Домашнее задание №4 по курсу:
«Введение в архитектуру вычислительных систем»

При проектировании вычислительных устройств важное значение отводится программному моделированию будущей архитектуры. С помощью функциональных и потактовых симуляторов можно до момента получения первого инженерного образца проверить функциональность и производительность будущего устройства. Также потактовые модели служат важным инструментом для анализа влияния различных улучшений / изменений микроархитектуры на производительность.
Более подробно о программном моделировании вы можете почитать в книге Григория Речистова «Программное моделирование вычислительных систем»

В данном задании вы будете исследовать влияние политик замещения и префетчинга на характеристики подсистемы памяти модели микропроцессора. Задача будет исследовательской, готовьтесь.

Моделирование будет производится с помощью Open-Source симулятора ChampSim.

Часть 1: Установка ChampSim

Для установки выполните следующие команды:

```
git clone-- recursive https : // github.com/ChampSim/ChampSim.git
cd ChampSim
vcpkg/bootstrap-vcpkg.sh
vcpkg/vcpkg install
```

Основной конфигурационный файл симулятора: *champsim_config.json*
В нем мы будем менять параметры подсистемы памяти (конкретно — L2 кэша).
~~Посмотрите в файл и попытайтесь найти знакомые слова~~.

```
"L2C": {
    "sets": 256,
    "ways": 8,
    "rq_size": 32,
    "wq_size": 32,
    "pq_size": 16,
    "mshr_size": 32,
    "latency": 10,
    "max_tag_check": 1,
    "max_fill": 1,
    "prefetch_as_load": false,
    "virtual_prefetch": false,
    "prefetch_activate": "LOAD,PREFETCH",
    "replacement": "lru",
    "prefetcher": "no"
},
```

После любого изменения конфигураций вам нужно будет скомпилировать симулятор следующими командами (Новая конфигурация — новый бинарник).

```
./config.sh champsim_config.json
make -j $(nproc)
cp bin/champsim/ bin/<name>
```

Последняя команда нужна, чтобы переименовать и сохранить бинарник определенной конфигурации для последующих экспериментов
(например, *bin/champsim_lru_no_prefetch_512_set_8_way*)

Часть 2: Бенчмарки (трассы, приложения)

В симулятор в качестве входа идут различные трассы/бенчмарки/приложения. На выходе — статистика, которую мы будем анализировать. В нашем случае трассы заботливо подготовлены: Link. Это стандартные бенчмарки из набора SPEC2017, SPEC2006, которые классически используются для анализа производительности CPU.
Методологически правильно было бы прогнать все трассы, чтобы получить более полную картину. Но, я боюсь, что у вас нет своего личного кластера и трех дней свободного времени. Поэтому, ограничимся **тремя трассами**. Вы можете выбрать любые три трассы по ссылке выше, но мои рекомендации — выбрать те, которые более интенсивно используют память, например:

x264_r (кодирование видео)

- Это приложение использует интенсивную обработку данных, что приводит к большому количеству обращений в кэш. Видео кодирование часто имеет очень высокие требования к локальности данных, и оно активно использует кэш, что делает его хорошим кандидатом для нагрузочных тестов L2.

hmmer_r (поиск в базах данных)

- HMMER занимается поиском в биологических базах данных, и алгоритмы, которые он использует, активно обращаются к памяти и кэшам. Это приложение генерирует большие объемы данных и часто вызывает промахи в кэше, что хорошо для тестирования.

bwaves_r (расчеты для гидродинамики)

- Приложение выполняет сложные вычисления, требующие больших объемов данных и частых обращений в память. Оно нагружает кэш и может быть полезно для анализа работы L2 кэша в условиях вычислительных интенсивных задач.

Gobmk_r (игра в го)

- Gobmk использует алгоритмы, которые требуют частых операций с большими структурами данных и активного обращения к памяти. Приложение может сильно нагружать кэш, особенно при больших размерах игры.

leslie3d_r (решение задач в области гидродинамики)

- Leslie3d занимается сложными вычислениями в области трехмерной гидродинамики и активно использует кэш для обработки больших массивов данных. Оно может быть полезно для наблюдения за работой L2 при интенсивных вычислениях.

povray_r (рендеринг изображений)

- PovRay генерирует изображения, требующие значительных вычислений и обработки больших объемов данных. Оно также часто обращается к памяти и будет активно использовать L2 кэш для ускорения работы.

Выбор конкретных трасс остается за вами, **но я буду очень скептически относиться к работам, у которых в наборе совпадут все три трассы.**

Скачать трассы можно в директорию с ChampSim командой wget:

```
wget https://dpc3.compas.cs.stonybrook.edu/champsim-traces/speccpu/625.x264_s-12B.champsimtrace.xz
```

Часть 3: Конфигурации для экспериментов:

Нужно будет проверить 3 конфигурации L2 кэша:

1) *“replacement” : “lru”,*
   “prefetcher” : “no”

2) *“replacement” : “ship”,*
   “prefetcher” : “no”

3) *“replacement” : “ship”,*
   “prefetcher” : “spp_dev”

Код соответствующих политик можете найти в директориях *replacement/* и *prefetcher/* соответственно

Статьи по политике замещения SHIP и политике префетчинга SPP прикреплены вам к письму с заданием. Внимательно ознакомьтесь. Можете использовать AI-помощников.

