Oregon State University

CS 472 - Computer Architecture

Spring 2014

Lab 5 - The Memory Hierarchy and Endian-Neutral Programming

Author:
Drake Bridgewater
Ryan Phillips

Professor:
Kevin McGrath

May 29, 2014

Part 1

Code

cache_size.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define KB 1024
#define MB 1024 * 1024

int main() {
    unsigned int steps = 256 * 1024 * 1024;
    static int arr[4 * 1024 * 1024];
    int lengthMod;
    unsigned int i;
    double timeTaken;
    clock_t start;
    int sizes[] = {
        1 * KB, 4 * KB, 8 * KB, 16 * KB, 32 * KB, 64 * KB, 128 * KB, 256 * KB,
        512 * KB, 1 * MB, 1.5 * MB, 2 * MB, 2.5 * MB, 3 * MB, 3.5 * MB, 4 * MB
    };
    int results[sizeof(sizes)/sizeof(int)];
    int s;

    // for each size to test for ...
    for (s = 0; s < sizeof(sizes)/sizeof(int); s++) {
      lengthMod = sizes[s] - 1;
      start = clock();
      for (i = 0; i < steps; i++) {
          arr[(i * 16) & lengthMod] *= 10;
            arr[(i * 16) & lengthMod] /= 10;
      }

      timeTaken = (double)(clock() - start)/CLOCKS_PER_SEC;
        printf("%d, %.8f \n", sizes[s] / 1024, timeTaken);
    }
    return 0;
}
```

Output

cache_size.txt

```

1, 16.52000000
4, 16.24000000
8, 16.31000000
16, 17.85000000
32, 18.74000000
64, 36.05000000
128, 78.88000000
256, 91.57000000
512, 92.83000000
1024, 92.88000000
1536, 92.84000000
2048, 92.94000000
2560, 92.83000000
3072, 92.86000000
3584, 92.83000000
4096, 92.86000000
```

From the output our program produces while running on a Beaglebone Black we can see that the size of the **L1 cache is about 32K** as the jump in time happened immediately after that. As for the **L2 cache it is likely to be around 128K** as there is no jump in timing after that.

Part 2

endian_neutral.c

```
#include <stdio.h>
#include <stdint.h>

#define IS_BIG_ENDIAN (*(uint16_t *)"\0\xff" < 0x100)

int main(int argc, char **argv)
{
        printf("%d\n",IS_BIG_ENDIAN); //0 if false

        short val;
        char *p_val;
        p_val = (char *) &val;

        if (IS_BIG_ENDIAN){
          p_val[0] = 0x12;
          p_val[1] = 0x34;
        } else {
          p_val[0] = 0x34;
          p_val[1] = 0x12;
        }
        printf("%x\n", val);

        return 0;
}
```