Embedded Operating Systems

Che-Wei Chang

chewei@mail.cgu.edu.tw

Department of Computer Science and Information Engineering, Chang Gung University

Introduction of Embedded Systems

Definition of Embedded Systems

- It is difficult to define embedded systems
  - An embedded system is a digital system
  - An embedded system has computing processors
  - An embedded system runs dedicated functions
  - An embedded system is frequently used as a controller

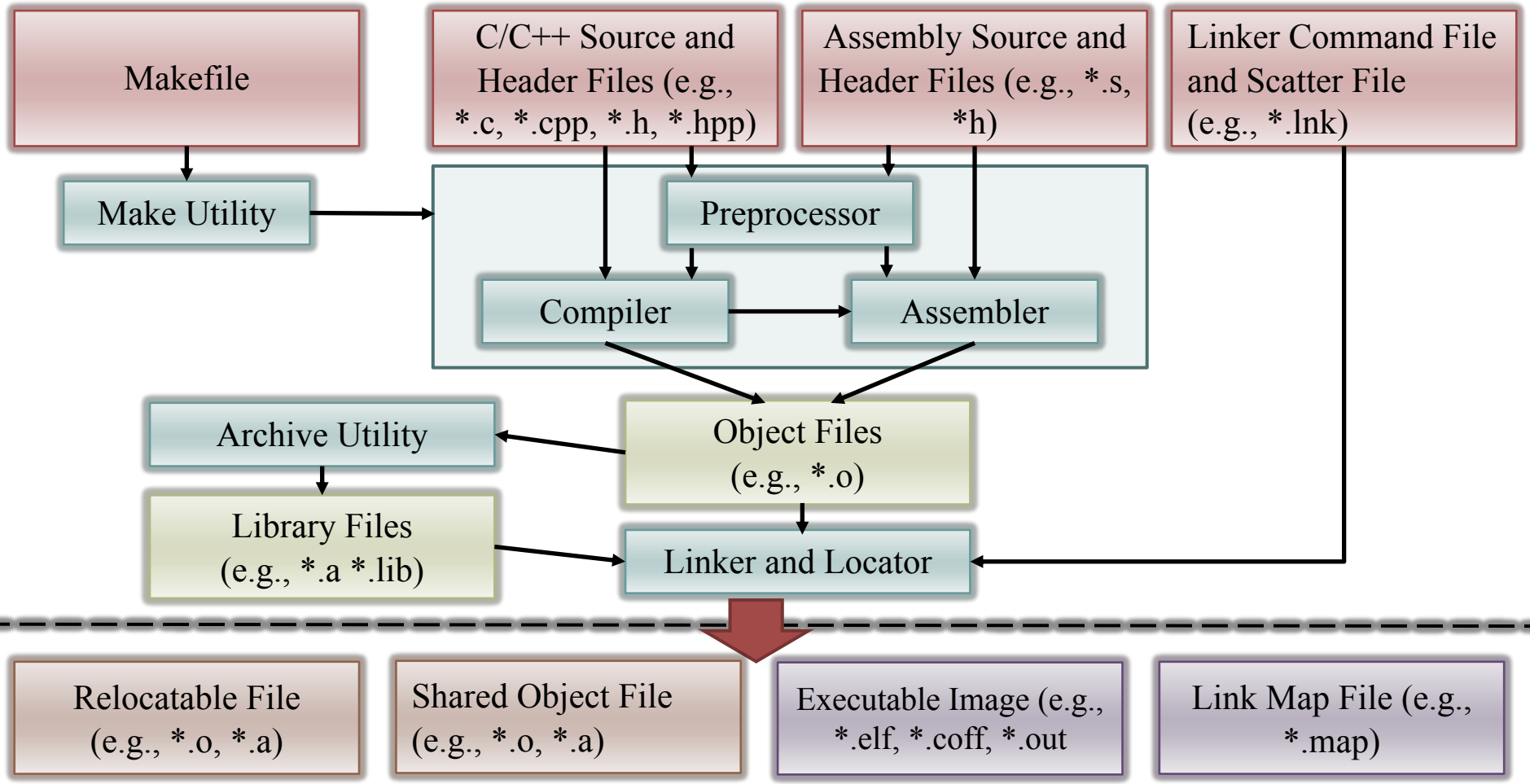
- An embedded system is a computer system with some dedicated functions within a larger mechanical or electrical system, often with real-time computing constraints. - From Wikipedia

Common Performance Metrics

- Unit Cost: the monetary cost of manufacturing each copy of the system
- NRE Cost (Non-Recurring Engineering cost): the one-time monetary cost of designing the system
- Size: the physical space required by the system
- Performance: the execution time or throughput of the system
- Power: the amount of power consumed by the system
- Flexibility: the ability to change the functionality of the system without incurring heavy NRE cost
- Time-to-Market: the time required to develop a system to the point that it can be released and sold to customers
- Maintainability: the ability to modify the system after its initial release

