CSE 5095 & ECE 4451 & ECE 5451 – Spring 2017

Lecture 3b

SGX Enclave Life Cycle Tracking TLB Flushes Security Guarantees

- Slide deck extracted from Kamran's tutorial on SGX and Chenglu's security analysis of SGX, both presented during ECE 6095 Spring 2017 on Secure Computation and Storage, a precursor to this course

Marten van Dijk

Syed Kamran Haider, Chenglu Jin, Phuong Ha Nguyen

Department of Electrical & Computer Engineering

University of Connecticut

UCONN

With the help of:

1. Intel SGX Tutorial (Reference Number: 332680-002) presented at ISCA 2015
2. "*Intel SGX Explained*", Victor Costan and Srinivas Devadas, CSAIL MIT

Outline

- SGX Enclave Life Cycle
- Tracking TLB Flushes
- SGX Security Properties
  - Misconceptions about SGX
  - Interaction with Anti-Virus Software

The Life Cycle of an SGX Enclave

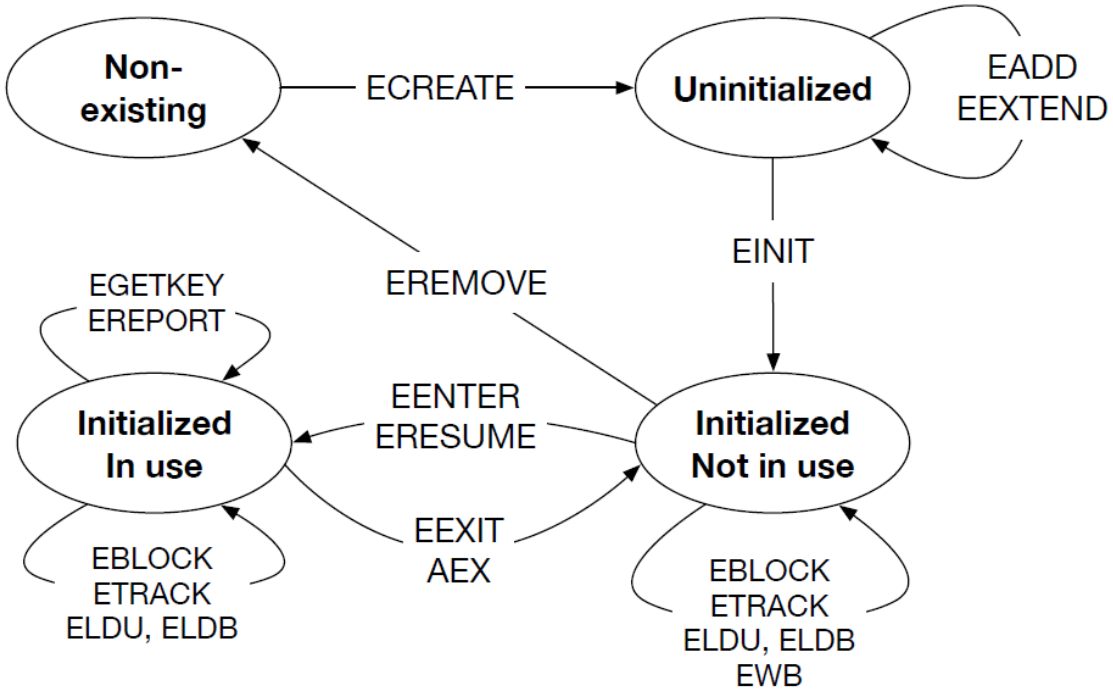
- Overview
- Relevant SGX Instructions
- Example

The Life Cycle of an SGX Enclave

An enclave's life cycle is all about the allocation of EPC pages.

Following are the major transitions during an SGX enclave's life cycle:

- Creation (ECREATE)
- Loading (EADD, EEXTEND)
- Initialization (EINIT)
- Enter/Exit the Enclave (EENTER/EEXIT)
- Teardown (EREMOVE)



Enclave Creation (ECREATE)

- Creates a unique instance of an enclave, establishes the linear address range, and serves as the enclave's root of trust
  - Enclave mode of operation (32/64)

- This information is stored within a Secure Enclaves Control Structure (SECS) generated by ECREATE.

Loading (EADD, EEXTEND)

EADD

- Add Regular (REG) or Thread Control Structure (TCS) pages into the enclave
  - System software responsible for selecting free EPC page, type, and attributes, content of the page and the enclave to which the page added to.
- Initial EPCM entry to indicate type of page (REG, TCS)
  - Linear address, RWX, associate the page to enclave SECS

EEXTEND

- Generates a cryptographic hash of the content of the enclave in 256Byte chunks
  - EEXTEND 16 times for measuring a 4K page

Initialization (EINIT)

- Verifies the enclave's content against the ISV's signed SIGSTRUCT and initializes the enclave – Mark it ready to be used
  - Validate SIGSTRUCT is signed using SIGSTRUCT public key
  - Enclave measurement matches the measurement specified in SIGSTRUCT.
- Enclave attributes compatible with SIGSTRUCT
- Record sealing identity (sealing authority, product id, SVN) in the SECS

Synchronous Enclave Entry (EENTER)

- Check that Thread Control Structure (TCS) is not busy and flush TLB entries for enclave addresses.
- Transfer control from outside enclave to pre-determined location inside the enclave
- Change the mode of operation to be in enclave mode
- Save RSP/RBP for later restore on enclave asynchronous exit
- Save XCR0 and replace it with enclave XFRM value

Synchronous Enclave Exit (EEXIT)

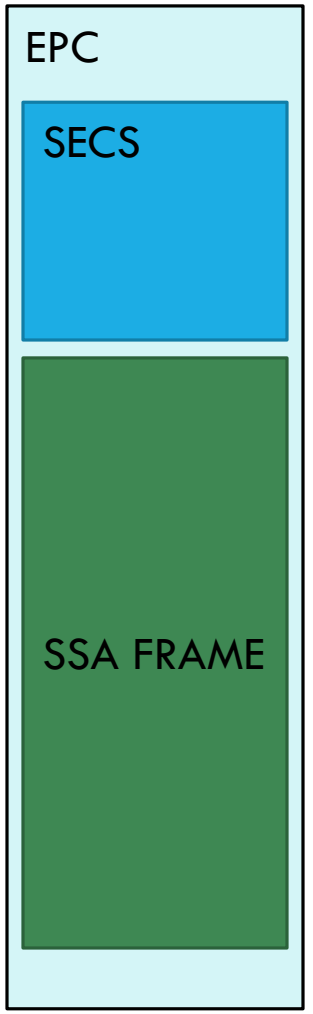
- Clear enclave mode and TLB entries for enclave addresses.
- Transfer control from inside enclave to a location outside the enclave
  - Mark TCS as not busy
- Responsibility to clear register state is on enclave writer!

Teardown (EREMOVE)

- EREMOVE deallocates/removes a 4KByte page permanently from the EPC
- A page cannot be removed until there is no thread executing code inside this enclave.
- A SECS page cannot be removed until all the regular pages of this enclave are removed.
- The SECS page is removed at the very last, and this also destroys the Enclave.

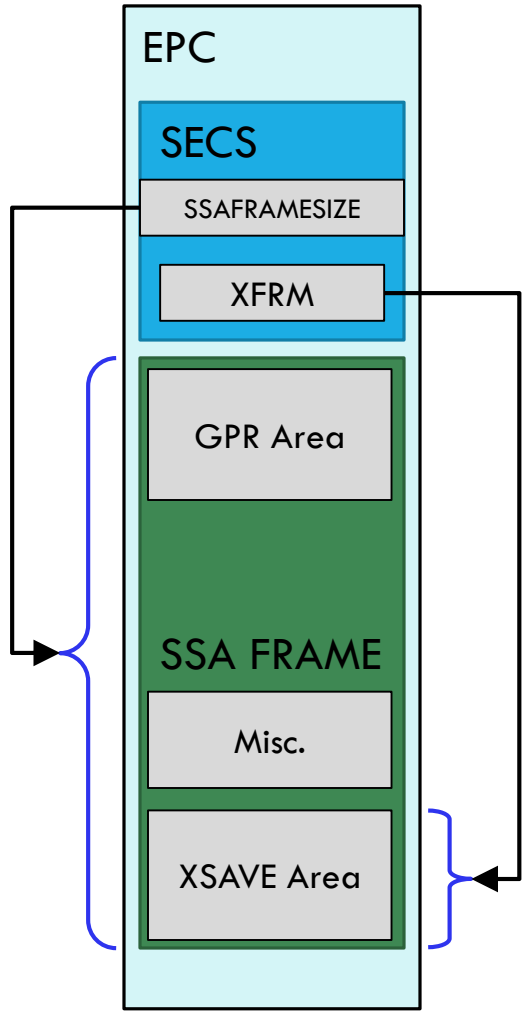
Context Switching/Exception Handling

- A context switch or exception (e.g. Interrupt) may occur during Enclave's code execution.
- Enclave's execution context must be stored
  - General Purpose Registers (GPRs)
  - Special Registers indicated by Requested-Feature BitMap (RFBM) register
- The area used to store an enclave thread's execution context while a hardware exception is handled is called **State Save Area (SSA)**
  - SSA is implemented by special EPC Page(s)
  - Notice that the saved context (i.e. SSA) is protected being inside the EPC

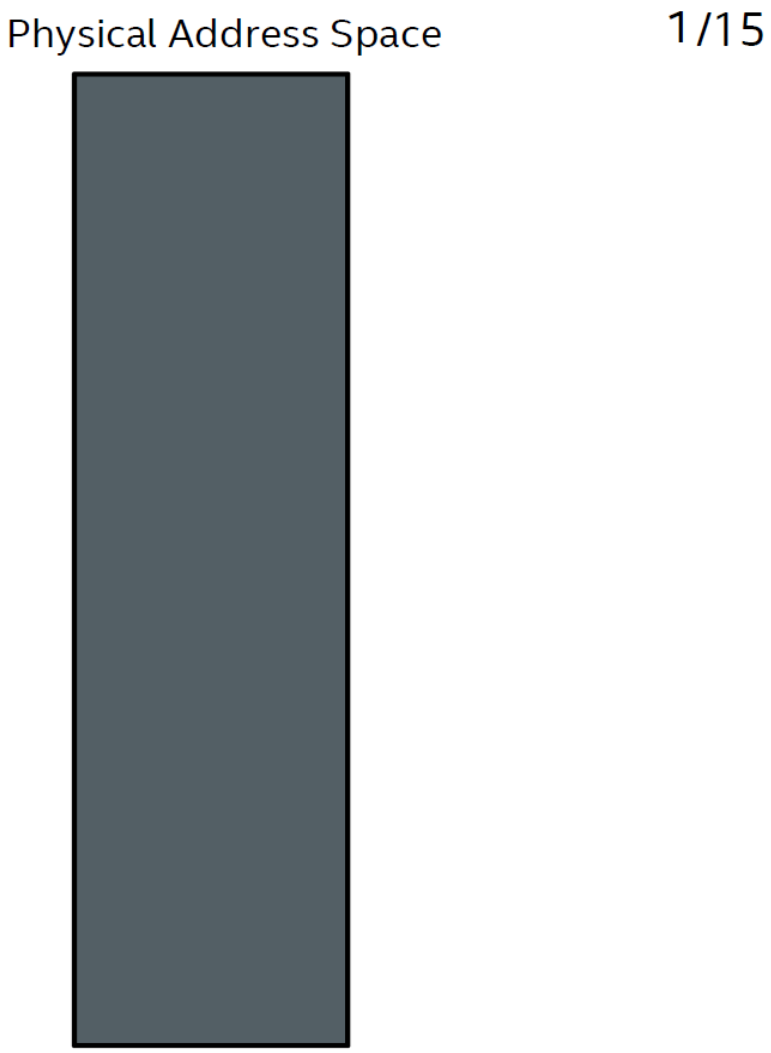


The State Save Area (SSA)

