Лабораторная работа №3

Кэш

Цель работы: моделирование системы “процессор-кэш-память”.

Инструментарий и требования к работе: работа выполняется на C/C++ (C11 и новее / C++20), Python (3.11.6) или Java (Temurin-17.0.8.1+1). Требования для всех работ: Правила оформления и написания работ.

Описание работы

Необходимо программно смоделировать работу кэша процессора в двух вариантах: с политикой вытеснения LRU и bit-pLRU.

Реализованную модель необходимо использовать для определения процента попаданий (число попаданий к общему числу обращений) и общего времени (в тактах), затраченного на выполнение задачи.

Вывод результата производится в стандартный поток вывода в формате (стиль Си для printf):

```
LRU:\thit perc. %3.4f%%\ttime: %d\npLRU:\thit perc. %3.4f%%\ttime: %d\n
```

Если что-то не реализовано, то выводится unsupported. Например, если нет реализации pLRU, то вывод про него будет в формате: `pLRU:\tunsupported\n`

Для вывода результата настоятельно рекомендуется использовать printf в C и C++, System.out.printf в Java и print в Python (https://stackoverflow.com/a/37848366). Использовать другие варианты не запрещается, но результат должен быть эквивалентным.

Задача

Имеется следующее определение глобальных переменных и функций (код ниже на C):

```
#define M 64
#define N 60
#define K 32
int8 a[M][K];
int16 b[K][N];
int32 c[M][N];

void mmul()
{
  int8 *pa = a;
  int32 *pc = c;
  for (int y = 0; y < M; y++)
  {
    for (int x = 0; x < N; x++)
    {
      int16 *pb = b;
      int32 s = 0;
      for (int k = 0; k < K; k++)
      {
        s += pa[k] * pb[x];
        pb += N;
      }
      pc[x] = s;
    }
    pa += K;
    pc += N;
  }
}
```

Моделируемый кэш используется только для работы с данными (не командами).

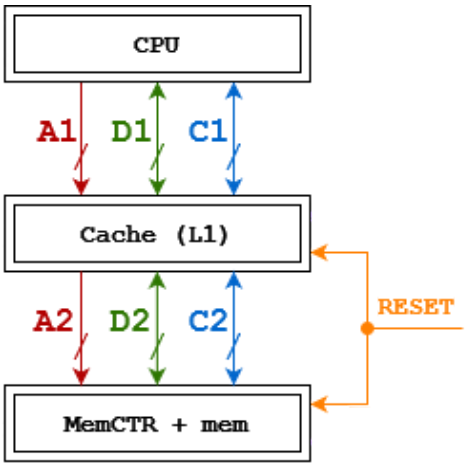
Сложение, инициализация переменных, переход на новую итерацию цикла и выход из функции занимают 1 такт. Умножение – 5 тактов. Обращение к памяти вида pc[x] считается за одну команду.

Массивы хранятся в памяти последовательно, первый из них начинается с адреса 0x40000.

Все локальные переменные лежат в регистрах процессора.

Инициализация по умолчанию (`RESET`) – все кэш-линии в состоянии invalid.

Система “процессор-кэш-память”



Переменные, параметры, константы (как они должны быть в коде):

- `MEM_SIZE` – размер памяти (в байтах)
- `CACHE_SIZE` – размер кэша, без учёта служебной информации (в байтах)
- `CACHE_LINE_SIZE` – размер кэш-линии (в байтах)
- `CACHE_LINE_COUNT` – кол-во кэш-линий
- `CACHE_WAY` – ассоциативность
- `CACHE_SETS_COUNT` – кол-во блоков кэш-линий
- `ADDR_LEN` – длина адреса (в битах)
- `CACHE_TAG_LEN` – длина тэга адреса (в битах)
- `CACHE_IDX_LEN` – длина индекса блока кэш-линий (в битах)
- `CACHE_OFFSET_LEN` – длина смещения внутри кэш-линии (в битах)

Устройство кэш-линии:

flags	**tag**	**data**
(сколько требуется)	CACHE_TAG_LEN	CACHE_LINE_SIZE

Интерпретация адреса кэшем (слева старшие биты, справа – младшие):

tag	**idx**	**offset**
CACHE_TAG_LEN	CACHE_IDX_LEN	CACHE_OFFSET_LEN

Размерность шин

Шина	Обозначение	Размерность
A1, A2	ADDR1_BUS_LEN, ADDR2_BUS_LEN	вычислить самостоятельно
D1, D2	DATA1_BUS_LEN, DATA2_BUS_LEN	16 бит
C1, C2	CTR1_BUS_LEN, CTR2_BUS_LEN	вычислить самостоятельно

Время

Время отклика – расстояние в тактах от первого такта команды до первого такта ответа.

Для моделируемой системы:

- 6 тактов – время, через которое в результате кэш попадания, кэш начинает отвечать.
- 4 такта – время, через которое в результате кэш промаха, кэш посылает запрос к памяти.
- 100 тактов – время, через которое память начинает отвечать.

Время передачи данных по шинам для моделируемой системы:

- По шинам A1 и A2 адрес передаётся за 1 такт.
- По шинам D1 и D2 в каждый такт передаётся по 16 бит данных.
- По шинам C1 и C2 команда передаётся за 1 такт.

Параметры системы

Везде, где не указаны значения, нужно самостоятельно вычислить. Все параметры должны быть явно заданы в коде – либо константы, либо переменные, значения которых вычисляются по формулам. Набор команд общий, остальное разделено по вариантам.

