Homework 9

1. How would you build a $2^{12} \times 8$ memory using the available 1K × 1 memory chips? Each such memory chip has one line for data input, one line for data output, a chip selection (CS) line and a set of 10 address lines. Each chip can be represented by a box with these input/output lines. You need to show how to form the $2^{12} \times 8$ memory by connecting as many as necessary such chips with minimum additional logic (multiplexers, decoders, etc.).

2. The CPU is executing a program which needs to access the data in the following blocks in the order shown:

   A C B E C D A B E A C

   Assume the cache memory can hold four blocks and that it is initially empty. Give the hit ratio during the execution of this program using the (a) optimal, (b) FIFO and (c) LRU replacement policies. Assume the cache memory is initially empty.

3. A computer system has a main memory consisting of 1M (2^{20}) 16-bit words and a 4K-word (2^{12}) cache organized in the set-associative manner, with four blocks per set and 64 words per block. Load-through strategy is not used.

   - Calculate the number of bits in each of the TAG, SET, and WORD fields of the main memory address format.
   - Assume that the cache is initially empty. Suppose that the CPU fetches 4352 words from locations 0, 1, 2, ..., 4351 (in that order). It then repeats this fetch sequence nine more times. If the cache is 10 times faster than the mian memory, estimate the improvement factor (defined as the ratio of time-without-cache to time-with-cache) resulting from the use of the cache. Assume that the LRU algorithm (replacing the least recently used block first) is used for block replacement.
   - Repeat the above with the assumption that the MRU algorithm (replacing the most recently used block first) is used for block replacement.

4. The three types of memory, cache memory, main memory and secordary memory, form a three-level memory hierarchy (M_1, M_2, M_3) that can be characterized by the following relationships:

   Content: $M_1 \subset M_2 \subset M_3$

   Capacity: $S_1 < S_2 < S_3$

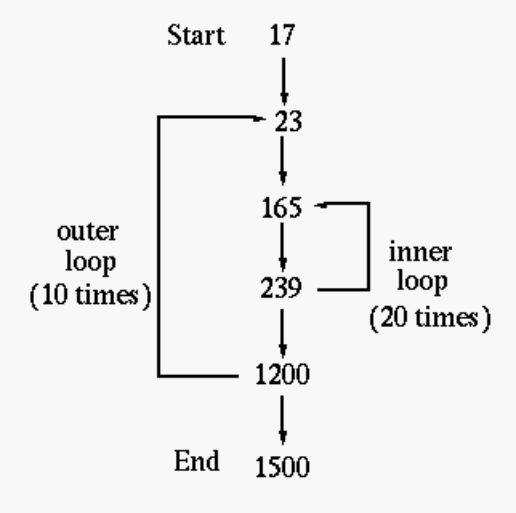
   Access time: $T_1 < T_2 < T_3$

   Hit ratio: $H_1 < H_2 < H_3$

   - Give the expression for H_i (i = 1, 2, 3) in terms of the N_i's, which are defined as the number of actual references to M_i (i = 1, 2, 3). Hint: refer to the expression for H in previous case for i = 1, 2.
   - P_i is defined as the probability that M_i has to be accessed to obtain the information requested by the CPU. Give the expression for P_i in terms of H_i's. Verify that $P_1 + P_2 + P_3 = 1$ (representing the fact that one of the M_i's must be accessed).
   - Let C_i denote the cost-per-bit of M_i. Give an expression for C_{average}, the average cost-per-bit of the memory hierarchy. Compute C_{average} using the data given in the table below.
   - Give the expression of the average access time T_{average}, defined as the average time for the CPU to access a word in the memory hierarchy (M_1, M_2, M_3). Consider both the cases with and without using the "load-through" technique. Assume that when "load-through" is used, the information from M_i is loaded to M_{i-1} and M_{i-2} (or the CPU) simultaneously. Determine T_{average} for the data given in the table below.

Level i	Capacity S_i	Cost C_i ($/bit)	Access Time t_i	Hit Ratio H_i
M_1 (cache)	2^{10}	0.1000	10^{-8}	0.9000
M_2 (main)	2^{16}	0.0100	10^{-6}	0.9999
M_3 (secondary)	2^{24}	0.0001	10^{-4}	1.0000

5. Suppose the program shown below is to be executed on a computer with a virtual memory system.



Assume the following:

- Each page in the system has 512 bytes, i.e., page 1 contains byte 0 through byte 511, page 2 contains byte 512 through byte 1023, etc.
- All addresses in this program are virtual byte addresses, and each instruction is in one byte.
- A translation buffer and the pre-translation technique are used (see definitions below).

Estimate the following:

- The total number of virtual-memory accesses requested by the CPU during execution of the program
- The percentage of these accesses handled by pre-translation, assuming the translation buffer is initially empty
- The numbers of translation buffer misses and hits

Pretranslation: The physical page number of the page containing the current instruction stream is maintained in the CPU. If the next instruction is still in the same page, its physical address can be obtained by concatenating this physical page number with the low-order bits of the program counter. A new page table translation is needed only if the next instruction is in a different page.

Translation Lookaside Buffer (TLB): Virtual memory requires extra main memory accesses to translate virtual addresses to physical addresses. To speed up the memory access time, a special cache memory, called a TLB, is used to store the translation table entries corresponding to pages currently in main memory. When a new page is brought in from secondary memory to main memory, its translation table entry is also brought in to the TLB (to replace an existing table entry if necessary).