Memory

Memory is one of the most valuable and essential components of any computer system.

When purchasing a new computer, the size of memory is an important consideration. Buyers should consider both RAM and cache.

Example: The **Asus Vivobook X541NA-GO513T** **provides 4G of RAM**

https://www.soundstore.ie/asus-15-6-celeron-4gb-1tb-silver-laptop-x541nag0513t.html

Memory is measured in bits, bytes, kilobytes, megabytes + gigabytes.

Memory: units of memory

bit: the smallest unit can hold a binary value either 1 or 0.

byte: 1 byte = 8 bits

kilobyte: 1K = 1024 bytes = 2^{10} bytes

megabyte 1M = 1024K = 2^{20} bytes

gigabyte 1G = 1024M = 2^{30} bytes

Terabytes (2^{40}) & petabytes (2^{50}) are generally used to measure storage.

Memory

- Memory retains programs and data to be used by the processor.
- Data is moved back and forth from memory to the processor. See notes on the fetch – execute – interrupt cycle.
- With reference to process states – the Ready and Blocked states refer to memory

Memory

Generally, the greater the amount of memory that a device has means:

- It’s more efficient
- It’s more expensive
- It’s faster

Much research is devoted to improving memory performance.

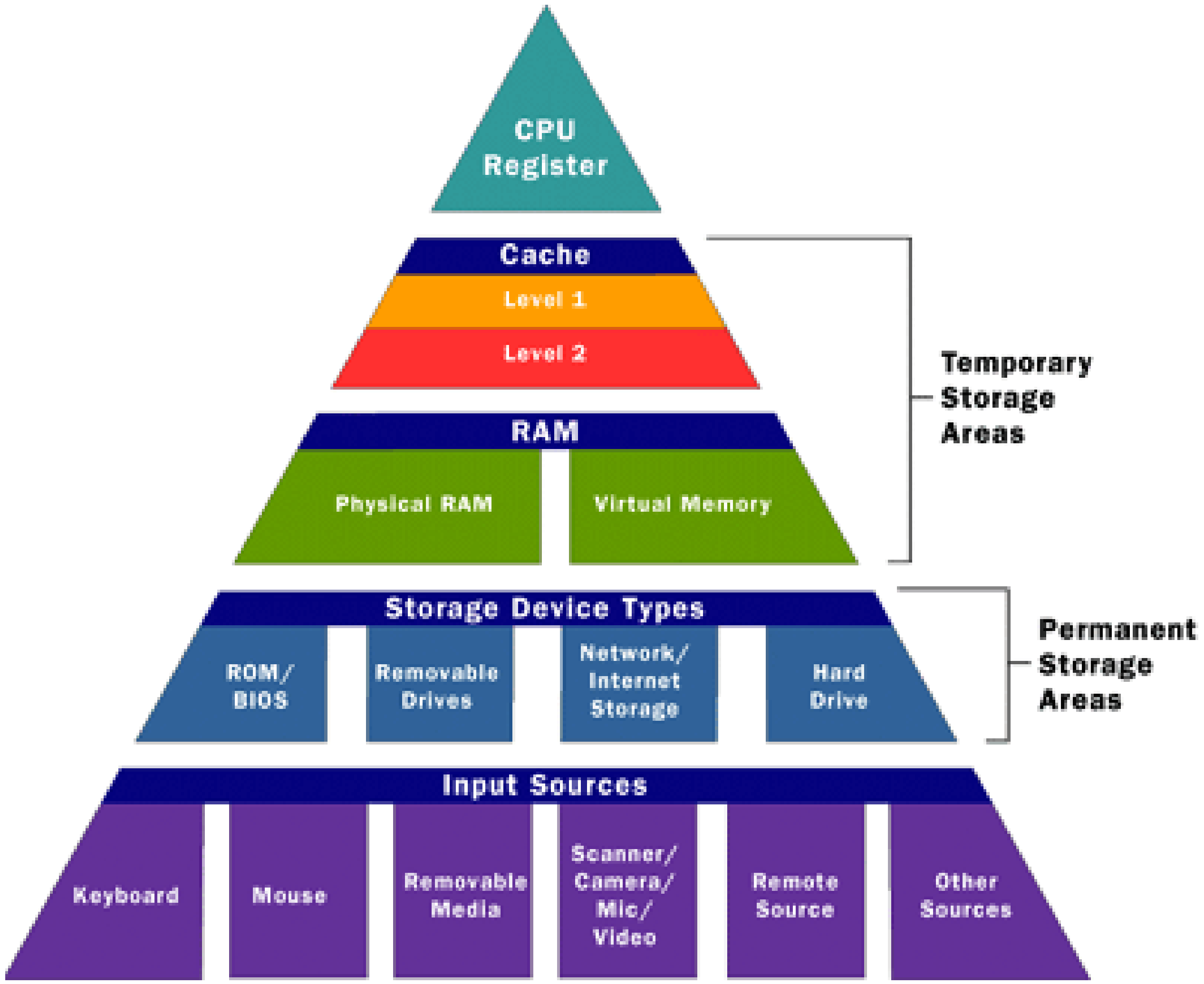
Memory v Storage

The terms memory and storage are often interchanged and can be confusing.

Memory is volatile: contents are lost when switched off.

Storage is non-volatile: contents can be saved when switched off.

Memory & Storage Hierarchy



Memory: Registers

Registers are part of the CPU.

Registers are very small & very fast memory components.

Examples: Memory Buffer Register, Memory Address Register, Program Counter, etc.

Memory: cache

