Problem 1

Describe the general characteristics of a program that would exhibit very little temporal and spatial locality with regard to instruction fetches. Provide an example program (pseudocode is fine).

Describe the general characteristics of a program that would exhibit very high amounts of temporal locality but very little spatial locality with regard to instruction fetches. Provide an example program (pseudocode is fine).

Describe the general characteristics of a program that would exhibit very little temporal locality but very high amounts of spatial locality with regard to instruction fetches. Provide an example program (pseudocode is fine).
Solution 1

Low temporal locality for code means no loops and no reuse of instructions.

High temporal locality for code means tight loops with lots of reuse.

Low spatial locality for code means lots of jumps to far away places.

High spatial locality for code means no branches/jumps at all.
Problem 2

Consider a 64K direct-mapped cache with a 16 byte blocksize. Show how a 32-bit address is partitioned to access the cache.
Solution 2

Consider a 64K direct-mapped cache with a 16 byte blocksize. Show how a 32-bit address is partitioned to access the cache. What is the total number of bits in the cache.

There are 64K/16 = 4K = 4096 = 2^12 lines in the cache.
Lower 2 bits ignored.
Next higher 2 bits = position of word within block.
Next higher 12 bits = index.
Remaining 16 bits = tag.

Total number of bits = 2^12 * (16*8 + 1 + 16)
Problem 3

Consider a 3-way set-associative write-through cache with an 8 byte blocksize, 128 sets, and random replacement. Assume a 32-bit address. How big is the cache (in bytes)? How many bits total are there in the cache (i.e. for data, tags, etc.).
Solution 3

Consider a 3-way set-associative write-through cache with an 8 byte blocksize, 128 sets, and random replacement. Assume a 32-bit address. How big is the cache (in bytes)? How many bits total are there in the cache (i.e. for data, tags, etc.).

The cache size is $8 \times 3 \times 128 = 3072$ bytes.
Each line has 8 bytes * 8 bits/byte = 64 bits of data.
A tag is 22 bits (2 address bits ignored, 1 for position of word, 7 for index).
Each line has a valid bit (no dirty bit 'cause it is write-through).
Each line therefore has $64 + 22 + 1 = 87$ bits.
Each set has $3 \times 87 = 261$ bits.
There are no extra bits for replacement.
The total number of bits is $128 \times 261 = 33408$. 

For More Practice

Problems 7.9, 7.10