Конструирование ядра операционных систем (VII)

Виртуальная память (продолжение)

План

- Актуальные архитектуры процессоров
- Конвейеризация в процессоре
- Атаки по побочным каналам

Архитектуры набора команд ЦПУ

- RISC — Reduced Instruction Set (ARC, ARM, MIPS, Power…):
  - Одинаковый размер инструкций (e.g. 4 байта), есть pipeline.
  - Не используют микрокод (hardwired control unit).
- CISC — Complex Instruction Set (x86, Zilog Z80):
  - Переменный размер инструкций, есть pipeline.
  - Используют микрокод (microprogram control unit).
- VLIW — Very Long Instruction Word (Elbrus, Itanium):
  - Явное параллельное исполнение инструкций, pipeline опционален.
- Экзотика в разных DSP (Digital Signal Processor):
  - Dataflow architecture, multiply-accumulate operations, etc.

Классический RISC конвейер

- Глубина конвейера — количество стадий (5).
- Ширина конвейера — количество юнитов на стадии (1).

IF — Instruction Fetch
ID — Instruction Decode
EX — Execute
MEM — Memory Access
WB — Reg. Write Back

Такт / № инст.	1	2	3	4	5	6	7
1	IF	ID	EX	MEM	WB		
2		IF	ID	EX	MEM	WB	
3			IF	ID	EX	MEM	WB
4				IF	ID	EX	MEM
5					IF	ID	EX

Конфликты конвейера

- Конфликты (зависимости) по данным:
  - Read after Read ☺
  - Read after Write
  - Write after Read
  - Write after Write
- Структурные конфликты:
  - Отсутствие свободного юнита (например, АЛУ)
- Конфликты управления:
  - Отсутствие информации об инструкции (ветвление, указатели).

Задержки при обращении к данным

- L1 менее 1.5 нс (0.8 для Intel Comet Lake, 2020)
- L2 менее 5 нс (2.4 для Intel Comet Lake, 2020)
- L3 менее 20 нс (11.6 для Intel Comet Lake, 2020)
- RAM 30-100 нс (ближе к 40 нс для 16 GB и 80 нс для 64 GB)*
- NVMe SSD более 7-150 мкс (ближе к 50 мкс)
- HDD 1-20 мс (ближе к 10 мс)

* Задержки обращения к RAM при трансляции виртуальных адресов компенсируются отдельным кэшем TLB (Translation Lookaside Buffer). Обычно они полностью ассоциативны и имеют несколько уровней. Количество записей TLB зависит от размера страницы (e.g. L1 128/14 для 4K/2M и L2 512 для 4K на Intel Nehalem).

