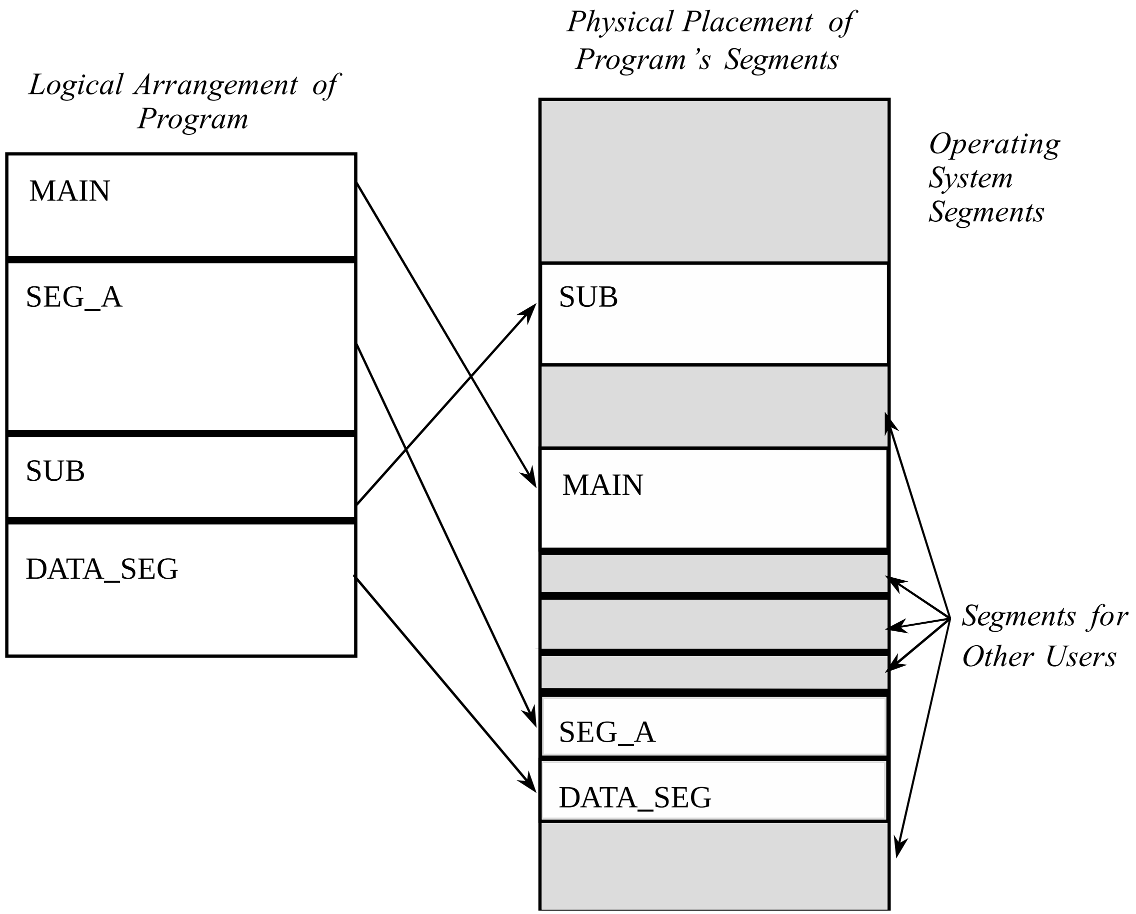
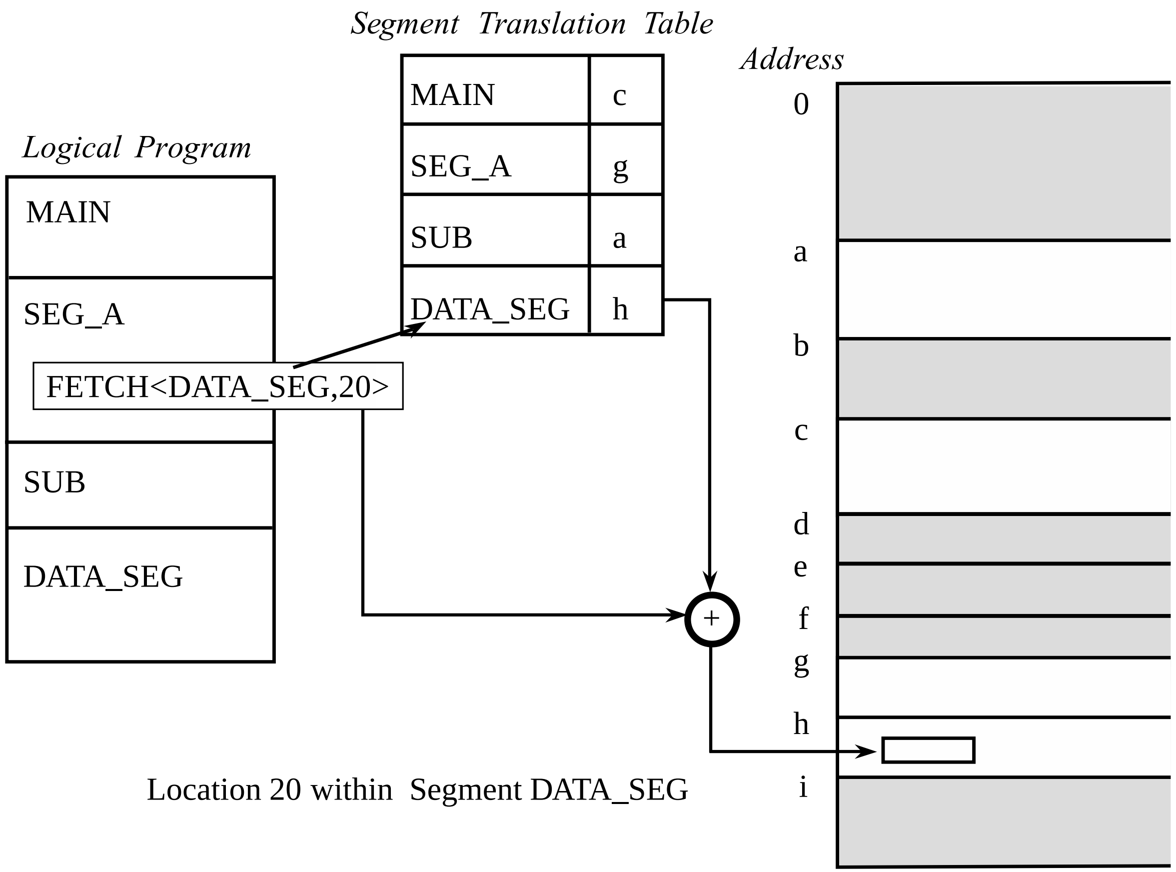
Segmentation

- A program is divided into separate, logical pieces.
- Each segment has its own set of access rights.
- OS maintains a table of each segment and its true memory address, and it translates calls to each segment using the table
- Advantages:
  - OS can move segments around as necessary
  - Segments can be removed from memory if they aren't being used currently



