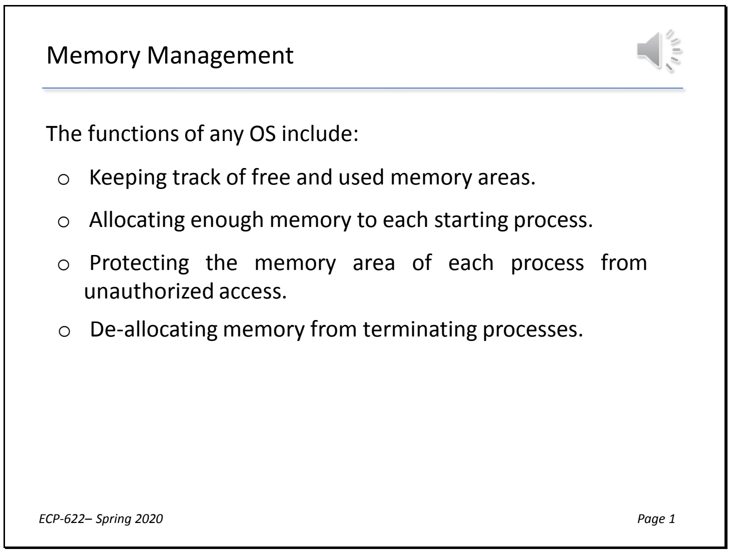


Memory Management

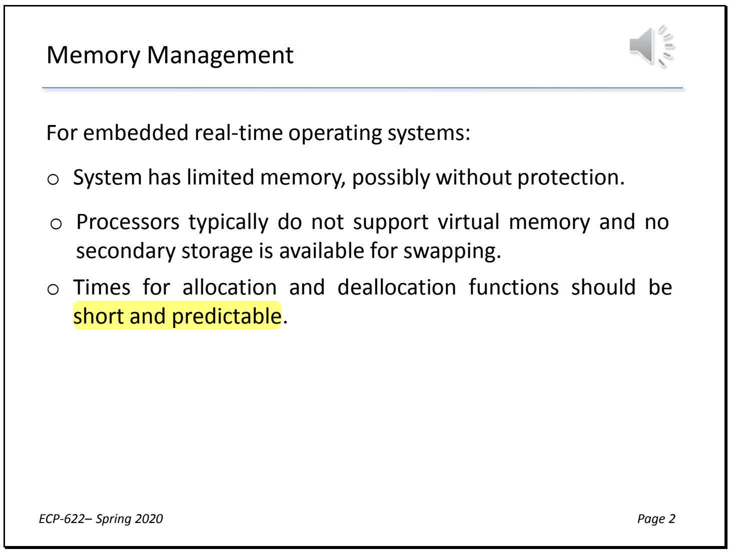
The functions of any OS include:

- Keeping track of free and used memory areas.
- Allocating enough memory to each starting process.
- Protecting the memory area of each process from unauthorized access.
- De-allocating memory from terminating processes.

ECP-622– Spring 2020 Page 1

Slide 1

The functions of keeping track of memory allocations, assigning enough separate memory area for each starting process, and releasing memory no longer needed by a process are common to all types of operating systems. Embedded and real-time systems are not an exception.



Memory Management

For embedded real-time operating systems:

- System has limited memory, possibly without protection.
- Processors typically do not support virtual memory and no secondary storage is available for swapping.
- Times for allocation and deallocation functions should be short and predictable.

ECP-622– Spring 2020 Page 2

Slide 2

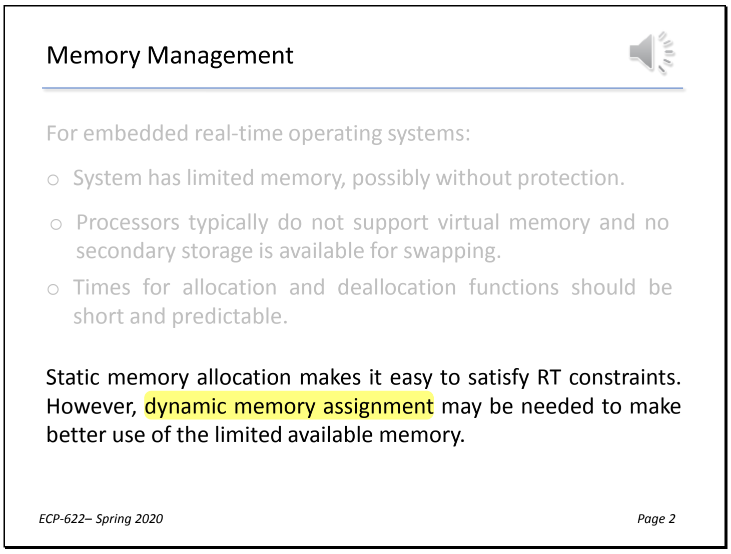
However, embedded systems will typically have to operate under more limited memory size. In addition, preventing a process from accessing some memory areas is not an option provided by the hardware in many architectures.