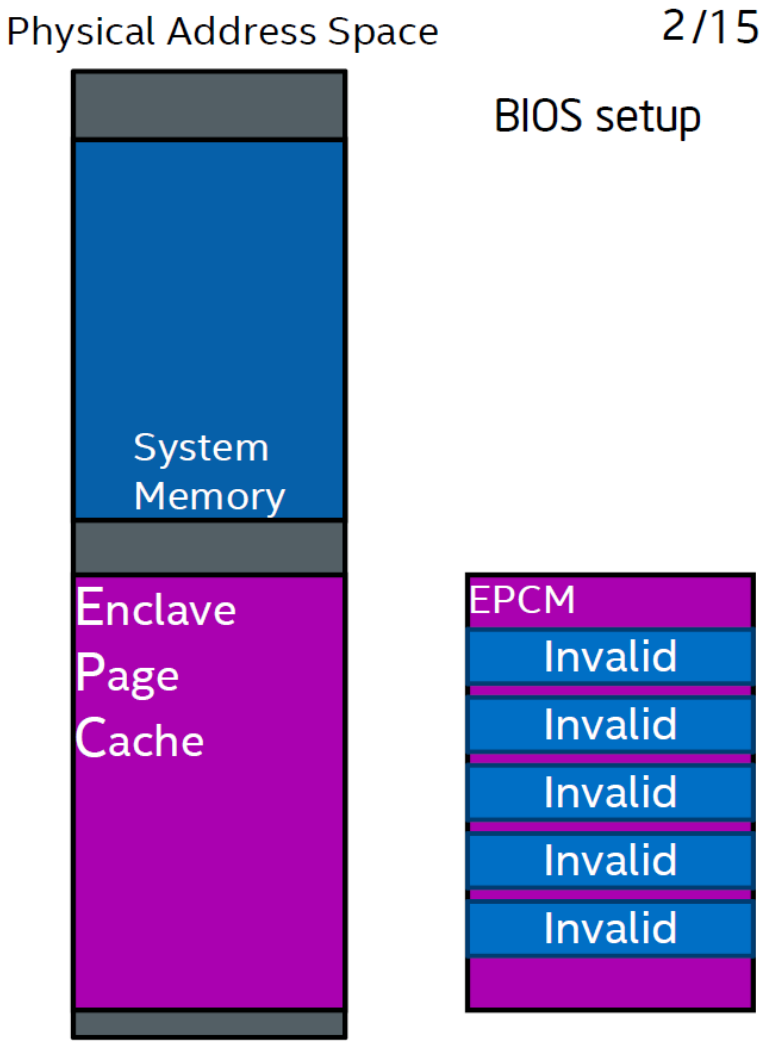
- **State Save Area (SSA)** stores the Enclave's execution context
- SSAFRAMESIZE field in SECS defines size of the SSA frame (in pages)
- GPRs are saved to GPR area on top of SSA frame
- Special Feature Registers are saved into XSAVE area at bottom of SSA frame
  - XFRM field in SECS controls the size of XSAVE area
- ECREATE instruction checks that all areas fit within an SSA frame



