«Исследование иерархии памяти, работы процессора в ЭВМ на языке Си»

Цель задания

- Научиться получать уникальные характеристики процессора и ОС Linux через /proc/cpuinfo, /proc/meminfo.
- Понять, как иерархия памяти влияет на производительность.
- Написать программу, которая корректно работает только на вашей системе, используя её уникальные параметры (модель CPU, объём RAM и т.п.).
- Закрепить навыки работы с буферизованным вводом (getchar, putchar) и циклическими вычислениями.

Общая часть

Часть 1 — Теоретическая (в отчёте)

Ответьте на следующие вопросы в контексте вашей системы и приведите команды/фрагменты вывода:

Какова модель процессора вашей машины? Сколько у него ядер и уровней кэш-памяти (L1, L2, L3)?

Какой объём оперативной памяти (RAM) установлен?

Где в Linux хранится информация о процессоре и памяти? (укажите пути к файлам)

Что входит в иерархию памяти?

Почему кэш-память компенсирует разницу в скорости между CPU и RAM?

Используйте команды:

cat /proc/cpuinfo | grep "model name\|cpu cores\|cache size"

cat /proc/meminfo | grep MemTotal

lscpu

Часть 2 — Программная

Все программы должны быть написаны только на языке Си, без scanf/printf для чисел, только getchar и putchar.
Программа должна выполнять нагрузку на CPU и RAM, а также проверять уникальные характеристики машины.

Уровень «Удовлетворительно»

Напишите программу, которая:

1. Читает число N (до 100 000) с помощью getchar (обработка многоцифровых чисел — через цикл).
2. Выполняет цикл от 1 до N, в котором:
- складывает числа → sum += i
- записывает текущее значение i в глобальный массив char memory[65536] по круговому принципу: memory[i % 65536] = i % 256
3. В конце программы выводит сумму с помощью putchar (цифры → символы).

Уровень «Хорошо» — учёт иерархии памяти и буферизации

Выполните всё из уровня «Удовлетворительно», +:

1. Программа должна учитывать размер кэша L1 вашей машины (в КБ).
2. В цикле используйте объём данных, превышающий L1-кэш, чтобы вызвать промахи кэша:
- Объявите массив char data[256 * 1024]; (256 КБ — больше типичного L1)
- В цикле: data[i % (256*1024)] = (sum + i) % 256;
3. Используйте буферизованный ввод:

   Перед чтением первого символа вызовите setbuf(stdin, NULL);

   Продемонстрируйте разницу: если ввод не буферизован — программа читает побайтово; если буферизован — ждёт <**Enter**>.

Уровень «Отлично» — уникальный алгоритм, привязанный к CPU

Выполните всё из уровня «Хорошо», плюс:

1. При помощи /proc/cpuinfo и команд awk и grep получить название процессора:

   Например: model name   : Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz

   Извлечь модель i7-9700K и перенаправить эту команду на вход программе.

   ./main < i7-9700K

   Программа должна читать модель из потока ввода и записать ее в массив char и использовать её название как ключ для шифрования результата.

2. Выводите не зашифрованную сумму через putchar
3. Вместо вывода чистой суммы — зашифруйте её:

   Переведите сумму в строку цифр

   Каждую цифру замените на (цифра + ASCII/Unicode-код первой буквы модели CPU) % 10

   Выводите зашифрованную сумму через putchar

   Пример:

   Модель CPU: i7-9700K → первая буква 'i' → ASCII = 105, '7' → ASCII = 55, '-' → ASCII = 45

   Сумма = 123 → шифр = (1+105)%10=6, (2+55)%10=7, (3+45)%10=8 → вывод: 678

!Если вы сдаёте программу не со своего компьютера. При запуске программы на другом ПК должен выводиться код, **полученный** на **вашем компьютере** и тот код который рассчитан на **нынешнем** (однако это может не произойти из-за различия написания команд Linux по извлечению модели из /proc/cpuinfo. В

таком случае выводить только тот код, что был рассчитан программой дома.

Неверная модель CPU → неверный результат

Требования к оформлению отчёта

Отчёт оформляется в электронной форме и должен содержать следующие разделы:

1. Титульный лист

- По шаблону СибГУТИ
- Название работы: *«Исследование иерархии памяти, работы процессора в ЭВМ на языке С»*
- ФИО студента, группа

2. Теоретические ответы с командами и выводом терминала.

3. Листинг программы

где происходит нагрузка на CPU и RAM
как реализована «защита»
где учтена иерархия памяти
как работает шифрование (для уровня 5)

4. Пример выполнения (скриншот терминала).

5. Вывод:

Почему программа работает только на вашей машине?

Как иерархия памяти влияет на производительность?

Что вы узнали о буферизации ввода?