Types of memories

a) Primary memory (RAM):
→ The main memory where the O.S, appl^n is data in current use are kept so they can be quickly reached by the device's processor.

b) Secondary Memory (HDD/SSD):
→ Non-Volatile memory used for storing data long term, including the O.S, appl^ns & files.

c) Cache Memory:
→ A small sized type of volatile computer memory that provides high speed data access to the processor & stores frequently used computer programs, applications and data.

d) Virtual Memory:
→ An extension of primary memory that provides an "illusion" of a very large main memory by using secondary storage.

e) Registers:
→ Small amounts of fast storage within CPU that store instructions and data currently being used.

Need for memory allocation
a) Efficient Memory utilization
→ Ensures that memory is used efficiently by allocating & deallocating memory as needed.

b) Process Isolation:
→ Protects each process from interfering with the memory space of another process.

c) Multitasking:
→ Allows multiple processes to run simultaneously by managing memory allocation & de allocation.

d) Virtual Memory:
→ Enables the execution of processes that require more memory than what is physically available.

Q.3) Contiguous & Dynamic Allocation in M.M
→ A) Contiguous Allocation.
→ Definition:
• Contiguous allocation is a M.M technique where each process is allocated a single contiguous block of memory.
• The entire process's memory needs are satisfied within one continous section of physical memory.

→ How it works?:
• When a process is to be loaded into memory, the O.S searches for a contiguous block of free memory that is large enough to accomodate the process.
• Once found, the entire process is loaded into this block.

→ Pro's:

• Simplicity: Accessing memory is straight forward because the entire process is stored in a single, continuous block.

• Efficient Memory Access: Since the entire process is stored contiguously there are fewer overheads related to memory access, resulting in faster access times.

→ Cons:

• External fragmentation:

- Over time, as processes are allocated & deallocated memory can become fragmented, making it difficult to find a large enough contiguous block for a new process.

• Memory Wastage:

- Even if there is enough total free memory, it may not be in a single contiguous block, leading to inefficient memory use.

• Fixed partitioning:

- In systems that use fixed sized partitions, a process might be allocated more memory than it needs, leading to internal fragmentation.

• Example: Suppose a sys has 600MB of free memo divided into blocks of 200MB each.

⇒ If a process requires 300MB of memory it will not be able to fit into any of these blocks even though there is enough total free memory, as the blocks are not contiguous.

B] Dynamic Allocation:

→ Definition:

- Dynamic memory allocation allows proces
to request memory as needed during exec
- Unlike contiguous allocation, dynamic allo
does not require the process to occupy a
single continous block of memory.

→ How it works?:

- Memory is allocated in small chunks that co
be scattered across different locations in
physical memory.
- The O.S keeps track of these chunks & provides
the necessary mappings between the process's
logical address space & the physical memory
locations.

→ Types of Dynamic Allocation:

- Paging:

- Divides the process's memory into fixed size
pages and maps them to frames in physical memo

- Segmentation:

- Divides the process memory into segments
based on logical divisions like code, stack, d
These segments can be located anywhere in
physical memory.

→ Pro's:

- flexible memory use:

- By allowing non contiguous allocation,
dynamic allocation makes better use of

available memory, reducing both internal & external fragmentation.

• Efficient use of memory:
- Memory can be allocated & deallocated as needed, leading to more efficient use of system resources.

→ con's:
• Complexity:
- Dynamic allocation requires the operating system to manage mappings between logical & physical memory, increasing complexity

• Overhead:
- Keeping track of scattered memory allocation & ensuring efficient access can introduce overhead.

Example: -
If a process requires 300MB of memory & the system has three non-contiguous blocks of 100MB each available, the process can still be allocated the required memory by dynamically allocating these non-contiguous blocks.

* • Comparison:

A] Contiguous allocation	B] Dynamic allocation
1) Simple & fast	1) More complex
2) prone to fragmentation	2) Reduces fragmentation
3) less flexible (Potential memory)	3) More flexible (sophisticated memory)

Conveyoff

Q.4] First Fit, Best Fit, Worst Fit

→ • These are memory allocation strats for placing processes into memory

* a] First Fit:
- Allocates the first block of memory that is large enough for the process.
• Pro's: Reduces wasted space fast allocation
• Con's: Can lead to fragmentation.

* b] Best fit:
- Allocates the smallest block of memory that is large enough for the process.
• Pro: Reduces wasted space.
• Con: Can lead to ext fragmentation & slower allocation

* c] Worst fit:
- Allocates the largest block of memory.
• Pro: Leaves larger chunks of memory for future allocation.
• Con: Can lead to inefficient memory usage & fragmentation.

Q.5) compaction:

→ i) It is a technique used to eliminate ext fragmentation by moving all allocated memory blocks to one end of the memory space, combining all free memory into one large block.

ii) Dis: It is time consuming process and is not feasible in all O.S, especially those requiring real-time performance.

Q.6) Internal & Ext fragmentation.

→ A] Internal frag

→ occurs when fixed sized memory block is allocated & the process does not use all of the allocated memory.

→ The unused space within the block is wasted.

B] Ext. frag

→ Occurs when free memory is scattered in small blocks throughout the system, making it difficult to allocate large contiguous memory spaces, even if sufficient total memory is available.

Q.7) Segmentation

→ • Segmentation is memory management scheme that supports the user view of memory.

• A program is divided into segments, each of different length.

* Hardware req:

- Requires a segment table that maps logical addresses to physical addresses.
- Each table entry has a base address & limit

* Segmentation Table:

- Contains the base address [starting address of seg in phy memory]
- the limit [limit of the segment]

Memo

Q.8) Paging

→ i) Paging is a memory management scheme that eliminates the need for contiguous allocation by ⇒
- dividing memory into fixed size page
- dividing processes into fixed size page fra

ii) It avoids ext. frag & supports the use of virtual memory.

* Hardware required:
→ Page tables are used to keep track of where pages are stored in physical memory.

* Translation lookaside buffer:
→ A cache used to reduce the time taken to access the page table, speeding up memory access in a paging system.

Q.9) Dirty bit

→ - It is a bit associated with block of memory that indicates whether the block has been modified.
- If dirty bit is set, the block needs to be written back to storage (disk) before it can be replaced or removed.

Q.10) Shared pages

→ Allows multiple to share the same physical pages of memory, often used for shared libraries or code segments.

Q.11) Reentrant code (Pure code)

→ • Does not modify itself & can be shared among multiple processes
• Since it is not self modifying it can be shared & executed by multiple process simultaneously

Q.12) Throttling (over clocking of CPU)

→ 1) Refers to controlling the execution speed of processes, typically to manage system load or reduce power consumption.
2) This can involve slowing down processes, reducing CPU speed or limiting the no. of processes running concurrently.

Q.13) I/O Management

→ It involves managing I/O devices & ops in computer system.
The OS provides an interface between the hardware & user level processes., managing the comms, data transfer & synchronization of I/O devices.