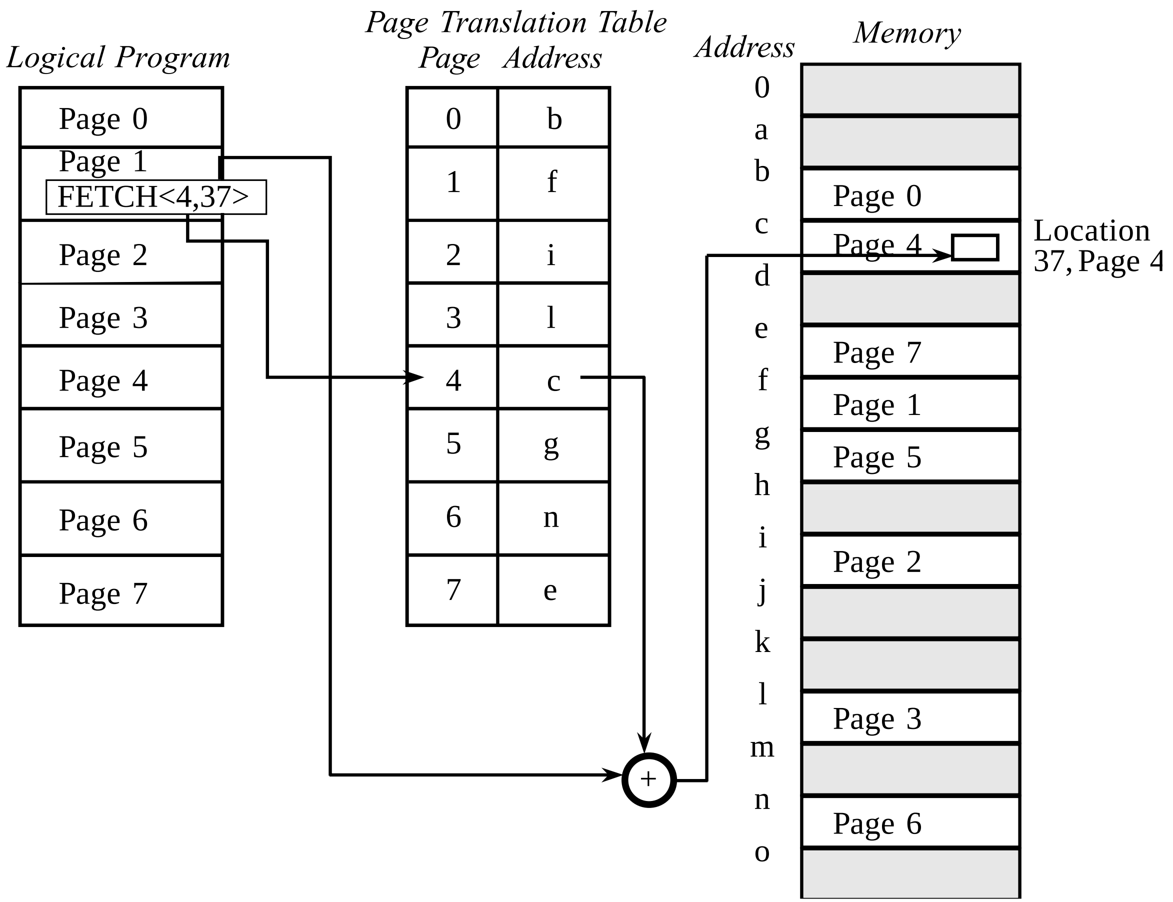
Segment Address Translation

- When program generates an address of <name, offset>, OS looks up name in segment directory and determines the real beginning address
- One table for each process
- Program knows nothing about actual address
- Security benefit
  - Each address reference is checked
  - Two or more users can share access to a segment, with different access rights
- Problem
  - How to decide segment size?



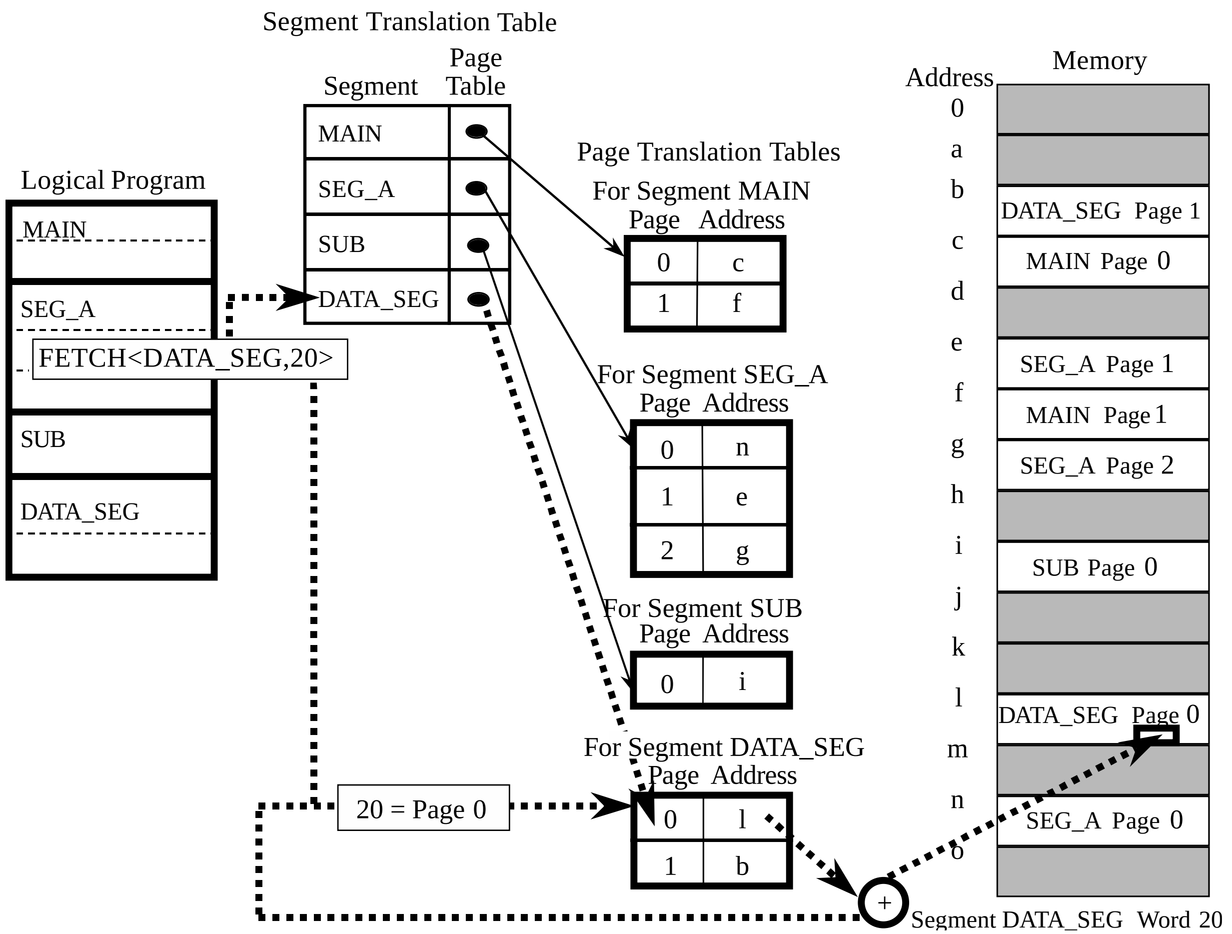
Paging

- Program divided to equal-sized pages
- Each address is <page, offset>
- OS maintains a page table
- Offset beyond the end of a page results in a carry into the page portion of address
- Problem:
  - No logical boundary



Paged Segmentation

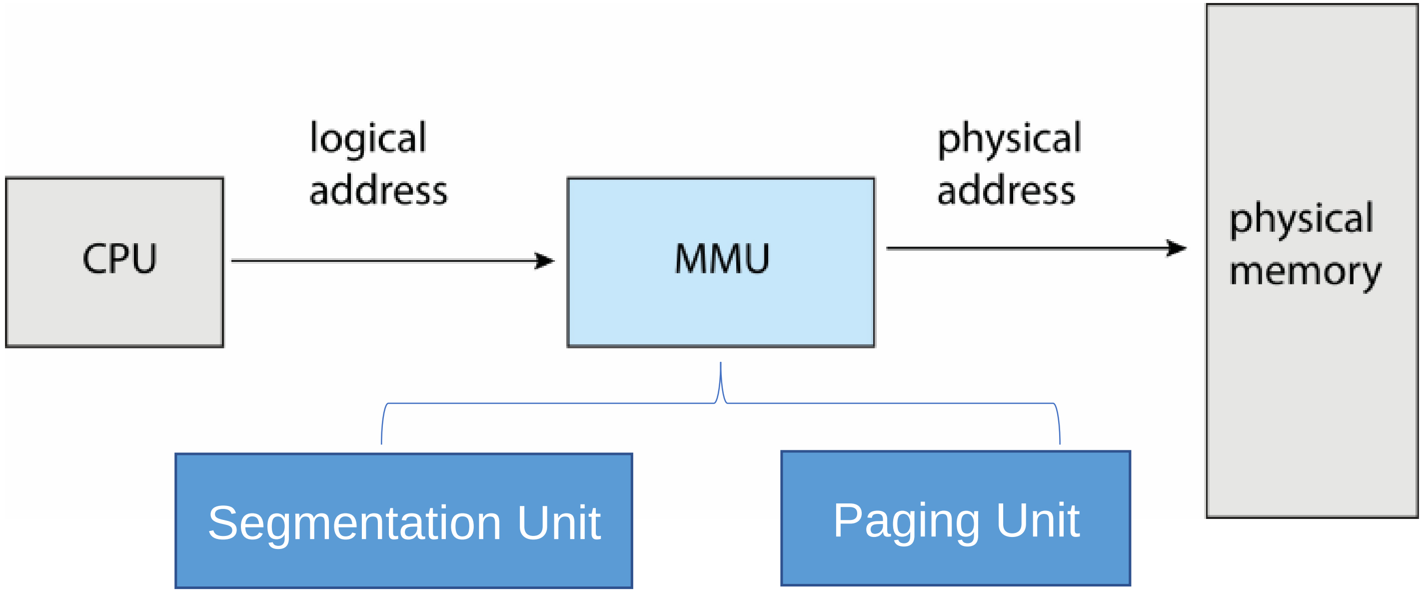
- Segmentation: logical protection
- Paging: efficiency
- Combined -> paged segmentation
- Programs can be broken into segments, and the segments are then combined to fill pages.



