ФИО	Группа	Название
Бобовский Кирилл Алексеевич	M3138	Кэш

Вариант 3
https://github.com/skkv-itmo/itmo-comp-arch-2023-cache-WannaSleep43

C++, cpp (GCC) 13.2.1

LRU: hit perc. 99.9335% time: 3290640
pLRU: hit perc. 99.9335% time: 3290640

Вычисления

MEM_SIZE	128 Кбайт
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	4 бит
CACHE_OFFSET_LEN	7 бит
CACHE_SIZE	8 Кбайт
CACHE_LINE_SIZE	128 байт
CACHE_LINE_COUNT	64
CACHE_SETS_COUNT	16

MEM_SIZE = 2^ADDR_LEN = 2^20 = 128 Кбайт
CACHE_OFFSET_LEN = log(CACHE_LINE_SIZE) = log(128) = 7
CACHE_IDX_LEN = ADDR_LEN — CACHE_OFFSET_LEN — CACHE_TAG_LEN = 20 — 9 — 7 = 4
CACHE_SETS_COUNT = 2^CACHE_IDX_LEN = 2^4 = 16
CACHE_LINE_COUNT = CACHE_WAY * CACHE_SETS_COUNT = 4 * 16 = 64
CACHE_SIZE = CACHE_LINE_COUNT * CACHE_LINE_SIZE =
64*128 = 2^13 = 8 Кбайт

Шина	Обозначение	Размерность
A1, A2	`ADDR1_BUS_LEN`, `ADDR2_BUS_LEN`	20 бит
D1, D2	`DATA1_BUS_LEN`, `DATA2_BUS_LEN`	16 бит
C1, C2	`CTR1_BUS_LEN`, `CTR2_BUS_LEN`	4 бит

Так как длина адреса 20 бит, а шины A передают его за 1 такт, то размерность шин A должна быть тоже 20 бит.

Шины C передают команды, всего команд у нас 10, поэтому чтобы их закодировать нужно как минимум 4 бита.

Что сделано и как сделано

В ходе выполнения работы я смоделирвоал работу системы процессор-кэш-память. Я реализовал политики вытеснения LRU и pLRU, а так же конфигурации look-through и write-back.

Как всё устроено

Система процессор-кэш-память написана при помощи классов. У меня есть класс Cache, который предстставляет собой кэш, он имеет методы read и write, которые позволяют обратиться к кэшу и попросить его считать/записать значение по адресу. Сам Cache включает в себя CACHE_SETS_COUNT блоков, когда к кэшу приходит запрос по какому-то адресу, он берет из этого адреса IND и находит блок с таким индексом, а дальше обработка запроса происходит в блоке.

Блок содержит в себе CACHE_WAY кэш-линий, линии изначально инициализируются флажком MSI = 2 (0 — Modify, 1 — Shared, 2 — Invalid). Когда в блок поступает запрос, он пытается найти нужную кэш-линию, если он находит кэш-линию с нужным тэгом, которая не является Invalid, происходит попадание в кэш и блок обращается к кэш-линии, чтобы она вернула ему значение байта с нужным адресом. При обращении к кэш-линии, ее флажок LRU обновляется на значение, которое больше всех остальных в этом блоке, а pLRU становится равным одному. Если изменение pLRU этой линии на 1 приведет к тому, что все pLRU в блоке равны одному, то pLRU всех кэш-линий в блоке зануляются, а затем pLRU текущей становится единицей.

Когда запрос приходит кэш-линии, исходно подразумевается, что в ней есть ячейка с нужным адресом(иначе мы бы в нее не зашли) и она с ней работает.

В случае, если в блоке не оказалось нужной валидной кэш-линии, он находит не валидную кэш-линию, либо, если такой не оказалось, он находит кэш-линию с самым маленьким флажком LRU(которая наиболее давно изменялась) для политики LRU и первую линию с pLRU = 0 для политики pLRU, далее блок говорит этой кэш-линии, что она должна перезаписать свои данные в память(если MSI=0 и она модифицирована), а затем он удаляет эту кэш-линию и записывает на ее место новую кэш-линию из памяти, содержащую нужный адрес. (конфигурация look-through - процессор обращается к кэшу, а кэш к памяти и конфигурация write-back — запись в память происходит при удалении кэш-линии из кэша).

Когда мне нужно что-то сделать с памятью, я обращаюсь к Кэшу, он обращается к нужному блоку и смотрит, есть ли нужная кэш-линия в блоке и считает попадание или промах.