Verkefni 2
Arnar Sigurðsson
ars98@hi.is

```c
/*
 * csim.c - A cache simulator that can replay traces from Valgrind
 *     and output statistics such as number of hits, misses, and
 *     evictions.  The replacement policy is LRU.
 *
 * Implementation and assumptions:
 *  1. Each load/store can cause at most one cache miss. (I examined the trace,
 *     the largest request I saw was for 8 bytes).
 *  2. Instruction loads (I) are ignored, since we are only interested in evaluating
 *     data cache performance.
 *  3. Data modify (M) is treated as a load followed by a store to the same
 *     address. Hence, an M operation can result in two cache hits, or a miss and a
 *     hit plus an possible eviction.
 *
 */
#include <getopt.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <assert.h>
#include <math.h>
#include <limits.h>
#include <string.h>
#include <errno.h>

/* Type: Memory address */
typedef unsigned long long int mem_addr_t;


/* Hér fyrir neðan eru ýmsar víðværar breytur skilgreindar.  Þið munuð nota þessar
   en þurfið líka að bæta við nokkrum í viðbót (t.d. skilgreiningu á línu og
   skilgreiningu á skyndiminninu sjálfu)
 */

/* Globals set by command line args */
int s = 0; /* set index bits */
int b = 0; /* block offset bits */
int E = 0; /* associativity */
char* trace_file = NULL;

/* Derived from command line args */
int S; /* number of sets */
int B; /* block size (bytes) */
```

```c
/* Counters used to record cache statistics */
int miss_count = 0;
int hit_count = 0;
int eviction_count = 0;
unsigned long long int lru_counter = 1;

/* Hluturinn sem táknar eina línu */
struct Cache {
    int valid;
    int tag;
    long lru;
};

struct Cache** skyndiminni;


/* Býr til gagnagrind sem hermir eftir skyndiminni. Býr til fylki af bendum af stærð S sem tákna
mengin og svo inni í hverju hólfi
er svo bendir á fylki af Cache hlutum af stærð E sem tákna línurnar */
void initCache()
{
    skyndiminni = malloc(S * sizeof(struct Cache*));
    for(int i =0; i<S; i++){
        skyndiminni[i] = malloc(E * sizeof(struct Cache));
        for(int l = 0; l<E; l++){
            skyndiminni[i][l].lru = 0;
            skyndiminni[i][l].tag = l;
            skyndiminni[i][l].valid = 0;
        }
    }
}


/*
 * freeCache - free allocated memory
 */
void freeCache()
{
    for(int i=0; i<E; i++){
        free(skyndiminni[i]);
    }
    free(skyndiminni);
}

/*
```

```
 * accessData - Access data at memory address addr.
 *   If it is already in cache, increast hit_count
 *   If it is not in cache, bring it in cache, increase miss count.
 *   Also increase eviction_count if a line is evicted.
 */
void accessData(mem_addr_t addr)
{
    mem_addr_t bbits = (((1 << b)-1)& (addr >> (1-1))); //blokkarhliðrun
    mem_addr_t sbits = (((1 << s)-1)& (addr >> b));     //mengi
    mem_addr_t tbits = addr >> (b+s);                   //tag
    int success = 0;

    /* Athuga hvort tag í einhverri línu sé eins og tag inntaks og er valid. Ef svo er, staðfesta
smell og hætta í lykkju */
    for(int i = 0; i<E; i++){
        if(skyndiminni[sbits][i].tag == tbits && skyndiminni[sbits][i].valid == 1){
            success = 1;
            skyndiminni[sbits][i].lru = lru_counter;
            lru_counter++;
            hit_count++;
            break;
        }
    }
    /* Athuga hvort einhver lína hafi valid=0 og ef svo er, uppfæra þá línu með núverandi inntaki,
staðfesta skell og hætta í lykkju.
       Ef engin lína er laus er núverandi inntak sett í þá línu sem er lengst síðan var breytt,
eða með minnsta lru. Svo er staðfest
       eviction og skellur. */
    if(success==0){
        unsigned long long int teljari = lru_counter;
        int lina = 0;

        for(int i=0; i<E; i++){
            if(skyndiminni[sbits][i].lru < teljari){
                lina = i;
                teljari = skyndiminni[sbits][i].lru;
            }
        }

        for(int i=0; i<E; i++){
            if(skyndiminni[sbits][i].lru < teljari){
                lina = i;
                skyndiminni[sbits][i].lru = teljari;
            }
        }
```

```
        if(skyndiminni[sbits][lina].valid == 1){
            eviction_count++;
        }
        skyndiminni[sbits][lina].valid = 1;
