FORESHADOW

Speculative Execution Attack

용어 정의

- Speculative Execution (= 예측 실행)

  최신 프로세서에서 효율적으로 작업을 수행하기 위한 기술

  다음에 수행할 연산을 미리 수행하는 것

Foreshadow

정의
인텔 프로세서에 탑재된 기술인 “예측 실행”을 이용한 공격

대상
개인 컴퓨터에 저장된 데이터, 클라우드에 저장된 데이터

버전
기본 버전, Next Generation (NG) 버전

Foreshadow

기본 버전 대상

- SGX Enclave (인클레이브)

NG 버전 대상

- 가상 머신
- 하이퍼바이저
- OS 커널 메모리
- 시스템 관리 모드 메모리

Foreshadow

기본 버전 대상

- SGX Enclave (인클레이브)

NG 버전 대상

- 가상 머신
- 하이퍼바이저
- OS 커널 메모리
- 시스템 관리 모드 메모리

용어 정의

- Software Guard eXtension (=SGX)
  최신 인텔 프로세서에 탑재된 기술

  공격자가 전체 시스템을 제어할 수 있는 상황에도 안전하게 사용자 데이터를 보호할 수 있는 기술

- Enclave (=인클레이브)
  SGX 구성 요소, 사용자 데이터의 안전을 보장할 수 있는 영역

Foreshadow

“예측 실행”을 이용하여..

1. 인클레이브 내부의 데이터 획득
   → 기밀성 공격
2. 장치가 가진 키 획득
   →무결성 공격 (키를 이용한 데이터 변조 가능)

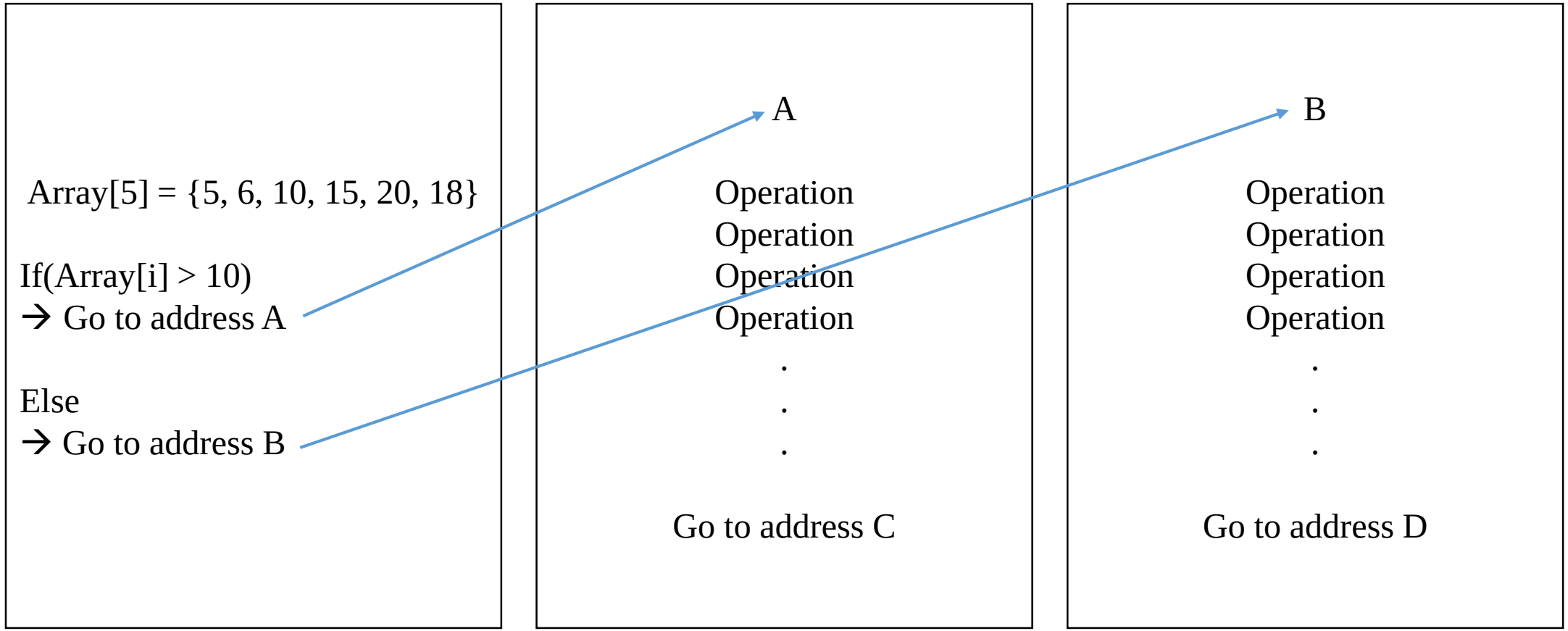
예측 실행 (Speculative Execution)

- 최신 프로세서에 탑재된 기술

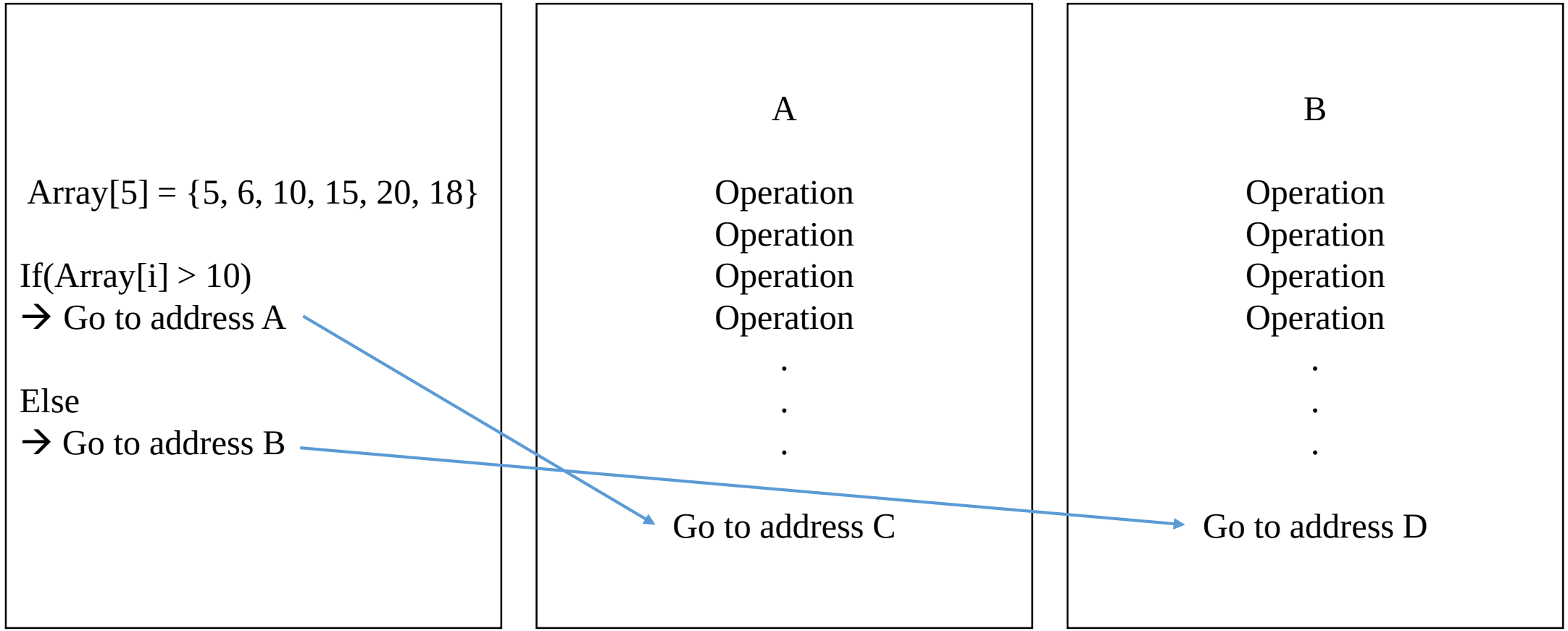
- 이후에 실행될 명령어를 예측하여 미리 계산
  → 옳은 예측 : 계산된 결과 사용
  → 틀린 예측 : 계산된 결과 버림, 제대로 된 계산 실행