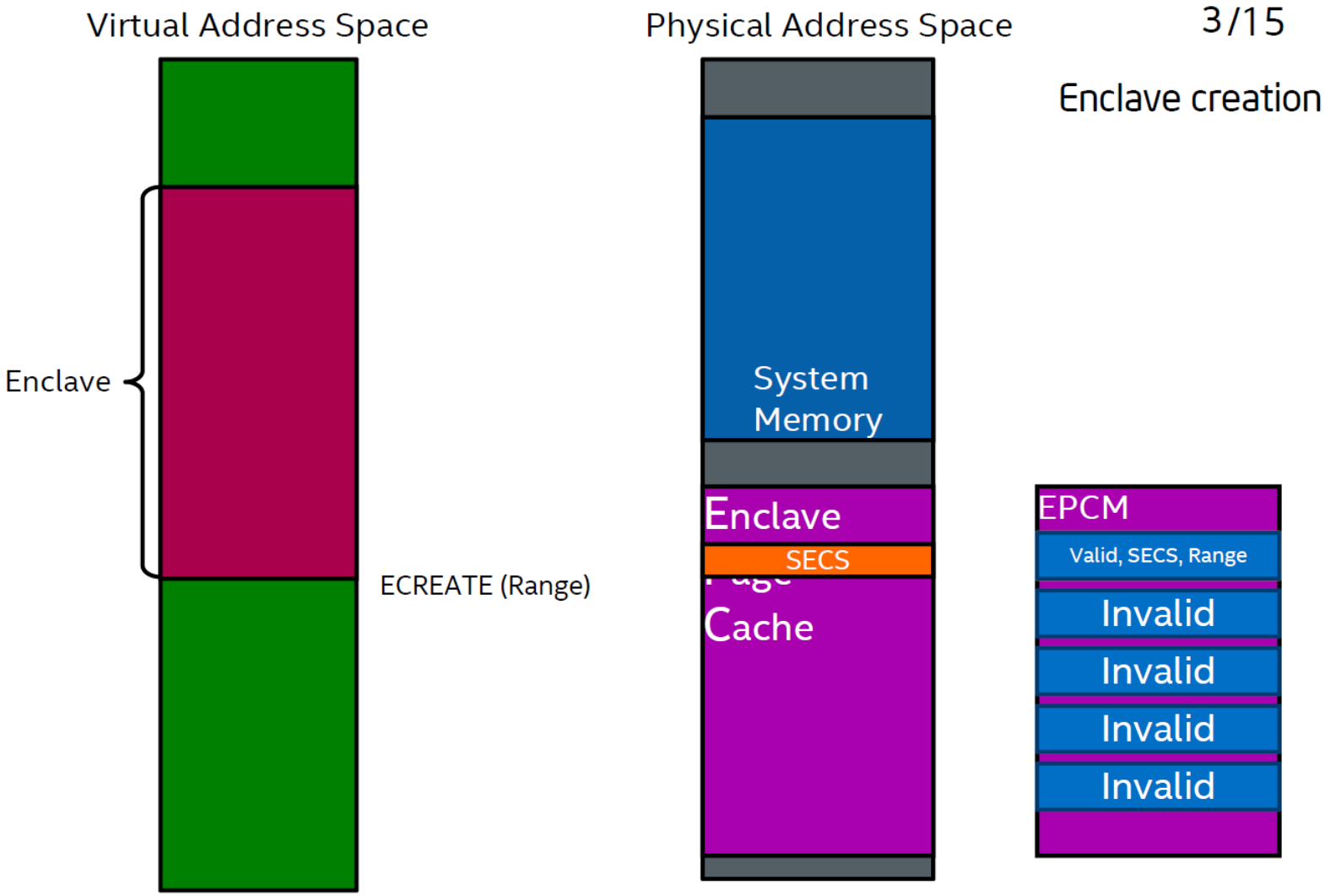
SGX Enclave Life Cycle Example



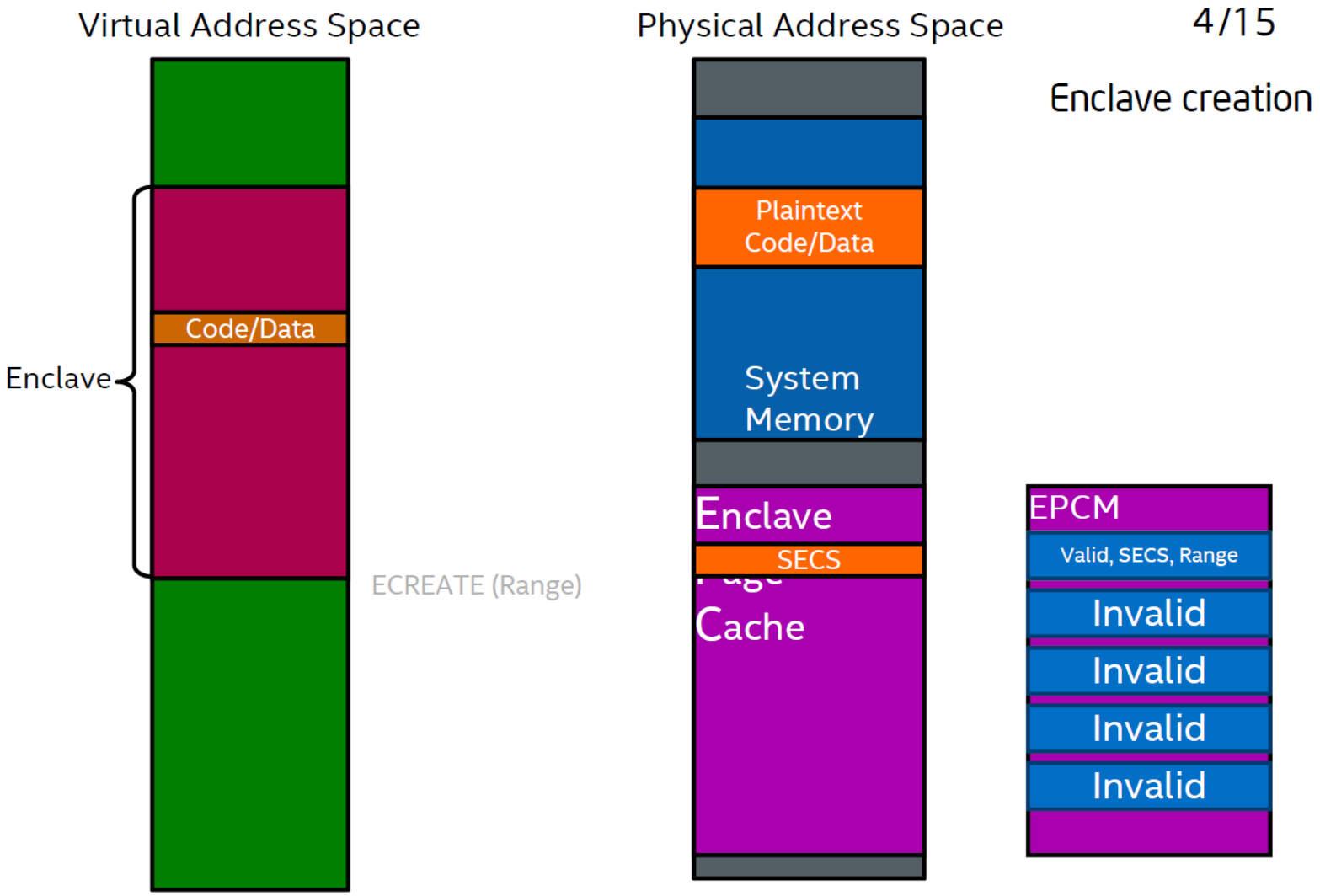
SGX Enclave Life Cycle Example



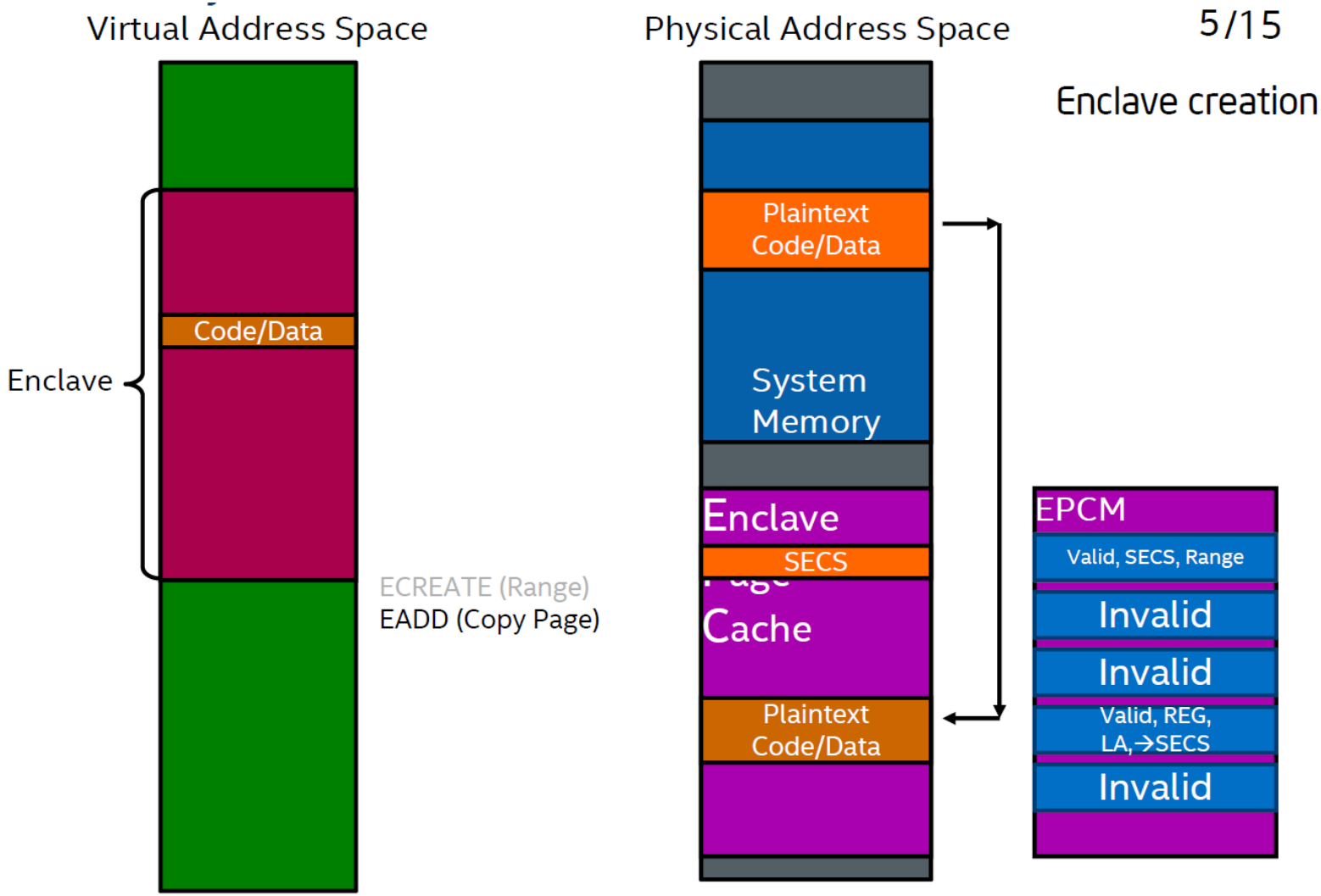
SGX Enclave Life Cycle Example



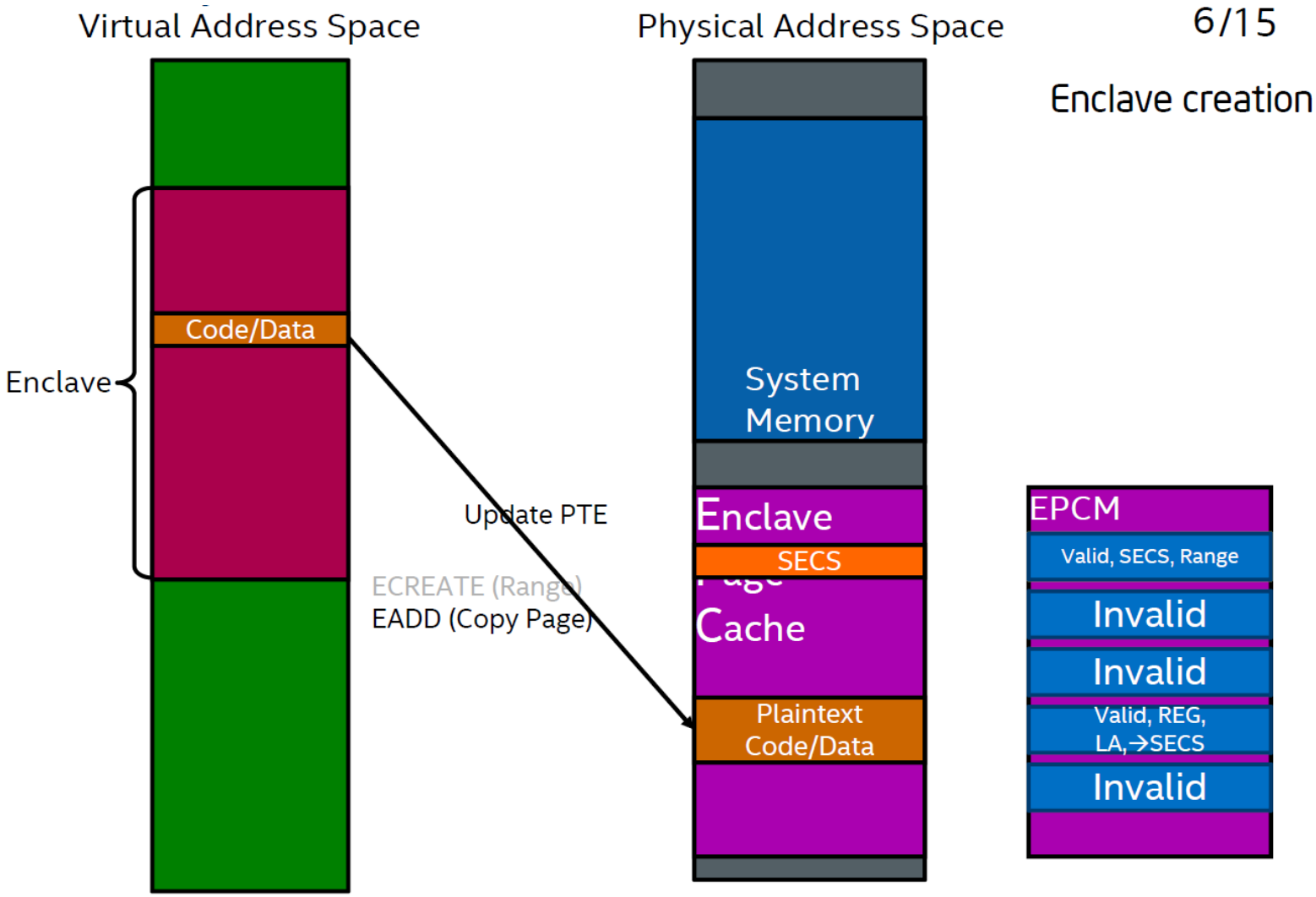
SGX Enclave Life Cycle Example



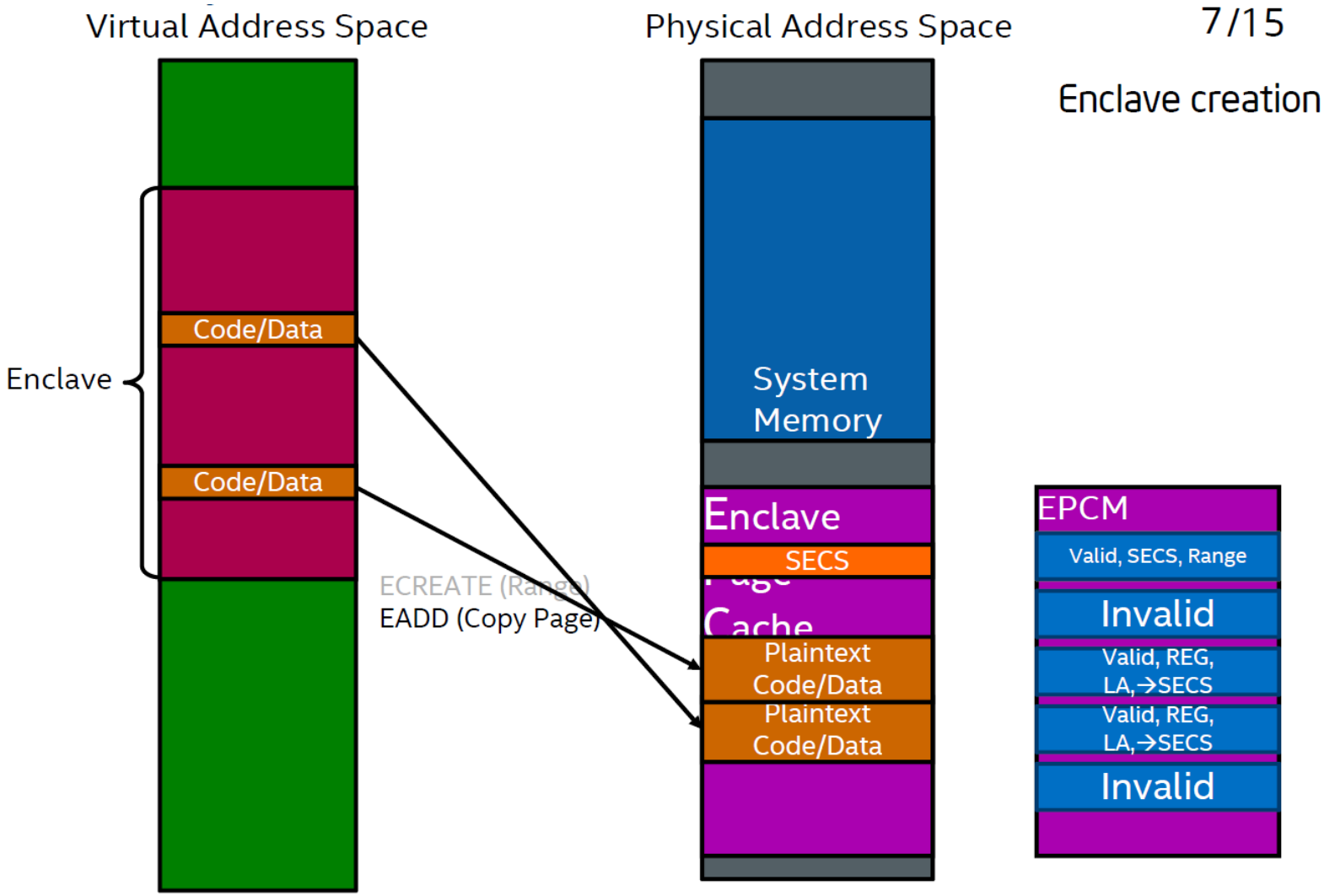
SGX Enclave Life Cycle Example



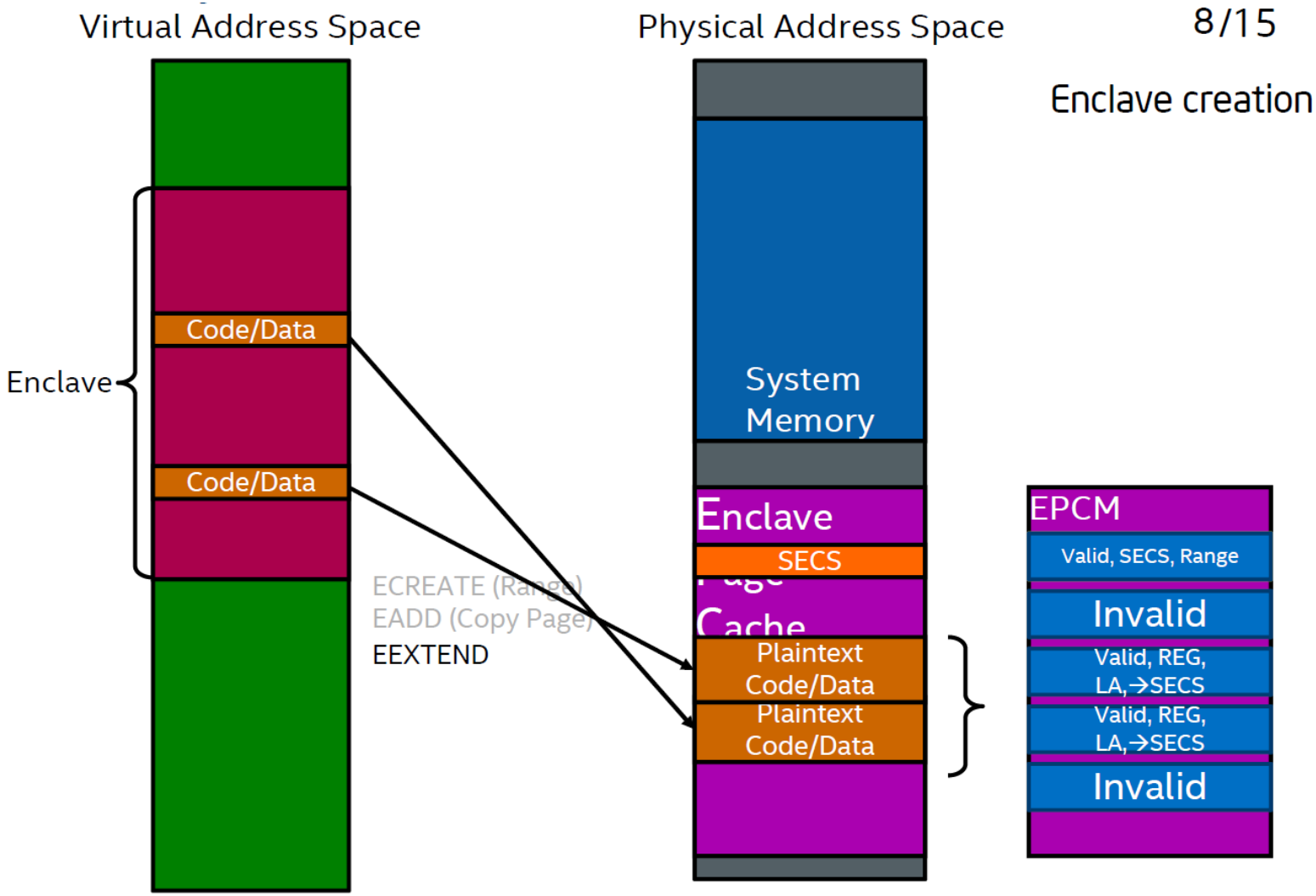
SGX Enclave Life Cycle Example



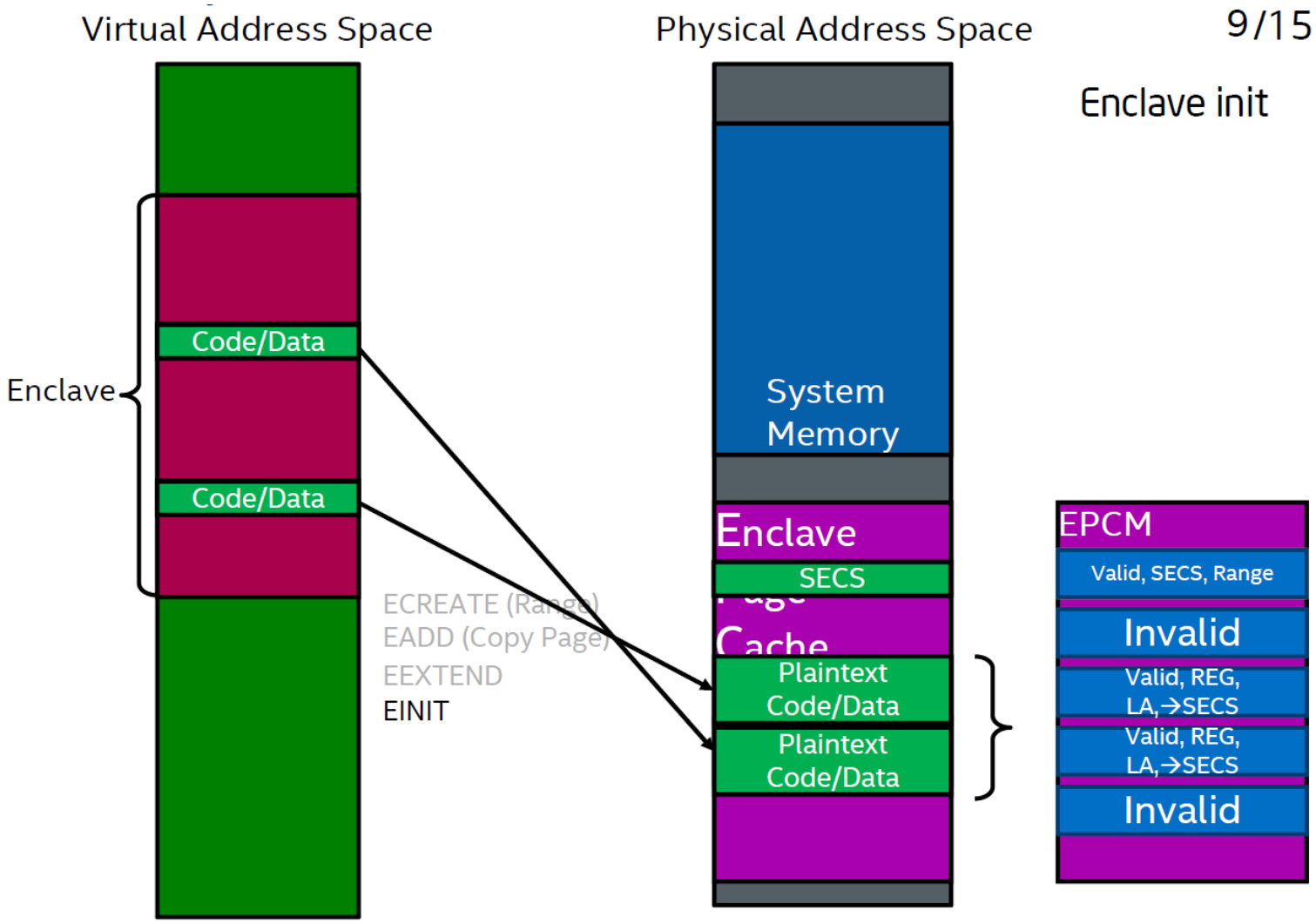
SGX Enclave Life Cycle Example



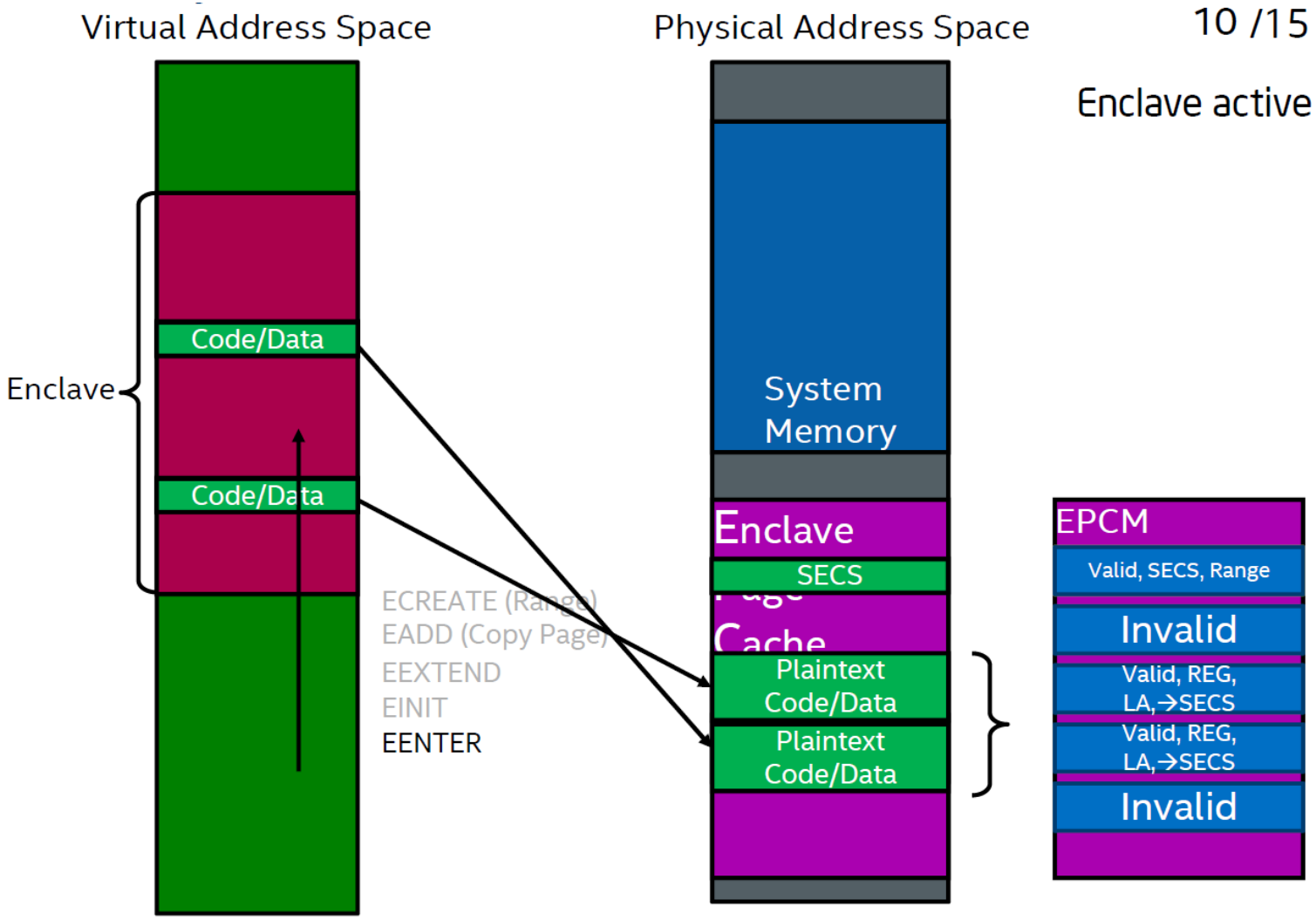
SGX Enclave Life Cycle Example



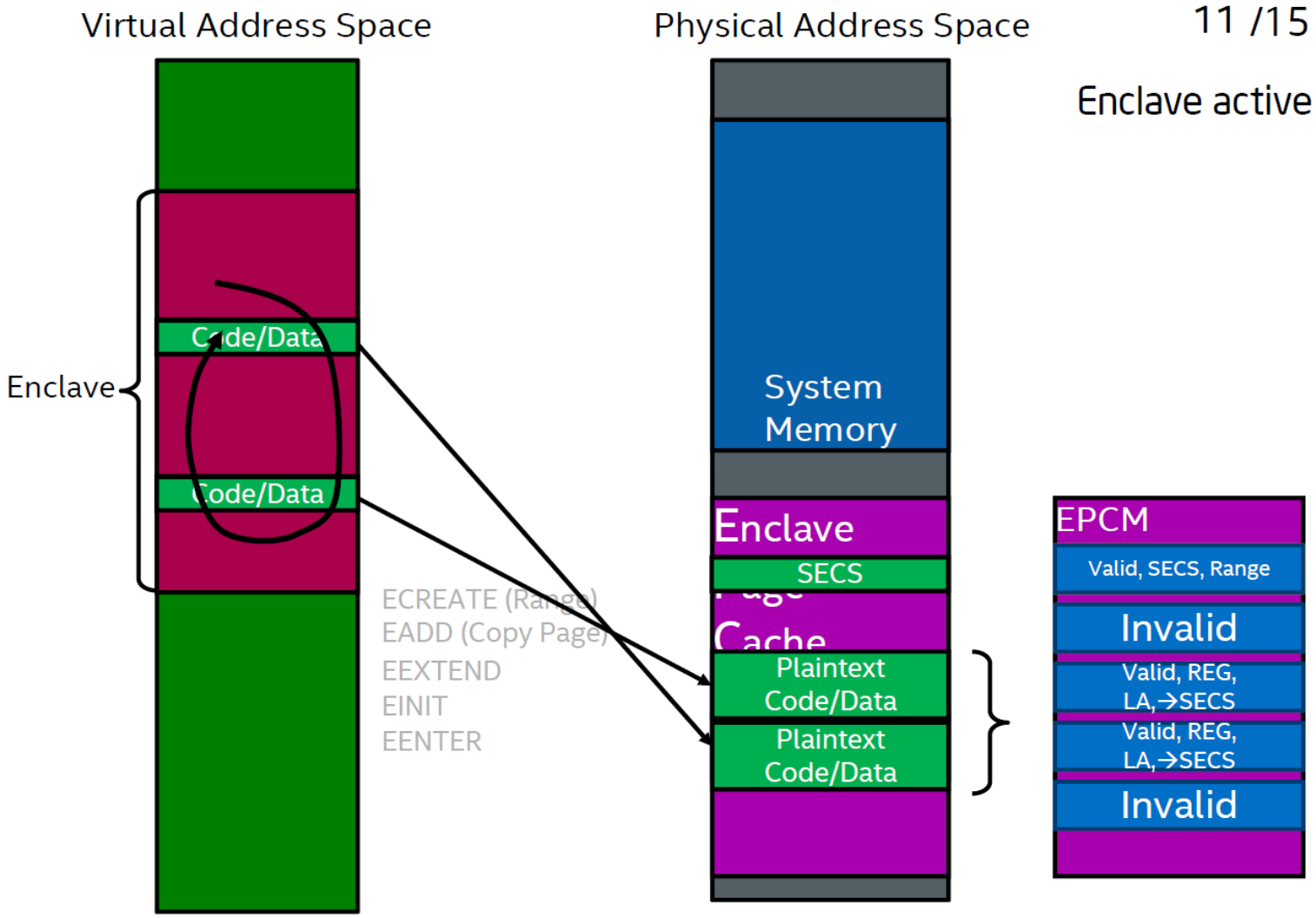
SGX Enclave Life Cycle Example



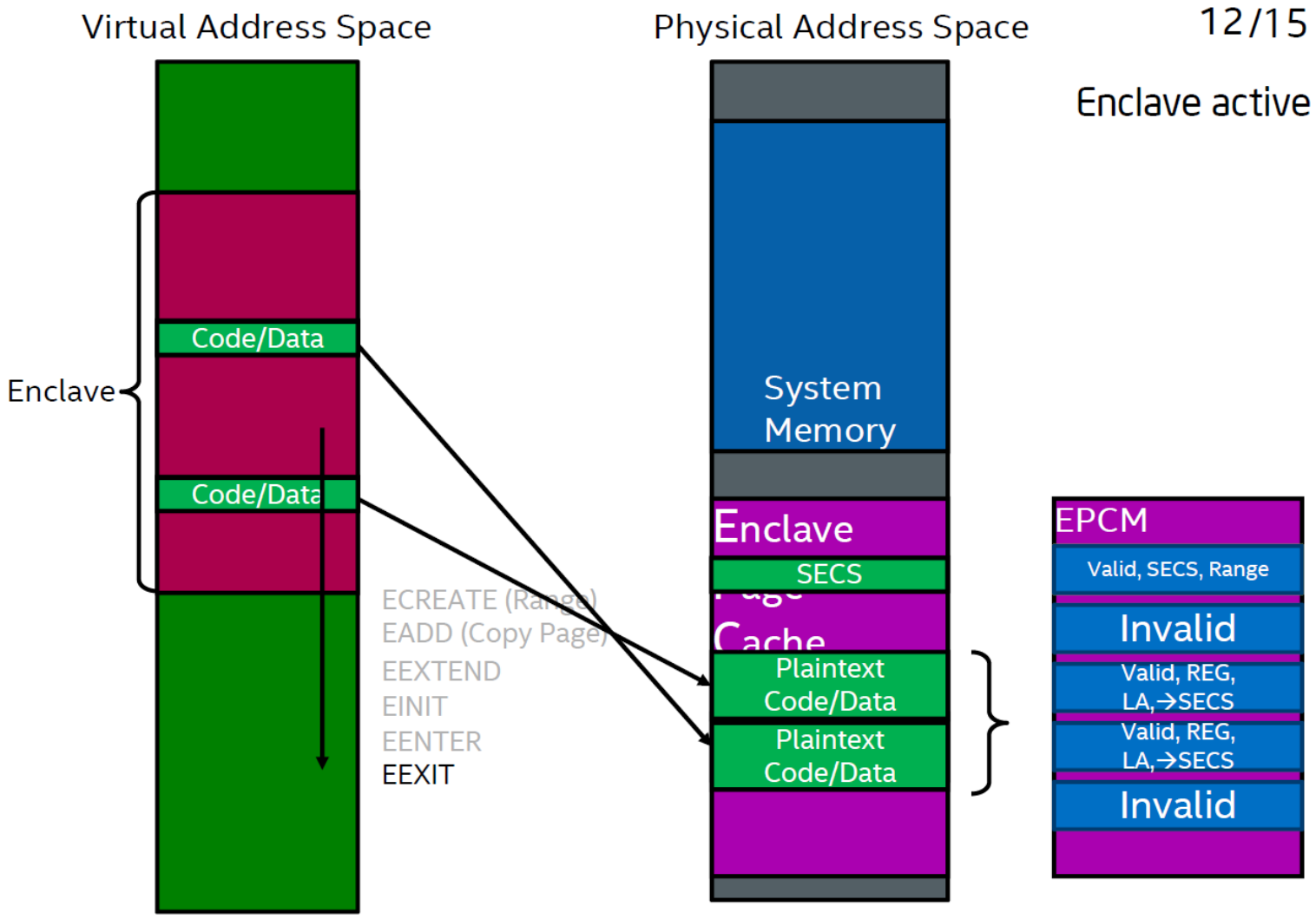
SGX Enclave Life Cycle Example