        skyndiminni[sbits][lina].tag = tbits;
        skyndiminni[sbits][lina].lru = lru_counter;
        lru_counter++;
        miss_count++;

    }
}


/*
 * replayTrace - replays the given trace file against the cache
 */
void replayTrace(char* trace_fn)
{
    char buf[1000];
    mem_addr_t addr=0;
    unsigned int len=0;
    FILE* trace_fp = fopen(trace_fn, "r");

    if(!trace_fp){
        fprintf(stderr, "%s: %s\n", trace_fn, strerror(errno));
        exit(1);
    }

    while( fgets(buf, 1000, trace_fp) != NULL) {
        if(buf[1]=='S' || buf[1]=='L' || buf[1]=='M') {
            sscanf(buf+3, "%llx,%u", &addr, &len);

            accessData(addr);

            /* If the instruction is R/W then access again */
            if(buf[1]=='M')
                accessData(addr);
        }
    }

    fclose(trace_fp);
}

/*
```

```c
 * printSummary - Summarize the cache simulation statistics
 */
void printSummary(int hits, int misses, int evictions)
{
    /* GAMLA
    printf("hits: %d  misses: %d  evictions: %d\n", hits, misses, evictions);
    printf("miss ratio: %.2f%%\n", 100.0*misses/(hits+misses));
    */
   printf("s: %d, E: %d, b: %d =", s, E, b);
   printf(" miss ratio: %.2f%%\n", 100.0*misses/(hits+misses));
}

/*
 * printUsage - Print usage infostruct Node {
    int line;
    int legal;
    struct Node* next;
}

 */
void printUsage(char* argv[])
{
    printf("Usage: %s [-h] -s <num> -E <num> -b <num> -t <file>\n", argv[0]);
    printf("Options:\n");
    printf("  -h         Print this help message.\n");
    printf("  -s <num>   Number of set index bits.\n");
    printf("  -E <num>   Number of lines per set.\n");
    printf("  -b <num>   Number of block offset bits.\n");
    printf("  -t <file>  Trace file.\n");
    printf("\nExamples:\n");
    printf("  linux>  %s -s 4 -E 1 -b 4 -t traces/yi.trace\n", argv[0]);
    printf("  linux>  %s -s 8 -E 2 -b 4 -t traces/yi.trace\n", argv[0]);
    exit(0);
}

/*
 * main - Main routine
 */
int main(int argc, char* argv[])
{
    char c;

    while( (c=getopt(argc,argv,"s:E:b:t:h")) != -1){
        switch(c){
        case 's':
```

```
            s = atoi(optarg);
            break;
        case 'E':
            E = atoi(optarg);
            break;
        case 'b':
            b = atoi(optarg);
            break;
        case 't':
            trace_file = optarg;
            break;
        case 'h':
            printUsage(argv);
            exit(0);
        default:
            printUsage(argv);
            exit(1);
        }
    }

    /* Make sure that all required command line args were specified */
    if (s == 0 || E == 0 || b == 0 || trace_file == NULL) {
        printf("%s: Missing required command line argument\n", argv[0]);
        printUsage(argv);
        exit(1);
    }

    /* Compute S, E and B from command line args */
    S = (unsigned int) (1 << s);
    B = (unsigned int) (1 << b);

    /* Initialize cache */
    initCache();

    /* Run the simulation */
    replayTrace(trace_file);

    /* Free allocated memory */
    freeCache();

    /* Output the hit and miss statistics */
    printSummary(hit_count, miss_count, eviction_count);
    return 0;
}
```

```
addi@addi:~/Desktop/Haskoli/Onn3/TolvutaekniOgForritun/Verkefni 2/verk2$ ./csim -s 4 -E 4 -b 6 -t traces/transpose.trace
hits: 233  misses: 5  evictions: 0
miss ratio: 2.10%
addi@addi:~/Desktop/Haskoli/Onn3/TolvutaekniOgForritun/Verkefni 2/verk2$ ./csim-ref -s 4 -E 4 -b 6 -t traces/transpose.trace
hits:233 misses:5 evictions:0
miss ratio: 2.10%
addi@addi:~/Desktop/Haskoli/Onn3/TolvutaekniOgForritun/Verkefni 2/verk2$ ./csim -s 4 -E 4 -b 6 -t traces/yi.trace
hits: 6  misses: 3  evictions: 0
miss ratio: 33.33%
addi@addi:~/Desktop/Haskoli/Onn3/TolvutaekniOgForritun/Verkefni 2/verk2$ ./csim-ref -s 4 -E 4 -b 6 -t traces/yi.trace
hits:6 misses:3 evictions:0
miss ratio: 33.33%
addi@addi:~/Desktop/Haskoli/Onn3/TolvutaekniOgForritun/Verkefni 2/verk2$ ./csim -s 4 -E 4 -b 6 -t traces/yi2.trace
hits: 16  misses: 1  evictions: 0
miss ratio: 5.88%
addi@addi:~/Desktop/Haskoli/Onn3/TolvutaekniOgForritun/Verkefni 2/verk2$ ./csim-ref -s 4 -E 4 -b 6 -t traces/yi2.trace
hits:16 misses:1 evictions:0
miss ratio: 5.88%
```