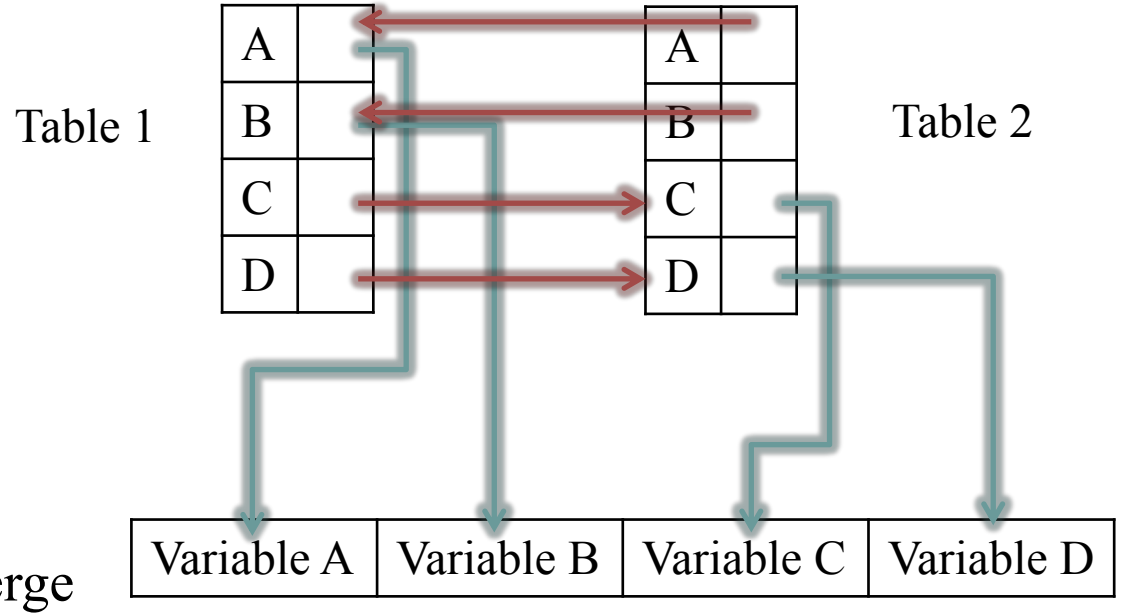
Development Tools of Embedded Systems

Creating an Image for a Target Embedded System



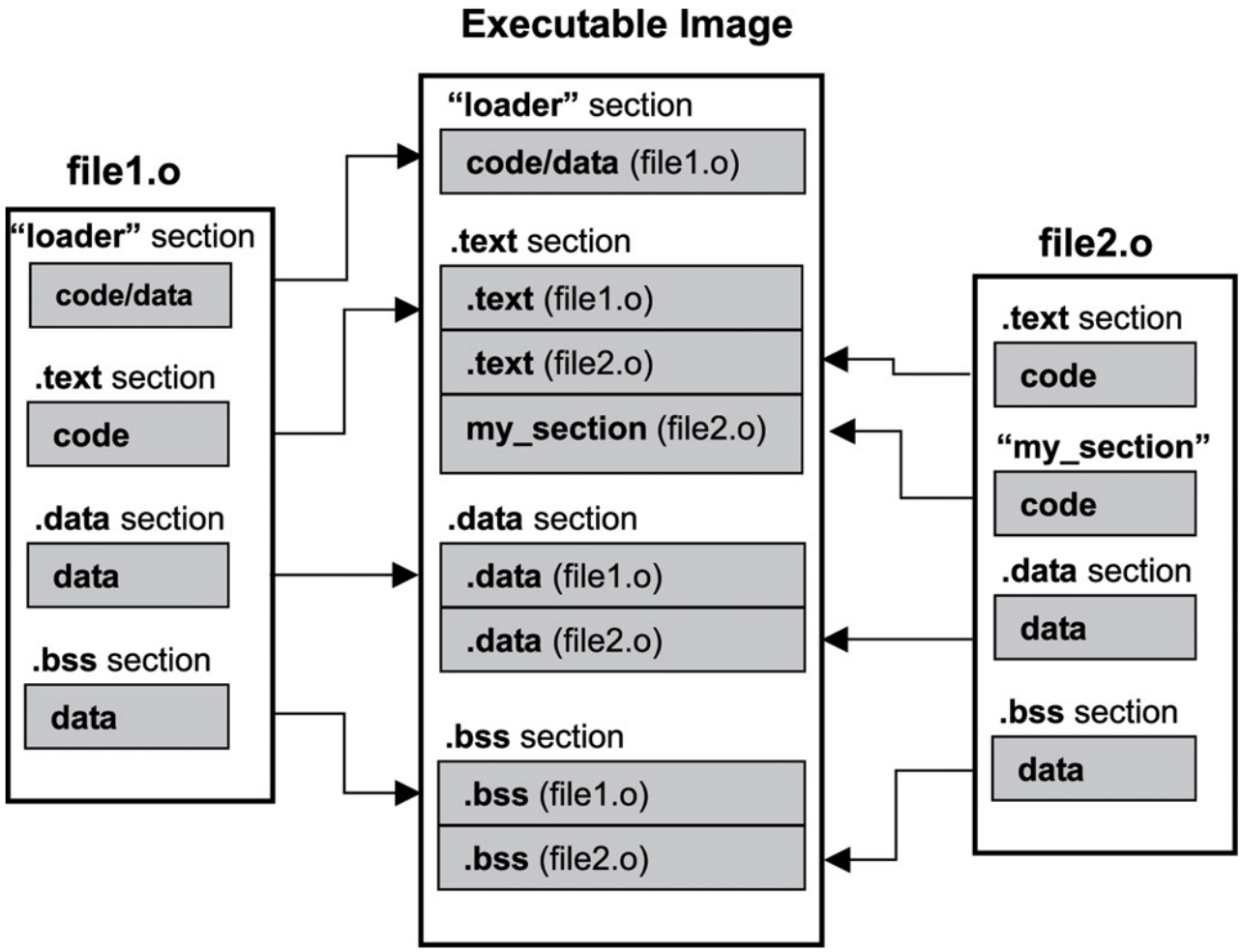
Symbol Resolution and Relocation

- Symbol Resolution
  - Resolving references across symbols
  - Merging multiple symbol tables into one



- Relocation
  - Performing section merge
  - Resolving all resolvable relocation
  - Replacing symbolic references with actual addresses (binding)

Combining Input Sections into an Executable Image



A Case Study: ADS Scatter File

```
LOAD_ROM 0x0000 0x8000
{
        ROM 0x0000 0x8000
        {
                part1.o (+RO)
        }
        SRAM 0x8000 0x8000
        {
                part2.o (+RO)
        }
        DRAM 0x200000 0x400000
        {
                * (+RW, +ZI)
        }
}
```