Read over notes on cache from week1

- ***Cache Memory*** is smaller and faster than main memory – here frequently accessed data can be stored for rapid access. It is possible to have multiple layers of cache.

- The processor first checks the cache - if not found in cache, the block of memory containing the needed information is moved to the cache.

- Data can be moved into cache before the processor needs it

- This means that there is a need to predict future requirements.

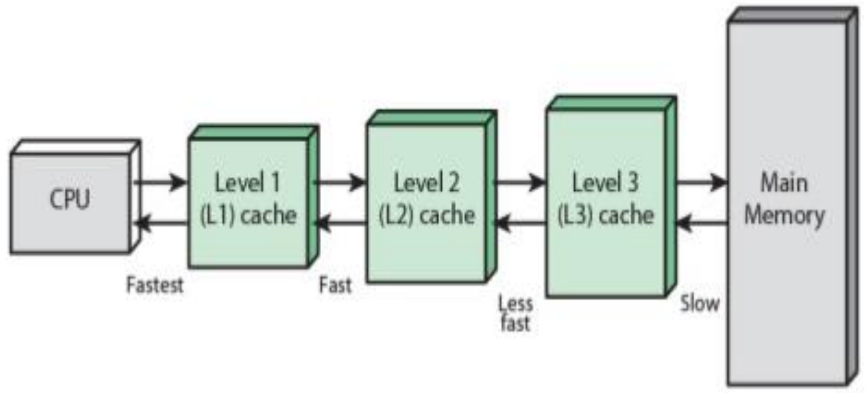
Memory: cache

There are different levels of cache e.g. Level 1, Level 2 and Level 3.

The CPU maintains L1 cache with data it anticipates. This is first to be searched for instructions. L1 cache is smaller and faster than L2 cache.

If the instructions are not found in L1 cache, then L2 cache is searched. If unsuccessful here, L3 is searched.

With each miss the search proceeds to the next (slower) level of cache.



Memory: RAM

Computer Memory:

See notes from Week 1 for overview of memory.

Computer memory is a mechanism that stores data temporarily.

The term RAM – Random Access Memory is used to describe memory. Often referred to as Main Memory or Physical Memory.

Virtual Memory

Virtual memory uses hard disk space to provide additional memory.

Programs can run that require more memory than available (in RAM)

With reference to process states – the Ready Suspend and Blocked Suspend states refer to virtual memory.

ROM, PROM + EPROM

Non-volatile memory components exist such as:

ROM Read only memory

PROM Programmable read only memory

EPROM Erasable programmable read only memory

These are used to store special software (such as firmware) and are non-volatile.

