Лекция 5

Кеш-память

- Кеш-память имеет смысл, так как программы на практике демонстрирую свойство **локальности**
- Кеш-память не дает улучшения в худшем случае (при случайных обращениях в память)

Свойство локальности

- Временная локальность – если программа обращается к некоторой ячейке памяти впервые, велика вероятность того, что скоро обращение к этой ячейке памяти повторится
  - Циклы в коде программы
  - Переменные в памяти
- Пространственная локальность – если программа обращается к некоторой ячейке памяти, велика вероятность того, что скоро программа обратится к соседним ячейкам
  - Код программы
  - Массивы/структуры в памяти

Промахи в кеше

- Обязательный промах (compulsory miss) – ячейка не была загружена в кеш – первое обращение к ней в программе
- Промах из-за емкости (capacity miss) – размер кеша слишком мал для одновременного хранения используемых данных
- Промах из-за конфликта (conflict miss) – нужные данные были в кеше, но оказались выгружены из-за ограниченной ассоциативности

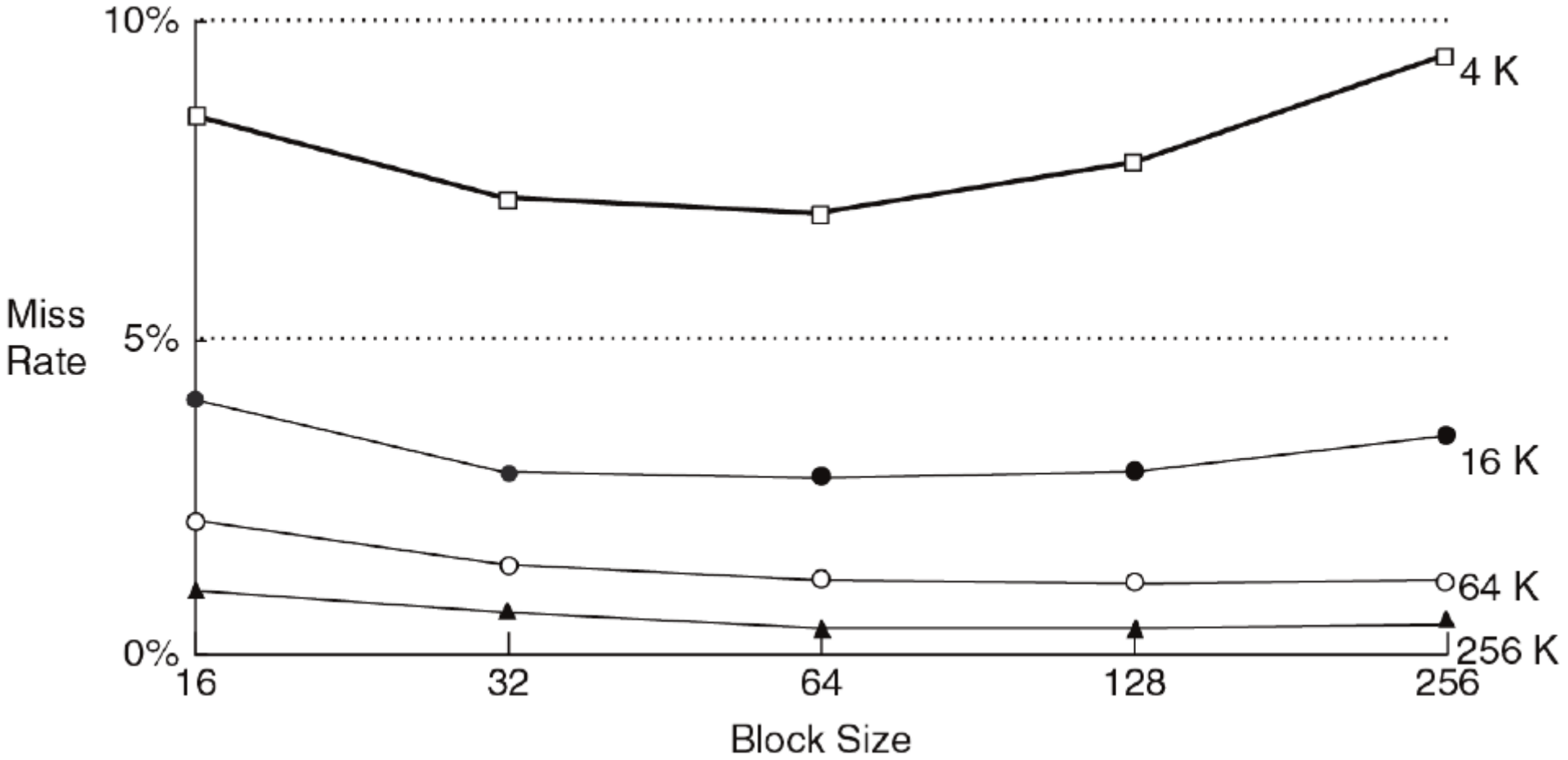
Среднее время доступа

- Оценка времени, которое процессор тратит, ожидая данные из памяти
- Для одноуровневого кеша
  $AMAT = t_c + M_c * t_M$
- t_c – время обращения к кешу
- M_c – вероятность промаха в кеше
- t_M – время обращения к памяти
- Если $t_c = 1$, $M_c = 0.10$, $t_M = 100$, то $AMAT = 11$