Instructions are kept in ROM

Critical instructions are moved to SRAM

All data are moved to DRAM

Embedded System Initialization

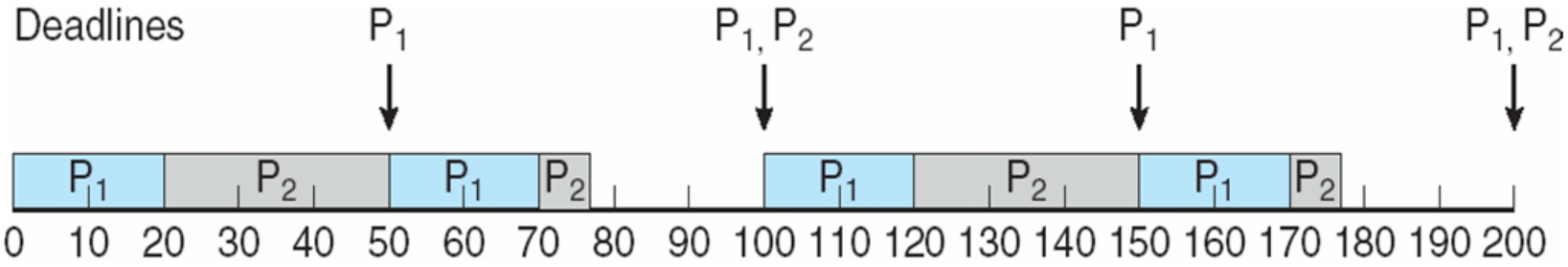
- Image Transfer from the Host to the Target System
- Target System Tools
  - Embedded Loader
  - Embedded Monitor
  - Target Debug Agent
- Target Boot Process
- Target System Software Initialization Sequence

Real-Time Scheduling

A Static Scheduling Algorithm—Rate Monotonic Scheduling

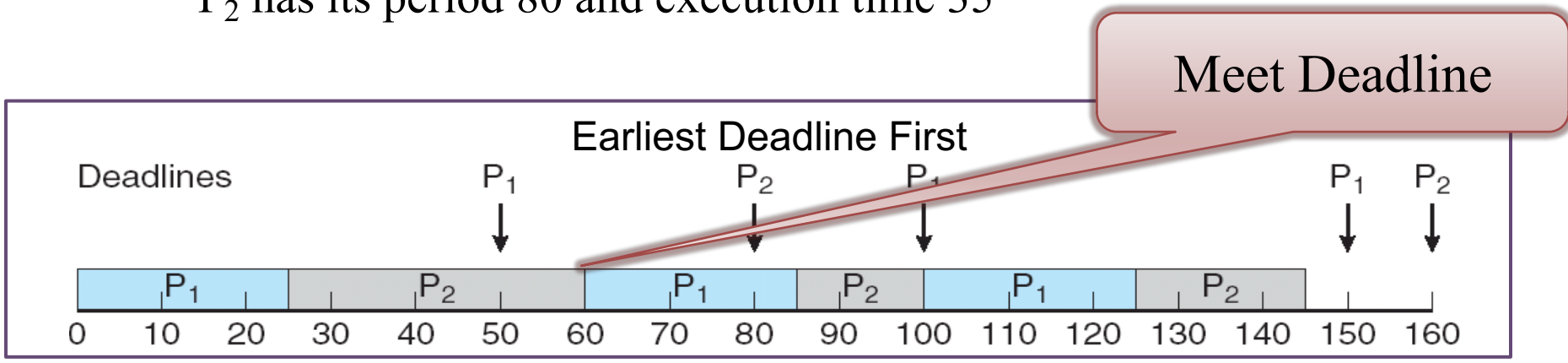
- A static priority is assigned to each task based on the inverse of its period
  - A task with shorter period ➔ higher priority
  - A task with longer period ➔ lower priority
  - For example:
    - P_1 has its period 50 and execution time 20
    - P_2 has its period 100 and execution time 37

    ➔P_1 is assigned a higher priority than P_2



A Dynamic Scheduling Algorithm—Earliest Deadline First Scheduling

- Dynamic priorities are assigned according to deadlines
  - The earlier the deadline, the higher the priority
  - The later the deadline, the lower the priority
  - For example:
    - P_1 has its period 50 and execution time 25
    - P_2 has its period 80 and execution time 35

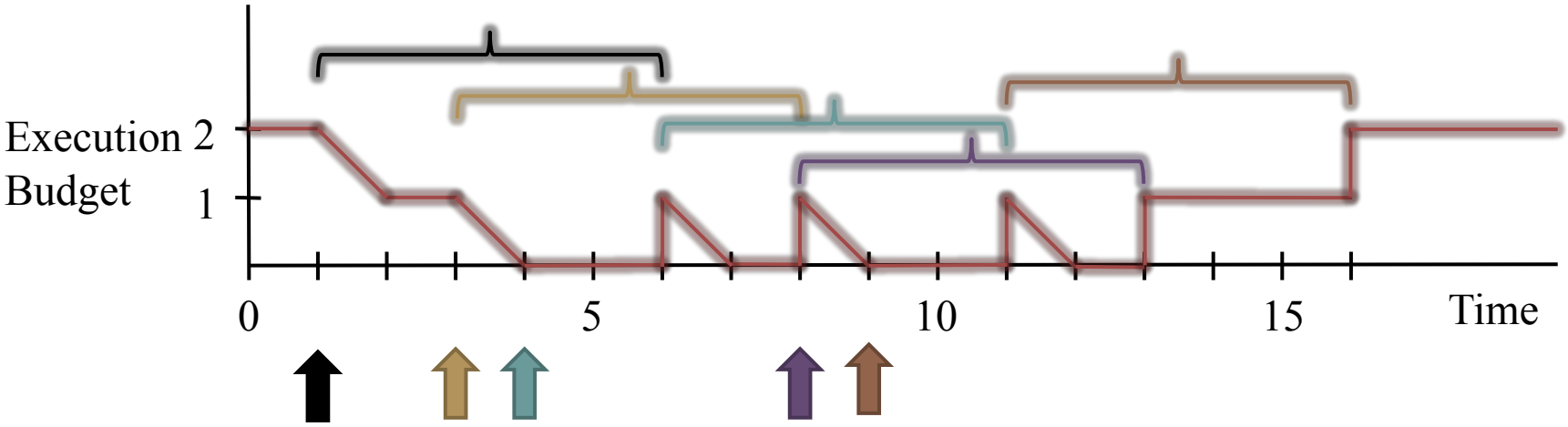


Priority Ceiling Protocol (PCP)

