ICS Homework 9

April 13, 2022

1 Cache

Assume we have a **64-bit machine** with a **2-way set** associative cache. There are **8 sets**, and each block is **8 bytes**. **LRU** policy is used for eviction and **write back** policy is used.
One of the TA write a program and test it on this cache. The size of int type is **4 bytes**. The cache is **empty** before the execution of the program. Please only consider **data cache** access and ignore instruction cache access. Other processes will not interfere status of memory and cache during this execution. (**NOTE**: Contents in buffers **WON'T** be saved in registers. That means, all operations to these buffers will cause memory accesses.)

```
1  int buf1[8][8][4];
2  int buf2[8][8][4];
3
4  void func(void) {
5          int i, j, k;
6
7      for (int j = 0; j < 8; j++) {
8          for (int k = 0; k < 4; k++) {
9              for (int i = 0; i < 8; i++) {
10                 buf2[i][j][k] = buf1[i][j][k];
11                 buf2[i][j][k] = buf2[i][j][k] + 1;
12             }
13         }
14     }
15 }
```

Assume the address of **buf1[0][0][0]** is **0x0**. The address of **buf2[0][0][0]** is **0x400**. Answer the following questions:
1. Please calculate the cache miss rate for func().

2. Assume the cache hit time is 4 cycles, miss penalty is 200 cycles. Please calculate the **average access time** for **func()**.

3. How to reduce the cache miss rate through reordering codes between **line 7 and 9**? Please write **i, j, k** in the optimized order and calculate the cache miss rate.

2 Web Server

Assume we want to write a tick program which prints a "BEEP" in console every second. If there is any client connected, the BEEP will be sent to client instead of being printed on server. When client closes the connection, "BEEP" should be printed in console again. Here is part of the program:

```c
void handler(int sig) {
    write(1, "BEEP\n", 5);
    alarm(1);
}

// This function will block and return
// After the client close the connection
void wait_disconnect(int fd) {
    char c;
    while (read(fd , &c, 1) > 0 || errno == EINIR);
}

int main(void) {
    int listenfd, connfd;
    // You are not allowed to use stderr
    close(2);
    signal(SIGALRM, handler);
    alarm(1);
    listenfd = open_listenfd(1234);
    while (1) {
        connfd = accept(listenfd , NULL, NULL);
        /* Your code here */
    }
    exit (0);
}
```

Please complete the program.