NTNU | Norwegian University of Science and Technology

# Operating Systems

Lecture overview and Q&A Session 6 – 21.2.2022

Michael Engel

# Lectures 9 and 10

## Memory management

- Memory allocation policies, strategies and problem def.
- Static vs. dynamic memory allocation
- Placement strategies and fragmentation
- Segmentation vs. paging

## Virtual memory

- Demand paging
- Replacement strategies
- Thrashing
- Working set model

# Memory allocation policies, strategies and problem def.

- Policies
  - Placement: Which area of memory should be allocated?
  - Fetch: When should we swap in memory contents?
  - Replacement: Which memory should be swapped out?

- Memory allocation problem
  - Find allocation of memory to satisfy the requirements of
    - User processes: code, data, stack, heap
    - Operating system: code and data, process control blocks, data buffers for input/output, …

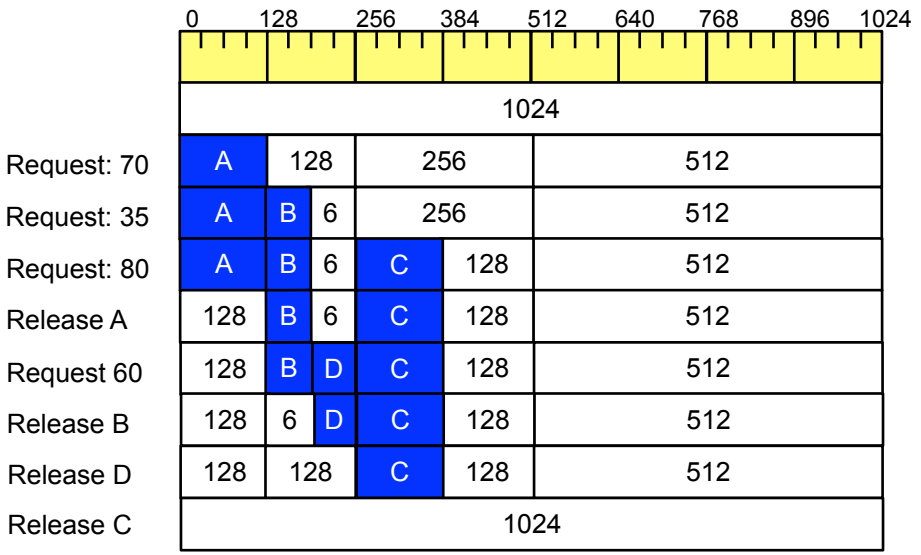
- Memory allocation methods are necessary!

# Static vs. dynamic memory allocation

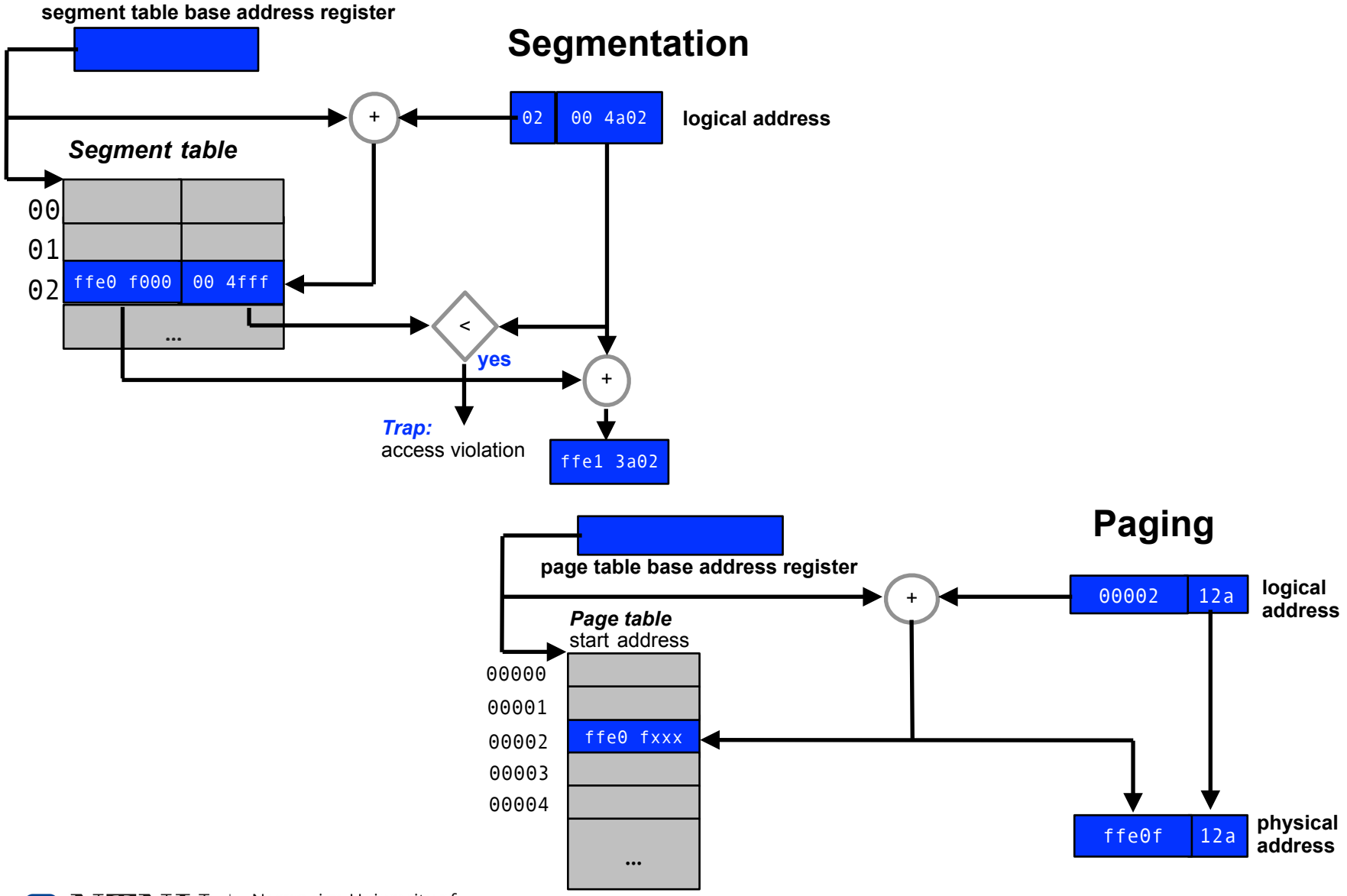
- Static allocation
  - use fixed memory areas for OS and user processes
- Problems:
  - Limited degree of multiprogramming
  - Limitation of other resources (e.g. I/O bandwidth)
  - Unused OS memory cannot be used by applications
- Dynamic allocation
  - Free list management: bitmaps
  - Linked lists / linked lists in free memory
  - Releasing memory
    - Problem: merging adjacent free gaps