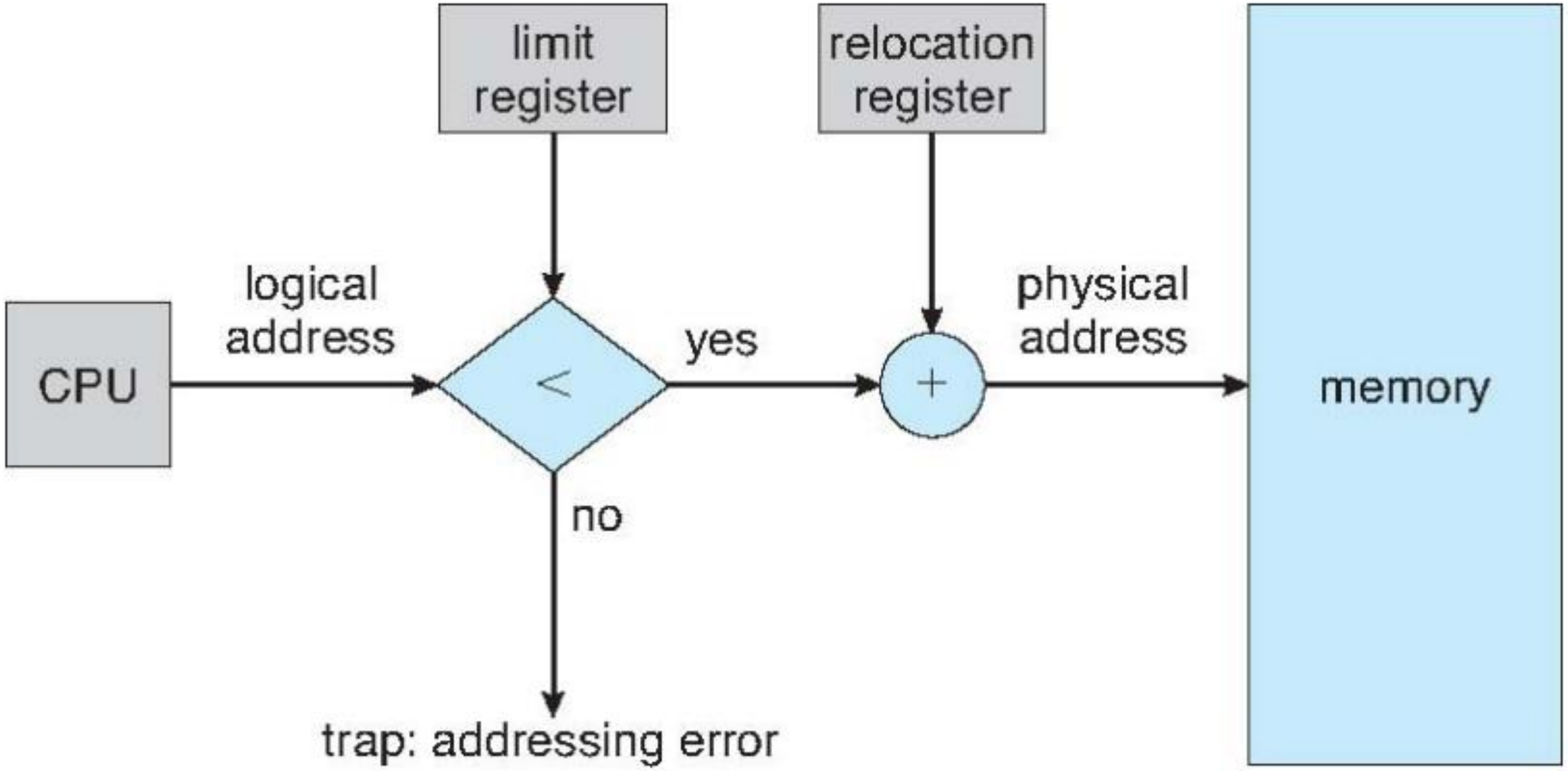
Memory management by contemporary hardware & OS

Memory-Management Unit (MMU)

- Hardware device that at run time maps virtual to physical address
- Base (relocation) register + logical address => physical address