Slide 3

Virtual memory techniques allow swapping unused memory pages to secondary storage, thus increasing the effective memory size. While used widely in general purpose computers, it is rarely used in embedded architectures.

Slide 4

All memory management operations frequently used by processes should have short and predictable execution times, which may not be the case in general purpose systems.



Memory Management

For embedded real-time operating systems:

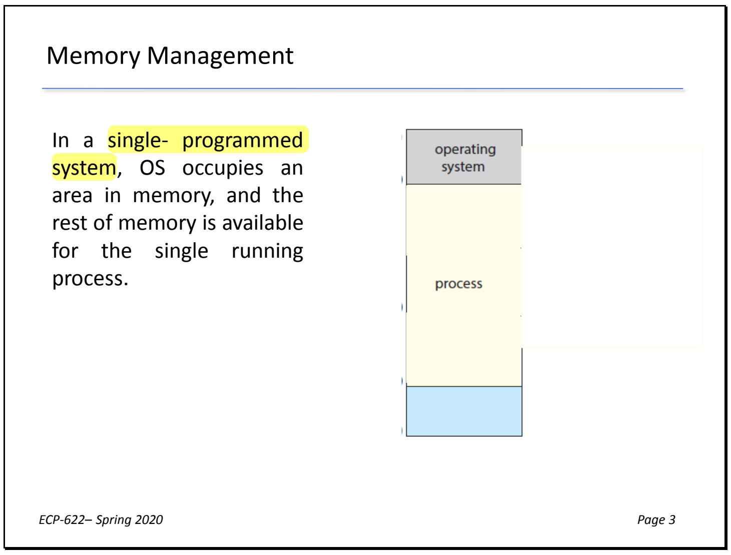
- System has limited memory, possibly without protection.
- Processors typically do not support virtual memory and no secondary storage is available for swapping.
- Times for allocation and deallocation functions should be short and predictable.

Static memory allocation makes it easy to satisfy RT constraints. However, dynamic memory assignment may be needed to make better use of the limited available memory.

ECP-622– Spring 2020 Page 2

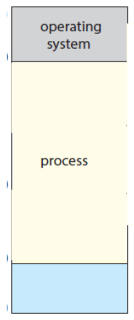
Slide 5

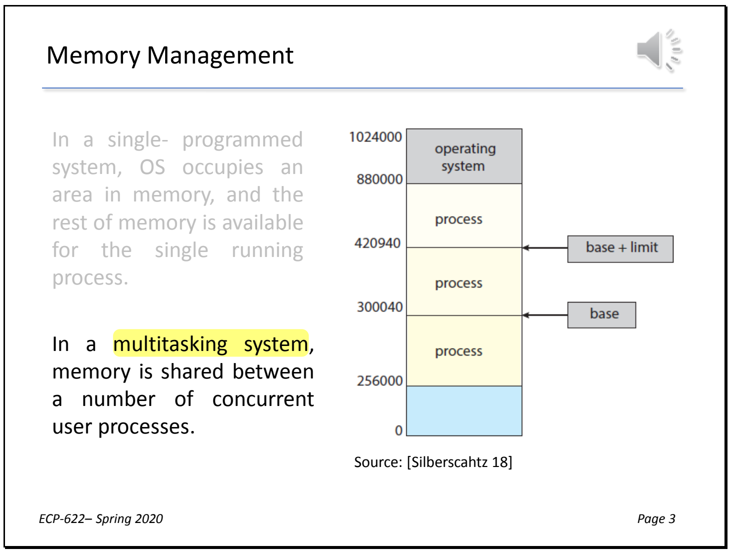
By static memory allocation we mean that memory is divided among a fixed set of tasks, each given a fixed area in memory. However, system can run using smaller physical memory if for example different sets of tasks run at different times. In this case memory assignment will be dynamic, i.e. change as tasks start and terminate.



Memory Management

In a single- programmed system, OS occupies an area in memory, and the rest of memory is available for the single running process.

