2020/04/07

ENCM 369 - B04

Laboratory # 10

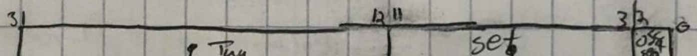
Michael Tagg - 30080581

Exercise A

(3) blocksize = 8 words
$C = 16\,kB = 2^{14}$

a) $C = S \times B_{pl}$
$B_{pl} = 8$ words $\times$ 4 bytes per word

$2^{14} = S \cdot 8 \cdot 4$
$S = \boxed{512 \text{ sets}}$

b) 

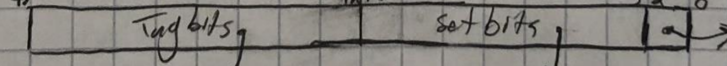
20 bits (all remaining)

(9 bits) $= \log_2 512$

(3 bits) for 8 word block
$3 = \log_2 8$

(4) Wordsize = 64 bits = 8 bytes

a) $C = 32KB = 2^{15}$ $\qquad 2^{15} = S \cdot 64$; $S = 512$ sets

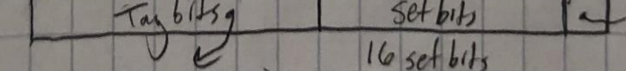


32 bits (remaining amount)

9 bits
$9 = \log_2(512 \text{sets})$

3 offset bits
$3 = \log_2(8\text{words})$

b) $C = 4MB$
$= 2^{22} = S \cdot 64$; $S = 65{,}536$ sets



(25 bits remaining)

16 set bits
$16 = \log_2(65\,536)$

still 3 bits
$3 = \log_2(8\text{words})$

c) total SRAM cells $= (2^{15} \text{sets})(64 \text{bytes per block} \cdot \frac{8 \text{bit}}{\text{byte}} + \text{Vbit} + 25 \text{bit tag})$
$\boxed{= 1.7629 \times 10^7 \text{ Cells}}$

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%
```