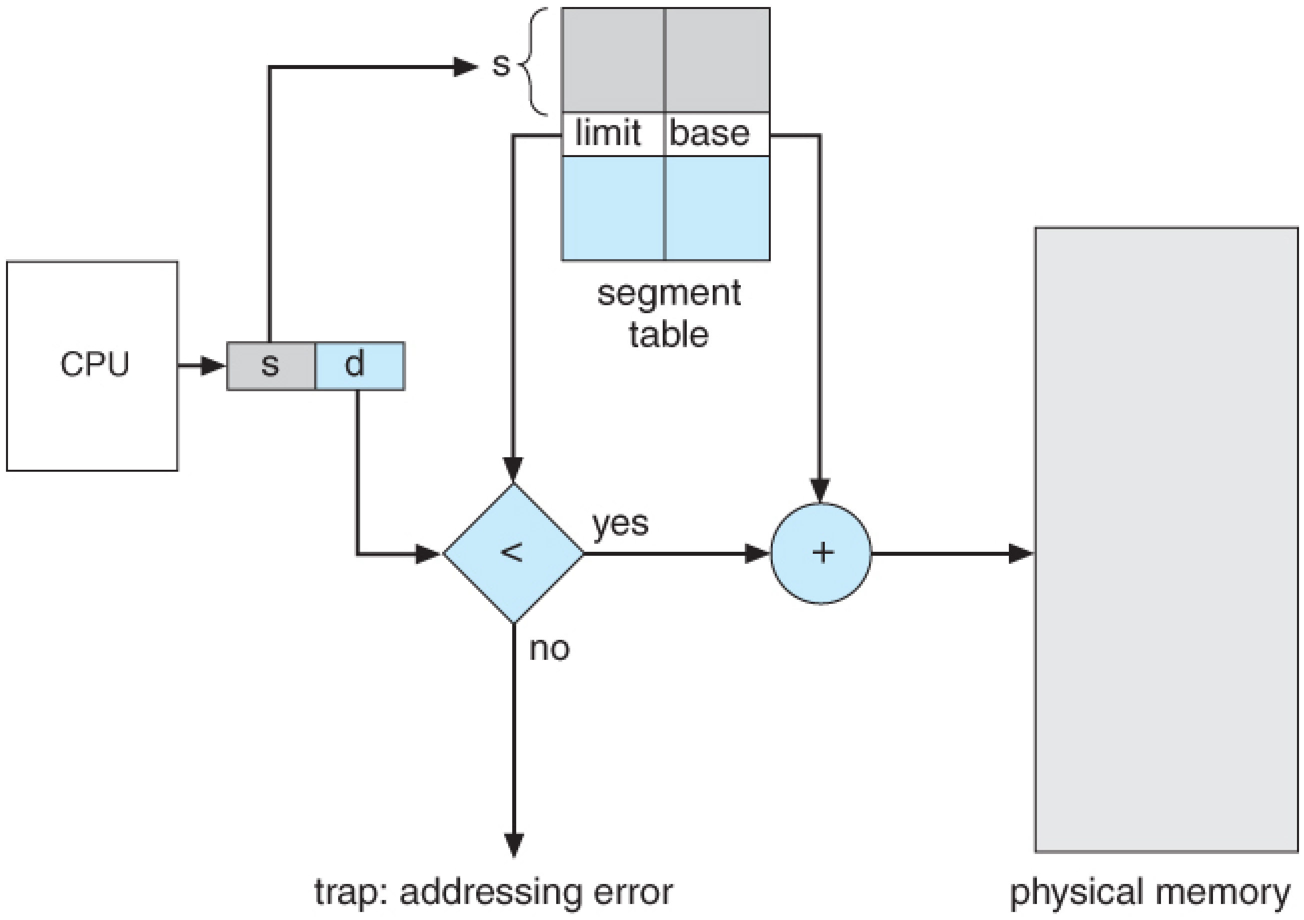


Relocation and Limit Registers

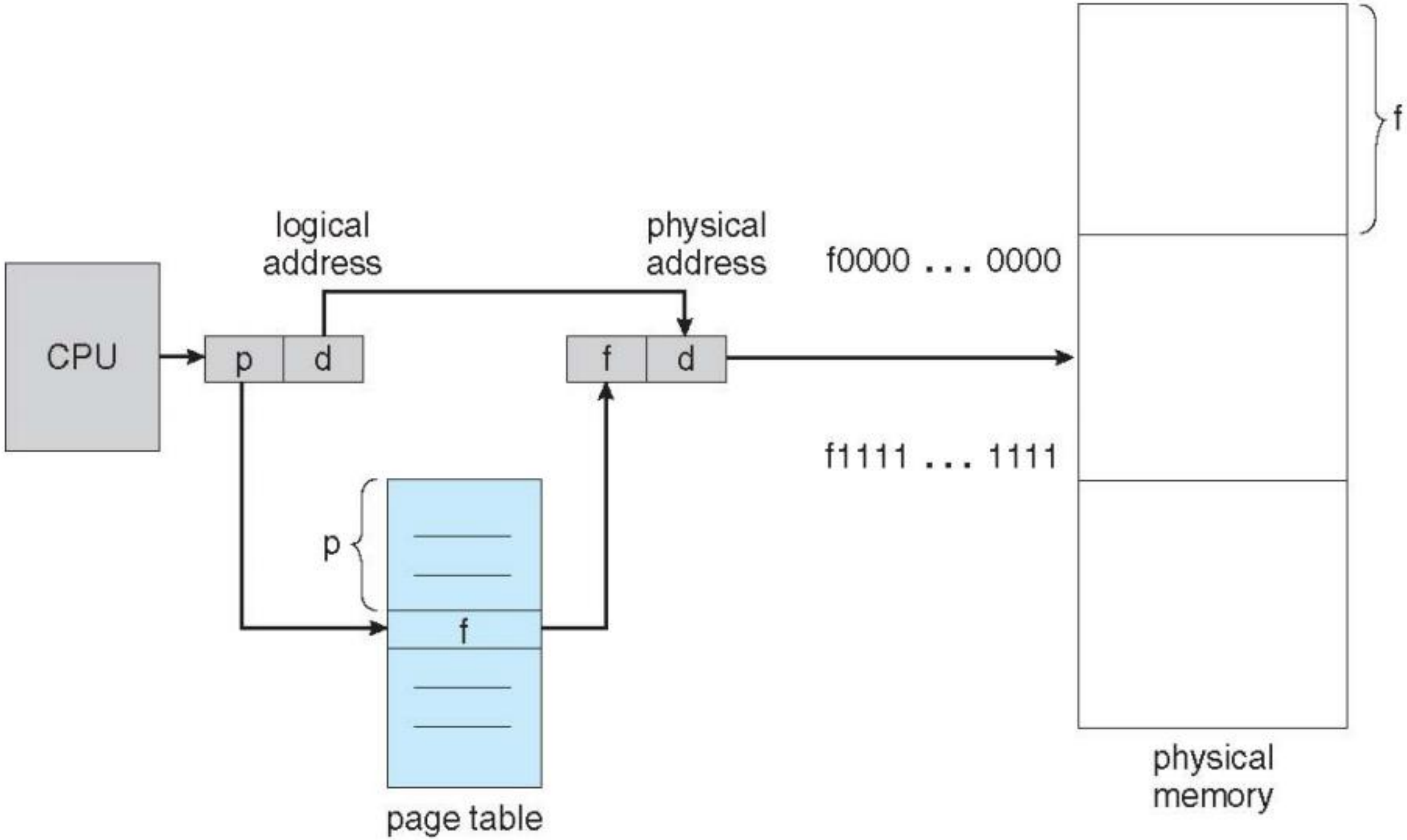


Segmentation Hardware

- Registers (Intel CPU)
  - CS: code segment
  - DS: data segment
  - SS: stack segment
  - ES: extra segment
  - FS and GS



Paging Hardware



TLB

- Page table could be kept in memory rather than paging hardware (if too big)
- TLB (translation look-aside buffer) caches the address translation (frequently accessed address)
  - if page number is in the TLB, no need to access the page table
  - if page number is not in the TLB, need to replace one TLB entry
  - TLB usually use a fast-lookup hardware cache called associative memory (memory that supports parallel search)
  - TLB is usually small, 64 to 1024 entries

Hardware-based Protection

- Something like tagged architecture
- Each page table entry has a present (aka. valid) bit
  - present: the page has a valid physical frame (block of same size as page in memory), thus can be accessed
- Each page table entry contains some protection bits
  - kernel/user, read/write, execution?, kernel-execution?
- Any violations of memory protection result in a trap to the kernel

How about other objects?

- File, external devices, network, …?
- We learnt some in “Access Control”
- Easier to control because they can separated logically & physically
  - E.g., Linux DAC for files
  - E.g., application-based network isolation through port