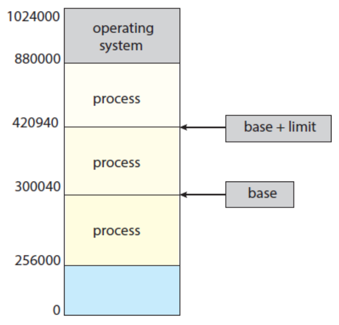




Memory Management

In a single- programmed system, OS occupies an area in memory, and the rest of memory is available for the single running process.

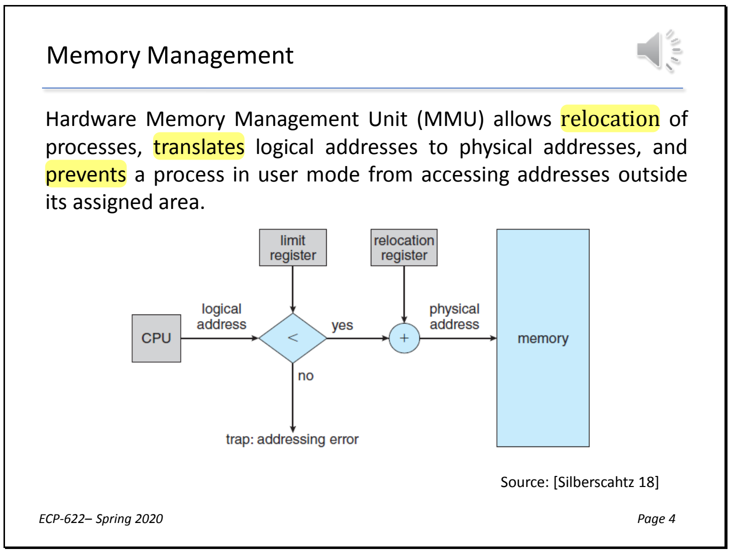
In a multitasking system, memory is shared between a number of concurrent user processes.



Source: [Silberscahtz 18]

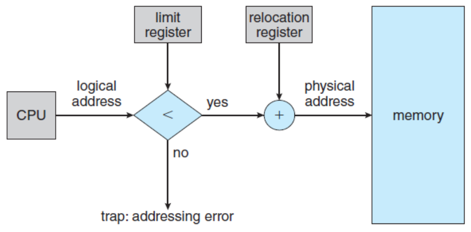
Slide 7

The idea of relocatable processes and base address was mentioned in first week lecture. Here, the limit is the threshold of memory addresses that process should not go across.



Memory Management

Hardware Memory Management Unit (MMU) allows relocation of processes, translates logical addresses to physical addresses, and prevents a process in user mode from accessing addresses outside its assigned area.

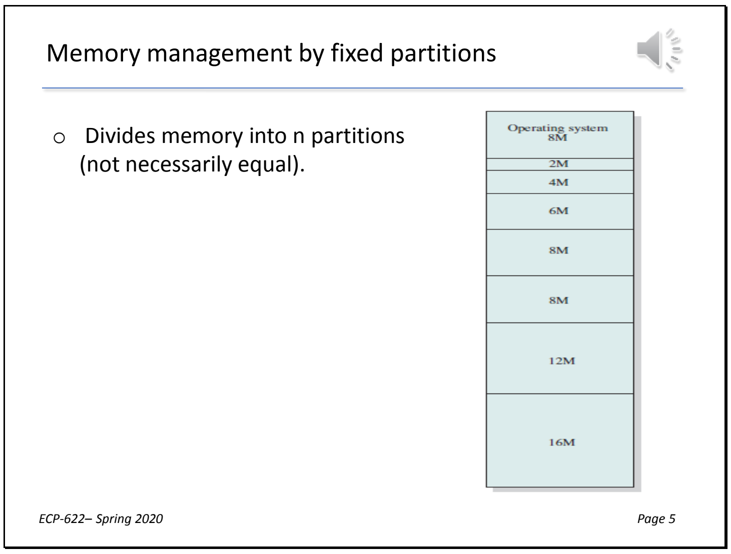


Source: [Silberschatz 18]

ECP-622– Spring 2020 Page 4

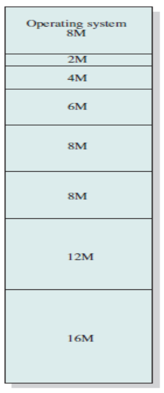
Slide 8

The logical address is the offset we mentioned before. The operation of checking if limit is passed, and adding base to offset will be needed for each memory access. For efficiency, these operations must be done in hardware MMU and not by software. Again, supporting multitasking by this hardware may not be available in small embedded architectures.