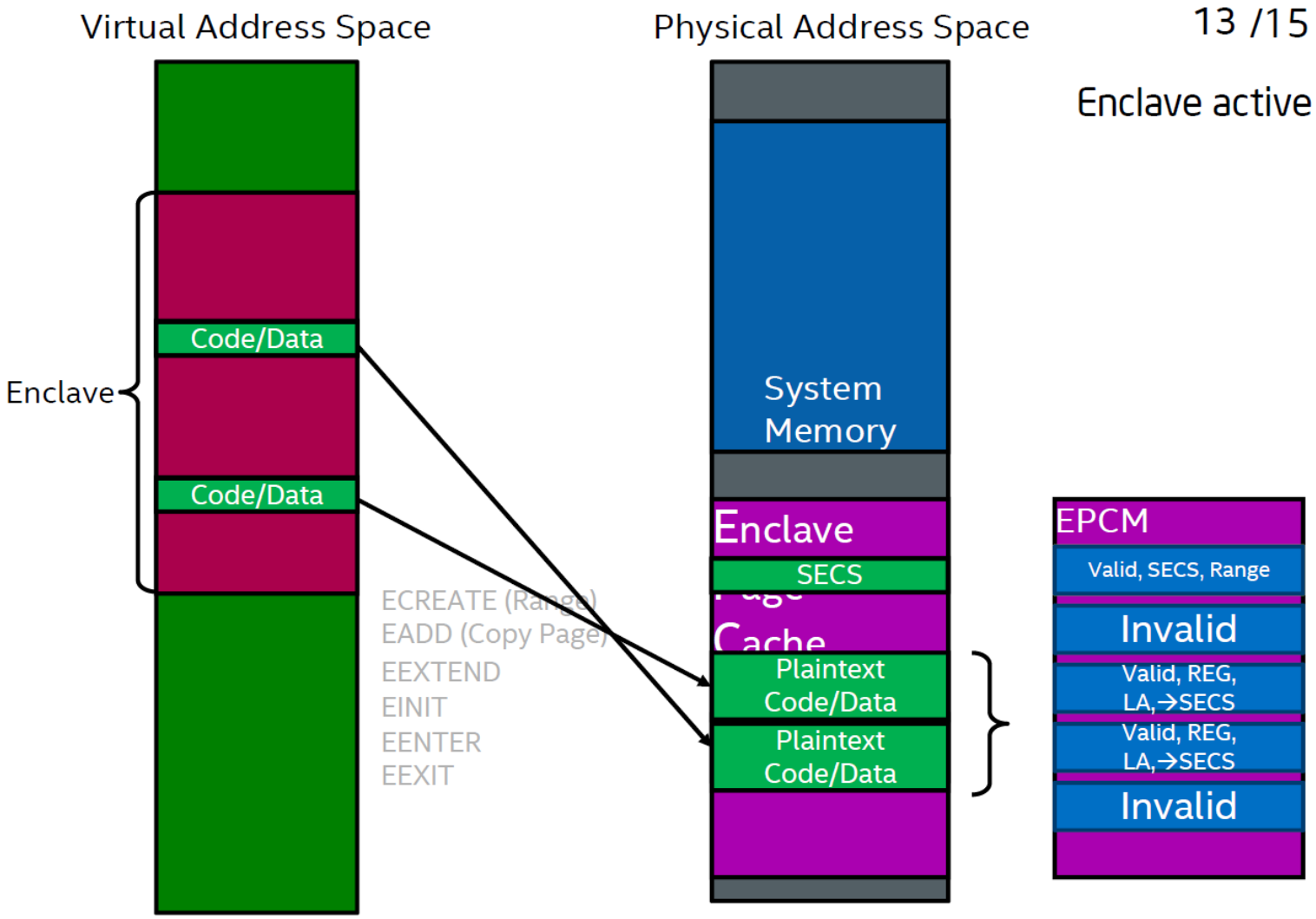
SGX Enclave Life Cycle Example



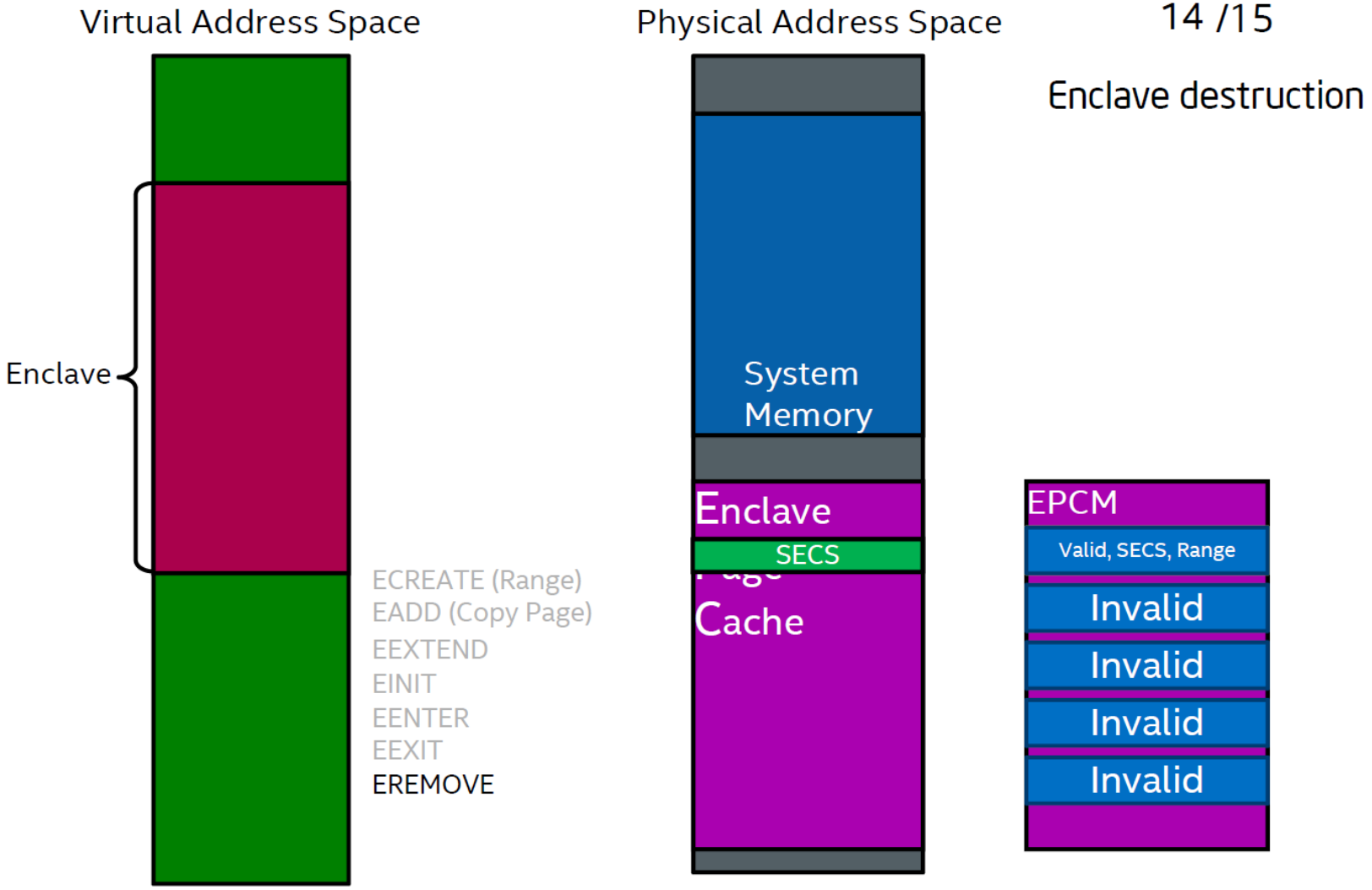
SGX Enclave Life Cycle Example



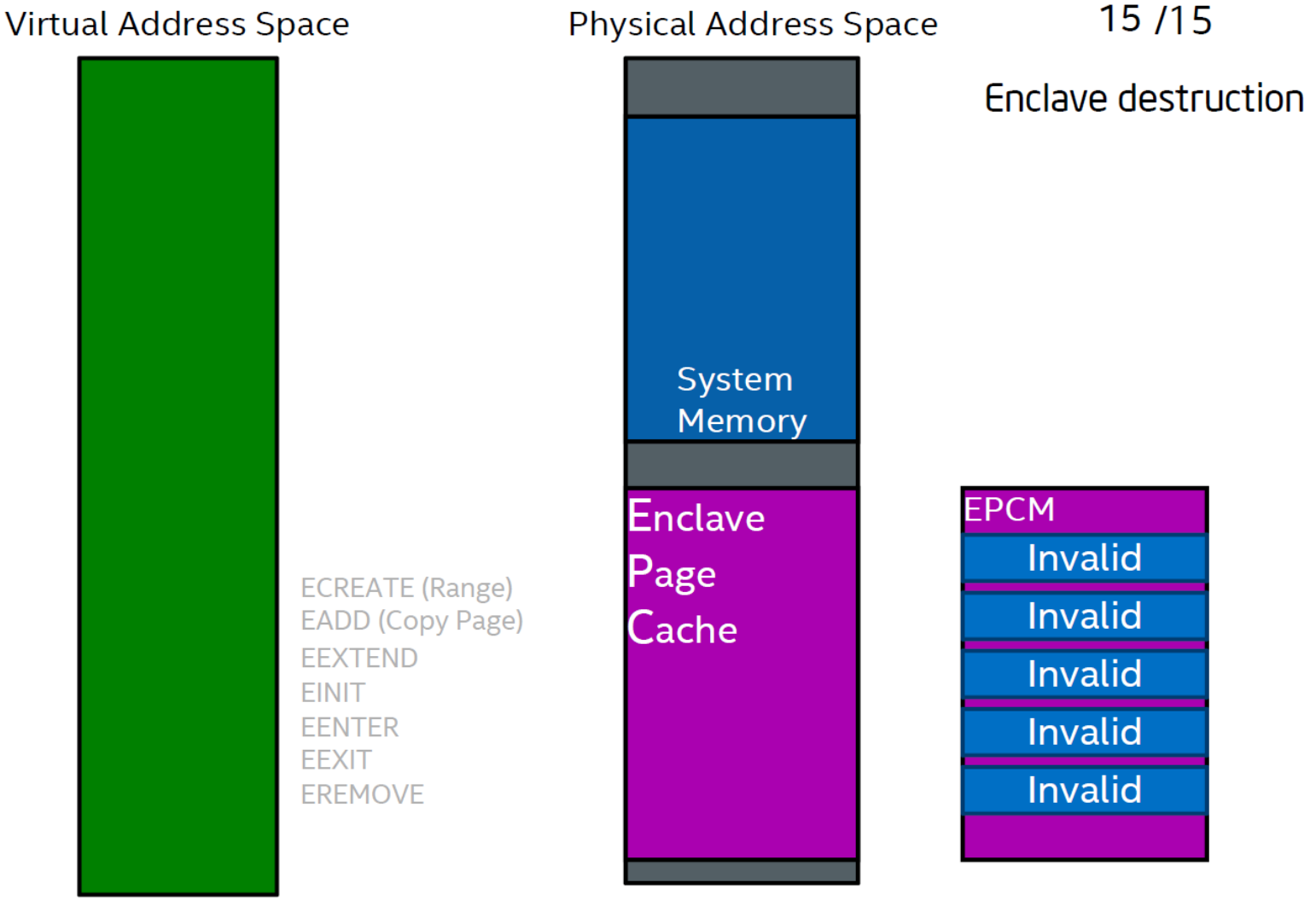
SGX Enclave Life Cycle Example



SGX Enclave Life Cycle Example



SGX Enclave Life Cycle Example



Outline

- SGX Enclave Life Cycle
- Tracking TLB Flushes
- SGX Security Properties
  - Misconceptions about SGX
  - Interaction with Anti-Virus Software

Tracking TLB Flushes

- In order to evict a batch of EPC pages, the OS kernel must first issue EBLOCK instructions targeting them. The OS is also expected to remove the EPC page's mapping from page tables, but is **not trusted** to do so.