Peripheral Memory: printers

See https://www.techwalla.com/articles/what-is-printer-memory

Some peripherals have independent memory.

Printers have a limited amount of volatile memory

Printer memory is often upgradable

This memory stores jobs that are sent to it.

Peripheral Memory: printers (cont.)

Printer memory helps to print faster and with better quality.

Once a job is done (i.e. printing is complete); the job is wiped from memory

Example:

The HP Color LaserJet Pro MFP M280nw A4 Colour Multifunction Laser Printer has 256M RAM

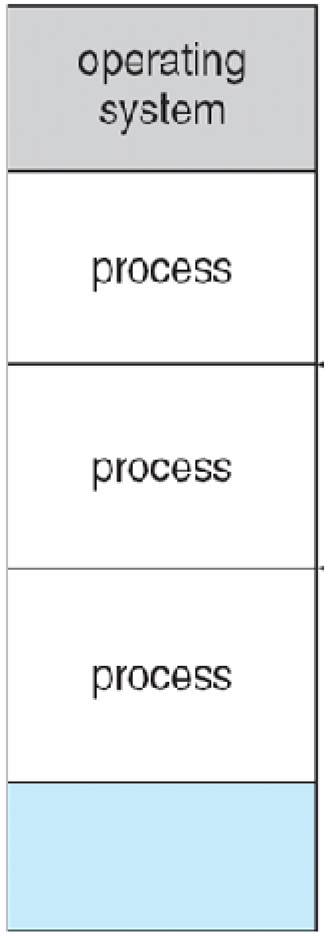
https://www.printerland.co.uk/HP-Color-LaserJet-Pro-MFP-M280nw-P140437.aspx

Memory & the operating system

Managing memory is a significant part of an operating system’s role. The operating system determines the amount of memory available for each activity and for how long that memory is available.

The operating system itself requires memory and is loaded into memory at boot time.

Memory & the operating system (cont.)



Memory is required for both the operating system and user programs.

Memory: Linux

In Linux, there are a number of commands that are used to view data on memory usage.

free: displays the amount of used and available memory. It can report in kilobytes, megabytes and gigabytes.

Memory: Linux

Output might look like:

free -m

```
                     total    used    free    shared   buffers    cached
Mem:                  748      485     263       0       28        313
-/+ buffers/cache:             144     604

Swap:                1128        0    1128
```

Memory: Linux

For a more comprehensive look at memory usage, open the dynamic file /proc/meminfo

```
cat  /proc/meminfo
```

```
MemTotal:        766736 kB
MemFree:         254408 kB
Buffers:          31284 kB
Cached:          312160 kB
SwapCached:           0 kB
Active:          206388 kB
Inactive:        262296 kB
          etc.
```

Memory: Linux

The vmstat command is similar:

```
vmstat –s

 766736  total memory
 512452  used memory
 206444  active memory
 262420  inactive memory
 254284  free memory
  31376  buffer memory
 332144  swap cache
1156092  total swap
      0  used swap
1156092  free swap
         etc.
```

Memory: Linux

The top command (used to look at process activity)

```
top - 13:54:03 up 17 min,  3 users,  load average: 0.00, 0.04, 0.12
Tasks: 133 total,   1 running, 132 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.7%us,  0.3%sy,  0.0%ni, 99.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st

Mem:    766736k total,   511344k used,   255392k free,    31456k buffers
Swap:  1156092k total,        0k used,  1156092k free,   332280k cached

  PID USER      PR  NI  VIRT  RES  SHR  S %CPU  %MEM    TIME+    COMMAND
 4510 root      20   0  2500  948  704  R  0.7   0.1     0:00.02  top
 1173 root      20   0 65168  18m 9008  S  0.3   2.5     0:04.15  Xorg
 1439 root      20   0 24848 4480 3420  S  0.3   0.6     0:00.31  polkitd
 1761 root      20   0 24088 3256 2692  S  0.3   0.4     0:01.19  vmtoolsd
 4223 root      20   0 65536  13m  10m  S  0.3   1.8     0:01.85  gnome-terminal
```

Linux: top command

The top command displays total memory for the entire system both real and swap.