Команды:

CPU → Cache		
C1_READ8 C1_READ16 C1_READ32	C1_WRITE8 C1_WRITE16 C1_WRITE32	Команды, запрашивающие несколько байт, не могут пересекать кэш-линию.
Cache → Mem		
C2_READ_LINE	C2_WRITE_LINE	Команды пишут и читают порциями, равными размеру кэш-линии.
CPU ← Cache и Cache ← Mem		
C1_RESPONSE	C2_RESPONSE	Ответ на команду.

Параметры (вариант 1)

MEM_SIZE	512 Кбайт
ADDR_LEN	вычислить самостоятельно
Конфигурация кэша	look-through write-back
Политика вытеснения кэша	LRU и bit-pLRU
CACHE_WAY	вычислить самостоятельно
CACHE_TAG_LEN	10 бит
CACHE_IDX_LEN	вычислить самостоятельно
CACHE_OFFSET_LEN	вычислить самостоятельно
CACHE_SIZE	вычислить самостоятельно
CACHE_LINE_SIZE	32 байт
CACHE_LINE_COUNT	64
CACHE_SETS_COUNT	вычислить самостоятельно

Параметры (вариант 2)

MEM_SIZE	1 Мбайт
ADDR_LEN	вычислить самостоятельно
Конфигурация кэша	look-through write-back
Политика вытеснения кэша	LRU и bit-pLRU
CACHE_WAY	вычислить самостоятельно
CACHE_TAG_LEN	вычислить самостоятельно
CACHE_IDX_LEN	4 бита
CACHE_OFFSET_LEN	6 бита
CACHE_SIZE	4 Кб
CACHE_LINE_SIZE	вычислить самостоятельно
CACHE_LINE_COUNT	вычислить самостоятельно
CACHE_SETS_COUNT	вычислить самостоятельно

Параметры (вариант 3)

MEM_SIZE	вычислить самостоятельно
ADDR_LEN	20 бит
Конфигурация кэша	look-through write-back
Политика вытеснения кэша	LRU и bit-pLRU
CACHE_WAY	4
CACHE_TAG_LEN	9 бит
CACHE_IDX_LEN	вычислить самостоятельно
CACHE_OFFSET_LEN	вычислить самостоятельно
CACHE_SIZE	вычислить самостоятельно
CACHE_LINE_SIZE	128 байт
CACHE_LINE_COUNT	[UPDATE] вычислить самостоятельно
CACHE_SETS_COUNT	вычислить самостоятельно

Модификация ППА

Необходимо программно смоделировать работу кэша процессора в трёх вариантах: с политикой вытеснения LRU, bit-pLRU и Round-robin (циклический).

Вывод результата производится в стандартный поток вывода в формате (стиль Си для printf):

```
LRU:\thit perc. %3.4f%%\ttime: %d\npLRU:\thit perc. %3.4f%%\ttime:
%d\nRR:\thit perc. %3.4f%%\ttime: %d\n
```

Обновляется конфигурация кэша. Теперь у всех новый единый вариант:

Параметры (вариант ППА)

MEM_SIZE	вычислить самостоятельно
ADDR_LEN	16 бит
Конфигурация кэша	look-through write-back
Политики вытеснения кэша	LRU, bit-pLRU, Round-robin
CACHE_WAY	4
CACHE_TAG_LEN	вычислить самостоятельно
CACHE_IDX_LEN	вычислить самостоятельно
CACHE_OFFSET_LEN	вычислить самостоятельно
CACHE_SIZE	вычислить самостоятельно
CACHE_LINE_SIZE	32 байт
CACHE_LINE_COUNT	вычислить самостоятельно
CACHE_SETS_COUNT	32

Размерность шин (ППА)

Шина	Обозначение	Размерность

A1	ADDR1_BUS_LEN	вычислить самостоятельно
A2	ADDR2_BUS_LEN	вычислить самостоятельно
D1	DATA1_BUS_LEN	16 бит
D2	DATA2_BUS_LEN	32 бит
C1	CTR1_BUS_LEN	вычислить самостоятельно
C2	CTR2_BUS_LEN	вычислить самостоятельно

Массивы хранятся в памяти последовательно, первый из них начинается с адреса 0x400.

Содержание отчета

1. Минититульник (таблица с ФИО, группой и названием работы из шаблона).
2. Ссылка на репозиторий.
3. Инструментарий (язык и версия компилятора/интерпретатора).
4. *Результат работы написанной программы* (то, что выводится в стандартный поток вывода).
5. *Результат расчёта параметров системы* (формулы для расчёта и какие числа по итогу получились).
6. *Описание работы написанного кода*: ЧТО БЫЛО РЕАЛИЗОВАНО (LRU и pLRU или только 1 из этого), как вы его реализовали (как у вас в коде представлен кэш, как реализованы политики вытеснения и конфигурация в целом).

При описании работы написанного кода может быть полезно приложить формулы или небольшие рисунки для иллюстрации пояснений.

Порядок сдачи работы

1. Выполнить работу.
2. Оформить отчет в формате pdf.

3. Загрузить файл отчета и файлы с исходным кодом (расширения *.c/*.h/*.cpp/*.hpp или .py или *.java) в выданный вам репозиторий в корень.
4. Запустить автотесты (проверка логов будет производиться вручную ближе к дедлайну). Подробнее: Автотесты на GitHub - comp-arch-cource (gitbook.io)

   ВАЖНО: закрытых тестов нет и проверяться будет то, что выводится в автотестах на GH. Если отчёт не в pdf или там код не собирается/падает в runtime – 0 баллов.
5. Отправить на проверку работу (в открытом PR отмечаем Assignee Викторию и ставим label submit) и ждём ответа.

В репозиторий необходимо загружать файл с отчётом в формате pdf, названным в формате **“Фамилия_Имя_Группа_НомерРаботы.pdf”**. Номер группы – только 2 последние цифры группы, номер работы – порядковый номер лабораторной работы: 1, 2 и т.д.