Современные конвейеры

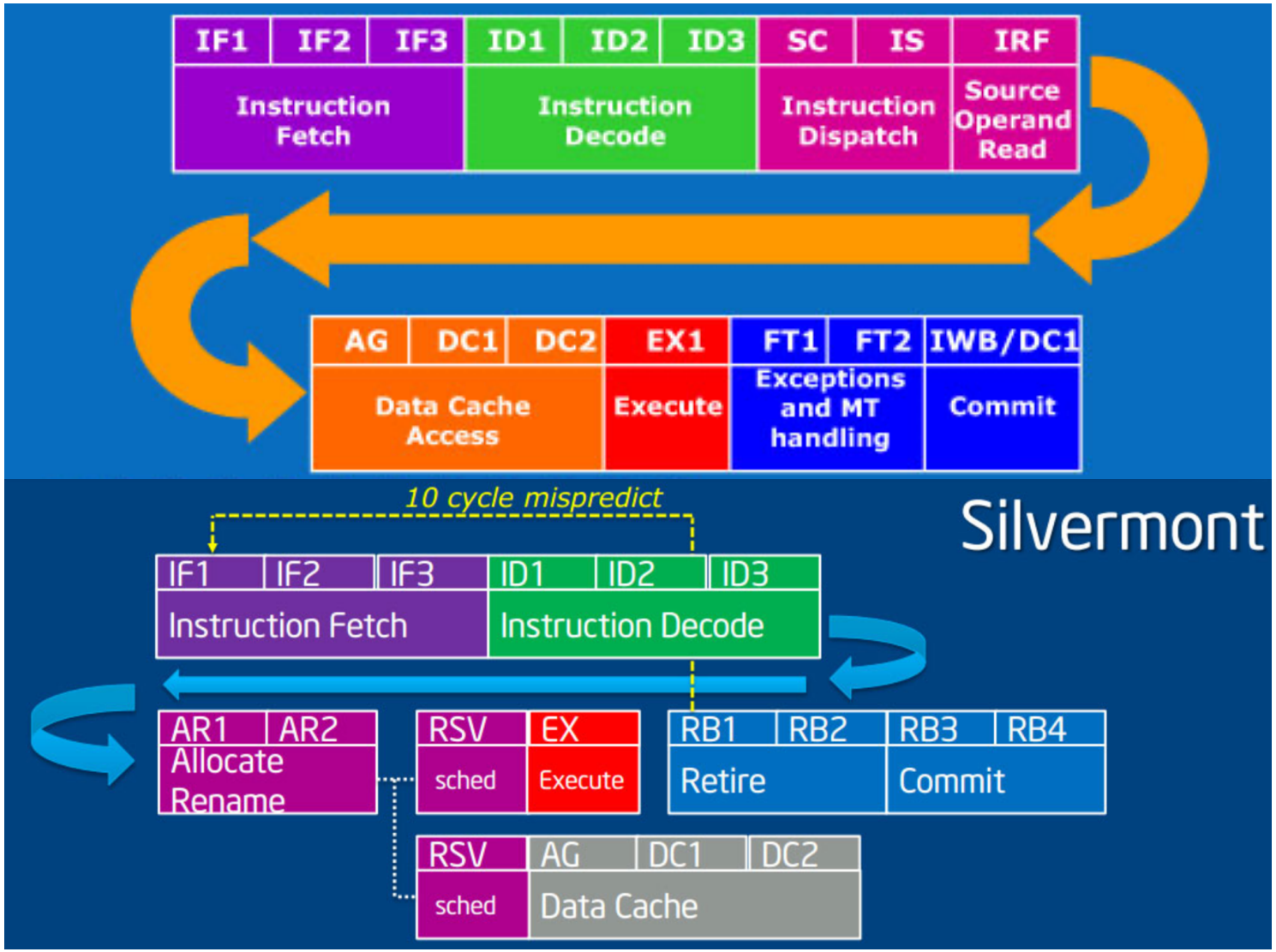
- Используют переменное количество микроопераций для инструкций (верно даже для RISC архитектур вроде ARM).
- Переименовывают регистры в микрооперациях.
- Применяют out-of-order исполнение инструкций для разрешения конфликтов без остановки конвейера.
- Speculative (branch/value prediction) исполнение инструкций вместо остановки конвейера при конфликте управления (192 микроинструкции для Intel Haswell).

Конвейеры Intel (2008, 2013)

1. Instruction Fetch
2. Instruction Decode
3. Instruction Dispatch
4. Operand Fetch
5. Execute
6. Handle exceptions
7. Commit

- 14-19 depth
- 4-5 width
- 6-10 ports (ALU, Read, Write)

[Intel Architecture Day 2018 (Sunny Cove Architecture)]



Атаки по побочным каналам

Атака по побочному каналу — любая атака на систему, в основу которой положено извлечение знаний за счёт особенностей внутреннего устройства системы, а не наличия слабых сторон в её непосредственном алгоритме работы (ошибок в ПО, уязвимостей криптопримитивов, etc.).

- Атаки по времени (e.g. TOCTOU, memcmp при сравнении хэша)
- Атака на остаточные данные (e.g. Cold Boot)
- DFA (Differential fault analysis) атаки (e.g. аппаратные глитчи, row hammer)
- Атаки по энергопотреблению
- Акустические атаки
- Атаки по ЭМИ
- Атаки на кэширующие элементы (e.g. Spectre, Meltdown)

ЭМИ атака из прошлого

Пассивный резонатор, установленный в посольстве США (1945-1952), модулирующий речь при подаче высокочастотного импульса.