One can also view how much memory each process is using – virtual, real and shared.

The percentage of memory used by individual process is also displayed

Memory Management

- CPUs need quick and easy access to large amounts of data in order to maximize their performance. If the CPU cannot get to the data it needs, it literally stops and waits for it. Memory is expensive and requires efficient management.

- Computer designers have solved the cost problem by "**tiering**" memory -- using expensive memory in small quantities and then backing it up with larger quantities of less expensive memory.

Memory Management

- Memory is divided into two parts: (1) For the operating system and (2) for the program or programs being executed.
- The user part of memory is subdivided to accommodate multiple processes.
- Memory needs to be allocated efficiently to pack as many processes into memory as possible.

Memory Management Requirements

- *Relocation*
- *Protection*
- *Sharing*
- *Logical organization*
- *Physical organization*

Relocation

Available memory is shared among a number of processes

Processes need to be swapped in and out of main memory to maximise processor utilisation.

When a process is swapped back it may be relocated to a different area of memory

Relocation (cont.)

- We cannot know where the program will be placed in memory when it is executed.
- While the program is executing, it may be swapped to disk and returned to main memory at a different location (relocated).
- Memory references must be translated in the code to actual physical memory address.

Protection

- Processes should not be able to reference memory locations in another process without permission.

- Impossible to check absolute addresses in programs since the program could be relocated.

- Must be checked during execution as the Operating system cannot anticipate all of the memory references a program will make.

Sharing

- The protection mechanism has the flexibility to allow several processes to access the same portion of memory.
- Better to allow each process (or person) access to the same copy of the program rather than have their own separate copy.

Logical Organisation

- Programs are written in modules.
- Modules can be written and compiled independently.
- Different degrees of protection given to modules (read-only, execute-only).
- Share modules.

Physical Organisation

- Memory available for a program plus its data may be insufficient so overlaying (see next slide) allows various modules to be assigned to the same region of memory.

- Programmer does not know how much space will be available.

Memory Overlaying

Memory Overlay

This is based on the principle that whenever a process is running – not all of it is required at once.

It’s a technique where a portion of memory holds only the part of the process it requires at any time.

That part can then be replaced by a different part to the process.

It means a program that’s bigger than the size of available memory can be executed.

Memory Management

The purpose of memory management is to bring processes into main memory for execution for the processor.

Almost all modern systems use virtual memory techniques based on paging and / or segmentation.

It is also worth looking at partitioning used in many (now) obsolete operating systems.

Memory Management Methods

- **Memory partitioning**
  - Fixed partitioning
  - Dynamic partitioning
- **Simple paging**
- **Simple segmentation**
- **Virtual Memory paging**
- **Virtual Memory segmentation**

Partitioning

- Memory can be divided up into partitions. Partitioning allows the memory to be divided up among several processes.

- These partitions can be static or dynamic:

Static:

- The partitions are decided at system configuration time and cannot be altered.