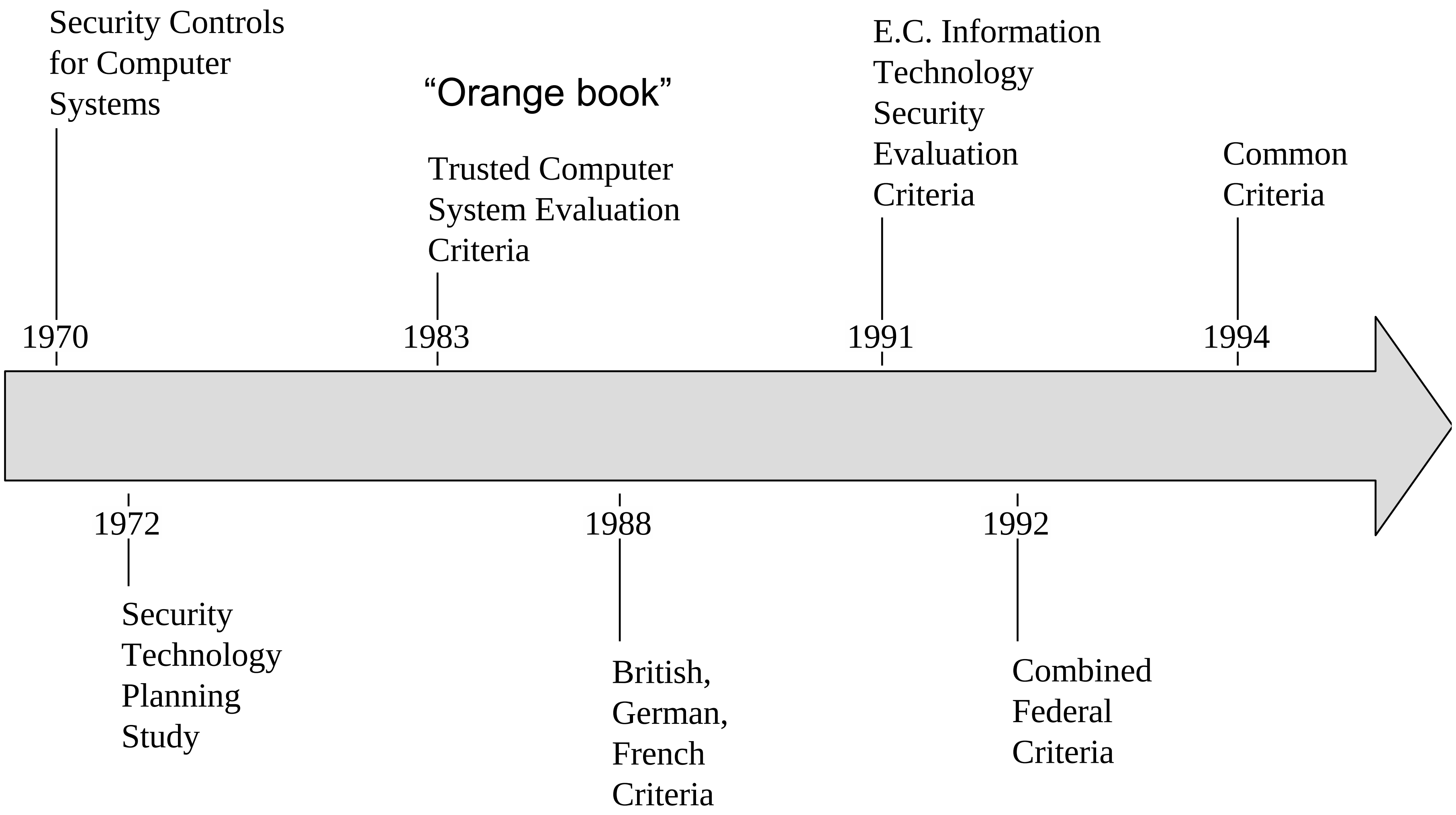
Objectives

- ~~Basic security functions provided by operating systems~~
- ~~System resources that require operating system protection~~
- ~~Operating system design principles~~
- ~~How operating systems control access to resources~~
- The history of trusted computing
- Characteristics of operating system rootkits
- *Formally verified kernel: seL4*

Trusted Systems

- A trusted system is one that has been shown to warrant some degree of trust that it will perform certain activities faithfully
- Characteristics of a trusted system:
  - A defined policy that details what security qualities it enforces
  - Appropriate measures and mechanisms by which it can enforce security adequately
  - Independent scrutiny or evaluation to ensure that the mechanisms have been selected and implemented properly

History of Trusted Systems

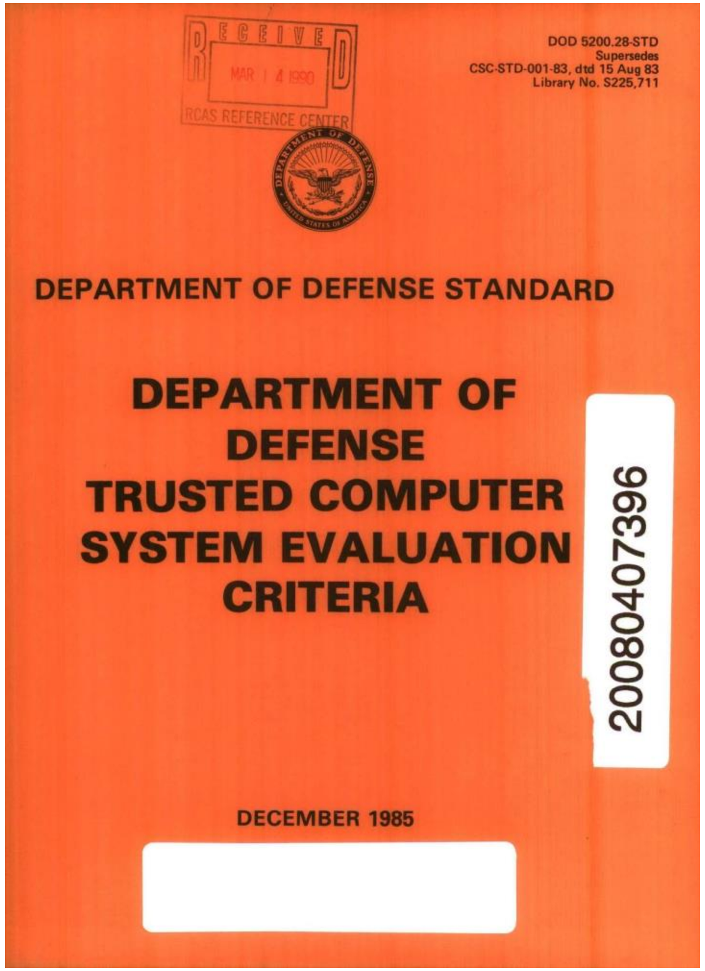


“Orange Book”

- Trusted Computer System Evaluation Criteria (TCSEC)
  - Drafted in late 1970s by DoD, released in 1980s
  - Specify functionality, design principles and evaluation methodology for trusted computer systems

Evaluation Assurance Level (for IT product)

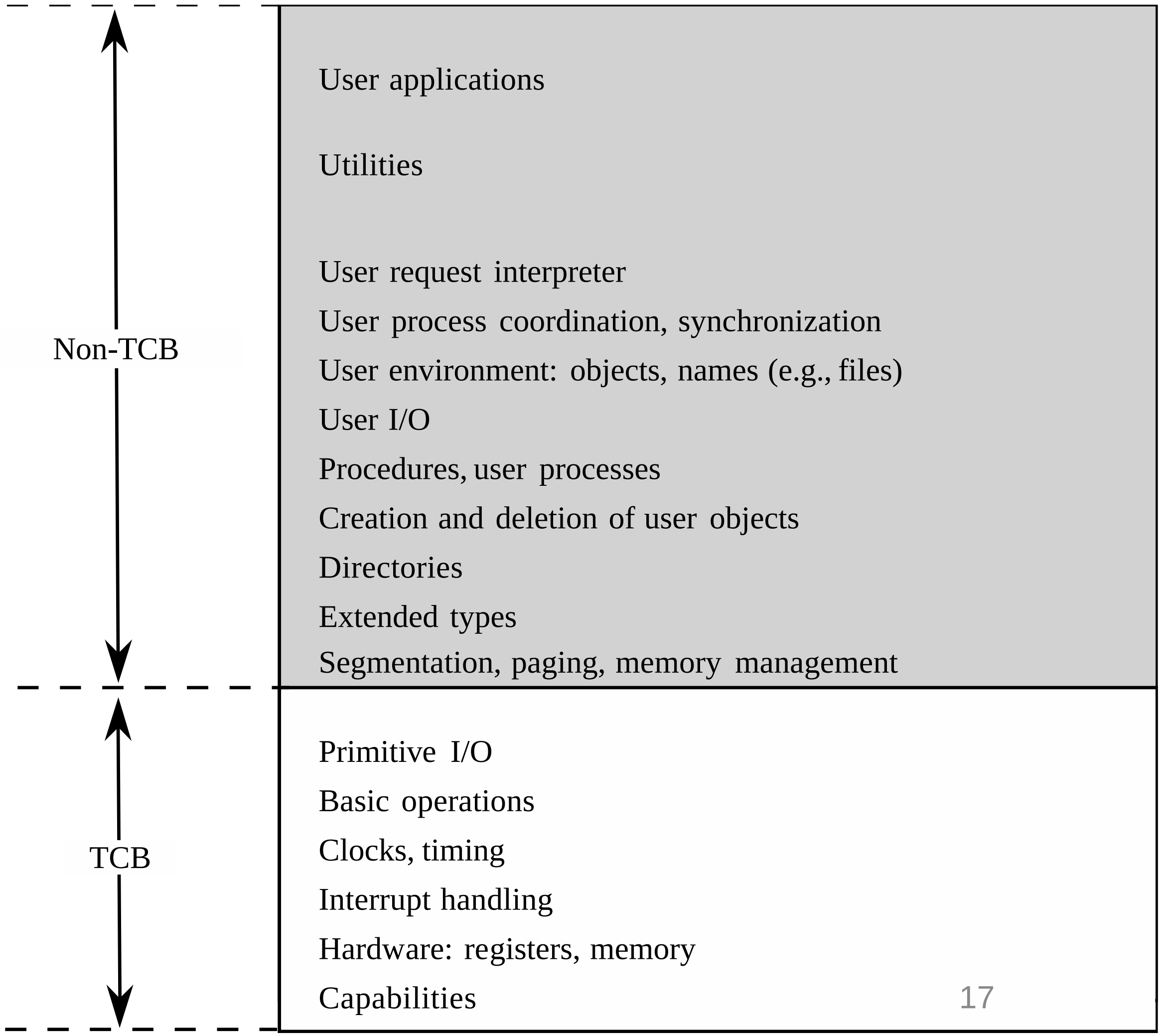
EAL	Name	*TCSEC
EAL1	Functionally Tested	
EAL2	Structurally Tested	C1
EAL3	Methodically Tested & Checked	C2
EAL4	Methodically Designed, Tested & Reviewed	B1
EAL5	Semiformally Designed & Tested	B2
EAL6	Semiformally Verified Design & Tested	B3
EAL7	Formally Verified Design & Tested	A1