Hluti 2:

Með því að keyra sh scriptuna fyrir öll möguleg gildi inntaksins s, E og b þannig að heildarútkoman er 32kb og setja úttakið í textaskrá er hægt að finna það inntak sem gefur fæstu skellina.

```
#!/usr/bin/bash

./csim -s 1 -E 1 -b 15 -t traces/gauss.trace > gildi.txt

for (( s = 1; s < 15; s++))
do
    for ((E = 1; E < 8192; E++))
    do
        for ((b = 1; b < 14; b++))
        do
            if (( 2**$s * E * 2**$b == 32768 ));
            then
                ./csim -s $s -E $E -b $b -t traces/ps.trace >> gildi.txt
            fi
        done
    done
done
```

Fyrir gauss.trace skránna komu eftirfarandi gildi:

Þarna sést að besta miss-ratioið er 1.23% og fæst með S = 2, E = 8 og B = 2048

s: 1, E: 1, b: 15 = miss ratio: 21.41%
s: 1, E: 2, b: 13 = miss ratio: 10.57%
s: 1, E: 4, b: 12 = miss ratio: 2.76%
s: 1, E: 8, b: 11 = miss ratio: 1.23%
s: 1, E: 16, b: 10 = miss ratio: 1.45%
s: 1, E: 32, b: 9 = miss ratio: 1.54%
s: 1, E: 64, b: 8 = miss ratio: 1.66%
s: 1, E: 128, b: 7 = miss ratio: 2.57%
s: 1, E: 256, b: 6 = miss ratio: 4.55%
s: 1, E: 512, b: 5 = miss ratio: 8.50%
s: 1, E: 1024, b: 4 = miss ratio: 33.30%
s: 1, E: 2048, b: 3 = miss ratio: 35.13%
s: 1, E: 4096, b: 2 = miss ratio: 35.03%
s: 2, E: 1, b: 13 = miss ratio: 18.61%
s: 2, E: 2, b: 12 = miss ratio: 4.31%
s: 2, E: 4, b: 11 = miss ratio: 1.31%
s: 2, E: 8, b: 10 = miss ratio: 1.45%
s: 2, E: 16, b: 9 = miss ratio: 1.55%