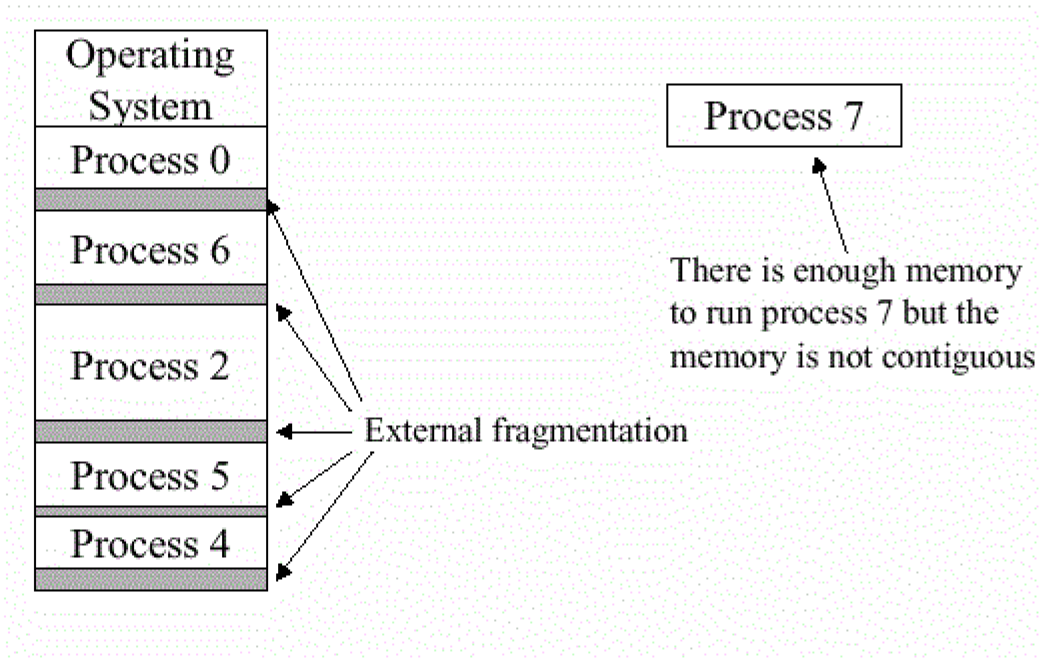
Dynamic:

- The partitions are made and remade as the system is running.

External Fragmentation

External Fragmentation

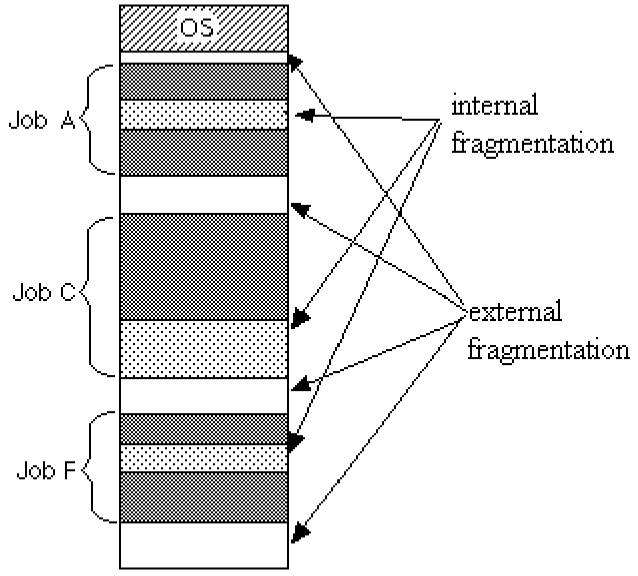
When a block of memory is unusable – usually because it's too small.



Internal Fragmentation

Internal Fragmentation

When an allocated block of memory is too large for what's required -it results in wasted memory that cannot be used elsewhere.

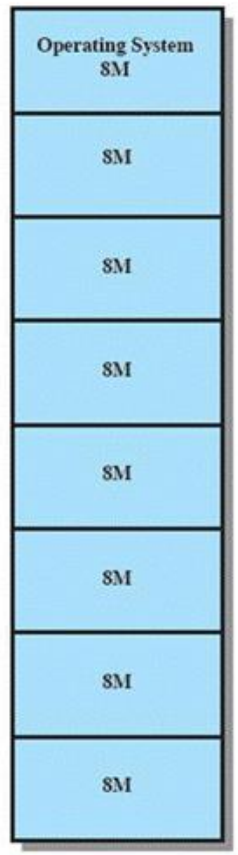


Static (Fixed) Partitioning

- Equal-size partitions - any process whose size is less than or equal to the partition size can be loaded into an available partition

- The operating system can swap out a process as required.