- The priority ceiling of a semaphore is the priority of the highest priority task that may lock the semaphore
- The Basic Priority Inheritance Protocol + Priority Ceiling
- A task *J* may successfully lock a semaphore S if S is available, and the priority of *J* is higher than the highest priority ceiling of all semaphores currently locked by tasks other than *J*
- Priority inheritance is transitive

An Example of Sporadic Server

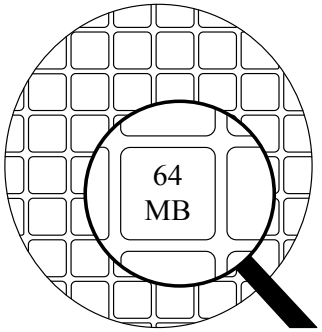
- A sporadic server has a replenishment period 5 and an execution budget 2
- Each event consumes the execution 1
- Events arrive at 1, 3, 4, 8, 9



Storage of Embedded Systems

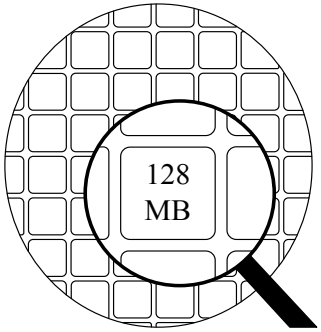
Single-Level Cell (SLC) vs Multi-Level Cell (MLC) Flash

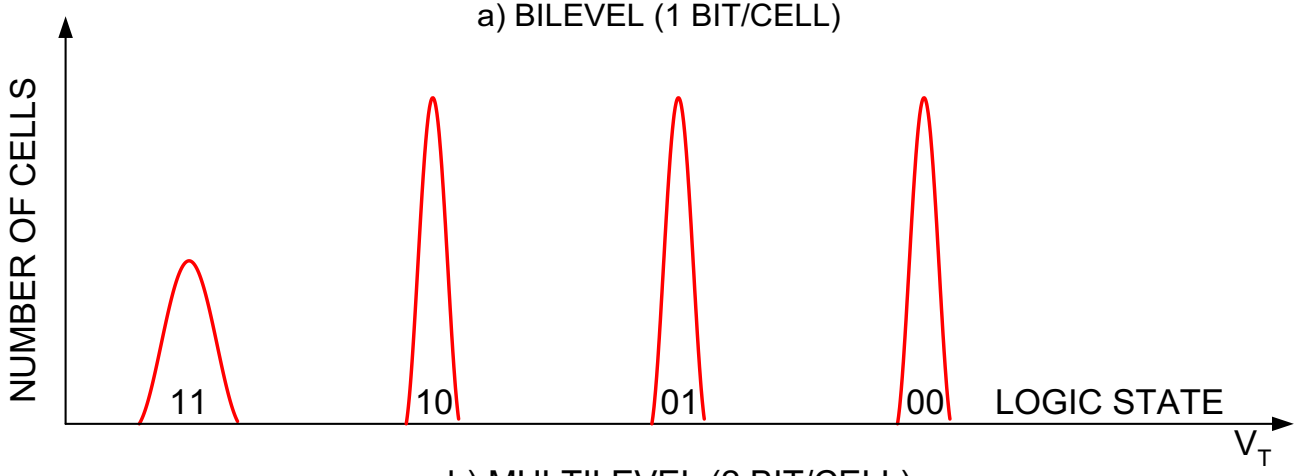
SLC Flash



a) BILEVEL (1 BIT/CELL)

