計算機結構 小考一

姓名： 學號：

1. (20 pnts) Please explain Temporal Locality and Spatial Locality. And give an example for each.

Temporal Locality (Locality in Time): If an item is referenced, it will tend to be referenced again soon (e.g., loops, reuse)

Spatial Locality (Locality in Space): If an item is referenced, items whose addresses are close by tend to be referenced soon   
(e.g., straight-line code, array access)

1. (15 pnts) There are numerous techniques for improving the performance of caches. Please explain the main idea behind the following techniques, i.e., nonblocking caches, critical word first, and early restart.
2. *Non-blocking cache*
   * + 1. allows data cache to continue to supply cache hits during a miss
3. *Critical Word First*—Request the missed word first from memory and send it to the CPU as soon as it arrives; let the CPU continue execution while filling the rest of the words in the block
4. Early restart-Request words in normal order. Send missed work to the processor as soon as it arrives
5. (20 pnts) In the class, we describe the following compiler optimization techniques, i.e., Merging Arrays, Loop Interchange, Loop Fusion, and Blocking, which can improve the cache performance. Given the following code, please rewrite them to improve cache miss. And give which locality is improved in each case.

for (j = 0; j < 100; j = j+1)

for (i = 0; i < 5000; i = i+1)

x[i][j] = 2 \* x[i][j];

for (i = 0; i < 5000; i = i+1)

for (j = 0; j < 100; j = j+1)

x[i][j] = 2 \* x[i][j]; spatial

(b)

int val[SIZE];

int key[SIZE];

struct merge {

int val;

int key;

};

struct merge merged\_array[SIZE]; spatial

(c)

for (i = 0; i < N; i = i+1)

for (j = 0; j < N; j = j+1)

a[i][j] = 1/b[i][j] \* c[i][j];

for (i = 0; i < N; i = i+1)

for (j = 0; j < N; j = j+1)

d[i][j] = a[i][j] + c[i][j];

for (i = 0; i < N; i = i+1)

for (j = 0; j < N; j = j+1)

{ a[i][j] = 1/b[i][j] \* c[i][j];

d[i][j] = a[i][j] + c[i][j];}

Spatial temporal

(d) Please explain the rationale behind the blocking mechanism.

Fetch a portion of data and reuse it until it is replaced. And it does not be refetched again.

1. (20 pnts) There are numerous techniques for improving the performance of caches. Some reduce the frequency of misses, some reduce the miss penalty, some reduce hit times, and so on. Many optimizations involve trade-offs, making one performance factor worse in return for reducing another. In some cases, the so-called optimization can actually hurt performance. For the following optimizations, briefly summarize the trade-offs involved and how it helps or can hurt performance.
2. Higher associativity
3. Reduce conflict misses
4. Larger block size
5. Reduce compulsory misses
6. Bigger caches
7. Reduce capacity misses
8. Avoiding address translation in cache indexing

Reduce hit time