Лабораторная работа №3

Сайфуллин Искандар M3136

6.12.2023
Репозиторий: https://github.com/skkv-itmo2/itmo-comp-arch-2023-cache-1sSay
Версия Python: 3.11.6

Вывод программы

```
LRU:  hit perc. 99.9551%  time: 3157071
pLRU: hit perc. 99.9551%  time: 3154411
```

Вычсление параметров

Дано:
`MEM_SIZE = 1 MB`
`CACHE_SIZE = 4 KB`
`CACHE_IDX_LEN = 4`
`CACHE_OFFSET_LEN = 6`

Вычисления:
Можно сразу же вычислить длину адреса, размер кэш-линии и количество наборов:
`ADDR_LEN = log2(MEM_SIZE) = 20`
`CACHE_LINE_SIZE = 2^CACHE_OFFSET_LEN = 64`
`CACHE_SETS_COUNT = 2^CACHE_IDX_LEN = 16`

Далее можно вычислить длину тэга, количество кэш-линий:
`CACHE_TAG_LEN = ADDR_LEN - CACHE_IDX_LEN - CACHE_OFFSET_LEN = 10`
`CACHE_LINE_COUNT = CACHE_SIZE / CACHE_LINE_SIZE = 64`

Теперь можно вычислить ассоциативность:
`CACHE_WAY = CACHE_LINE_COUNT / CACHE_SETS_COUNT = 4`

Раз по `A1` и `A2` адрес передается за 1 такт, то ширина шины должна быть `20`
Размер шины данных нам дан.
Размер шины комманд равны соотвественно 3 и 2.

Описание работы программы

- Функция `print_list_in_memory`, чтобы вычислить адреса начала массивов, так как они у нас хранятся последовательно.
  `a - 0x40000`
  `b - 0x40800`
  `c - 0x41700`
- Функция `calc_program_time` эмулирует работу задачи.
  Сначала заводим счётчик для тактов, кэш-попаданий и кэш-промахов. Далее инициализируем все переменные. Всего есть 3 момента, где мы обращаемся к кэшу:
  1. достать значение в массиве по указателю `pa`
  2. достать значение в массиве по указателю `pb`
  3. записать значение в массив по указателю `pc`
- Класс CacheLine. Просто хранит в себе тэг, последнее время последнего использования для LRU и бит для Bit-pLRU.

- Класс Cache. Он принимает на вход тип: LRU или Bit-PLRU. Изначальные инициализируем все сеты, кэш-линии изначально пусты (для LRU их время последнего использование равно бесконечности (`10^20`), для Bit-PLRU бит равен 0). Также реализованы следующие методы:
  1. `loadLRU(addr, size)`, где `addr` – адрес первого байта числа, `size` – размер числа (8 или 16, но в принципе это не важно). В нём мы пытаемся сначала найти тэг адреса в соответсвующем сете. Если не находим, то обращаемся к памяти и заменяем самую старую кэш-линию
  2. `writeLRU(addr, size)`. Если находим тэг, то записываем, иначе отправляем самую старую кэш-линию в память, и пишем уже на её место в кэше.
  3. `loadBitPLRU(addr, size)`. Работает также, но работает уже по принципу Bit-pLRU. То есть у кэш-линии есть флажочек, который говорит, использовали ли её недавно. Если нашли тэг, то в бит ставим 1 (при этом сбрасывая остальные, если все флажки в сете стали единицами), иначе подгружаем кэш-линию из памяти и заменяем ею самую первую кэш-линию со значение 0 у флажка.
  4. `writeBitPLRU(addr, size)`. Если нашли тэг, то в бит ставим 1 (как таковых данных у нас, кстати, нет), иначе отправляем в память самую первую кэш-линию с флажком 0 и записываем новые данные туда и ставим бит в 1.