

**Exercise B**

**Part 2**

// sim2.c

// ENCM 369 Winter 2020 Lab 10 Exercise B

// Author: S. Norman

//

// If you build an executable using gcc -Wall sim1.c -o sim1

// you can run it by redirecting input to come from a data file,

// as in

//         ./sim1 < heapsort\_trace.txt

#include <stdio.h>

#include <stdlib.h>

int read\_one\_line(unsigned \*p)

// Read one line of the input stream.

// Return value is normally 'r' or 'w' to indicate read or write.

// In that case, \*p contains the address read from the input line.

// Return value is 'e' to indicate that input failed at the end of

// the input stream.

{

  int nscan, rw;

  char buf[2];

  nscan = scanf("%1s%x", buf, p);

  if (nscan == EOF)

    return 'e';                 /\* indicate end-of-file \*/

  else if (nscan != 2) {

    fprintf(stderr, "Format error in input stream.\n");

    exit(1);

  }

  rw = buf[0];

  if (rw != 'r' && rw != 'w') {

    fprintf(stderr, "Read/write character was neither r nor w.\n");

    exit(1);

  }

  return rw;

}

// These two arrays keep track of all the V-bits and stored tags in

// the array.  We don't need an array for data to count hits and misses.

// Because these arrays are external variables, it's safe to assume

// that they will be initialized to all zeros before main starts.

char v\_bit[128];

unsigned stored\_tag[128];

int main(void)

{

  int read\_count = 0, read\_hits = 0;

  int write\_count = 0, write\_hits = 0;

  int access\_count, miss\_count;

  int rw;

  unsigned address, search\_tag, set\_bits;

  int hit;

  while (1) {

    rw = read\_one\_line(&address);

    if (rw == 'e') break;

    set\_bits = (address & 0x3f8) >> 3;   // bits 9-3

    search\_tag = address >> 10;          // bits 31-10

    // Note: Next line results in either hit == 1 or hit == 0.

    hit = v\_bit[set\_bits] == 1 && stored\_tag[set\_bits] == search\_tag;

    if (rw == 'r') {

      read\_count++;

      read\_hits += hit;

    }

    else {

      write\_count++;

      write\_hits += hit;

    }

    if (!hit) {                 // On a miss, update V-bit and search\_tag.

      v\_bit[set\_bits] = 1;

      stored\_tag[set\_bits] = search\_tag;

    }

  }

  printf("%d reads\n", read\_count);

  printf("%d read hits\n", read\_hits);

  printf("%d writes\n", write\_count);

  printf("%d write hits\n", write\_hits);

  access\_count = read\_count + write\_count;

  miss\_count = access\_count - read\_hits - write\_hits;

  printf("overall miss rate: %.2f%%\n",

         100.0 \* (double) miss\_count / access\_count);

  return 0;

}

$ ./a <mergesort\_trace.txt

104298 reads

90653 read hits

73410 writes

63504 write hits

overall miss rate: 13.25%

mmmta@LAPTOP-35G9NI35 /cygdrive/e/encm369/lab10/exB

$ ./a <heapsort\_trace.txt

64705 reads

37830 read hits

60419 writes

60225 write hits

overall miss rate: 21.63%

Part 1 Answer:

The miss percentage of Merge-sort hardly changed from sim1 to sim2. Heap-sort’s miss percentage increased to almost double. Neither of the results from simulation 2 showed a decrease of miss percentage which would show a low spatial locality of reference. If the memory accessed by these two functions used a significant level of spatial locality, sim2 would have yielded lower miss percentages, as local instructions would have been saved in sequential blocks after an instruction led to a cache miss.

**Part 3**

// sim3.c

// ENCM 369 Winter 2020 Lab 10 Exercise B

// Author: S. Norman

//

// If you build an executable using gcc -Wall sim1.c -o sim1

// you can run it by redirecting input to come from a data file,

// as in

//         ./sim1 < heapsort\_trace.txt

#include <stdio.h>

#include <stdlib.h>

int read\_one\_line(unsigned \*p)

// Read one line of the input stream.

// Return value is normally 'r' or 'w' to indicate read or write.

// In that case, \*p contains the address read from the input line.

// Return value is 'e' to indicate that input failed at the end of

// the input stream.

{

  int nscan, rw;

  char buf[2];

  nscan = scanf("%1s%x", buf, p);

  if (nscan == EOF)

    return 'e';                 /\* indicate end-of-file \*/

  else if (nscan != 2) {

    fprintf(stderr, "Format error in input stream.\n");

    exit(1);

  }

  rw = buf[0];

  if (rw != 'r' && rw != 'w') {

    fprintf(stderr, "Read/write character was neither r nor w.\n");

    exit(1);

  }

  return rw;

}

// These two arrays keep track of all the V-bits and stored tags in

// the array.  We don't need an array for data to count hits and misses.

// Because these arrays are external variables, it's safe to assume

// that they will be initialized to all zeros before main starts.

char v\_bit[256];

unsigned stored\_tag[256];

int main(void)

{

  int read\_count = 0, read\_hits = 0;

  int write\_count = 0, write\_hits = 0;

  int access\_count, miss\_count;

  int rw;

  unsigned address, search\_tag, set\_bits;

  int hit;

  while (1) {

    rw = read\_one\_line(&address);

    if (rw == 'e') break;

    set\_bits = (address & 0x7f8) >> 3;   // bits 10-3

    search\_tag = address >> 11;          // bits 31-11

    // Note: Next line results in either hit == 1 or hit == 0.

    hit = v\_bit[set\_bits] == 1 && stored\_tag[set\_bits] == search\_tag;

    if (rw == 'r') {

      read\_count++;

      read\_hits += hit;

    }

    else {

      write\_count++;

      write\_hits += hit;

    }

    if (!hit) {                 // On a miss, update V-bit and search\_tag.

      v\_bit[set\_bits] = 1;

      stored\_tag[set\_bits] = search\_tag;

    }

  }

  printf("%d reads\n", read\_count);

  printf("%d read hits\n", read\_hits);

  printf("%d writes\n", write\_count);

  printf("%d write hits\n", write\_hits);

  access\_count = read\_count + write\_count;

  miss\_count = access\_count - read\_hits - write\_hits;

  printf("overall miss rate: %.2f%%\n",

         100.0 \* (double) miss\_count / access\_count);

  return 0;

}

$ ./a < mergesort\_trace.txt

104298 reads

93632 read hits

73410 writes

66168 write hits

overall miss rate: 10.08%

mmmta@LAPTOP-35G9NI35 /cygdrive/e/encm369/lab10/exb

$ ./a < heapsort\_trace.txt

64705 reads

45795 read hits

60419 writes

60354 write hits

overall miss rate: 15.16%