Static (Fixed) Partitioning



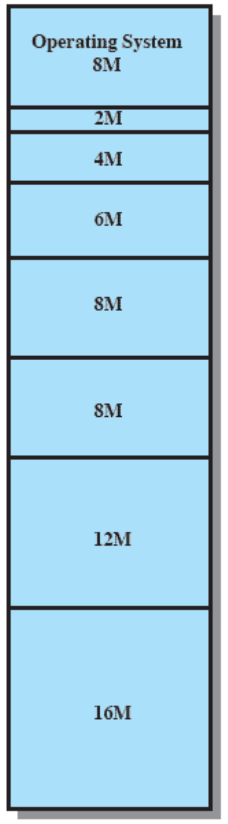
(a) Equal-size partitions

Disadvantages

A program may be too big to fit in a partition that means a program needs to be designed with the use of overlays.

Main memory utilization is inefficient - any program, regardless of size, occupies an entire partition/. This results in ***internal fragmentation*** i.e. wasted space due to the block of data loaded being smaller than the partition

Unequal Sized Partitions



(b) Unequal-size partitions

Using unequal size partitions helps lessen the problems. In the following scheme, programs up to 16M can be accommodated without overlays.

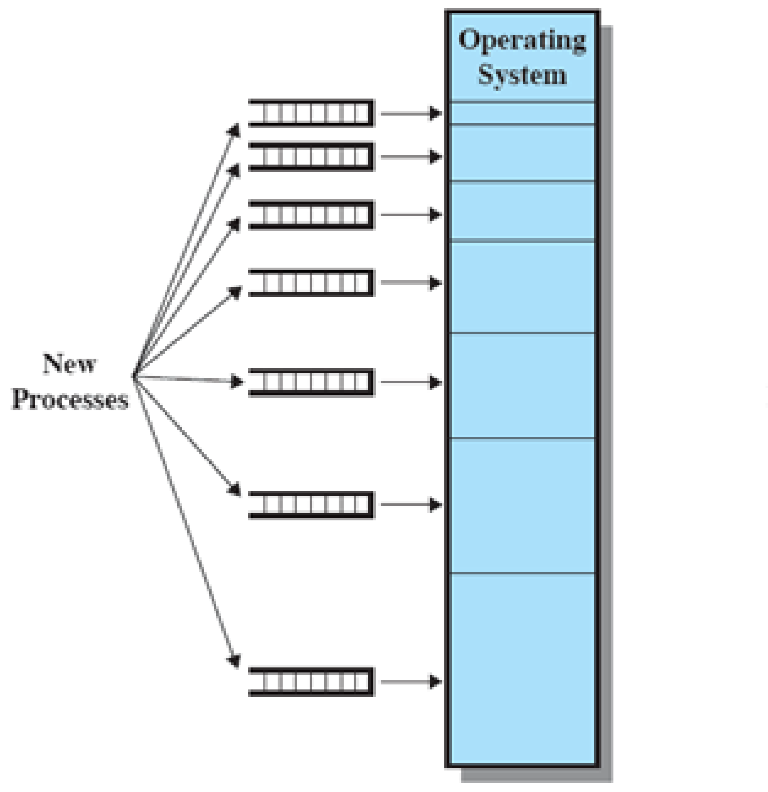
Partitions smaller than 8M allow smaller programs to be accommodated with less internal fragmentation

PLACEMENT ALGORITHM

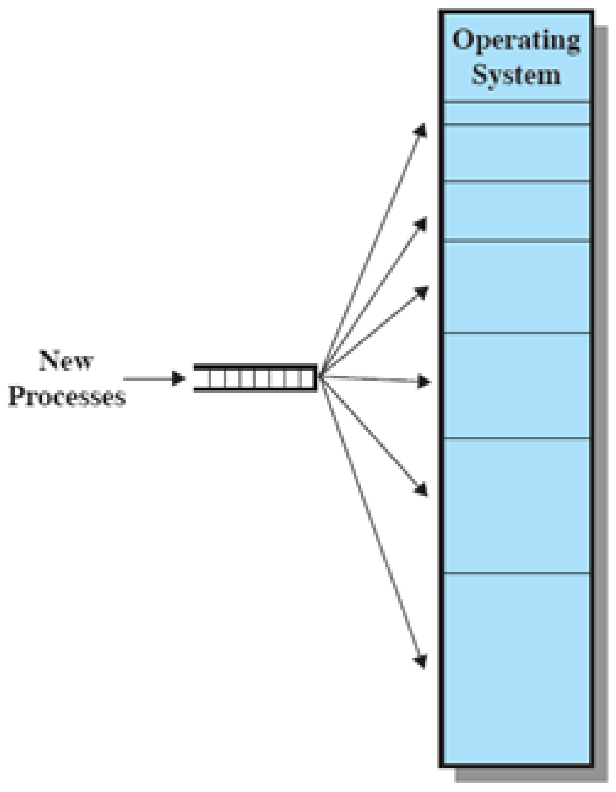
- Assign each process to the smallest partition within which it will fit. In this case, a scheduling queue is needed for each partition, to hold swapped-out processes destined for that partition (see below).