Строка (блок) кеша

- Cache Line (cache block) – единица хранения данных в кеше
- Загрузка одного байта приводит к загрузке всего блока кеша

Зависимость от размера блока



Кеш прямого отображения

- Каждый из адресов в ОЗУ отображается в единственную ячейку кеша
- Пусть размер блока – 16 байт, размер кеша – 256 байт, размер ОЗУ – 4 KiB: в кеше 16 блоков, в ОЗУ – 256 блоков
- Адрес в ОЗУ состоит из трех частей (адреса – hex): XYZ
  X – игнорируется при отображении, Y – номер блока, Z – смещение в блоке
  9A7 – адрес отобразится в блок кеша A
  2A3 – адрес отобразится в блок кеша A - конфликт

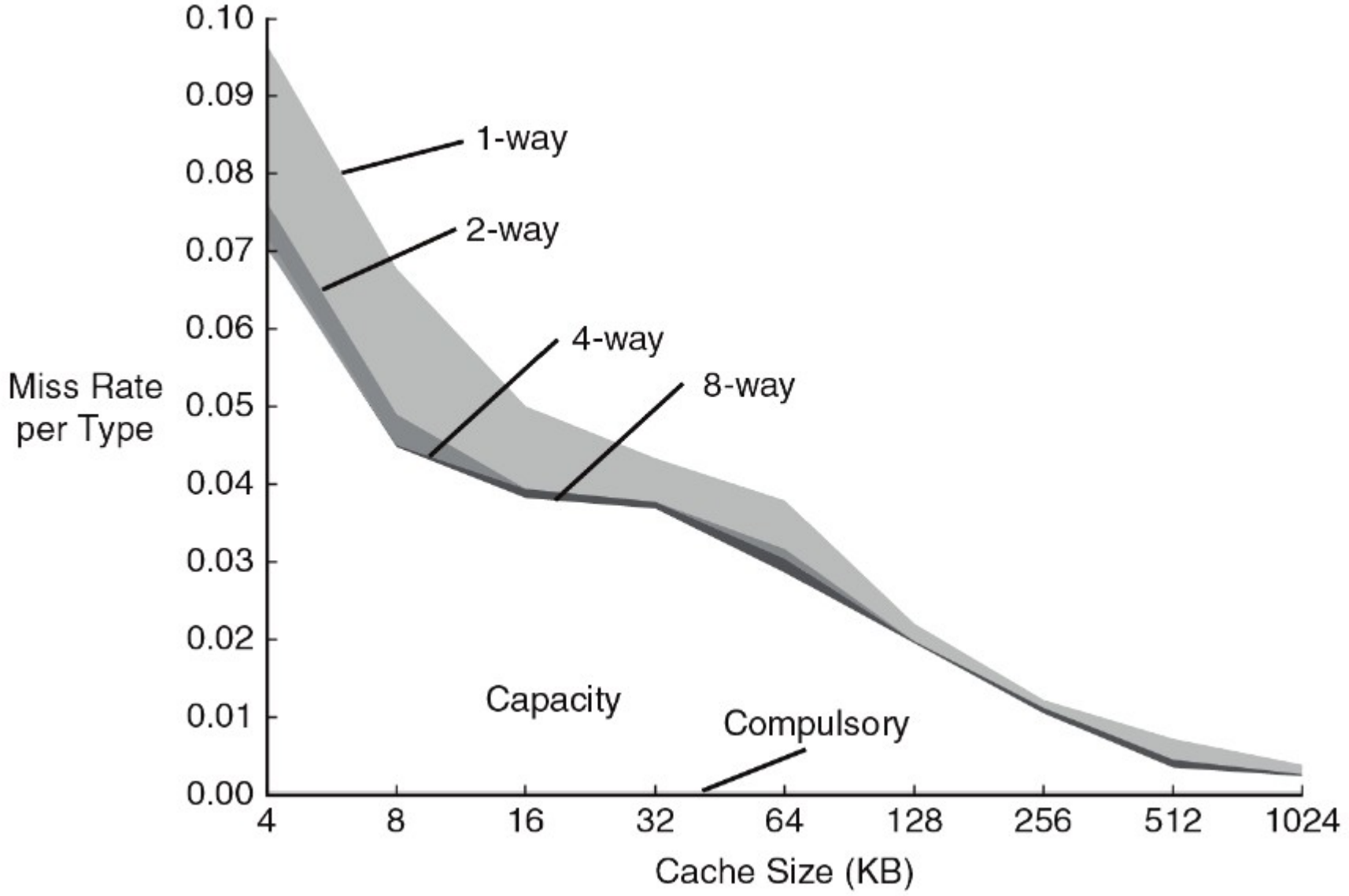
N-ассоциативный кеш

- N-секционный наборно-ассоциативный кеш (N-way set associative cache)
- Кеш делится на наборы, каждый набор содержит N блоков
- Каждый адрес памяти отображается в строго определенный набор, но в наборе может попасть в любой блок