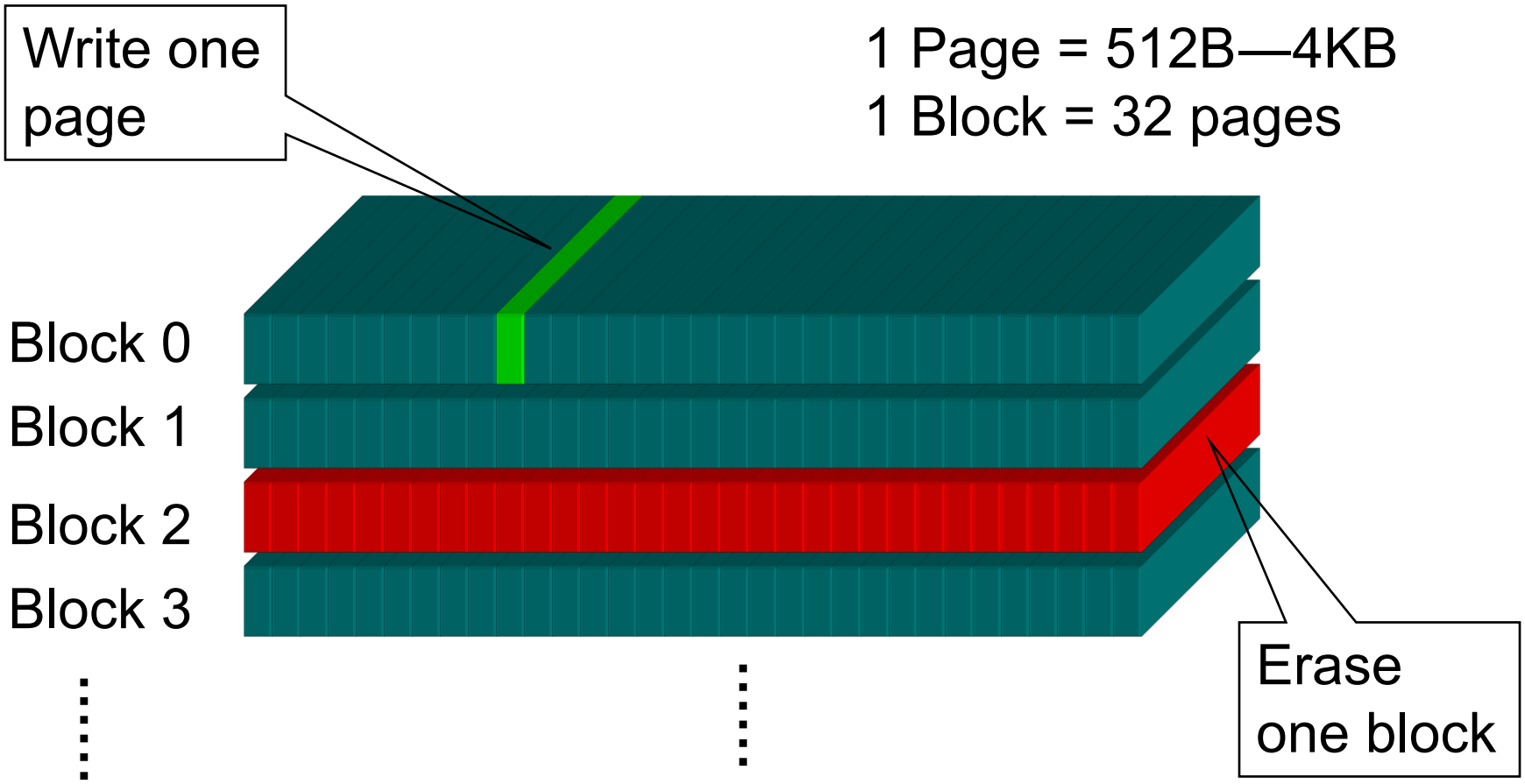
MLC Flash



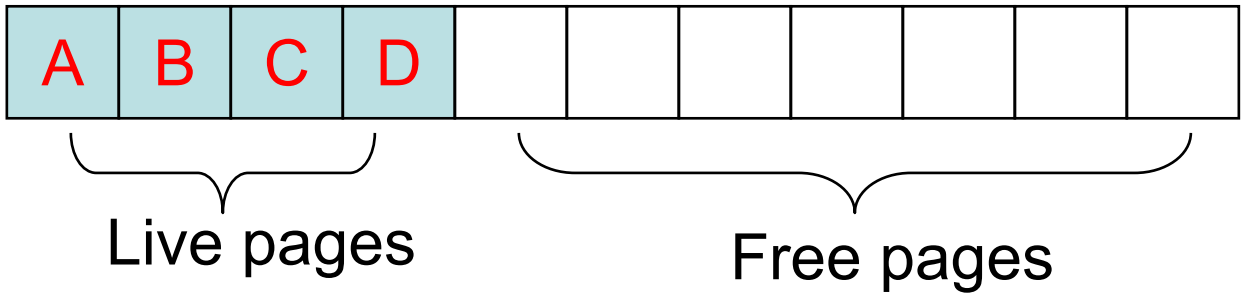


b) MULTILEVEL (2 BIT/CELL)