- Employ a single queue for all processes

Placement Algorithm



(a) One process queue per partition



(b) Single queue

Disadvantages (Fixed)

- The number of partitions specified at system generation time limits the number of active processes in the system.

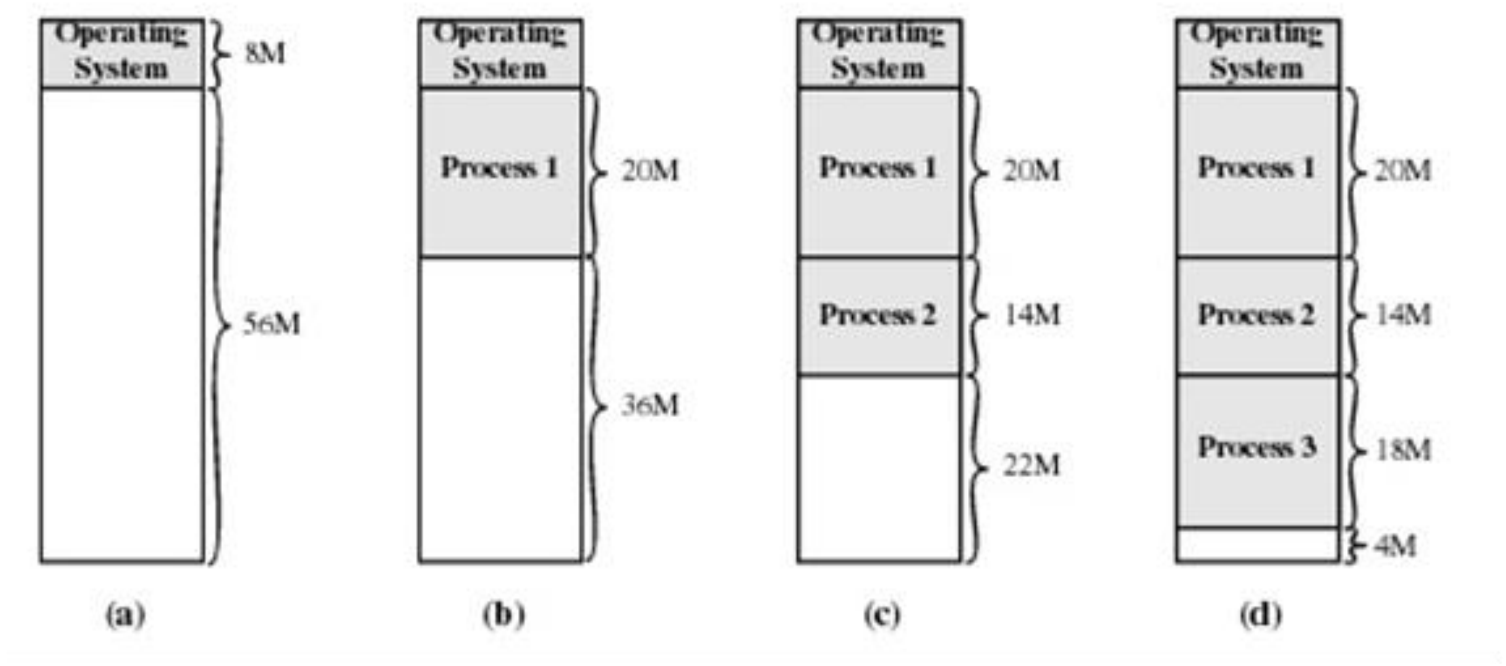
- Small jobs will not utilize partition space efficiently.