- After all the desired pages have been blocked, the OS kernel must execute an ETRACK instruction, which directs the SGX implementation to keep track of which logical processors have had their TLBs flushed.
- If the OS wishes to evict a batch of EPC pages belonging to multiple enclaves, it must issue an ETRACK for each enclave. (Next slides show an example for a single enclave.)
- Following the ETRACK instructions, the OS kernel must induce enclave exits on all the logical processors that are executing code inside the enclaves that have been ETRACKed. The SGX design expects that the OS will use IPIs (interrupt processor instructions) to cause AEXs (asynchronous enclave exits) in the logical processors whose TLBs must be flushed.
- The EPC page eviction process is completed when the OS executes an EWB instruction for each EPC page to be evicted (see previous lecture).

Tracking TLB Flushes

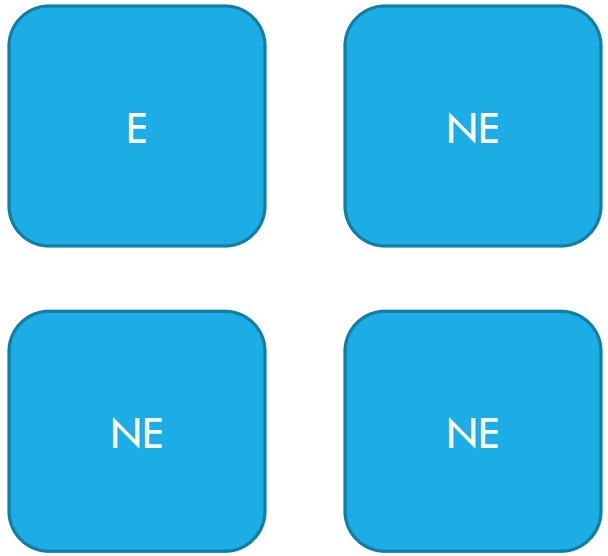
- Tracking TLB flushes is equivalent to verifying that all the logical processors (i.e., all the execution threads within the SGX Enclave) have exited Enclave mode at least once after we start tracking.
  - When a thread exits enclave mode (EEXIT or AES), the corresponding enclave address in the TLB are flushed.

- We rely on the SECS to store variables for tracking.

Tracking TLB Flushes

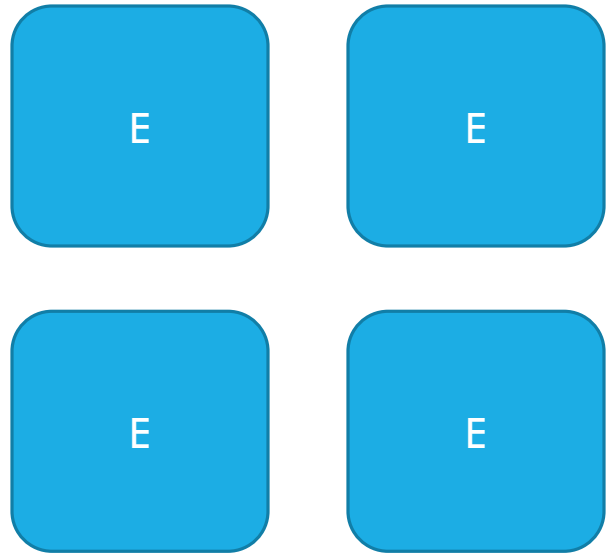
- ECREATE

- SECS.tracking = False
- SECS.done-tracking = False
- SECS.active-threads = 1
- SECS.tracked-threads = 0
- SECS.lp-mask = [.,.,.,.]