- CPU 성능 최적화

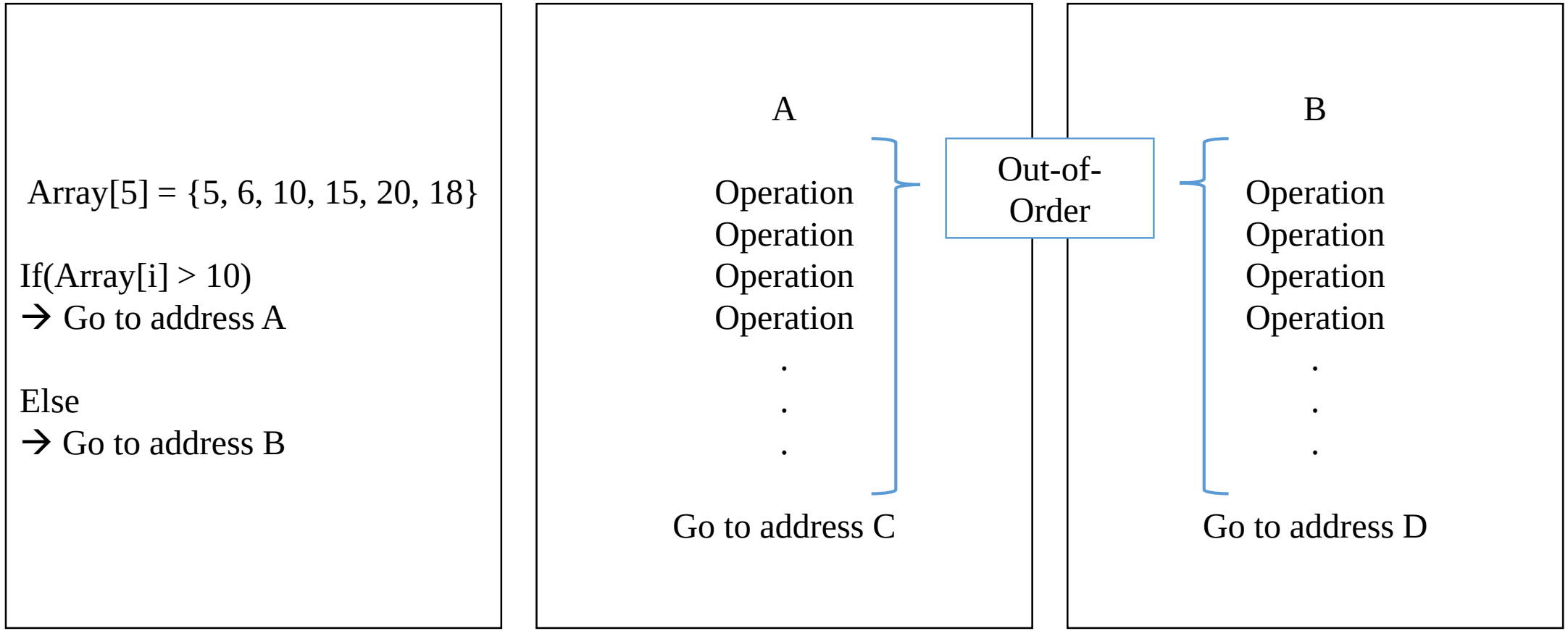
예측 실행 (Speculative Execution)



예측 실행 (Speculative Execution)



예측 실행 (Speculative Execution)



예측 실행 (Speculative Execution)

Array[5] = {5, 6, 10, 15, 20, 18}

If(Array[i] > 10)
→ Go to address A(C or D)

Else
→ Go to address B(C or D)

1. 정적 분기 예측
→ 정해진 규칙에 따라 예측

2. 동적 분기 예측
→ 최근 실행 기록을 기반으로 훈련된 예측
(보통 바로 전 실행을 따름)

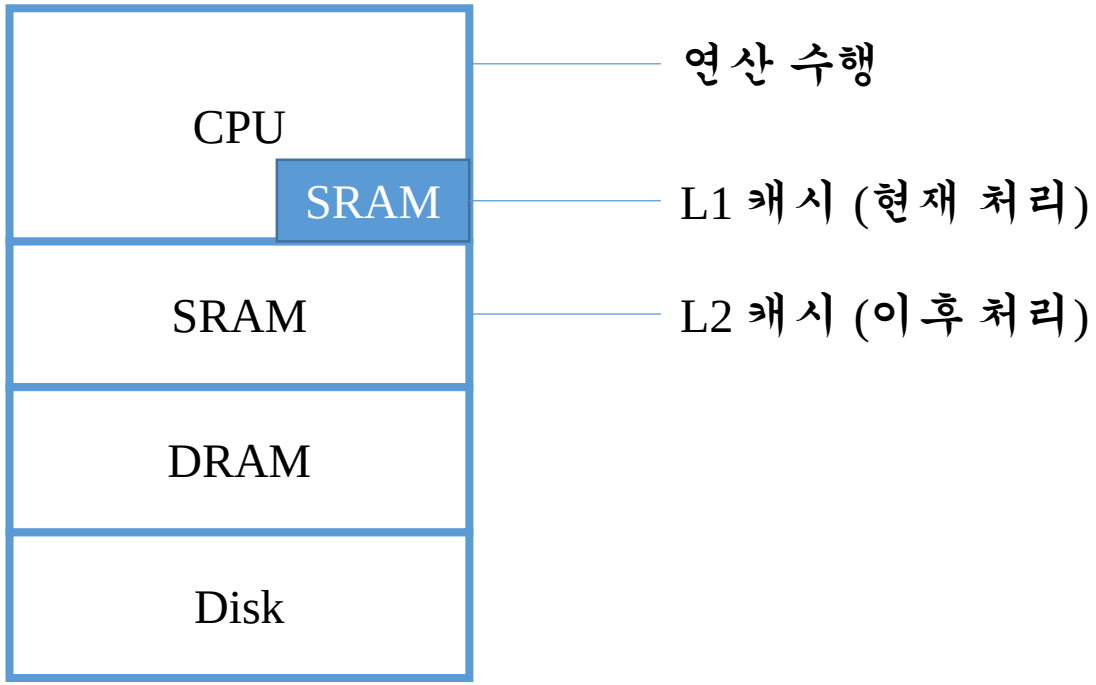
예측 실행 (Speculative Execution)

