МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет информационных технологий

Кафедра параллельных вычислений

ОТЧЕТ

О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

«Экспериментальное определение ltd»

студентки 2 курса, 24202 группы

Корсун Дарья Андреевна

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:
Кандидат технических наук
В. А. Перепелки

Новосибирск 2025

СОДЕРЖАНИЕ

ЦЕЛЬ

Экспериментальное определение tlb

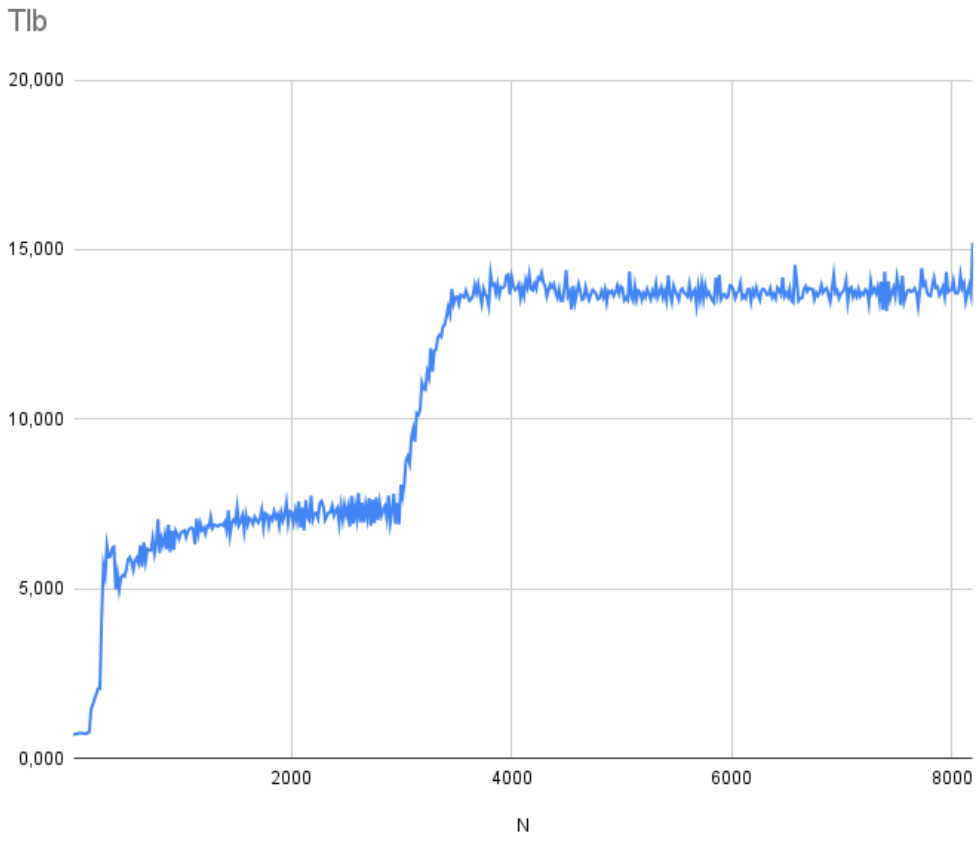
ЗАДАНИЕ

Написать программу которая позволит определить tld разных уровней.

ОПИСАНИЕ РАБОТЫ

Описание программы: программа создаёт циклическую цепочку из страниц памяти и измеряет время доступа к ним. При превышении размера TLB возникают промахи, что вызывает скачки времени доступа, позволяя определить ёмкость TLB.

После написания соответствующей программы был получен следующей график



Скачки соответствуют размерам таблицы в записях:

1) L1 Tlb: 160 = 32+128
2) L2 Tlb: 3072 = 1024+2048

Код программы

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/mman.h>
#include <unistd.h>

#define PAGE_SIZE 4096
#define CACHE_LINE 128
#define STRIDE (PAGE_SIZE + CACHE_LINE)
#define MAX_ENTRIES 8192
#define ITERATIONS 1000000
#define REPEAT_COUNT 100

static inline uint64_t read_tsc(void) {
  uint64_t val;
  asm volatile("mrs %0, cntvct_el0" : "=r"(val));
  return val;
}

void flush_cache(void *addr, size_t size) {
  asm volatile("dsb sy");
  volatile char *flush_buffer = malloc(1024 * 1024);
  if (flush_buffer) {
    for (size_t i = 0; i < 1024 * 1024; i += CACHE_LINE) {
      asm volatile("ldr xzr, [%0]" ::"r"(&flush_buffer[i]));
    }
    free((void *)flush_buffer);
  }
  asm volatile("dsb sy");
}

double measure_single_run(void **pages, int entries, int iterations,
                          int stride_multiplier) {
  int effective_stride = stride_multiplier * STRIDE;

  for (int i = 0; i < entries - 1; i++) {
    int next_idx = (i + 1) % entries;
    *(void **)pages[i * stride_multiplier] =
        pages[next_idx * stride_multiplier];
  }
  *(void **)pages[(entries - 1) * stride_multiplier] = pages[0];

  void *ptr = pages[0];

  for (int i = 0; i < entries; i++) {
    ptr = *(void **)ptr;
  }

  flush_cache(pages[0], entries * effective_stride);

  uint64_t start = read_tsc();

  for (int i = 0; i < iterations; i++) {
    ptr = *(void **)ptr;
  }

  uint64_t end = read_tsc();

  asm volatile("" : : "r"(ptr));
```

```
  uint64_t total_cycles = end - start;
  return (double)total_cycles / iterations;
}

void measure_tlb_large_stride(void **pages, int entries, int
iterations,
                              int stride_multiplier, FILE *csv_file) {
  double total_cycles = 0.0;

  for (int run = 0; run < REPEAT_COUNT; run++) {
    double cycles =
        measure_single_run(pages, entries, iterations,
stride_multiplier);
    total_cycles += cycles;
  }

  double avg_cycles = total_cycles / REPEAT_COUNT;

  if (csv_file) {
    fprintf(csv_file, "%d,%.3f\n", entries, avg_cycles);
    fflush(csv_file);
  }
}

int main() {
  FILE *csv_file = fopen("tlb_results.csv", "w");
  if (!csv_file) {
    return 1;
  }

  fprintf(csv_file, "entries,cycles_per_access\n");
  fflush(csv_file);

  size_t total_size = MAX_ENTRIES * 8 * STRIDE;

  void *buffer;
  if (posix_memalign(&buffer, PAGE_SIZE, total_size) != 0) {
    fclose(csv_file);
    return 1;
  }

  memset(buffer, 0, total_size);

  void **pages = (void **)malloc(MAX_ENTRIES * 8 * sizeof(void *));
  if (!pages) {
    free(buffer);
    fclose(csv_file);
    return 1;
  }

  for (int i = 0; i < MAX_ENTRIES * 8; i++) {
    pages[i] = (char *)buffer + i * STRIDE;
  }

  for (int entries = 8; entries <= 256; entries += 8) {
    measure_tlb_large_stride(pages, entries, ITERATIONS, 4, csv_file);
  }

  for (int entries = 272; entries <= MAX_ENTRIES; entries += 16) {
    measure_tlb_large_stride(pages, entries, ITERATIONS, 4, csv_file);
  }

  free(pages);
  free(buffer);
  fclose(csv_file);

  return 0;
}
```

ЗАКЛЮЧЕНИЕ

Получилось достигнуть цели и экспериментально определит размер tlb.