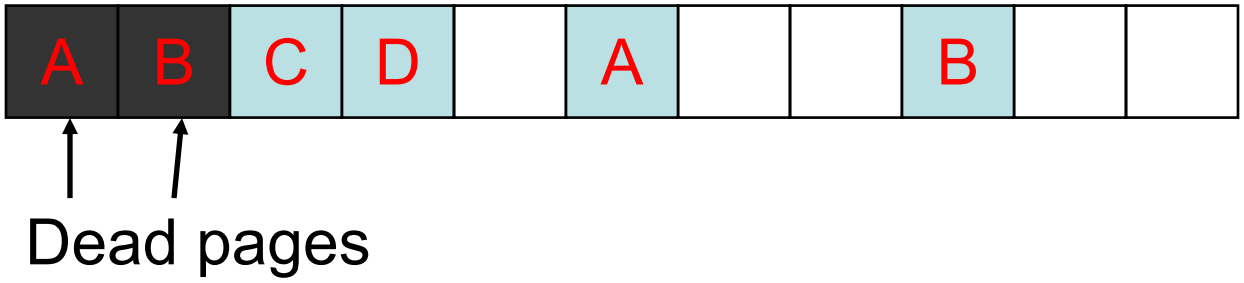
Page Write and Block Erase



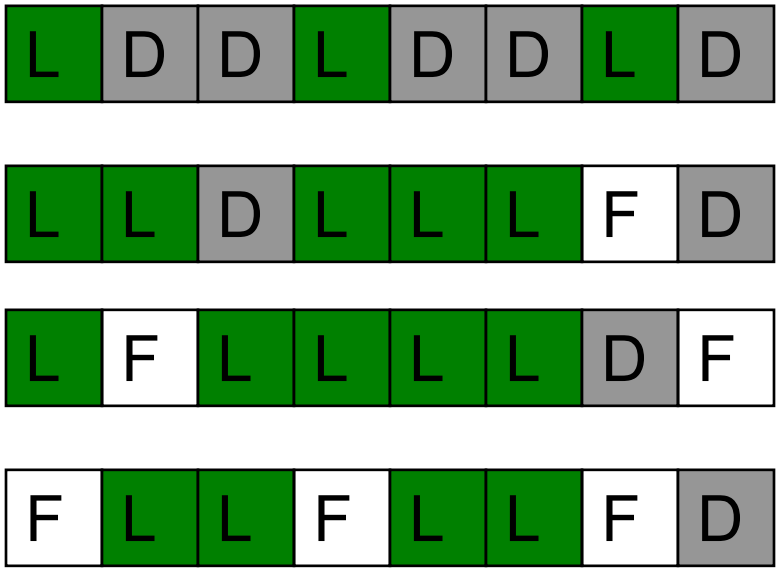
Out-Place Update



Suppose that we want to update data A and B…



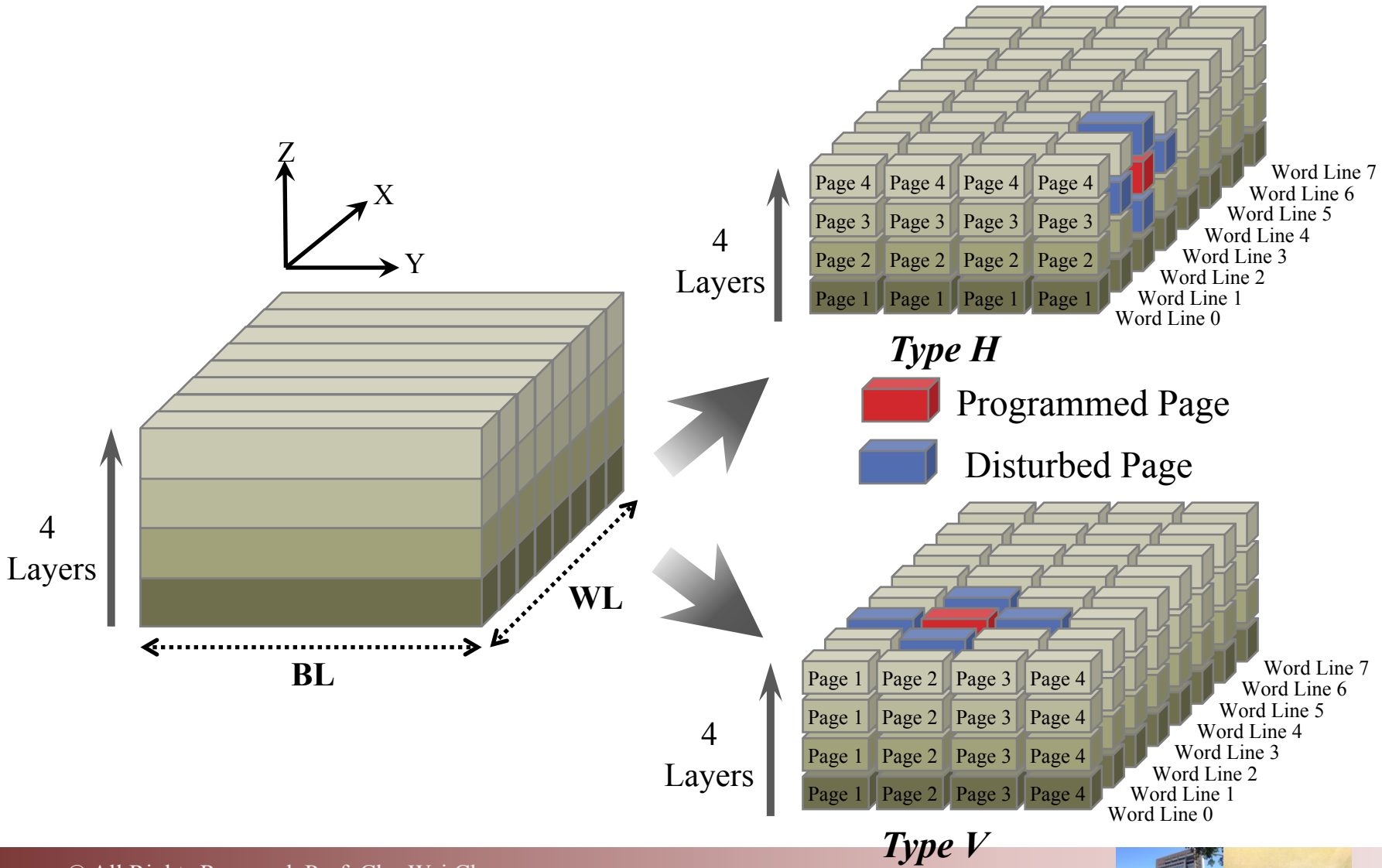
Garbage Collection



This block is to be recycled
(3 live pages and 5 dead pages)

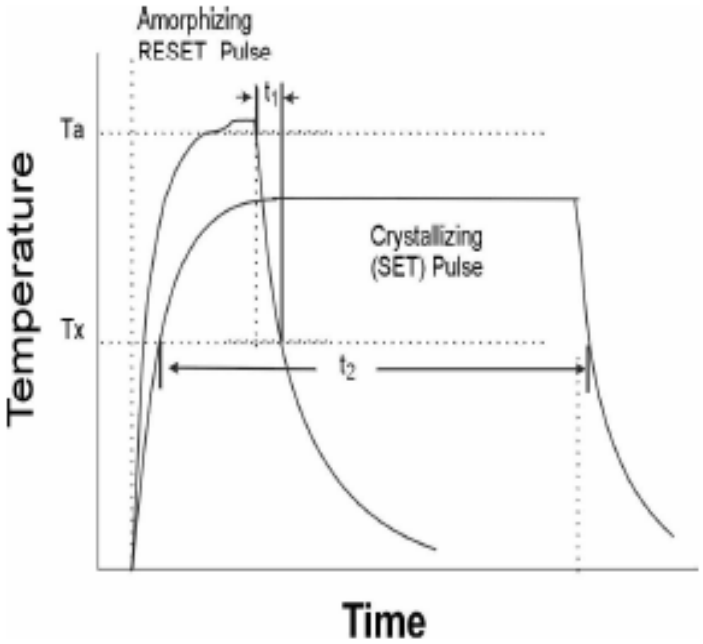
- A live page
- A dead page
- A free page

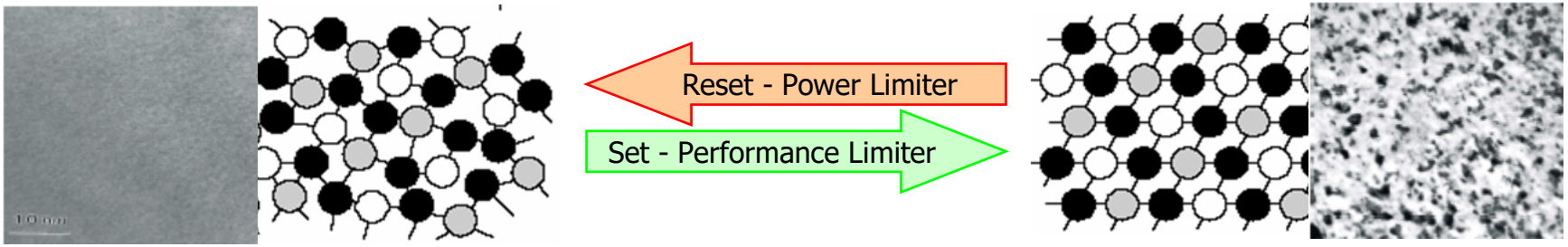
Deteriorated Disturb on 3D Flash



Phase Change Memory (PCM)