- 예측 실행 공격 (Speculative Execution Attack)
  대표적인 공격 기법 : Meltdown, Spectre

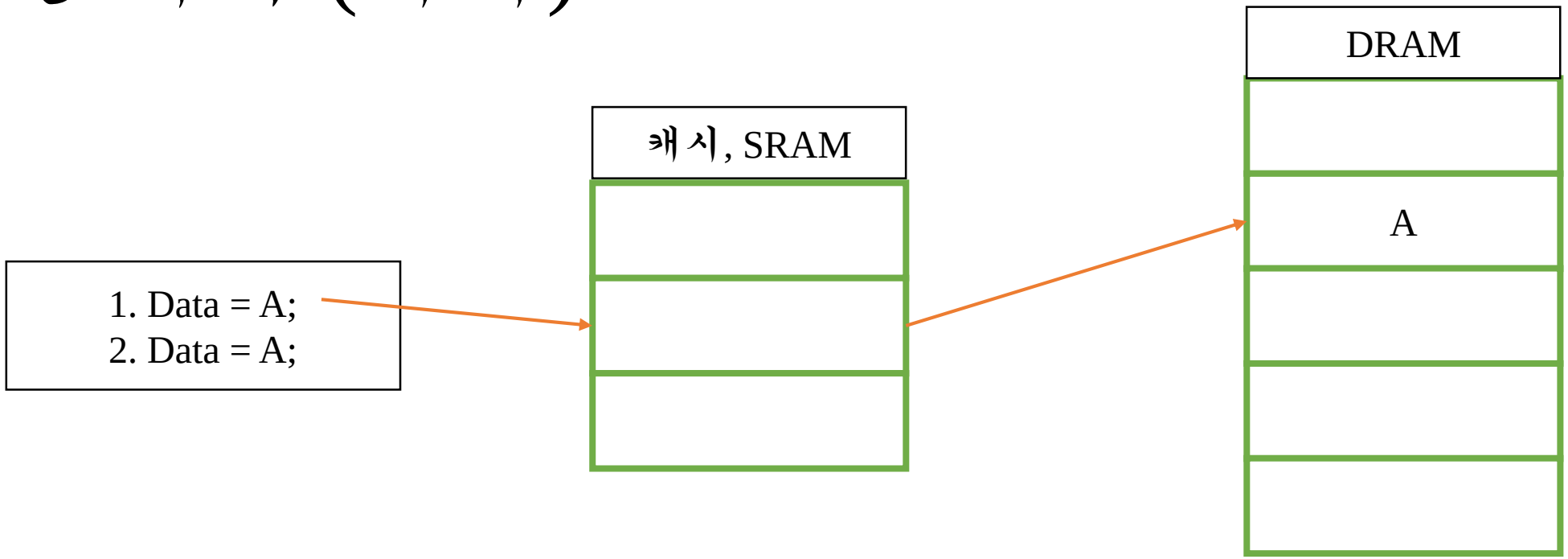
- Spectre, Meltdown 에서는 어떻게 예측 실행을 이용했는가?

사전 지식 (캐시)

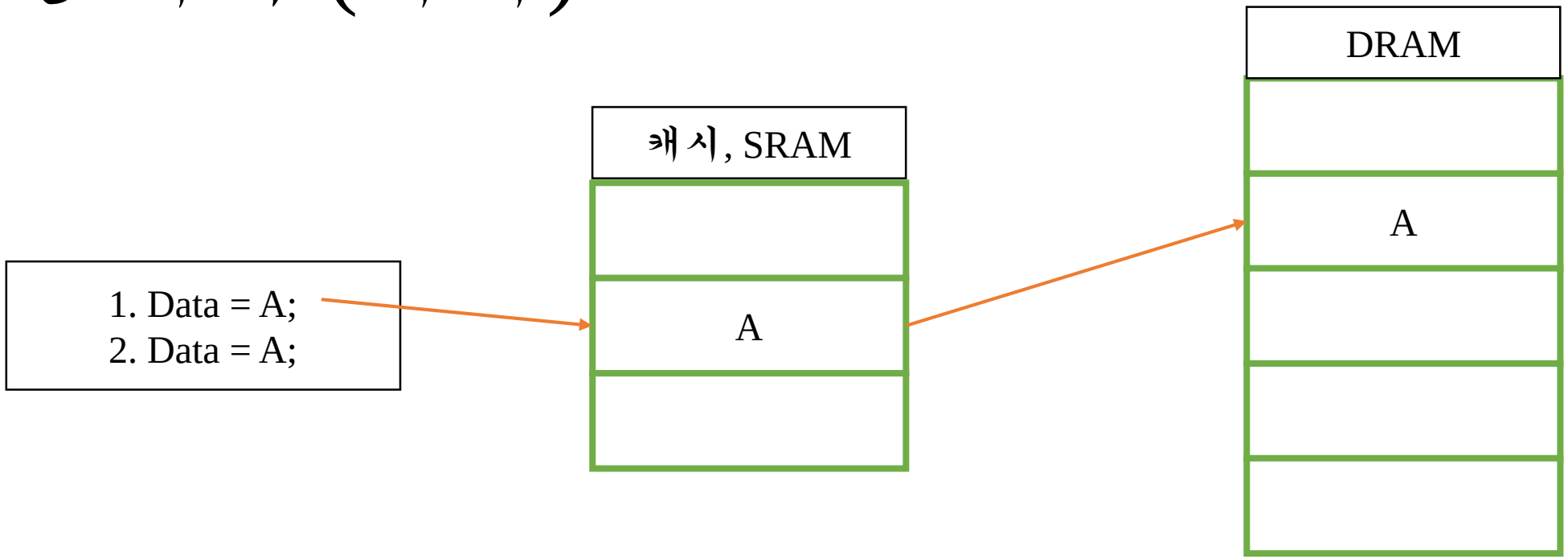
CPU와 가까울수록
- 속도 빠름
- 용량 작음
- 가격 비쌈



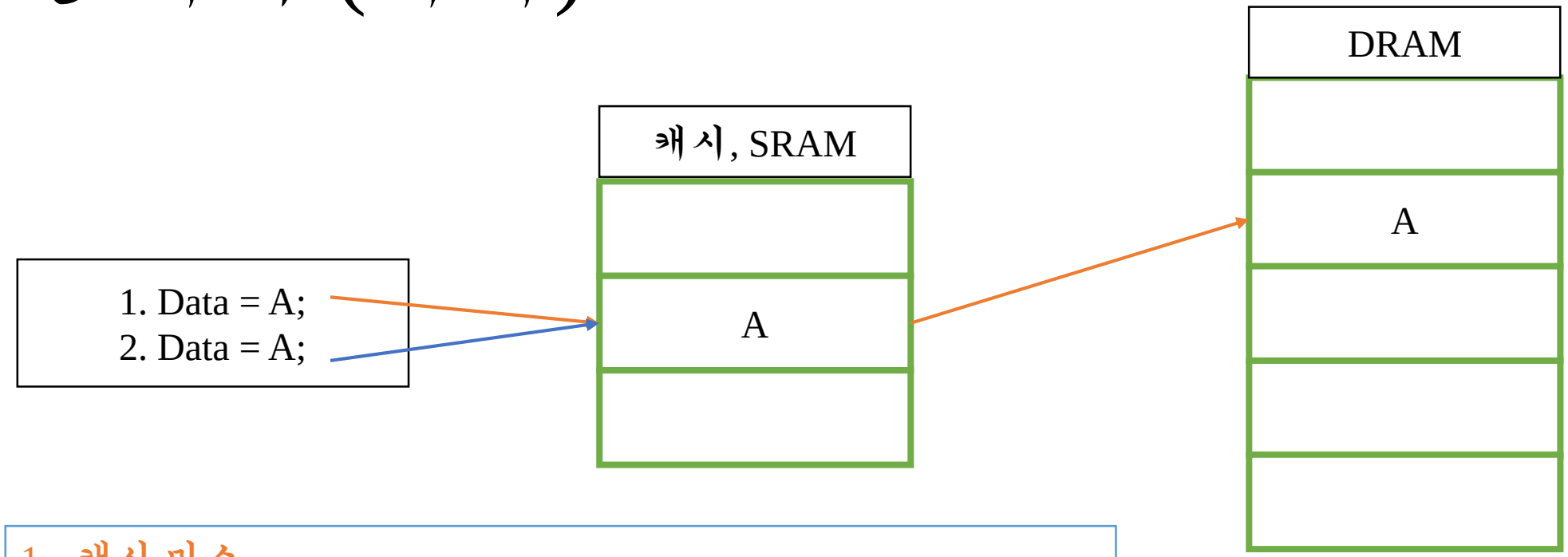
사전 지식 (캐시)