# Placement strategies and fragmentation

- Different algorithms: first fit, next fit, best fit, worst fit
- ***Buddy method***
  - split memory dynamically into areas of a size $2^n$
- *Problem:* External fragmentation
  - Allocations create memory fragments outside of the allocated memory areas which cannot be used
  - Problem with all list based strategies, e.g. first fit, best fit, ...
- *Problem:* Internal fragmentation
  - Unused memory inside of allocated memory areas
  - Problem e.g. with the buddy allocator
  - since request sizes are rounded up to the next power of 2

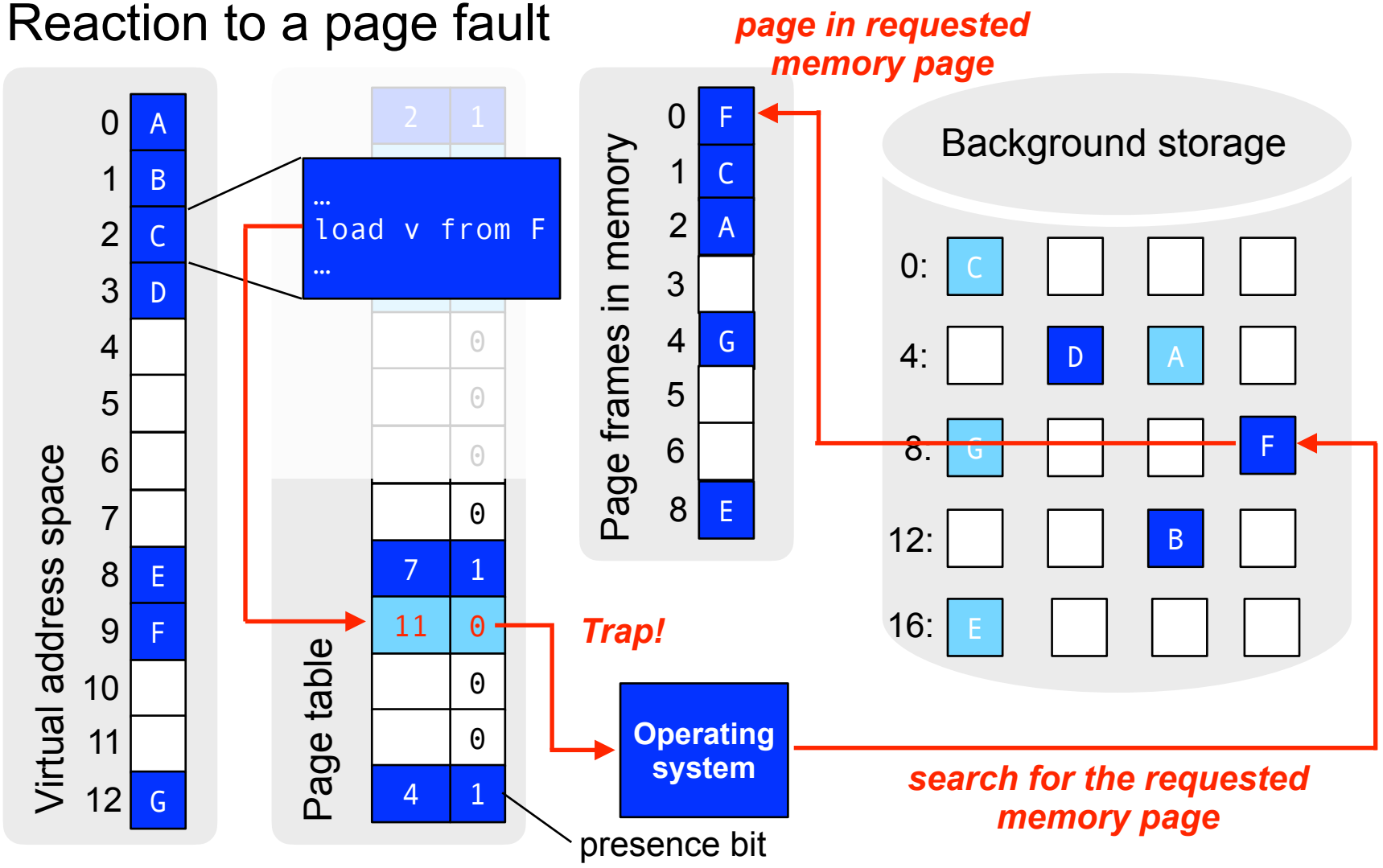


# Segmentation vs. paging



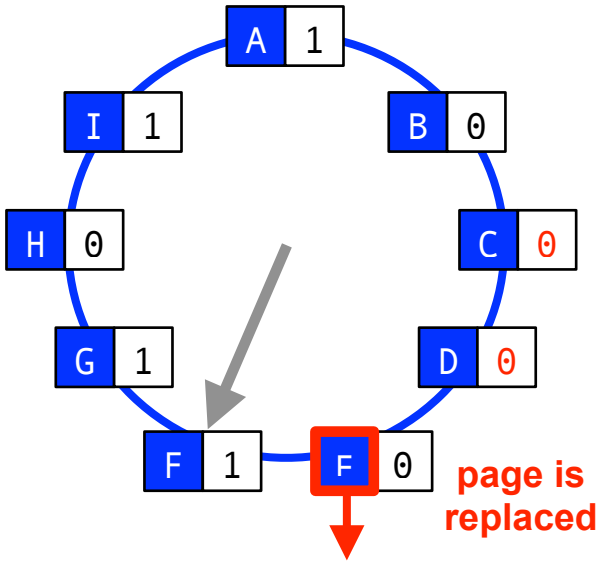
# Virtual memory: demand paging

- Reaction to a page fault



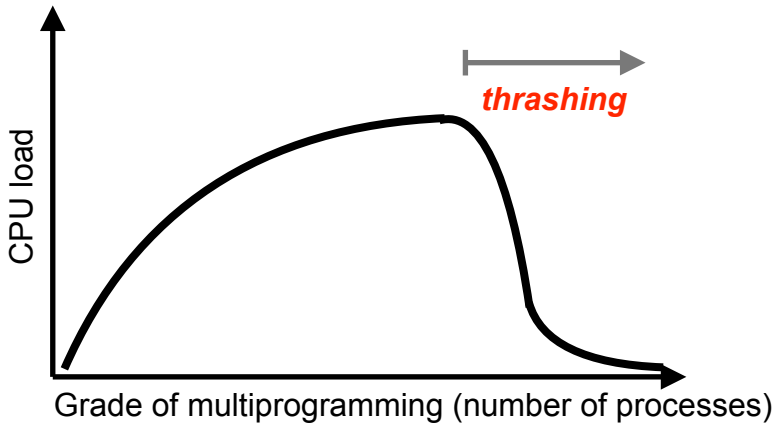
# Replacement strategies

- What if no free page frame available when a request comes in?
- **Sequence of events:**
  - *page fault, page out a page frame, page in requested page*
  - Repeat the memory access
- **Problem:**
  - Which page to choose to be paged out (the “victim”)?
- Replacement strategies:
  - first in, first out: Replace the *oldest page*
  - optimal: Time until next access to the respective page
  - LRU: Time since last access to the page
  - Second chance: clock and reference bits



# Thrashing

- A page that was paged out is accessed immediately after the page out happened
  - The process spends more time waiting to handle the page faults than with its own execution




- Solution 1: swapping of processes
  - Inactive processes do not require page frames
  - Can be temporarily swapped out to disk
- Solution 2: working set model
  - 80/20 rule: 80% of code uses only 20% of memory pages
  - *approximate* set of pages really needed by a process (***working set***): **WSclock** algorithm

# Overview Theoretical Exercise 4

Memory management and virtual memory

**Why?**

- Virtual memory is the basis of all modern computers
  - Flexibility *and*
  - Protection
- Different approaches and algorithms have significant impact on the performance
- Multi-level virtual memory schemes enable *efficient virtualization* as basis of Cloud computing

# The forum, finally!

- Discourse server is running since last Tuesday
  - open source solution (https://github.com/discourse/discourse)
  - IDI has provided us with a VM (thanks!)
  - https://tdt4186.idi.ntnu.no
- You should have received an invitation email
  - If not – let us know!
  - Currently "only" about 180 users registered
    - but ~420 on Blackboard