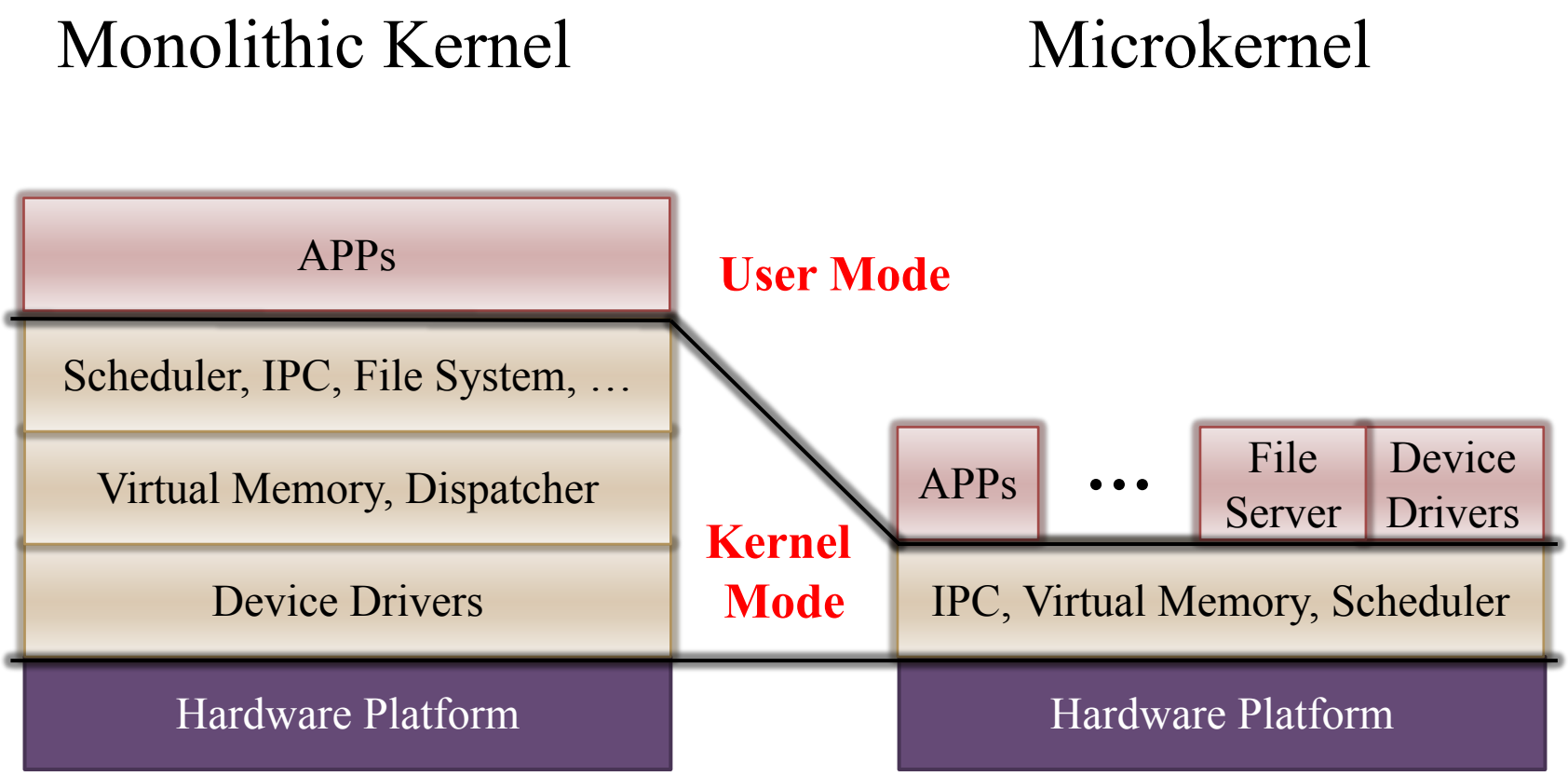
- PCM is a non-volatile memory (NVRAM)
- PCM employs a reversible phase change in materials to store information.
- PCM exploits differences in the electrical resistivity of a material in different phases



