 points] Suppose we want to apply the same convolution kernel to many independent input sequences, can we take advantage of *multithreading* to increase the performance? Please describe how such multithreading can be done, or why it cannot be done.

(i) [5 points] Suppose we want to apply different convolution kernels to one input sequence, can we take advantage of a *multiprocessor* to increase the performance? Please describe how such multiprocessing can be done, or why it cannot be done.

(j) [5 points] Let us estimate the processor performance with a *roofline model* shown in the figure blow. If you know the arithmetic intensity of a computing kernel, then you know the attainable performance would not be higher than the roofline. For example, Kernel 1 in the figure can attain no more than 8 GFLOPS (Floating-Point Operations Per Second), and Kernel 2 can attend up to 16 GFLOPS.

接次頁



Please calculate the arithmetic intensity of the convolution kernel and estimate the attainable performance in case there is no data cache. Then, discuss what would happen to the attainable performance of our convolution kernel when a data cache is added to the processor. Furthermore, discuss what would happen to the roofline and the attainable performance of our convolution kernel if a vector unit is added to the processor to provide 4 times of attainable performance.

Part II: Operating System

NOTE that in the question, it is intended to provide redundant or miss certain assumption to disguise you. Please make your own assumption if necessary to answer the questions.

2. (20 points) For each of the following statements, answer **Yes** if it is TRUE or **a brief description on why it is wrong.**
    (a) [2 points] The more threading, the more performance.
    (b) [2 points] The more frames, the less page fault rates.
    (c) [2 points] The bigger time quantum, the less average turnaround time.
    (d) [2 points] Multiprocessor system increases throughput.
    (e) [2 points] Firmware can be executed faster in RAM.
    (f) [2 points] Modularity is one of the reasons to support process cooperation.
    (g) [2 points] A safe state is not a deadlocked state.
    (h) [2 points] UNIX Semantics: Writes to an open file by a user are visible immediately to other users who have this file open.
    (i) [2 points] A real-time scheduler schedules tasks according to their priority in a fastest manner.
    (j) [2 points] The two-phase locking protocol ensures conflict serializability and prevents deadlock.

3. (5 points) What is the different between sector sparing and sector slipping? How is disk scheduling affected?

4. (10 points) For the working-set model used to prevent thrashing while keeping the degree of multiprogramming as high as possible, what other application can be also effective using the model? And how?

5. (15 points) Please write (pseudo) codes with comments to implement a collector for network packets transmitted any time from many other nodes with timestamp at transmission, so that the earliest transmitted packet received (not in the same order of transmission) in every 10 seconds is printed out on monitor on the goal to get the most credits. Note that the earlier printed timestamp gets the more credits. Hint: Is network delay bounded? Asynchronous/synchronous I/O both might be needed.