7.7

Reference	Hit or miss
1	Miss
4	Miss
8	Miss
5	Miss
20	Miss
17	Miss
19	Miss
56	Miss
9	Miss
11	Miss
4	Miss
43	Miss
5	Hit
6	Miss
9	Hit
17	Hit

Here is the final state of the cache:

Block #	Address
0	
1	17
2	
3	19
4	4
5	5
6	6
7	
8	56
9	9
10	
11	43
12	
13	
14	
15	

7.9 There are 4K entries and each entry needs 128 (data) + 16 (tag) + 1 (valid) = 145 bits. This results in 593,920 bits, or 74,240 bytes. A bit more than the 64-KB cache size as described in the figure caption!

7.11 The miss penalty is the time to transfer one block from main memory to the cache. Assume that it takes one clock cycle to send the address to the main memory.

a. Configuration (a) requires 16 main memory accesses to retrieve a cache block and words of the block are transferred 1 at a time.

   Miss penalty = $1 + 16 \times 10 + 16 \times 1 = 177$ clock cycles.

b. Configuration (b) requires 4 main memory accesses to retrieve a cache block and words of the block are transferred 4 at a time.

   Miss penalty = $1 + 4 \times 10 + 4 \times 1 = 45$ clock cycles.

c. Configuration (c) requires 4 main memory accesses to retrieve a cache block and words of the block are transferred 1 at a time.

   Miss penalty = $1 + 4 \times 10 + 16 \times 1 = 57$ clock cycles.

7.20

Reference	Hit or miss
1	Miss
4	Miss
8	Miss
5	Miss
20	Miss
17	Miss
19	Miss
56	Miss
9	Miss
11	Miss
4	Hit
43	Miss
5	Hit
6	Miss
9	Hit
17	Hit

Here is the final state of the cache with LRU order shown right to left:

Block #	Element #1 address	Element #1 address
0	56	8
1	17	9
2		
3	43	11
4	4	20
5	5	
6	6	
7		

7.22

Reference	Hit or miss
1	Miss
4	Miss
8	Miss
5	Hit
20	Miss
17	Miss
19	Hit
56	Miss
9	Miss
11	Hit
4	Miss
43	Miss
5	Hit
6	Hit
9	Hit
17	Miss

Here is the final state of the cache LRU order (left to right):

Address	Address	Address	Address
40	4	8	16

7.29 If direct-mapped and `stride` = 132, then we can assume without loss of generality that `array[0]` is in slot 0, along with `array[1]`, `[2]`, `[3]`. Then, slot 1 has `[4]`, `[5]`, `[6]`, `[7]`, slot 2 has `[8]`, `[9]`, `[10]`, `[11]`, and so on until slot 15 has `[60]`, `[61]`, `[62]`, `[63]` (there are 16 slots × 16 bytes = 256 bytes in the cache). Then wrapping around, we find also that slot 0 has `[64]`, `[65]`, `[66]`, `[67]` and also `[128]`, `[129]`, `[130]`, `[131]`, etc. Thus if we look at `array[0]`, `array[132]`, we are looking at two different slots. After filling these entries, there will be no misses. Thus the expected miss rate is about 0. If the stride equals 131, we are looking at slot 0 and slot 0 again. Each reference will bounce out the other, and we will have 100% misses. If the cache is two-way set-associative, even if two accesses are in the same cache line they can coexist, so the miss rate will be 0. Alternative explanation: 132 in binary is 10000100, and clearly when this is added to the word address of `array[0]`, bit 2 will change and thus the slots are different. 131 in binary is 10000011, and in this case the last two bits are used for the block offset within the cache. The addition will not change the index, so the two entries will map to the same location.

7.32 The total size is equal to the number of entries times the size of each entry. The number of entries is equal to the number of pages in the virtual address, which is

$$\frac{2^{40}\ \text{bytes}}{16\ \text{KB}} = \frac{2^{40}\ \text{bytes}}{2^4\ 2^{10}\ \text{bytes}} = 2^{26}$$

The width of each entry is 4 + 3 bits = 40 bits = 8 bytes. Thus the page table contains 2^{29} bytes or 512 MB!