사전 지식 (캐시)



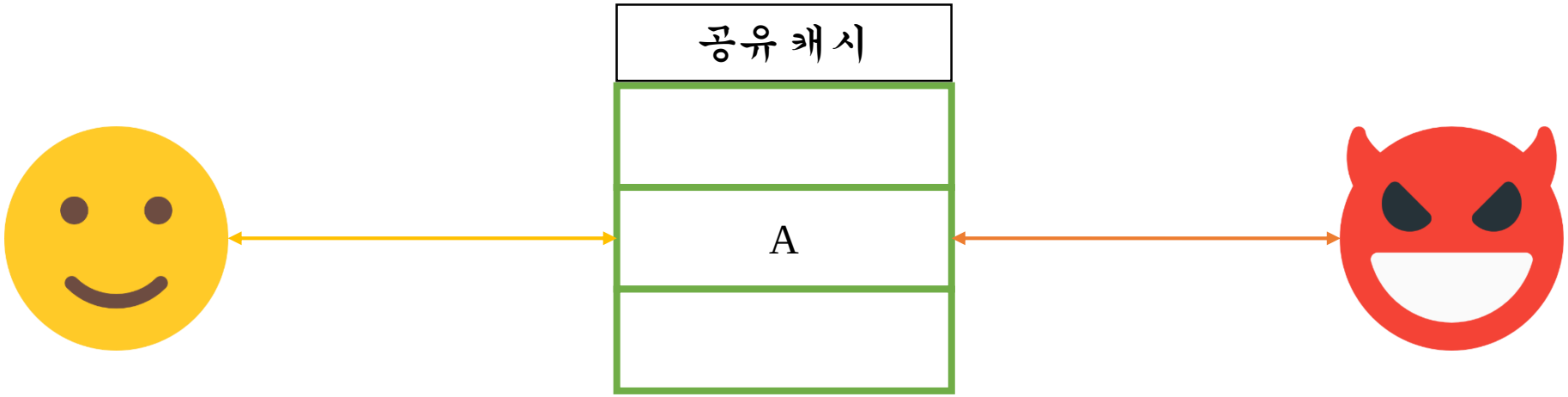
사전 지식 (캐시)



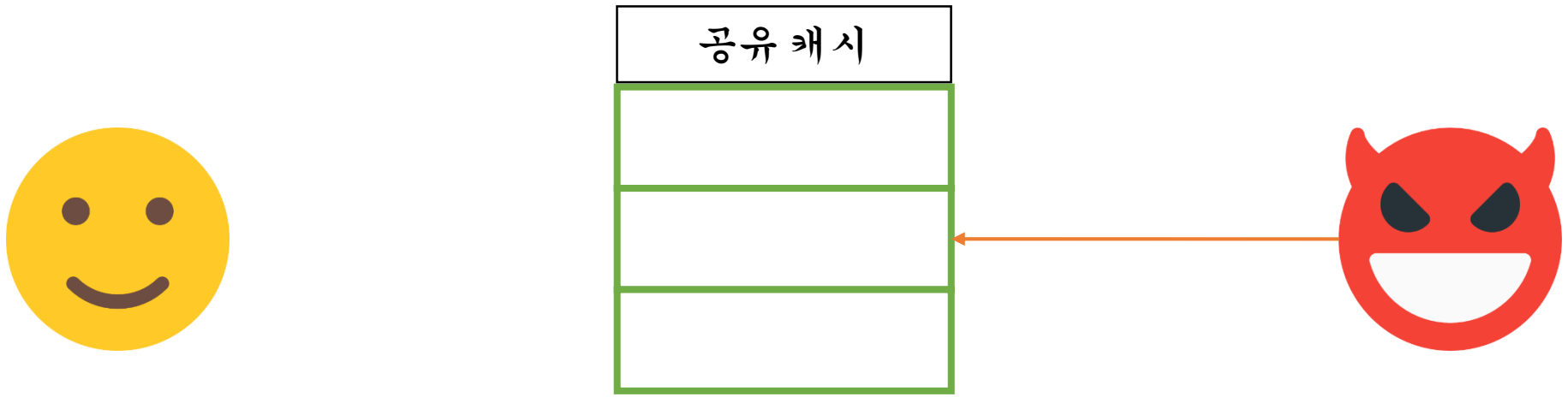
1. 캐시 미스
   - 캐시에 내용이 존재
   - 매우 빠른 속도

2. 캐시 히트
   - 캐시에 내용이 없음
   - 느린 속도

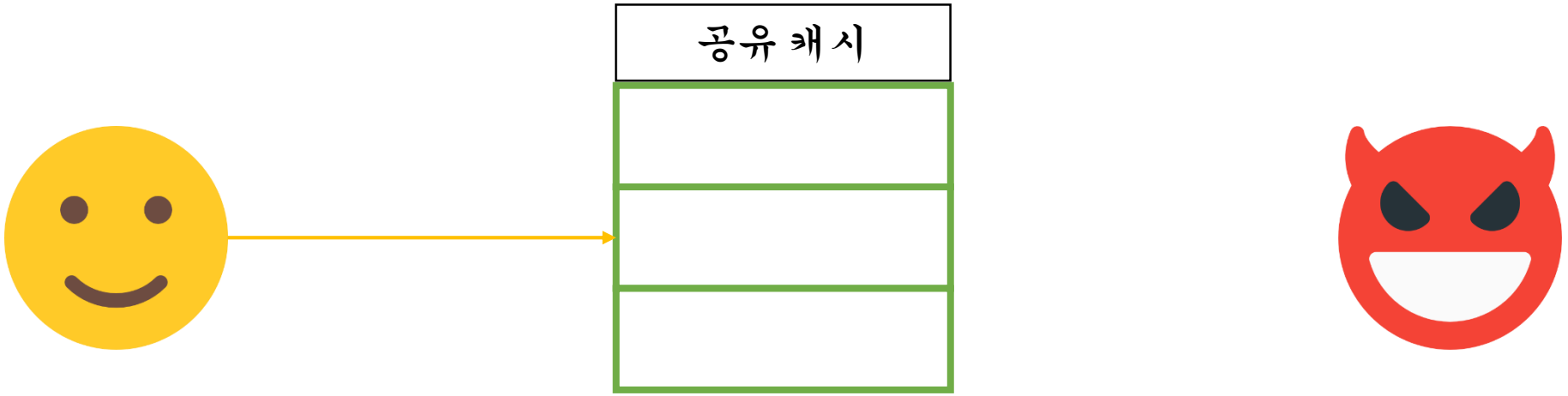
사전 지식 (Flush and Reload)