N-ассоциативный кеш

- Пусть размер блока – 16 байт, размер кеша – 256 байт, размер ОЗУ – 4 KiB: в кеше 16 блоков, в ОЗУ – 256 блоков;
- Кеш – 4-наборный, каждый набор – 4 блока
- Рассмотрим адрес XYZ, Y определяет местоположение в кеше: Y & 3 – номер набора; 6 битов адреса (XY >> 2) являются тегом в наборе
- Совпадение части X в адресах не означает конфликта
- Меньше промахов типа conflict miss

Оценка % промахов на SPEC2000

- 

Язык C

Этапы развития языка

- Разработка – 1970е
- 1978 – Kernighan, Ritche. The C Programming Language
- 1989 – ANSI C
- 1999 – стандарт ISO C99
- 2011 – стандарт ISO C11
- C и C++ развиваются параллельно и согласованно (например, memory model)

Ядро ОС и C++

- В основном, ядра операционных систем написаны на Си
- Symbian — практически полностью C++
- Windows, MacOS — драйвера можно разрабатывать на подмножестве C++

- Причины: 1) исторические — разработка началась в то время, когда C++ не было
- Объем кода — десятки млн. строк кода — затраты на перенос на C++

Ядро ОС и C++

- Технические причины: «Узкие места» C++
  - Исключения:
    - Либо накладные расходы на таблицы обработки исключений (раздувание кода)
    - Либо накладные расходы на поддержку стековых фреймов при работе (замедление работы)
    - «непрозрачные» передачи управления при работе
  - Динамическая память (STL):
    - Страницы памяти, занятые ядром, никогда не смогут использоваться в приложениях — требуется контроль за использованием памяти
  - «Непрозрачная» генерация кода

Реализации Си

- Freestanding реализация – поддерживается ограниченный набор заголовочных файлов и стандартных функций (например, memcpy)
  - Для ядер операционных систем
  - Для встроенных систем (embedded) без управления ОС
- Hosted реализация – полный набор (возможно, кроме опциональных) заголовочных файлов и библиотечных функций
  - Программирование на уровне пользовательских программ ОС

Стандартная библиотека Си (hosted в Linux)

- В Unix системах традиционно называется libc, является частью ОС
- Заголовочные файлы размещаются в /usr/include
- Бинарный динамически загружаемый файл: /lib/libc.so.6 (Linux)
- Помимо функций библиотеки Си содержит и функции POSIX и расширения
- Библиотека математических функций отдельно – libm – требуется опция -lm при компиляции

Взаимодействие программы на Си с окружением

- Стандартные потоки ввода и вывода stdin, stdout, stderr
- Аргументы командной строки
- Переменные окружения
- Код завершения программы

Обработка ошибок

- Библиотечные функции и системные вызовы в случае ошибки возвращают специальное значение (например, fopen возвращает NULL, часто возвращается -1)
- В этом случае переменная errno содержит код ошибки, например, EPERM, EAGAIN
- Переменная errno и коды ошибок определены в <errno.h>
- strerror из <string.h> возвращает строку, соответствующую ошибке
- Сообщения об ошибках должны выводиться на stderr

Взаимодействие со средой

- Процесс завершается системным вызовом _exit(exitcode)
- Или возвращаемое значение return из main
- Значение в диапазоне [0;127] — код завершения процесса, он доступен процессу-родителю
- Код 0 — успешное завершение (/bin/true)
- Ненулевой код — ошибка (/bin/false)

Аргументы командной строки

- Функция main получает аргументы командной строки:

  int main(int argc, char *argv[])

- argv — массив указателей на строки Си

  ./prog foo 1 bar

  argv[0] →"./prog"; путь к программе
  argv[1] →"foo";
  argv[2] →"1";
  argv[3] →"bar";
  argv[4] →NULL;

- Передаются на стеке процесса

argv[0]

- Обычно argv[0] – путь, использованный для запуска программы
- Некоторые программы анализируют argv[0] и модицифируют свое поведение (например, busybox)

Переменные окружения

- Именованные значения доступные процессу
- По умолчанию передаются неизменными порождаемым процессам

  char *getenv(const char *name);
- В процесс передаются на стеке
- Глобальная переменная environ содержит указатель на массив переменных