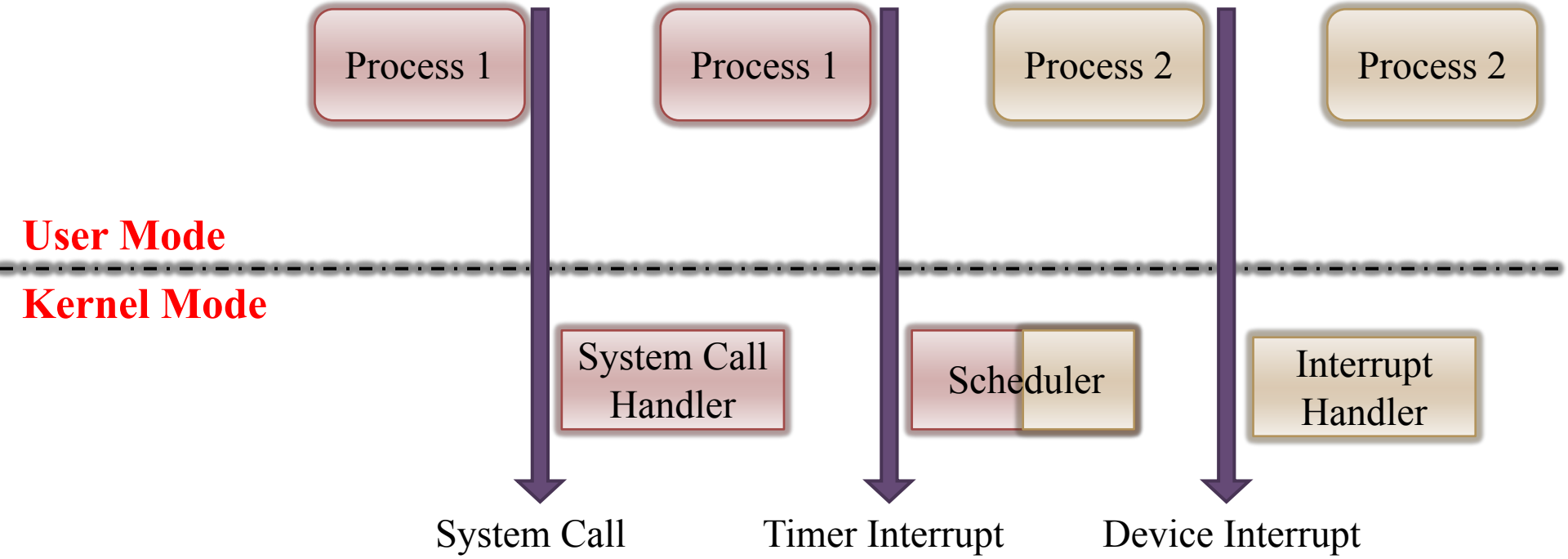


Linux and uC/OSII Environments

Monolithic Kernel and Microkernel



Transitions between User and Kernel Modes in Linux



Access to I/O Hardware

- Devices registers which can be accessed by the host
  - The data-in register is read by the host to get the input
  - The data-out register is written by the host to send the output
  - The status register contains bits which indicate device states
  - The control register is written by the host to send commands
- Methods to access devices with their addresses
  - Direct I/O instructions
  - Memory-mapped I/O
    - Device data and command registers mapped to processor address space
    - Especially for large address spaces (graphics)

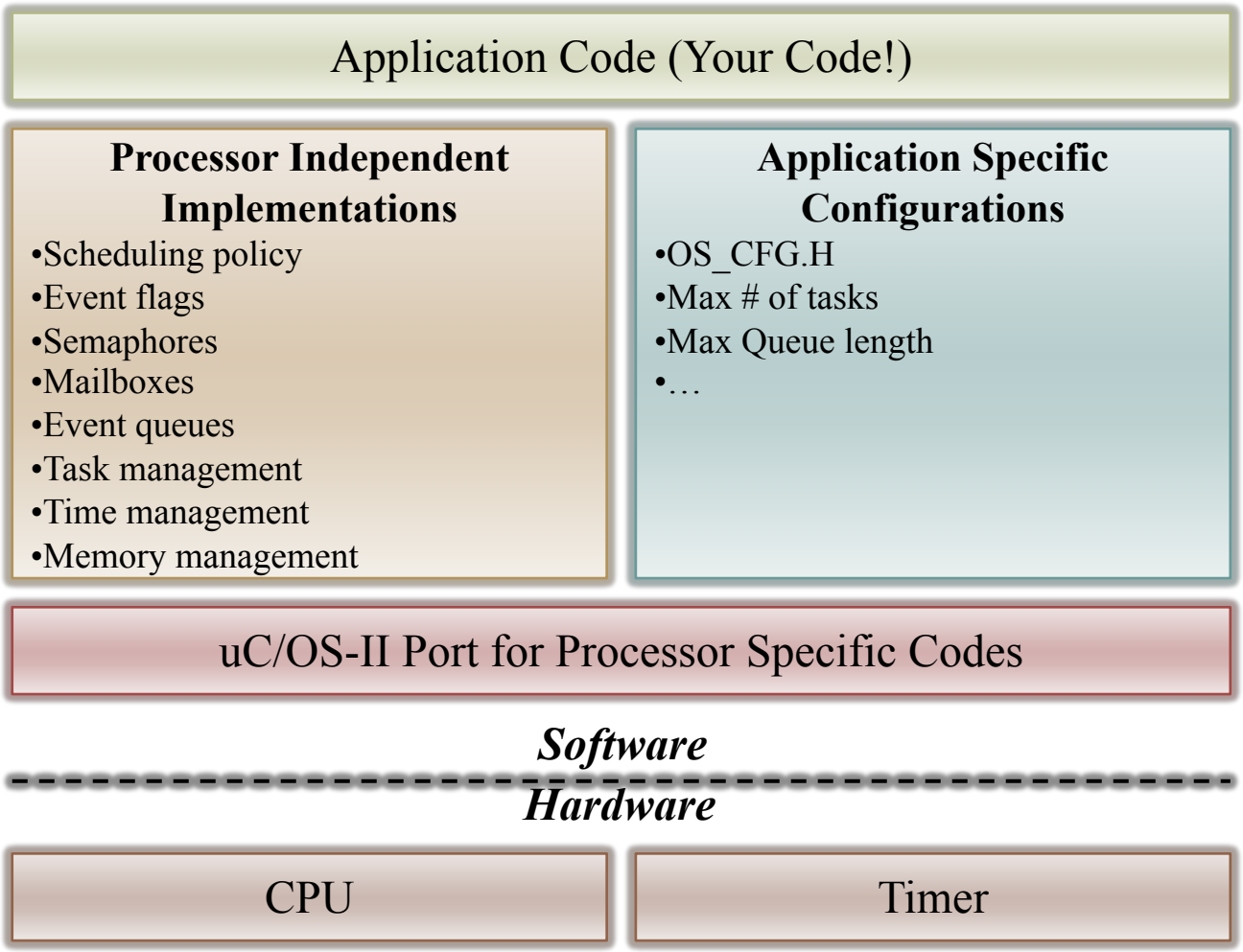
Introduction of µC/OS-II (1/2)

- The name is from micro-controller operating system, version 2
- µC/OS-II is certified in an avionics product by FAA in July 2000 and is also used in the Mars Curiosity Rover
- It is a very small real-time kernel
  - Memory footprint is about 20KB for a fully functional kernel
  - Source code is about 5,500 lines, mostly in ANSI C
  - It's source is open but not free for commercial usages
- Preemptible priority-driven real-time scheduling
  - 64 priority levels (max 64 tasks)
  - 8 reserved for µC/OS-II
  - Each task is an infinite loop

Introduction of μC/OS-II (2/2)

- Deterministic execution times for most μC/OS-II functions and services
- Nested interrupts could go up to 256 levels
- Supports of various 8-bit to 64-bit platforms: x86, 68x, MIPS, 8051, etc.
- Easy for development: Borland C++ compiler and DOS (optional)
- However, uC/OS-II still lacks of the following features:
  - Resource synchronization protocol
  - Soft-real-time support

The µC/OS-II File Structure



Thank You!