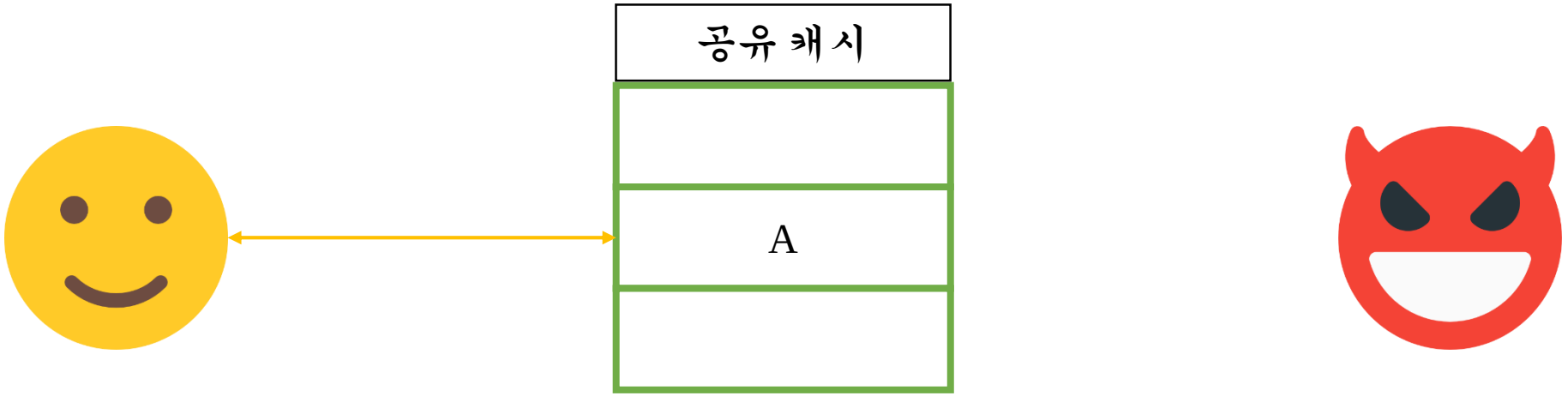
사전 지식 (Flush and Reload)



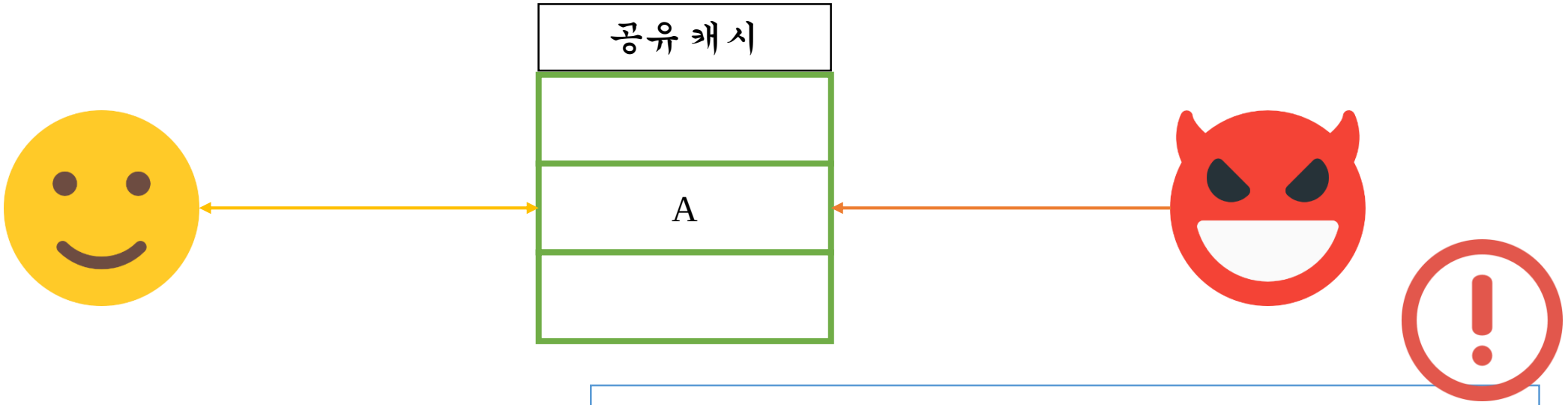
사전 지식 (Flush and Reload)



사전 지식 (Flush and Reload)



사전 지식 (Flush and Reload)



1. 캐시 미스
   피해자가 데이터에 접근하지 않음
2. 캐시 히트
   피해자가 데이터에 접근

Spectre (CVE-2017-5753)

- Bounds Check Bypass
  - 조건문 우회
  - 다음의 코드 실행 (Spectre 논문 예제 코드)

```
if (x < array1_size)
        y = array2[array1[x] * 4096];
```

  - x 는 공격자가 조작할 수 있는 데이터

Spectre (CVE-2017-5753)

1. x에 if문을 참으로 만드는 입력을 넣어 여러 번 수행
→ 동적 분기 예측 훈련 과정

2. array1의 길이를 넘어가는 입력을 수행
→ 참으로 예측하고 두 번째 줄 실행, 캐시에 적재

3. CPU는 실제 분기문을 검사하고 틀린 예측값은 버려짐
→ 캐시의 값은 버려지지 않음

```
if (x < array1_size)
        y = array2[array1[x] * 4096];
```

Spectre (CVE-2017-5715)

- Branch Target Injection
  - 특정 지점으로 이동 (Branch Target Buffer 내용 조작)
  - 분기된 곳에서 실행된 정보는 캐시에 올라오고 이를 통해 데이터 획득

Branch Target Buffer (BTB) : 분기문 실행 뒤 이동 될 곳을 저장하는 버퍼

Meltdown (CVE-2017-5754)

- Rogue Data Cache Load
  - 시스템 메모리 내용 탈취
  - 인텔 프로세서에 대해서만 영향
    - 다른 프로세서는 권한에 대한 문제가 있을 시 예측 실행을 하지 않음
  - 개념
    - 커널 주소 읽기 시도
    - 권한을 체크하기 전 예측 실행
    - 체크 뒤 캐시 내용이 남아있음

Foreshadow

- SGX는 “Abort page semantics”라는 기술을 가짐
  - 허가되지 않은 사용자의 접근에 대해서 0xFF 값을 반환
  - Heap, Stack, Code, Data 모든 영역에 적용
  - 예외 처리 타입에 관계없이 적용
- 이를 통해, 예측 실행과 관련된 공격을 차단

Abort page semantics:

An attempt to read from a non-existent or disallowed resource returns all ones for data (abort page). An attempt to write to a non-existent or disallowed physical resource is dropped. This behavior is unrelated to exception type abort (the others being Fault and Trap).