Tracking TLB Flushes

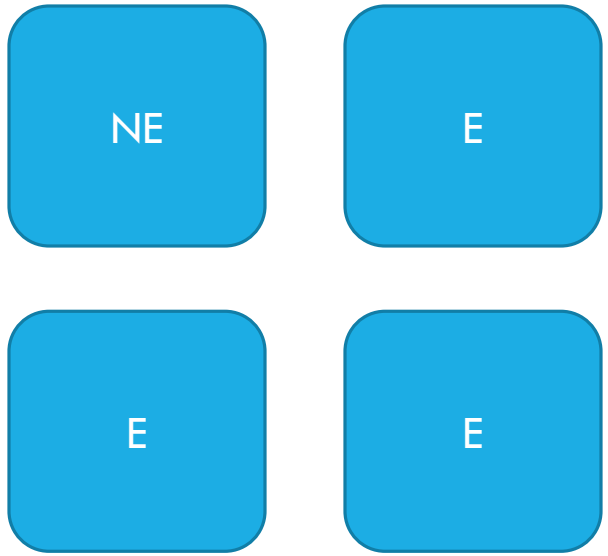
- EBLOCK
  - Targeting a batch of EPC pages to be swapped out
- ETRACK
  - Start of a TLB tracking cycle
- SECS.tracking = True
- SECS.done-tracking = False
- SECS.active-threads = 4
- SECS.tracked-threads = 4
- SECS.lp-mask = [0,0,0,0]



Tracking TLB Flushes

- EEXIT

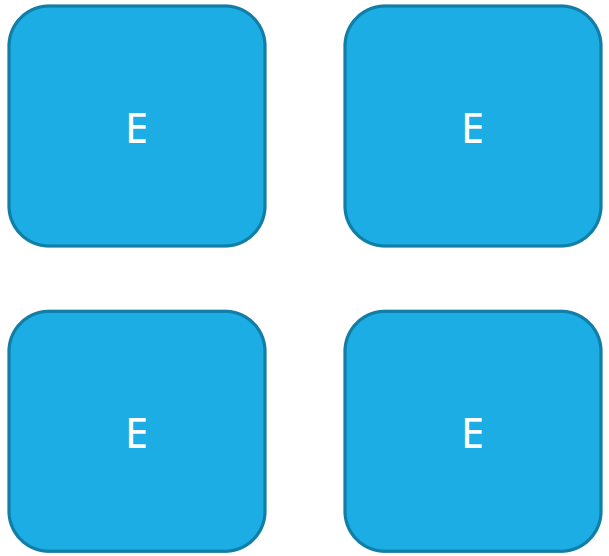
- SECS.tracking = True
- SECS.done-tracking = False
- SECS.active-threads = 3
- SECS.tracked-threads = 3
- SECS.lp-mask = [1,0,0,0]



Tracking TLB Flushes

- EENTER

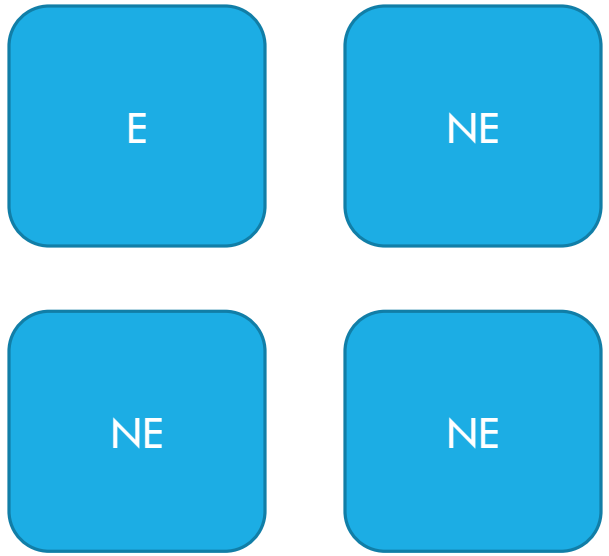
- SECS.tracking = True
- SECS.done-tracking = False
- SECS.active-threads = 4
- SECS.tracked-threads = 3
- SECS.lp-mask = [1,0,0,0]



Tracking TLB Flushes

- EEXIT / AES

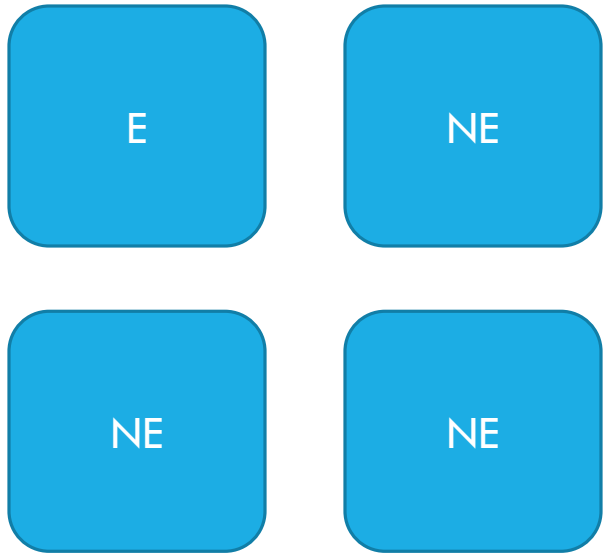
- SECS.tracking = True
- SECS.done-tracking = True
- SECS.active-threads = 1
- SECS.tracked-threads = 0
- SECS.lp-mask = [1,1,1,1]



Tracking TLB Flushes

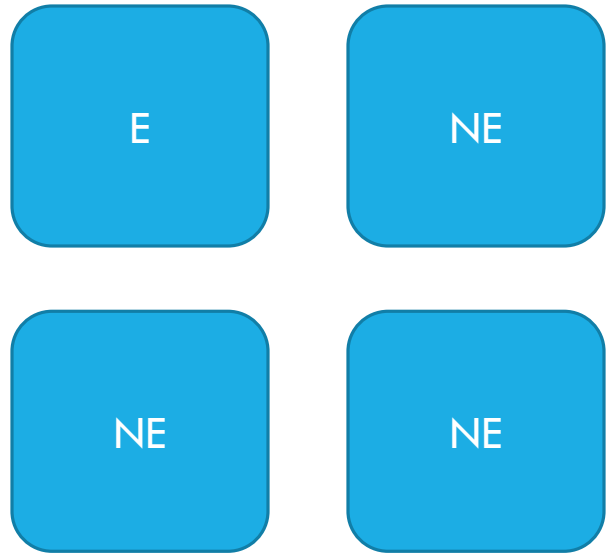
- EWB-VERIFY

- SECS.tracking = True
- SECS.done-tracking = True
- SECS.active-threads = 1
- SECS.tracked-threads = 0
- SECS.lp-mask = [1,1,1,1]



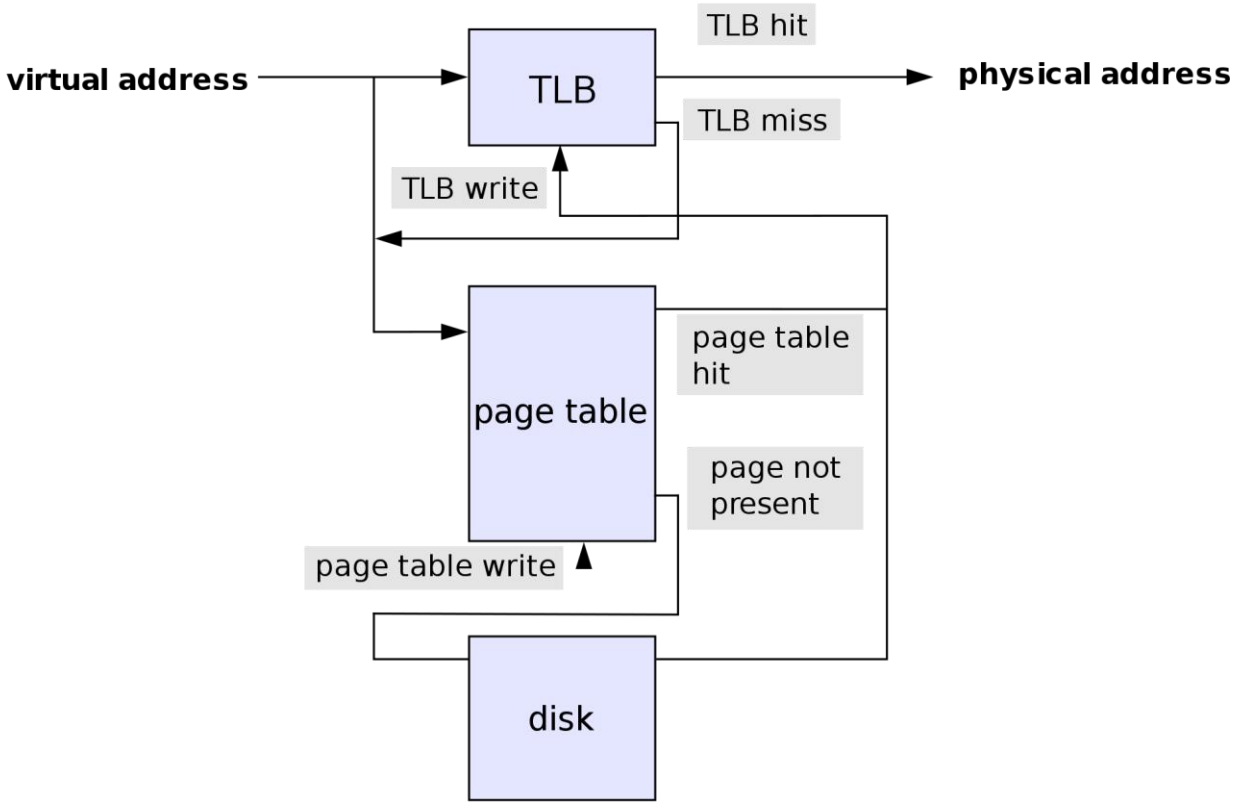
Tracking TLB Flushes

- EBLOCK
  - End of a TLB tracking cycle
- SECS.tracking = False
- SECS.done-tracking = True
- SECS.active-threads = 1
- SECS.tracked-threads = 0
- SECS.lp-mask = [1,1,1,1]



Passive Attacks

- Address translation used for page swapping
- An untrusted page table manager can swap pages using page faults and leak information



- Successful practical attacks on SGX!
  - Image inferred even though it was isolated by SGX
  - "Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems", IEEE S&P 2015, by Y. Xu, W. Cui, and M. Peinado
  - Intel's response (by Matt Hoekstra and Frank McKeen) puts blame on software developers
  - https://software.intel.com/en-us/blogs/2015/05/19/look-both-ways-and-watch-out-for-side-channels

Other Buffers which Need Flushing?

- Processors do branch prediction and store a history of branch selections in a branch history buffer
- SGX does not flush this buffer → Infer control flow → Leaks privacy
- Demonstrated "Inferring Fine-grained Control Flow Inside SGX Enclaves with Branch Shadowing", eprint Nov 2016, by S. Lee, M.-W. Shih, P. Gera, T. Kim, H. Kim, M. Peinado
- EEXIT and EAS should have updated microcode which includes a flush of this buffer

Outline

- SGX Enclave Life Cycle
- Tracking TLB Flushes
- SGX Security Properties
  - Misconceptions about SGX
  - Interaction with Anti-Virus Software

SGX Security Properties

- An isolated container whose contents receive special hardware protection. This translates to privacy, integrity and freshness guarantees.
- Offers a certificate-based identity system that can be used to migrate secrets between enclaves that have certificates issued by the same authority. More on this when we talk about attestation (next lecture).