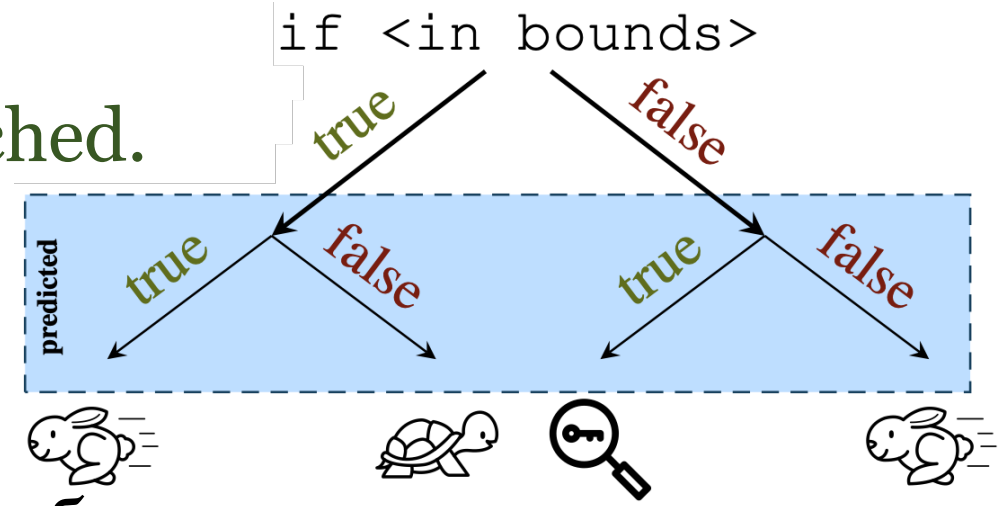
Атаки по кэшам

В основном направлены на несанкционированное получение доступа к данным через особенности устройства конвейера, кэшей, организации доступа к данным, шин и прочих компонентов для повышения производительности.

Современные уязвимости (e.g. Spectre, Meltdown) в основном направлены на чтение чужой оперативной памяти.

Уязвимость Spectre

```
uint32_t array_size = ARRAY_SIZE; // uncached.
uint8_t array1[ARRAY_SIZE]; // k = array1[x] — cached.
uint8_t sensitive_data[N];
uint8_t array2[1024*1024]; // uncached.
// Значение x контролируется.
if (x < array1_size)
   y = array2[array1[x] * 4096];
```



1. Передаём значение x, которое необходимо прочесть.
2. Передаём значение x’ менее array1_size.
3. Если k и k’ совпали, то время выполнения сильно меньше.

Защита от Spectre

- Разделение процессов для защиты доступа к чужой памяти.
- Отключение спекулятивного выполнения ☺
- Обратимые преобразования на операции (e.g. XOR со случайным значением) при вычислении адреса.
- Установка барьеров (e.g. lfence) для сброса кэшей в коде.
- Ограничения по времени (e.g. случайные задержки).
- Аппаратная защита (например, трекинг значений).

REF: Lab7-SPCT.pdf

Уязвимость Meltdown

```
raise_exception(); // Строка ниже никогда не вызывается.
// kaddr указывает на байт в памяти ядра.
access(probe_array[(*kaddr) *4096]); // access читает из RAM.
```

1. Выбираем значение kaddr, которое необходимо прочесть.
2. Вызываем код выше и позволяем процессору спекулятивно прочесть *kaddr и (незакешированный) probe_array[...].
3. Происходит исключение и завершение потока.
4. В другом потоке замеряем время доступа по k от [0, 255] к probe_array[k*4096]. Время будет наименьшим у k = *kaddr.

Защита от Meltdown

- Отображение ядерного и пользовательского кода в разных адресных пространствах.
- Аппаратная защита (спекулятивный доступ не должен обходить права доступа к памяти).

REF: Lab7-MLTD.pdf

Спасибо за внимание!

Вопросы?