Intel Whitepaper : https://download.01.org/intel-sgx/linux-2.4/docs/Intel_SGX_Developer_Reference_Linux_2.4_Open_Source.pdf

Foreshadow

- Foreshadow는 이러한 대응기법을 우회하는 방법
- L1 Terminal Fault 라고 명명
- CVE-2018-3615 취약점으로 등록

Foreshadow Background

- Intel SGX
  - Memory Isolation
  - Enclave Measurement
  - Architectural Enclaves

Foreshadow Background

- Intel SGX
  - Memory Isolation
  - Enclave Measurement
  - Architectural Enclaves

Memory Isolation

- 인클레이브
  - 가상 주소 공간
  - 물리 메모리는 하드웨어에 의해 완전히 분리

  → 인클레이브 메모리는 외부에서 접근 불가능

  → 인클레이브 메모리 관리는 시스템 소프트웨어가 함
- CPU
  - 주소 변환을 검증
  - 문제가 있을 시 page fault
  - 연속적인 주소 변환은 캐시 이용 → Translation Lookaside Buffer (TLB)

Memory Isolation

- TLB
  - 인클레이브 진입, 이탈 시 비워지는 캐시 공간
  - 외부 읽기 (0x FF 반환)
  - 외부 쓰기 (무시)
- Memory Encryption Engine (MEE)
  - CPU 캐시에 옮겨지기 전 인증, 복호화 과정을 거침

Memory Isolation

- 인클레이브 접근
  - 정해진 명령어를 통해 가능
  - EENTER, EEXIT (호스트 어플리케이션과 인클레이브 간의 제어 명령어)
  - 오류, 외부 인터럽트 발생 시 → AEX
- Asynchronous Enclave Exit (AEX)
  - 인터럽트 위치에서 CPU 레지스터 내용을 SSA에
  - CPU 레지스터를 지우고 제어권을 외부로 넘김
  - ERESUME 명령어를 통해 다시 인클레이브 복귀
  - SSA에 저장된 내용을 바탕으로 상태 복귀

State Save Area
- 인클레이브 상태 보관

Foreshadow Background

- Intel SGX
  - Memory Isolation
  - Enclave Measurement
  - Architectural Enclaves

Enclave Measurement

- 인클레이브 빌드 시
  - 해시값 생성
  - MRENCLAVE
    인클레이브 내용에 기반하여 신원값 생성
  - MRSIGNER
    개발자(소프트웨어 공급자)에 기반하여 신원값 생성
    개발자의 공개키, 버전 정보가 들어감
- 인클레이브 초기화 시
  - 해시값 비교
  - MRENCLAVE, MRSGINER 비교

→ 인클레이브 무결성 보장, + ~~(Attestation, Sealing)~~

Enclave Measurement

- CPU Processor
  - 플랫폼 마스터 키가 존재
  - 키 분배 하드웨어에 유일하게 접근 가능
  - 키 분배는 CPU 보안 버전, 랜덤한 KEYID에 의해 분배
  - EREPORT, EGETKEY 명령어를 통해 키 분배 이용

Enclave Measurement

- Attestation
  - 인클레이브 A : EREPORT 명령어를 통해 REPORT 생성
    REPORT(MRENCLAVE, MRSIGNER, 어플리케이션 데이터)
  - 인클레이브 B : EGETKEY 명령어를 통해 키 분배 실행
    REPORT 키를 통해 REPORT 검증
- Sealing
  - EGETKEY 명령어를 통해 SEALING 키 생성
  - 장기간 저장할 데이터를 암호화하여 외부에 저장
    (MRENCLAVE 기반, MRSIGNER 기반)

Foreshadow Background

- Intel SGX
  - Memory Isolation
  - Enclave Measurement
  - Architectural Enclaves

Architectural Enclaves

- 몇몇 기능을 하드웨어에서 구현하기에 비효율적
- 인텔이 작성한 인클레이브
  - Launch Enclave
    플랫폼에서 실행 가능한 인클레이브를 결정
  - Provisioning Enclave
    플랫폼 Attestation 키 공급에 이용
  - Quoting Enclave
    Remote Attesttation에 사용
- Debug/ Release 모드
  - 전용 디버거로 디버그 모드 인클레이브 조사 가능
  - Release 모드만이 배포용으로 이용 가능

Foreshadow

- 권한
  - 사용자 권한 공격자
  - 커널 권한 공격자
    - 공격 최적화 가능
- 가정
  - 부채널 취약점 X
  - 이용가능한 코드 X
  - 인클레이브 코드 X
  - 인클레이브 실행 O (인클레이브 주소 공간에 비밀이 있다면 X)
  - 캐시 이용 O

Foreshadow

- 사용자 권한
  - 캐시에 비밀이 올라옴
  - 인클레이브 실행 시 비밀 가져오기

- 커널 권한
  - 페이징 명령을 활용하여 데이터 가져옴
  - 인클레이브 실행 X
  - 전체 메모리를 가져올 수 있음

→ 기밀성 훼손

Foreshadow

- 키 추출 가능
  - Sealing Key
    - Sealed 데이터들의 기밀성, 무결성 훼손 가능
  - Report Key
    - Attestation 과정 훼손 가능

→ 무결성 훼손

Instruction Pipeline

- 복잡한 명령어 집합에서 각 명령어들은 디코딩 시 더 작은 명령어(Micro-operations)로 쪼개진다.
  - 프로세서의 효율적 설계 가능
  - 결함 발생 시 수정 가능
  - Superscalar 프로세서 최적화 기법 사용 가능

Superscalar 프로세서
한 주기에 하나 이상의 명령어를 수행 가능한 프로세서

Instruction Pipeline

- 병렬화 (3 단계)
  - Fetch-decode
    - 명령어 적재, 마이크로 연산으로 변환

    → 명령어를 미리 적재하여 효율성 높임
  - Scheduling
    - 마이크로 연산들은 실행 단위로 스케쥴링
    - 하이퍼 스레딩이 사용될 수 있음
  - Retirement
    - 레지스터, 메모리에 저장될 수 있는 형태로 변환

Hyper Threading
매 사이클마다 여유 자원에 연산을 할당하는 것, 마치 여러 개 코어가 작동하는 것으로 보임

Out-of-Order, Speculative Execution

- Out-of-Order Execution, Speculative Execution
  - 마이크로 연산을 순차대로 실행하는 것이 아니라 (in-order) 자원이 허용하는 대로 실행 (out-of-order)
  - 다음에 실행될 연산을 예측
  - 결과는 Reorder Buffer (ROB) 라는 공간에 저장 retirement 단계에서 사용
- 예측에 실패할 경우
  - 폐기하고 다시 실행 (pipeline, ROB 비움)

Cache in Intel Processor

- L1, L2 캐시 (하이퍼스레딩 코어 간 공유)
- L3 캐시 (물리적 코어 간 공유)
- 캐시 구성 단위 : 캐시 라인 (64 바이트)

Transient Execution Attacks

- 마이크로 연산이 예측되어 실행되고 이후에 사라지는 과정에서 의도치 않게 CPU 내부 상태(캐시)를 변환하게 됨 이러한 관계를 이용한 공격이 “Transient Execution Attack”