s: 2, E: 32, b: 8 = miss ratio: 1.66%
s: 2, E: 64, b: 7 = miss ratio: 2.57%
s: 2, E: 128, b: 6 = miss ratio: 4.55%
s: 2, E: 256, b: 5 = miss ratio: 8.50%
s: 2, E: 512, b: 4 = miss ratio: 16.34%
s: 2, E: 1024, b: 3 = miss ratio: 35.05%
s: 2, E: 2048, b: 2 = miss ratio: 35.18%
s: 2, E: 4096, b: 1 = miss ratio: 35.30%
s: 3, E: 1, b: 12 = miss ratio: 14.05%
s: 3, E: 2, b: 11 = miss ratio: 2.35%
s: 3, E: 4, b: 10 = miss ratio: 1.50%
s: 3, E: 8, b: 9 = miss ratio: 1.56%
s: 3, E: 16, b: 8 = miss ratio: 1.67%
s: 3, E: 32, b: 7 = miss ratio: 2.57%
s: 3, E: 64, b: 6 = miss ratio: 4.53%
s: 3, E: 128, b: 5 = miss ratio: 8.49%
s: 3, E: 256, b: 4 = miss ratio: 16.34%
s: 3, E: 512, b: 3 = miss ratio: 18.29%
s: 3, E: 1024, b: 2 = miss ratio: 35.18%
s: 3, E: 2048, b: 1 = miss ratio: 35.46%
s: 4, E: 1, b: 11 = miss ratio: 9.18%
s: 4, E: 2, b: 10 = miss ratio: 1.72%
s: 4, E: 4, b: 9 = miss ratio: 1.61%
s: 4, E: 8, b: 8 = miss ratio: 2.00%
s: 4, E: 16, b: 7 = miss ratio: 2.58%
s: 4, E: 32, b: 6 = miss ratio: 4.53%
s: 4, E: 64, b: 5 = miss ratio: 8.48%
s: 4, E: 128, b: 4 = miss ratio: 16.33%
s: 4, E: 256, b: 3 = miss ratio: 18.28%
s: 4, E: 512, b: 2 = miss ratio: 18.51%
s: 4, E: 1024, b: 1 = miss ratio: 35.45%
s: 5, E: 1, b: 10 = miss ratio: 5.29%
s: 5, E: 2, b: 9 = miss ratio: 1.73%
s: 5, E: 4, b: 8 = miss ratio: 2.00%
s: 5, E: 8, b: 7 = miss ratio: 2.73%
s: 5, E: 16, b: 6 = miss ratio: 4.54%
s: 5, E: 32, b: 5 = miss ratio: 8.49%
s: 5, E: 64, b: 4 = miss ratio: 16.32%
s: 5, E: 128, b: 3 = miss ratio: 18.25%
s: 5, E: 256, b: 2 = miss ratio: 18.50%
s: 5, E: 512, b: 1 = miss ratio: 18.80%
s: 6, E: 1, b: 9 = miss ratio: 4.67%
s: 6, E: 2, b: 8 = miss ratio: 2.00%
s: 6, E: 4, b: 7 = miss ratio: 2.72%
s: 6, E: 8, b: 6 = miss ratio: 4.70%% Intel Core i7
s: 6, E: 16, b: 5 = miss ratio: 8.47%
s: 6, E: 32, b: 4 = miss ratio: 16.37%
s: 6, E: 64, b: 3 = miss ratio: 18.22%
s: 6, E: 128, b: 2 = miss ratio: 18.48%

s: 6, E: 256, b: 1 = miss ratio: 18.79%
s: 7, E: 1, b: 8 = miss ratio: 2.51%
s: 7, E: 2, b: 7 = miss ratio: 2.69%
s: 7, E: 4, b: 6 = miss ratio: 4.68%
s: 7, E: 8, b: 5 = miss ratio: 8.79%
s: 7, E: 16, b: 4 = miss ratio: 16.30%
s: 7, E: 32, b: 3 = miss ratio: 18.25%
s: 7, E: 64, b: 2 = miss ratio: 18.44%
s: 7, E: 128, b: 1 = miss ratio: 18.77%
s: 8, E: 1, b: 7 = miss ratio: 2.89%
s: 8, E: 2, b: 6 = miss ratio: 4.63%
s: 8, E: 4, b: 5 = miss ratio: 8.73%
s: 8, E: 8, b: 4 = miss ratio: 16.94%
s: 8, E: 16, b: 3 = miss ratio: 18.18%
s: 8, E: 32, b: 2 = miss ratio: 18.48%
s: 8, E: 64, b: 1 = miss ratio: 18.73%
s: 9, E: 1, b: 6 = miss ratio: 4.83%
s: 9, E: 2, b: 5 = miss ratio: 8.64%
s: 9, E: 4, b: 4 = miss ratio: 16.84%
s: 9, E: 8, b: 3 = miss ratio: 18.91%
s: 9, E: 16, b: 2 = miss ratio: 18.41%
s: 9, E: 32, b: 1 = miss ratio: 18.76%
s: 10, E: 1, b: 5 = miss ratio: 8.66%
s: 10, E: 2, b: 4 = miss ratio: 16.64%
s: 10, E: 4, b: 3 = miss ratio: 18.78%
s: 10, E: 8, b: 2 = miss ratio: 19.13%
s: 10, E: 16, b: 1 = miss ratio: 18.70%
s: 11, E: 1, b: 4 = miss ratio: 16.35%
s: 11, E: 2, b: 3 = miss ratio: 18.56%
s: 11, E: 4, b: 2 = miss ratio: 19.00%
s: 11, E: 8, b: 1 = miss ratio: 19.42%
s: 12, E: 1, b: 3 = miss ratio: 18.21%
s: 12, E: 2, b: 2 = miss ratio: 18.78%
s: 12, E: 4, b: 1 = miss ratio: 19.29%
s: 13, E: 1, b: 2 = miss ratio: 18.43%
s: 13, E: 2, b: 1 = miss ratio: 19.07%
s: 14, E: 1, b: 1 = miss ratio: 18.72%