- Internal fragmentation occurs when a process’s allocated space is not all used and thus cannot be used by another process.

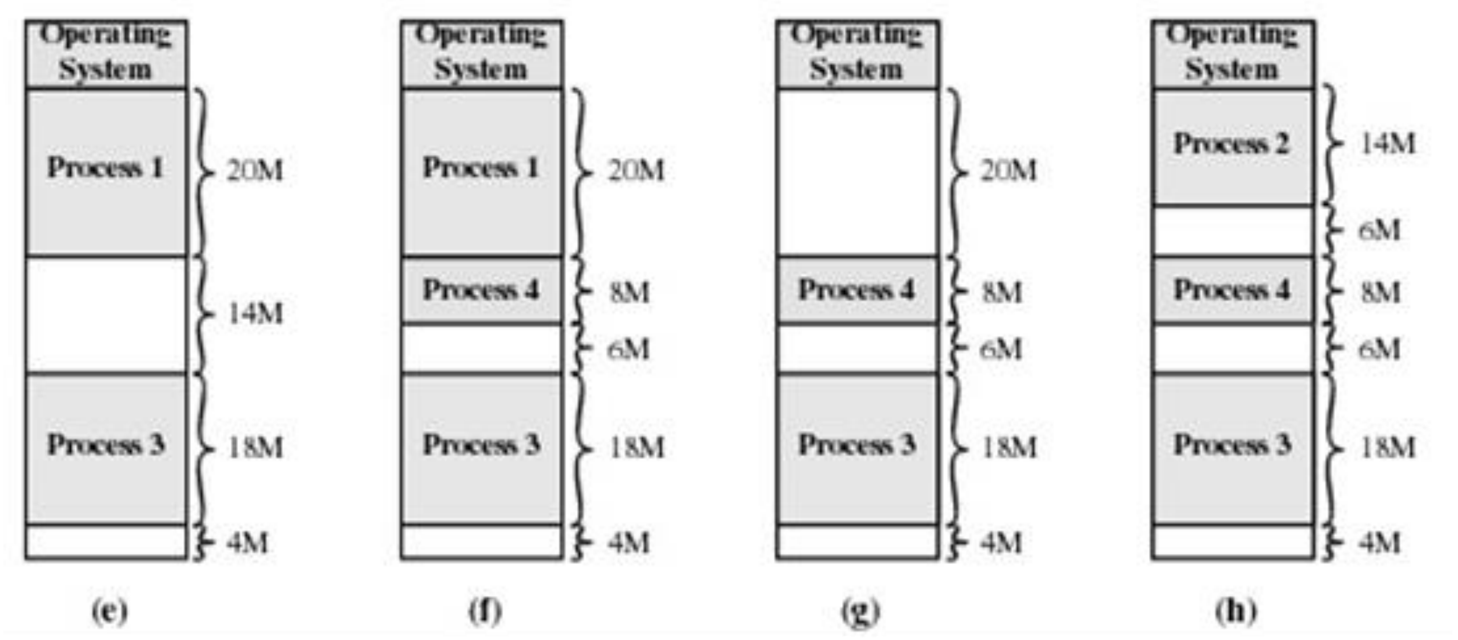
Dynamic Partitioning

- The size of a partition is determined at process load time based on the processes' exact requirement.
- Partitions are of variable length and number.
- Process is allocated exactly as much memory as required.
- **Eventually get holes in the memory: external fragmentation**.
- Must use compaction to shift processes so they are contiguous and all free memory is in one block.

Dynamic Partitioning



Dynamic Partitioning


Operating System
Process 1
20M
14M
Process 3
18M
4M
(e)
Operating System
Process 1
20M
Process 4
8M
6M
Process 3
18M
4M
(f)
Operating System
20M
Process 4
8M
6M
Process 3
18M
4M
(g)
Operating System
Process 2
14M
6M
Process 4
8M
6M
Process 3
18M
4M
(h)