- 대표적인 공격이 Spectre, Meltdown

Foreshadow

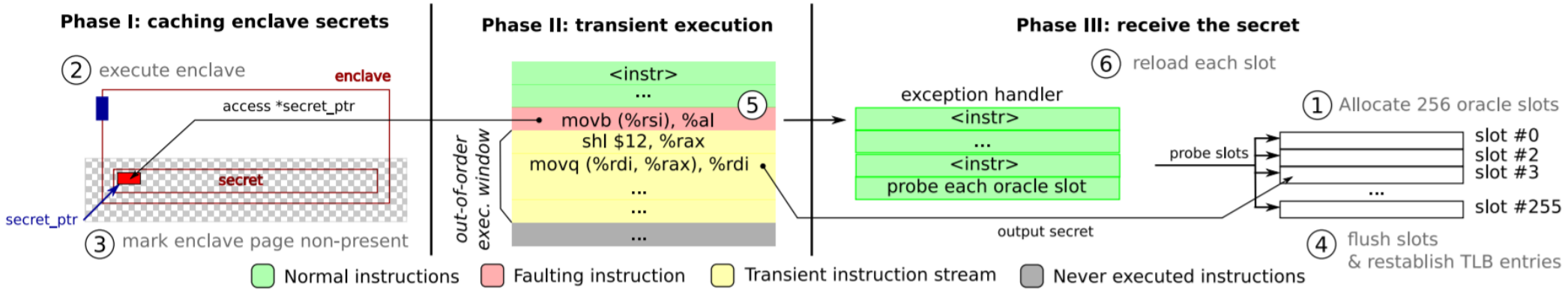
1. 캐시에 옮겨진 인클레이브 데이터 읽는 방법
2. 공격 최적화 기법
3. L1 캐시에 인클레이브 옮기는 방법

Foreshadow

① 256 개의 오라클 버퍼 할당

② 인클레이브 실행

이 과정에서 데이터가 평문의 형태로 캐싱
MEE 범위가 캐시를 포함하기 때문에, DRAM으로의 출입만이 암복호화 됨
즉, Transient execution을 이용할 수 있다는 뜻

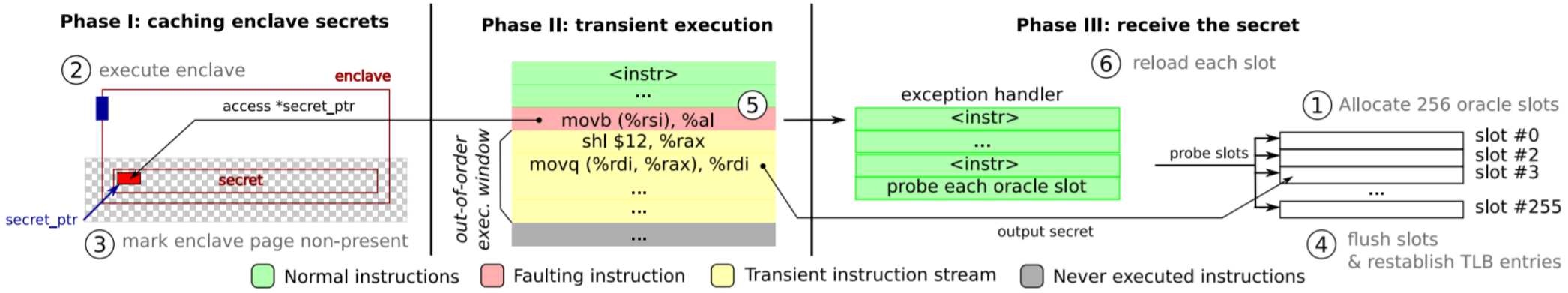


Foreshadow

Abort Page Semantics

역참조를 통한 데이터 읽기를 시도 → 불가능

페이지 테이블이 성공적으로 승인된 이후 APS 실행된다는 점 이용



부록 : 페이지 테이블

- 페이지 기본 주소
- 플래그 비트
  - 접근 비트
  - 변경 비트
  - 존재 비트(할당된 프레임이 있는지)
  - 읽기, 쓰기 비트

→ 로 구성된 물리 주소 매핑 테이블

Foreshadow

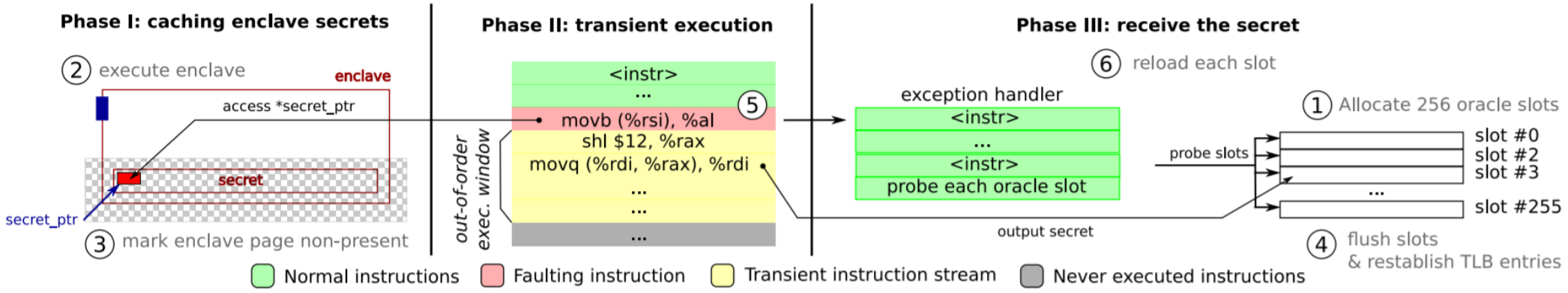
Abort Page Semantics

역참조를 통한 데이터 읽기를 시도 → 불가능

페이지 테이블이 성공적으로 승인된 이후 APS 실행된다는 점 이용

③ mprotect(secret_ptr &~0xfff, 0x1000, PROT_NONE);