Fyrir skránna ps.trace komu eftirfarandi gildi:

Þarna má sjá að nokkur mismunandi inntök gefa lægsta miss-hlutfallið en þau eru:

S = 2, E = 64, B = 256,

S = 4, E = 32, B = 256

S = 8, E = 16, B = 256

s: 1, E: 1, b: 15 = miss ratio: 21.41%
s: 1, E: 2, b: 13 = miss ratio: 13.31%
s: 1, E: 4, b: 12 = miss ratio: 4.01%
s: 1, E: 8, b: 11 = miss ratio: 0.57%
s: 1, E: 16, b: 10 = miss ratio: 0.34%
s: 1, E: 32, b: 9 = miss ratio: 0.12%
s: 1, E: 64, b: 8 = miss ratio: 0.11%
s: 1, E: 128, b: 7 = miss ratio: 0.13%
s: 1, E: 256, b: 6 = miss ratio: 0.18%
s: 1, E: 512, b: 5 = miss ratio: 23.88%
s: 1, E: 1024, b: 4 = miss ratio: 23.45%
s: 1, E: 2048, b: 3 = miss ratio: 23.24%
s: 1, E: 4096, b: 2 = miss ratio: 96.32%
s: 2, E: 1, b: 13 = miss ratio: 16.58%
s: 2, E: 2, b: 12 = miss ratio: 4.18%
s: 2, E: 4, b: 11 = miss ratio: 1.55%
s: 2, E: 8, b: 10 = miss ratio: 0.35%
s: 2, E: 16, b: 9 = miss ratio: 0.13%
s: 2, E: 32, b: 8 = miss ratio: 0.11%
s: 2, E: 64, b: 7 = miss ratio: 0.13%
s: 2, E: 128, b: 6 = miss ratio: 0.18%
s: 2, E: 256, b: 5 = miss ratio: 0.28%
s: 2, E: 512, b: 4 = miss ratio: 23.51%
s: 2, E: 1024, b: 3 = miss ratio: 23.24%
s: 2, E: 2048, b: 2 = miss ratio: 23.29%
s: 2, E: 4096, b: 1 = miss ratio: 96.38%
s: 3, E: 1, b: 12 = miss ratio: 4.79%
s: 3, E: 2, b: 11 = miss ratio: 2.18%
s: 3, E: 4, b: 10 = miss ratio: 0.43%
s: 3, E: 8, b: 9 = miss ratio: 0.14%
s: 3, E: 16, b: 8 = miss ratio: 0.11%
s: 3, E: 32, b: 7 = miss ratio: 0.13%
s: 3, E: 64, b: 6 = miss ratio: 0.18%
s: 3, E: 128, b: 5 = miss ratio: 0.28%
s: 3, E: 256, b: 4 = miss ratio: 0.48%
s: 3, E: 512, b: 3 = miss ratio: 23.26%
s: 3, E: 1024, b: 2 = miss ratio: 23.29%
s: 3, E: 2048, b: 1 = miss ratio: 23.35%
s: 4, E: 1, b: 11 = miss ratio: 3.62%
s: 4, E: 2, b: 10 = miss ratio: 0.46%
s: 4, E: 4, b: 9 = miss ratio: 0.27%