Physical Attacks

- Lack of publicly available details about the hardware implementation of SGX => some avenues for future exploration
  - Note recent NDSS'16 paper "OpenSGX: An Open Platform for SGX Research"
- Port attack, especially Generic Debug eXternal Connection.
- Bus attack, because the data in cache is in plaintext.
- Bus tapping attack, because SGX does not hide the memory access patterns.
- Cache timing attack.
- Intel Management Engine may not be protected. (Will discuss ME in a next lecture.)
- Fused seal key. -> May use PUF technology instead (a later lecture)
- Power analysis

Privileged Software Attacks

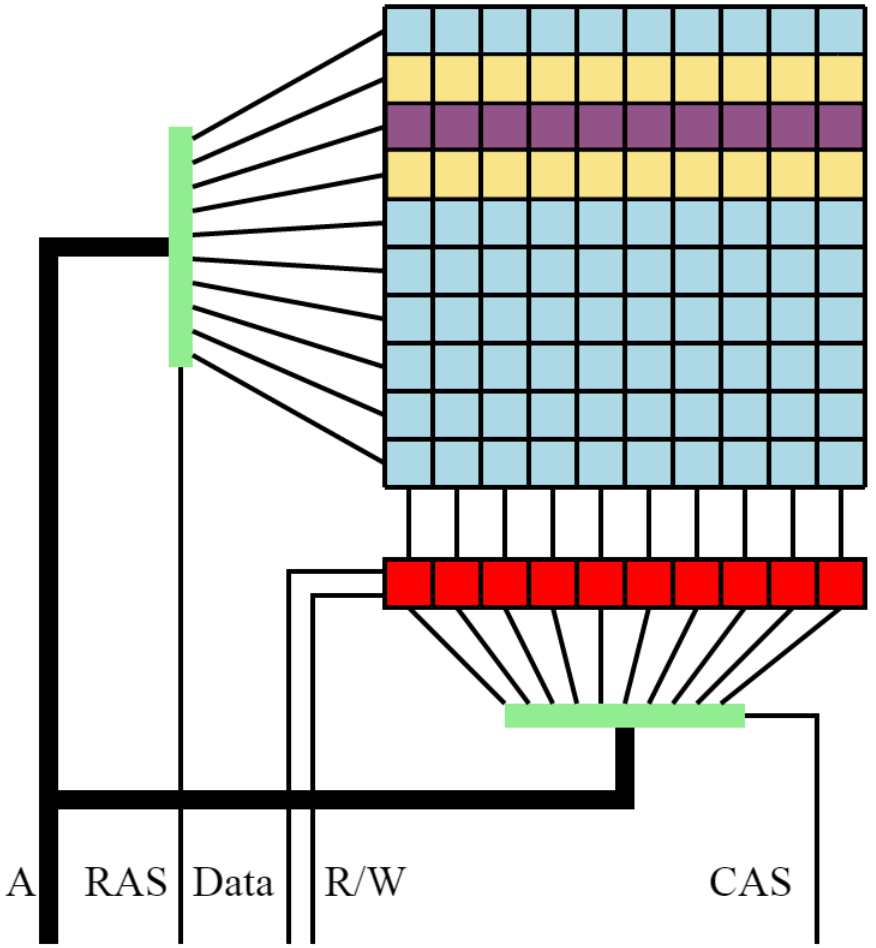
- The SGX design prevents malicious software from directly reading or from modifying the EPC pages that store an enclave's code and data.
- This relies on two pillars (isolation principle):
- First, the SGX implementation runs in the processor's microcode, which is effectively a higher privilege level to which system software has no access.
- Second, SGX's microcode is always involved when a CPU transitions between enclave code and non-enclave code, and therefore regulates all interactions between system software and an enclave's environment

Memory Mapping Attacks

- SGX can prevent active attacks by rejecting undesirable address translations before they reach the TLB. Also, it prevents the active attacks using page swapping or stale TLB entries.
- Passive address translation attacks can learn the memory access patterns (as discussed when explaining page swapping).

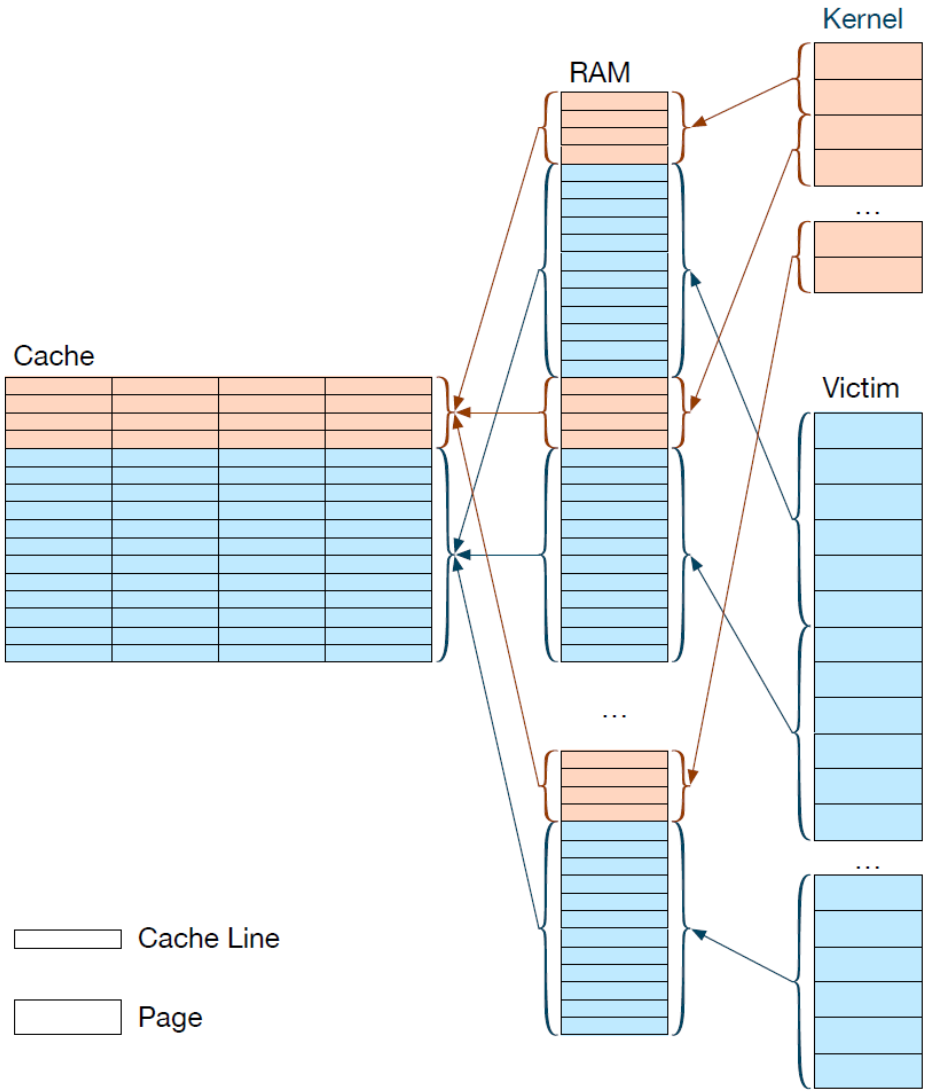
Software Attacks on Peripherals

- PCI (Peripheral Controller Interface) Express attacks are prevented, because MC rejects any DMA transfer that falls within the Processor Reserved Memory
- DRAM attacks (e.g. Rowhammer) are prevented due to MEE.
- Firmware attacks (especially, ME's firmware) are not mentioned in the documents. (ME compromise = DRAM attacks)
- SGX does not protect against software side-channel attacks that rely on performance counter (e.g. cache misses, branch predictors).



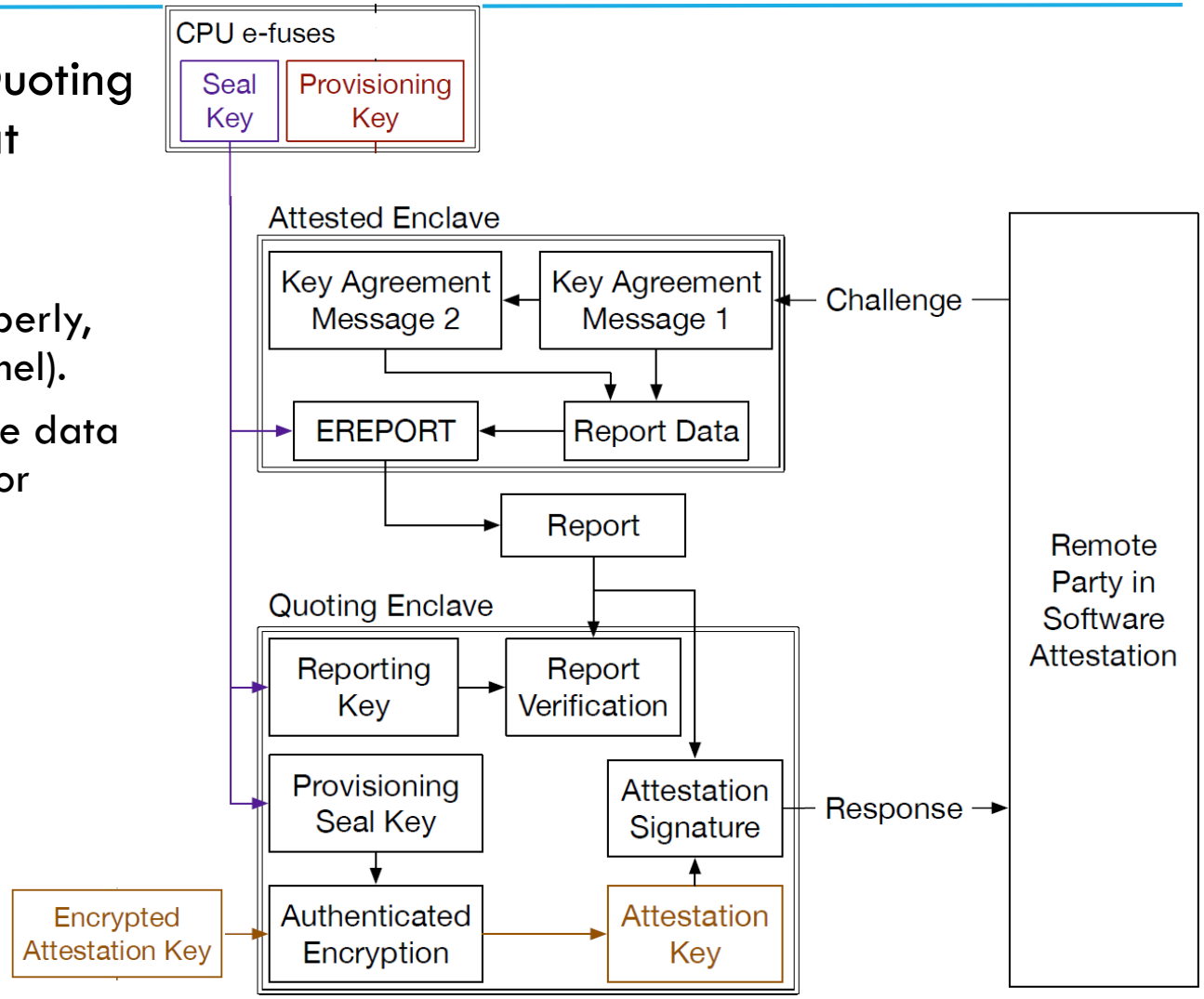
Cache Timing Attacks

- Cache timing attacks are not mentioned in the threat model.
- A malicious system software can make it worse.
  - Control the enclave scheduling
  - Control address translation
- SGX does not prevent this attack, but increases the difficulties: SGX's enclave entry implementation could flush the core's private caches.
- The Last Level Cache is still vulnerable, because it is shared among all the cores.



Misconceptions about SGX

- Remote attestation relies on the Quoting Enclave with special privileges that allows it to access the processor's attestation key.
  - This assumes the Enclave is isolated properly, but this is not true (e.g. cache side channel).
  - Intel suggests the programmer to remove data dependent memory access, especially for crypto algorithms.



Misconceptions about SGX

- Enclaves Can DOS (Denial-of-service) the System Software 
  - The SGX design provides system software the capability to protect itself from enclaves that engage in CPU hogging and DRAM hogging.
  - System software needs to reserve at least one Logical Processor (LP) for non-enclave computation.
- SGX is tamper-resistant 
  - The chip itself does not prevent physical tampering.

Interaction with Anti-Virus Software

- Today's anti-virus (AV) systems are pattern matchers.

- 1. A generic loader that is undetectable by AV's pattern matcher.
- 2. Load encrypted malicious payload from Internet.
- 3. Execute malicious code inside the Enclave. (botnets?)

- Possible solutions:
  - recording and filtering the I/O performed by software
  - Static analysis