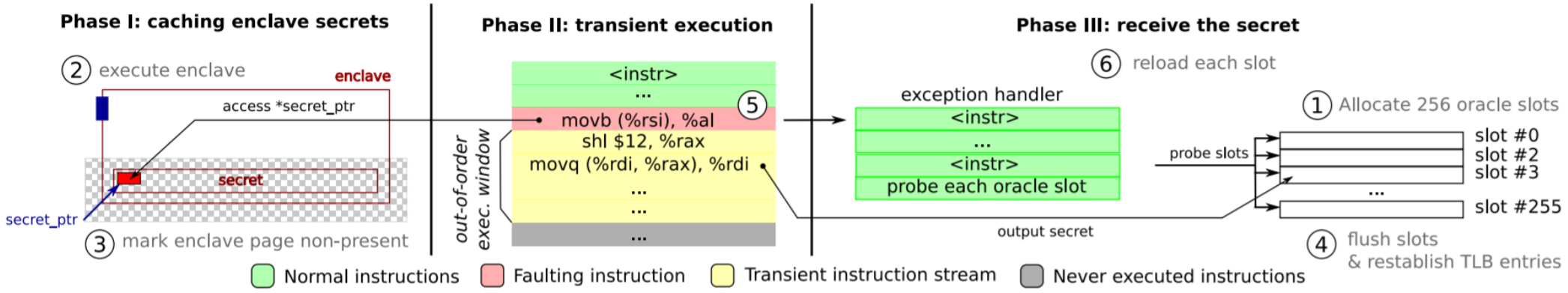
존재 비트를 0으로 만듦으로써 우회



Foreshadow

인클레이브 entry, exit 시 TLB 내용을 비움
→오랜 시간이 걸림, OoOE에 충분한 시간이 부족

④ clflush;
진입점 재설정, 진입점 캐시에서 삭제



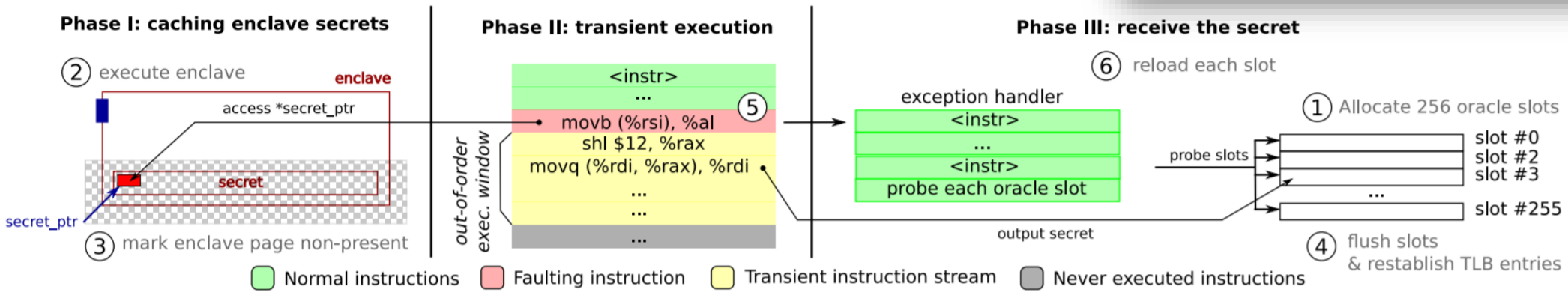
Foreshadow

⑤ Transient Execution 실행

오라클 버퍼로의 포인터, secret_ptr이 호출되면 5번째 줄에서 비밀을 읽는다.

6, 7은 OoOE에 의해 실행된다.

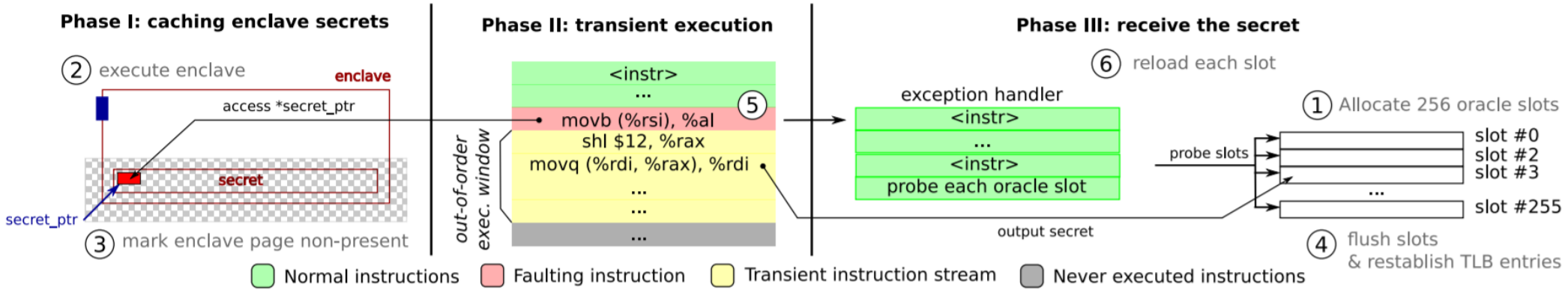
```
1 void foreshadow(
2     uint8_t *oracle,
3     uint8_t *secret_ptr)
4 {
5   uint8_t v = *secret_ptr;
6   v = v * 0x1000;
7   uint64_t o = oracle[v];
8 }
```



Foreshadow

페이지 폴트 발생 이후, 예외 핸들러가 발생한다.

⑥ Reload를 통해 각 슬롯 접근 시간을 측정한다.
캐시 히트가 일어날 시 접근 시간이 매우 빠를 것



Foreshadow 예제 코드 실행

조건

1. SGX 지원 프로세서
2. 커널
3.

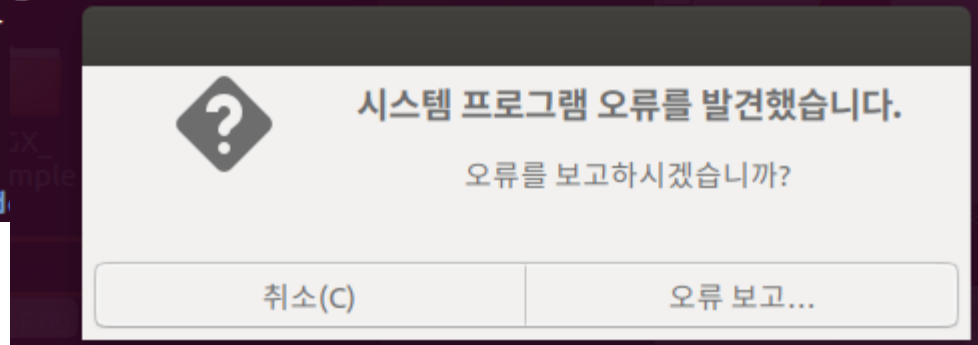
Model name	CPU	Base frequency	APIC timer interval
Skylake	i7-6700	3.4 GHz	19
Skylake	i7-6500U	2.5 GHz	25
Skylake	i5-6200U	2.3 GHz	28
Kaby Lake R	i7-8650U	1.9 GHz	34
Coffee Lake R	i9-9900K	3.6 GHz	21

```
파일(F)  편집(E)  보기(V)  검색(S)  터미널(T)  도움말(H)
vexyong@vexyong:~/FORESHADOW/sgx-step-master/app/foreshadow$ sudo ./app
[main.c] Creating enclave...
[sched.c] continuing on CPU 1
[pt.c] /dev/sgx-step opened!
==== Victim Enclave ====
    Base:   0x7f4871800000
    Size:   4194304
    Limit:  0x7f4871c00000
    TCS:    0x7f4871b7a000
    SSA:    0x7f4871b7bf48
    AEP:    0x7f4873973805
    EDBGRD: debug
[pt.c] /dev/mem opened!
[main.c] Randomly generated enclave secret at 0x7f4871a196c0 (page 0x7f4871a19000); alias at 0x7f4873db86c0 (revoking alias access rights)
+-----------------------------------------------------------------------------------------+
| XD | PK | IGN | RSVD | PHYS ADRS      | IGN | G | PAT | D | A | PCD | PWT | U/S | R/W | P |
| 0  | x  | x   | 0    | 0x000070542000 | x   | x | x   | 1 | 1 | x   | x   | 1   | 1   | 1 |
+-----------------------------------------------------------------------------------------+
+-----------------------------------------------------------------------------------------+
| XD | PK | IGN | RSVD | PHYS ADRS      | IGN | G | PAT | D | A | PCD | PWT | U/S | R/W | P |
| 0  | x  | x   | 0    | 0x000070542000 | x   | x | x   | 0 | 1 | x   | x   | 1   | 1   | 0 |
+-----------------------------------------------------------------------------------------+
[foreshadow.c] cache hit/miss=58/260; reload threshold=142

-------------------------------------------------------
[main.c] Foreshadow secret extraction
-------------------------------------------------------

[main.c] prefetching enclave secret (EENTER/EEXIT)...
[main.c] extracting secret from L1 cache..
명령어가 잘못됨
vexyong@vexyong:~/FORESHADOW/sgx-step-master/app/foreshad
```

1




```
enclave secret at 0x7f4ef1e196c0 (page 0x7f4ef1e19000); al
```

Model name	CPU	Base frequency	APIC timer interval
Skylake	i7-6700	3.4 GHz	19
Skylake	i7-6500U	2.5 GHz	25
Skylake	i5-6200U	2.3 GHz	28
Kaby Lake R	i7-8650U	1.9 GHz	34
Coffee Lake R	i9-9900K	3.6 GHz	21