s: 4, E: 8, b: 8 = miss ratio: 0.12%
s: 4, E: 16, b: 7 = miss ratio: 0.13%
s: 4, E: 32, b: 6 = miss ratio: 0.18%
s: 4, E: 64, b: 5 = miss ratio: 0.28%
s: 4, E: 128, b: 4 = miss ratio: 0.48%
s: 4, E: 256, b: 3 = miss ratio: 0.77%
s: 4, E: 512, b: 2 = miss ratio: 23.29%
s: 4, E: 1024, b: 1 = miss ratio: 23.35%
s: 5, E: 1, b: 10 = miss ratio: 1.14%
s: 5, E: 2, b: 9 = miss ratio: 0.32%
s: 5, E: 4, b: 8 = miss ratio: 0.19%
s: 5, E: 8, b: 7 = miss ratio: 0.13%
s: 5, E: 16, b: 6 = miss ratio: 0.18%
s: 5, E: 32, b: 5 = miss ratio: 0.28%
s: 5, E: 64, b: 4 = miss ratio: 0.48%
s: 5, E: 128, b: 3 = miss ratio: 0.77%
s: 5, E: 256, b: 2 = miss ratio: 0.92%
s: 5, E: 512, b: 1 = miss ratio: 23.35%
s: 6, E: 1, b: 9 = miss ratio: 0.67%
s: 6, E: 2, b: 8 = miss ratio: 0.27%
s: 6, E: 4, b: 7 = miss ratio: 0.17%
s: 6, E: 8, b: 6 = miss ratio: 0.18% Intel Core i7
s: 6, E: 16, b: 5 = miss ratio: 0.28%
s: 6, E: 32, b: 4 = miss ratio: 0.48%
s: 6, E: 64, b: 3 = miss ratio: 0.77%
s: 6, E: 128, b: 2 = miss ratio: 0.92%
s: 6, E: 256, b: 1 = miss ratio: 1.12%
s: 7, E: 1, b: 8 = miss ratio: 0.56%
s: 7, E: 2, b: 7 = miss ratio: 0.28%
s: 7, E: 4, b: 6 = miss ratio: 0.20%
s: 7, E: 8, b: 5 = miss ratio: 0.29%
s: 7, E: 16, b: 4 = miss ratio: 0.48%
s: 7, E: 32, b: 3 = miss ratio: 0.77%
s: 7, E: 64, b: 2 = miss ratio: 0.92%
s: 7, E: 128, b: 1 = miss ratio: 1.12%
s: 8, E: 1, b: 7 = miss ratio: 0.52%
s: 8, E: 2, b: 6 = miss ratio: 0.25%
s: 8, E: 4, b: 5 = miss ratio: 0.30%
s: 8, E: 8, b: 4 = miss ratio: 0.48%
s: 8, E: 16, b: 3 = miss ratio: 0.77%
s: 8, E: 32, b: 2 = miss ratio: 0.92%
s: 8, E: 64, b: 1 = miss ratio: 1.12%
s: 9, E: 1, b: 6 = miss ratio: 0.46%
s: 9, E: 2, b: 5 = miss ratio: 0.33%
s: 9, E: 4, b: 4 = miss ratio: 0.49%
s: 9, E: 8, b: 3 = miss ratio: 0.77%
s: 9, E: 16, b: 2 = miss ratio: 0.92%
s: 9, E: 32, b: 1 = miss ratio: 1.12%
s: 10, E: 1, b: 5 = miss ratio: 0.49%

s: 10, E: 2, b: 4 = miss ratio: 0.52%
s: 10, E: 4, b: 3 = miss ratio: 0.78%
s: 10, E: 8, b: 2 = miss ratio: 0.92%
s: 10, E: 16, b: 1 = miss ratio: 1.12%
s: 11, E: 1, b: 4 = miss ratio: 0.66%
s: 11, E: 2, b: 3 = miss ratio: 0.80%
s: 11, E: 4, b: 2 = miss ratio: 0.93%
s: 11, E: 8, b: 1 = miss ratio: 1.12%
s: 12, E: 1, b: 3 = miss ratio: 0.95%
s: 12, E: 2, b: 2 = miss ratio: 0.95%
s: 12, E: 4, b: 1 = miss ratio: 1.13%
s: 13, E: 1, b: 2 = miss ratio: 1.09%
s: 13, E: 2, b: 1 = miss ratio: 1.14%
s: 14, E: 1, b: 1 = miss ratio: 1.27%

Core i7 er í 4.70% miss-rate í gauss.trace og 0.18% í ps.trace sem er nokkuð gott fyrir gauss og frekar mjög gott fyrir ps.