Memory management by fixed partitions

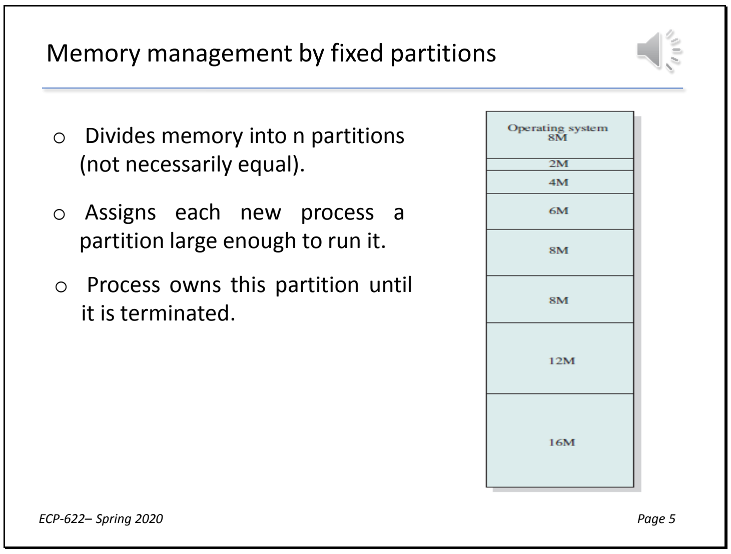
- Divides memory into n partitions (not necessarily equal).



ECP-622– Spring 2020 Page 5

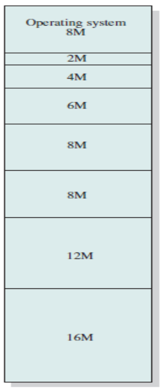
Slide 9

The first memory management technique we consider is the method of fixed partitions. Here, memory is divided into a fixed number of partitions with fixed sizes and limits. Each of these partitions can be assigned to a process. Giving different sizes to partitions allow running processes with different memory requirements.



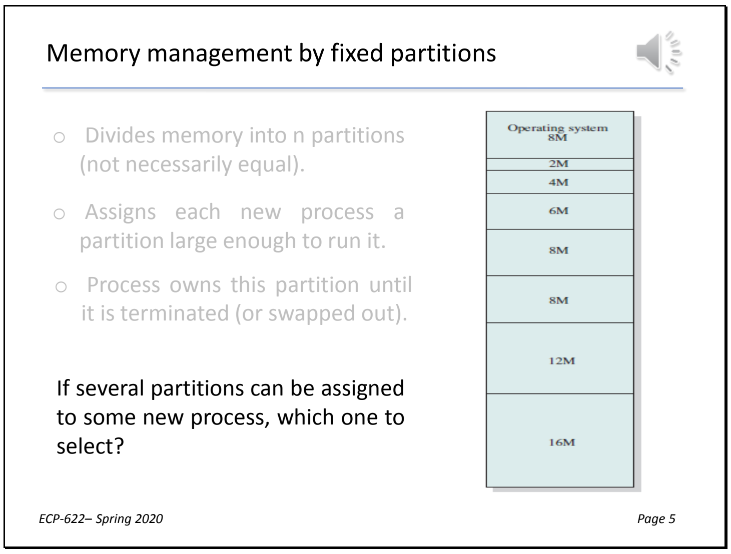
Memory management by fixed partitions

- Divides memory into n partitions (not necessarily equal).
- Assigns each new process a partition large enough to run it.
- Process owns this partition until it is terminated.



Slide 10

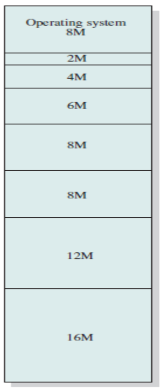
We generally assume that the memory needed by the process both to hold its instructions and its data structures (stack, heap,…etc) are known to the system.



Memory management by fixed partitions

- Divides memory into n partitions (not necessarily equal).
- Assigns each new process a partition large enough to run it.
- Process owns this partition until it is terminated (or swapped out).

If several partitions can be assigned to some new process, which one to select?



Slide 11

The obvious answer here is the smallest one. Size of partition assigned to a process must be equal to or larger than process requirements. If larger, the additional area cannot be used by any other process, and is hence wasted. This can be minimized if we select the smallest partition that can be used by the process.

Memory management by fixed partitions

<u>Disadvantages of fixed partitions:</u>

- Limits the degree of multi-programming: i.e number of concurrent processes that the system can handle.

- Typically results in large wasted memory area as a result of internal fragmentation*:* unused areas inside partitions.