Trusted Computing Base (TCB)

- TCB: everything necessary for a system to enforce its security policy

- Assuming you allow attacker to write all the non-TCB code, TCB won't be impaired.



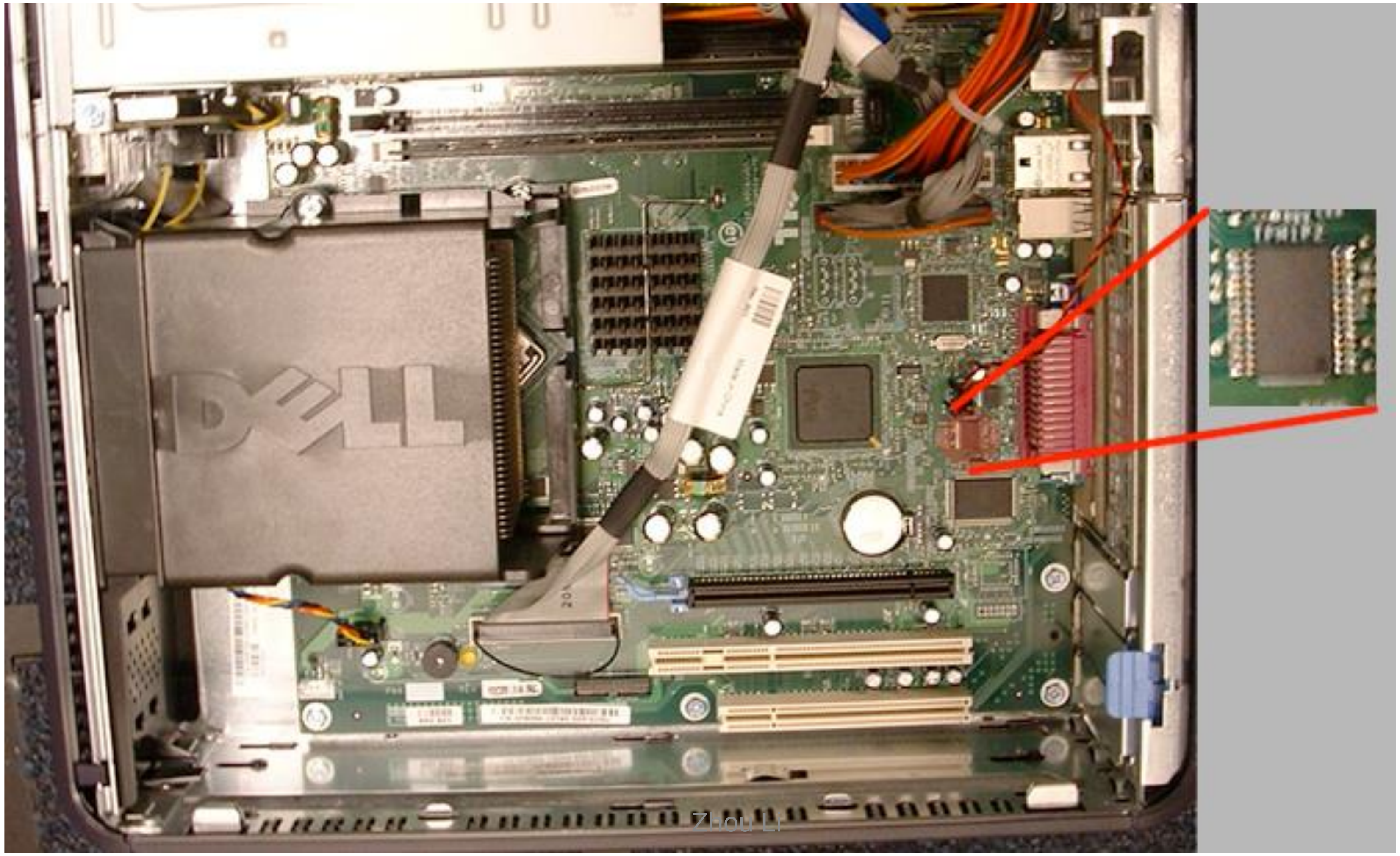
TCB implementations

- Security kernel
  - Small kernel (~10K LoC) between OS and hardware
- Secure startup
  - Ensure no malicious code can block or interfere with security enforcement
- Trusted path
  - An unforgeable connection by which the user can be confident of communicating directly with the OS
- Object reuse control
  - OS clears memory before reassigning it to ensure that leftover data doesn’t become compromised
- Audit
  - Trusted systems track security-relevant changes
  - Audit logs must be protected against tampering and deletion

Trusted Platform Module (TPM)

- Trusted hardware to support TCB
- Measure and attest the software running on a computer
- TPM brought *authenticated boot (secure startup)* into the mainstream
- It provides hardware support for *remote attestation (trusted path + verifying secure startup)*
- TPM offers few primitives
  - Measurement, cryptography, key generation, PRNG
  - Controlled by physical presence of the machine
  - BIOS is Core Root of Trust for Measurement (CRTM)
- More information: www.trustedcomputinggroup.org

Where are the TPMs?



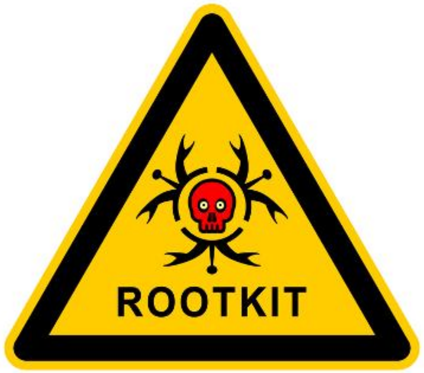
UCI Samueli
School of Engineering

Objectives

- ~~Basic security functions provided by operating systems~~
- ~~System resources that require operating system protection~~
- ~~Operating system design principles~~
- ~~How operating systems control access to resources~~
- ~~The history of trusted computing~~
- Characteristics of operating system rootkits
- *Formally verified kernel: seL4*