Для каждой из трех конфигураций вам нужно будет провести 5 замеров с различными размерами L2 кэша.

Напоминаю, что размер кэша вычисляется по формуле
*#set * #way * cache_line_size (64 bytes)*

Мои предложения:

256 set, 8 way -> 128 Кб
512 set, 8 way -> 256 Кб
512 set, 16 way -> 512 Кб
1024 set, 8 way -> 512 Кб
1024 set, 16 way -> 1024 Кб

Но можете выбрать и другие значения. Главное, чтобы можно было построить зависимость интересующей нас метрики от размера кэша.

Я рекомендую вам сразу скомпилировать все эти конфигурации, чтобы у вас были бинарники вида *bin/champsim_lru_no_prefetch_512_set_8_way,* и в дальнейшем не путаться.

Часть 4: Запуск трасс:

Запуск симулятора осуществляется командой

```
bin/champsim_lru_no_prefetch_512_set_8_way --warmup-instructions 50000000 --simulation-instructions 100000000 <path_to_trace>.xz
```

Здесь флаг *--warmup-instructions* (50 млн) означает количество инструкций для прогрева кэша. То есть, в кэш будут поступать обращения, и он будет заполняться, но в статистике данные холодные (compulsory) промахи не будут учтены. Это важная часть любого эксперимента на симуляторе, так как непрогретые кэши/BPU могут дать погрешность при измерениях.

Флаг *--simulation-instructions* (100 млн) означает число инструкций, которые будут исполнены после прогрева и с них будет сниматься статистика по обращениям в кэш.

Результатом нашего запуска будет лог в *STDIN*, в котором нам нужна ровно одна строчка:

```
cpu0->cpu0_L2C TOTAL      ACCESS: 11165    HIT: 4993    MISS: 6172    MSHR_MERGE: 0
```

Из нее, поделив *HIT* на *ACCESS* мы получим L2_Hit_Rate для одной трассы.

Для каждой из трех конфигураций на всех выбранных размерах кэша вы должны прогнать так три трассы, которые вы выбрали в самом начале. Каждая трасса «бежит» примерно 10 минут.

Я вам **настоятельно рекомендую** запускать различные конфигурации в параллель. Вы же сделали себе отдельный бинарник под каждую конфигурацию, чтобы не компилировать каждый раз заново? (Вы можете наплодить терминалов, либо использовать *tmux,* либо написать скрипт на bash/python, который запустит все за Вас)

Когда вы сделали эту ~~самую душную~~ часть работы и записали все результаты со значениями L2_Hit_Rate себе в excel-таблицу, настало время анализа результатов….

Часть 5: Анализ результатов:

Первое, что вы должны сделать — это взять geomean (геометрическое среднее) L2_Hit_Rate для ваших трех трасс и посчитать это значение для каждой конфигурации.

Второе — составить 3 графика (для 1 конфигурации один график), где по оси Y будет посчитанный L2_Hit_Rate для вашего набора трасс, а по оси X размер кэша.

Третье — творчески проанализировать результаты и ответить на вопросы:

- Насколько улучшила или ухудшила результаты смена политики замещения с LRU на SHIP?
- Какой прирост Hit_Rate дало включение префетчера?
- Какой размер кэша «достаточный» (Значение, где дальнейшее увеличение не даст прироста)
- На каком размере кэша больше проявляются улучшения Hit_Rate — на большом или на малом?
- Что выгоднее увеличивать — значение set или way при равном размере кэша (512 set, 16 way vs 1024 set, 8 way)?
- И другие выводы, которые вы можете и хотели бы сделать по вашим данным.

Ваши тезисы должны подкрепляться конкретными числами.
Пример: «Смена политики замещения улучшила процент попадания в кэш на 4,5%.»

Часть 6: Отчет

Отчет должен быть оформлен в виде Word/PDF файла и отправлен на почту zaitsev.vl@phystech.edu ответным письмом.
ДЕДЛАЙН 13.05.2025 в 23:59 МСК

Структура отчета:

1. ФИО
2. Номер группы
3. Краткие описания политик LRU, SHIP, префетчера SPP. (По краткому абзацу на каждый — описать смысл алгоритма)
4. Выбранные трассы для анализа (3 штуки)
5. 3 графика зависимости L2_Hit_Rate от размера кэша (по 1му на каждую конфигурацию)
6. Выводы, которые вы сделали по вашим данным.

* Если не сложно, поделитесь, какое домашнее задание показалось вам самым интересным на ваш субъективный взгляд, а какое не понравилось. Это очень поможет нам при составлении курса для следующих поколений.

Успехов!