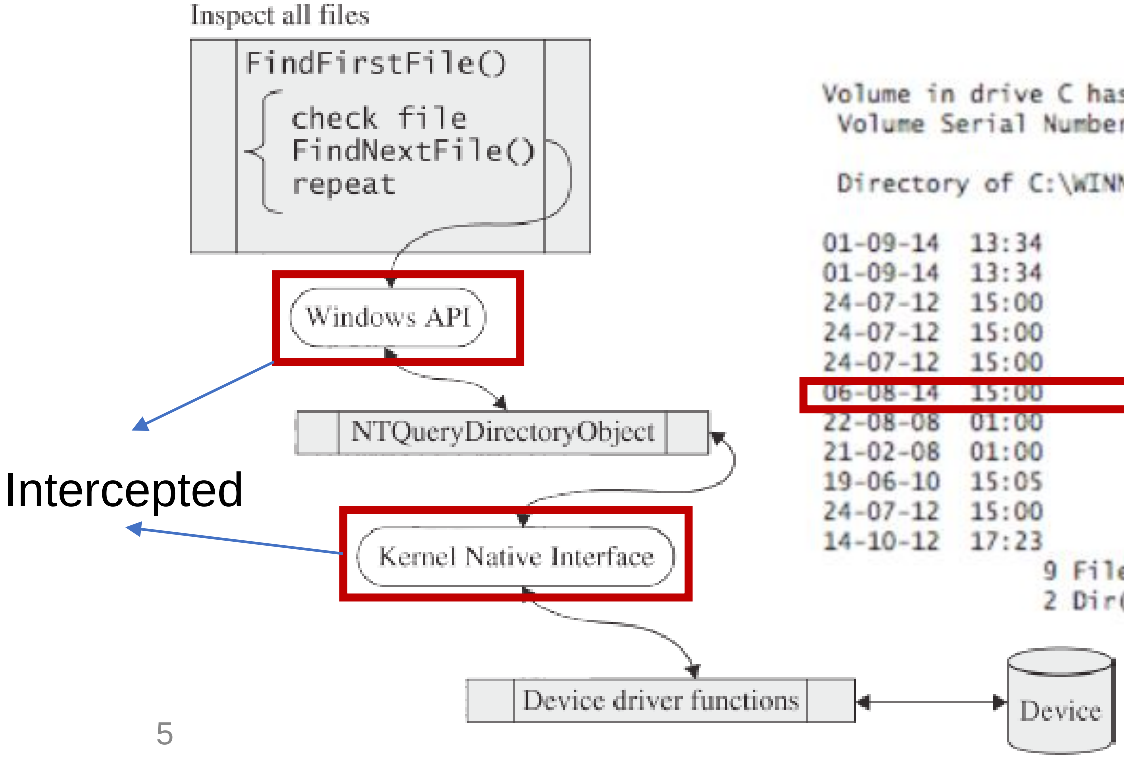
Rootkits

- A rootkit is a malicious software package that attains and takes advantage of root status or effectively becomes part of the OS
- Rootkits often go to great length to avoid being discovered or, if discovered and partially removed, to reestablish themselves
  - This can include intercepting or modifying basic OS functions



Hiding Rootkits

- If rootkit is at c:/winnt/apps/mal_code.exe
  - You or Anti-Virus can find it using Windows API and clean it
  - What if Windows API is changed by the rootkit?




```
Volume in drive C has no label.
 Volume Serial Number is E4C5-A911

 Directory of C:\WINNT\APPS

01-09-14  13:34         <DIR>          .
01-09-14  13:34         <DIR>          ..
24-07-12  15:00                82,944 CLOCK.AVI
24-07-12  15:00                17,062 Coffee Bean.bmp
24-07-12  15:00                    80 EXPLORER.SCF
06-08-14  15:00               256,192 mal_code.exe
22-08-08  01:00               373,744 PTDOS.EXE
21-02-08  01:00                   766 PTDOS.ICO
19-06-10  15:05                73,488 regedit.exe
24-07-12  15:00                35,600 TASKMAN.EXE
14-10-12  17:23               126,976 UNINST32.EXE
               9 File(s)        966,852 bytes
               2 Dir(s)  13,853,132,800 bytes free
```

Unmodified

```
Volume in drive C has no label.
 Volume Serial Number is E4C5-A911

 Directory of C:\WINNT\APPS

01-09-14  13:34         <DIR>          .
01-09-14  13:34         <DIR>          ..
24-07-12  15:00                82,944 CLOCK.AVI
24-07-12  15:00                17,062 Coffee Bean.bmp
24-07-12  15:00                    80 EXPLORER.SCF
22-08-08  01:00               373,744 PTDOS.EXE
21-02-08  01:00                   766 PTDOS.ICO
19-06-10  15:05                73,488 regedit.exe
24-07-12  15:00                35,600 TASKMAN.EXE
14-10-12  17:23               126,976 UNINST32.EXE
               8 File(s)        710,660 bytes
               2 Dir(s)  13,853,472,768 bytes free
```

Modified

Sony XCP Rootkit

- Identified by a security expert Mark Russinovich with a rootkit revealer that intercepts *NTQueryDirectoryObject* API.
- XCP rootkit was installed automatically from Sony music CD to prevent a user from copying the tunes.
  - Only Sony's music player can play the music
  - Blocks display of any program (Sony's) whose name begins with sys
  - It can be abused to hide virus like sysvirus-1
- Sony issued an uninstaller, but has serious bug

TDSS Rootkit

- A family of rootkit (TDL-1 through TDL-4)
- It installed filter code in the drivers
  - Drops all references to files begins with “tdl” (hide malicious files)
  - Blocks access to any disk volume, certain network ports
- It modifies a Windows registry API NTEnumerateKey to hide added registry keys
  - Modifying the first several bytes by inserting a jump to an extension (called splicing)
- Turned into botnet for TDL-3 (3 million infection)

Objectives

- ~~Basic security functions provided by operating systems~~
- ~~System resources that require operating system protection~~
- ~~Operating system design principles~~
- ~~How operating systems control access to resources~~
- ~~The history of trusted computing~~
- ~~Characteristics of operating system rootkits~~
- *Formally verified kernel: seL4*

What is seL4 and Why Should I Care?

- seL4, the secure embedded L4 microkernel, is the first formally verified microkernel, which offers fundamental software separation properties, and provides new opportunities to build assured computer systems.
- The seL4 Center of Excellence is a multi-organization group that was formed to increase collaboration between seL4 contributors, adoption of seL4, and the maturity of seL4 by increasing stability, continuing adoption of modern software engineering practices, and adding formally verifiable features and software libraries.
- seL4 is one of the fastest operating system kernels that has been designed for security and safety, and opportunities for those with seL4 expertise exist in several areas:
  - seL4 is the basis for many next generation secure hardware-software stacks
  - Cyberattack protection for autonomous vehicles (drones, helicopters, land robots, trucks)
  - ISOSCELES architecture – a reference implementation for mixed-criticality medical and Internet of Things (IoT) systems designs has been developed over seL4
  - Securing self-driving vehicles in the commercial sector
  - These are just a few examples. For detailed information about exciting seL4-related efforts in government, industry, and academia, please see the presentations from the 2018 seL4 Summit:

    https://www.sel4-us.org/summit/#agenda

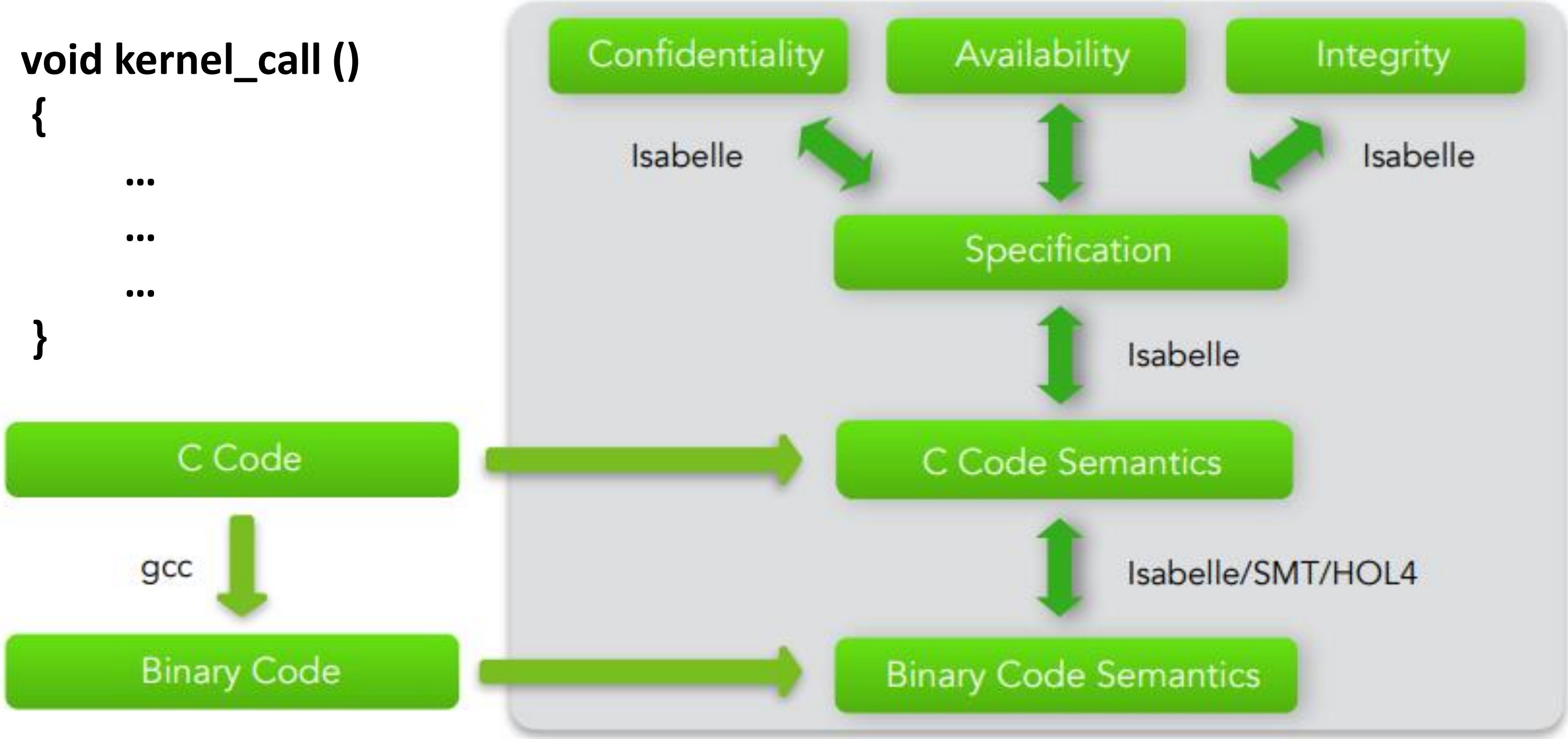
seL4 formal proof

$$\{\lambda s.\ s=s_0\}\ \mathit{kernel\_call\_A}\ ()\ \{\lambda s.\ \mathit{integrity}\ p\ s\ s_0\}$$

$$s \sim_p t \ \wedge\ \mathit{reachable}\ A\ s\ s' \wedge\ \mathit{reachable}\ A\ t\ t' \Longrightarrow s' \sim_p t'$$

formal, high-level, functional description of the expected behaviour

```
void kernel_call ()
{
        …
        …
        …
}
```

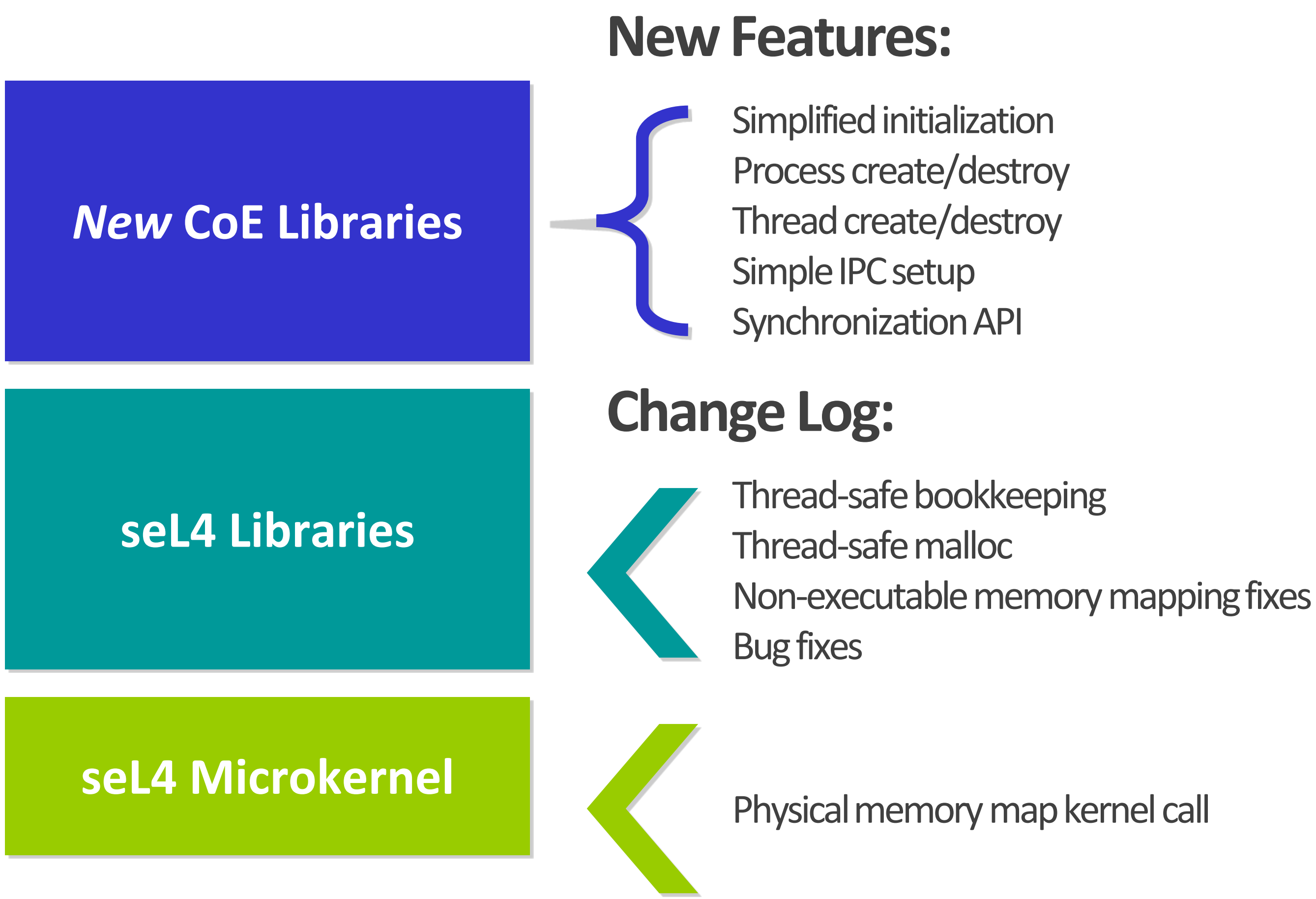


binary is correct with respect to spec and enforces isolation

Accomplishments of First Year seL4 Interns

- Interns in the 2018 seL4 Internship Program worked side-by-side with engineers to produce the first official U.S.-based release of the seL4 software
  https://github.com/sel4-us/

- Design Goals Included:
  - **Simplicity** – Clear and readable API
  - **Security** – Reduce chances for error/misconfiguration
  - **Generic** – Do not strictly enforce a specific architecture
  - **Maintainable** – Leave ample room for project growth



seL4 and the CoE 2019 Internship Program

- Where can I find additional seL4 information?
  - U.S.-based seL4 Center of Excellence: https://www.sel4-us.org/
  - Wiki containing an overview of seL4 technology: https://www.sel4-us.org/wiki/doku.php
  - seL4 Summit Hands-on Raspberry Pi3 training: https://www.sel4-us.org/summit/training/
  - U.S.-based seL4 public GitHub repository: https://github.com/sel4-us/
  - Australian based seL4 website: http://sel4.org/

- The seL4 CoE plans to continue its Internship Program during Summer 2019. For more information, please contact: mail@sel4-us.org (U.S. citizenship or Permanent Residency required).

Summary

- OSs have evolved from supporting single users and single programs to many users and programs at once
- Resources that require OS protection: memory, I/O devices, programs, and networks
- OSs use layered and modular designs for simplification and to separate critical functions from noncritical ones
- Resource access control can be enforced in a number of ways, including virtualization, segmentation, hardware memory protection, and reference monitors
- Rootkits are malicious software packages that attain root status or effectively become part of the OS
- New OS security models enabled by new platforms

Slides Credit

- Operating Systems (Fall/Winter 2018), Yajin Zhou, ZJU
- Security in computing 5th edition, Textbook Slides