Date 25-01-24.

MEMORY MANAGEMENT

- In uniprogramming system.
  main memory is divided into two parts.
  (1) one part for OS
  (2) one part for program currently executing.
- In multiprogramming systems, the user part of memory is further subdivided to acomodate multiple processes.
- This subdivision is dynamically carried out by OS, it is called memory management.

Memory Management terms

Frame: A fixed-length block of main memory.

Page: A fixed-length block of data that resides in secondary memory (disk).
A page of data may ~~contain~~ temporarily be copied into a frame of main memory.

Date

Segment: A variable-length block of data that resides in secondary memory. An entire segment may temporarily be copied into an available region of main memory (segmentation) or the segment may be divided into pages which can be individually copied into main memory (combined segmentation & paging).

Memory:

- Collection of data in a specific format
- Consists of a large array of words or bytes, each with its own address.
- Internal storage area of the computer.
- It stores all kind of data & information

Types of Memory:

1. Physical Memory:
   - primary or main memory, hardware component inside a computer
   - tangible & volatile (loses its contents when

MY SCHOOL

Date

the power is turned off.

- directly accessible by CPU or other hardware components.
- Ex: RAM, ROM.

2. Logical Memory:

- a memory space that a program can use.
- an abstraction of physical memory that the OS ~~can~~ provides to applications & processes
- Programs can access to memory addresses
- Provides a logical address space for each process, independent of actual physical memory.

Ex: CPU memory ⇒ cache, embedded memory.

3. Virtual Memory:

- An imaginary memory area, supported by some OS (eg Windows).
- An extension of logical memory.
- VM is a memory management technique that provides an illusion to applications that they have access to large, contiguous block of memory even if the physical memory is fragmented / or insufficient.

Date

- allow running processes to use more memory than physically available.
- using disk storage as an extension of physical RAM, memory

→ If the size of program is > available memory (Physical one) then virtual memory is used.

Available Memory:

- Amount of Physical memory not in use.
- Loading the OS takes up memory as well.
- When system boots up, available memory amount drops.
- Each time you open a program, it is loaded into your computer's memory, & therefore reduces ~~your~~ available memory.

Memory Management:

- Functionality of an OS to handle & manage Primary memory (RAM eg).
- Subdividing available memory among different processes is called memory management.

Date

- keep track of all available memory locations that either it is allocated to some process or not. (allocation or deallocation)
- decides which process will get which part of memory at any time.
- OS need to achieve to broad tasks

Memory Management Unit (MMU)

1. Each process must have enough memory space to execute in a processes can ~~neither be~~ not run into memory space of another process and not be run into by another process.
2. There are different types of memory in the system or all must be used properly to to achieve effectiveness.

Memory Management Unit (MMU)

How OS handles RAM ?...

→ It keeps track of parts of memory in use. (memory space)

→ Allocate or deallocate memory to or from processes when needed

→ Swapping & paging b/w main memory or disk

Date

- The principal operation of memory management is to bring processes into main memory for execution by the processor.

Static & Dynamic Memory:

① Static Memory: SRAM – Static RAM
faster, less volatile.
requires more power & it is expensive.

② Dynamic Memory: DRAM – Dynamic RAM.
RAM must be constantly refreshed, reenergized or it'll loose its contents.

Partitioning:

① Fixed Partitioning: The available memory, for use by processes. The simplest scheme is to partition available memory into fixed size regions with fixed boundaries.
It is static. A process may be loaded into equal or greater size of a partition.
It is simple & little OS overhead.

Internal fragmentation due to fixed sizes, inefficient use of memory.

~~(2) Dyna.~~

③ Dynamic Partitioning:

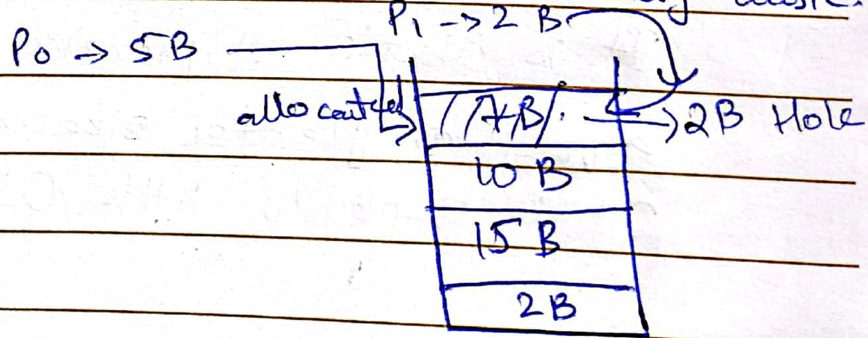
- Partitions are created dynamically, a process loaded into partition will be of same size.
- No internal fragmentation.
- Inefficient use of processor due to compaction to counter external fragmentation.

3 Ways in which data is put in RAM

Placement algorithms:

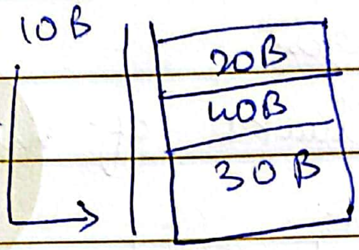
① First Fit:

- Begins to scan memory from the begining and chooses the first available block that is large enough.
- Faster in allocation but leads to memory waste.



2. Worst Fit:

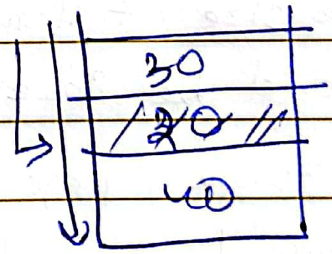
- It scans the entire available memory in ~~place~~ allocate the process in the largest block.
- Least efficient, waste of Memory & time
- It make big wholes $P_1 \to 10B$ in memory. cy slow as it traverse whole list.



3. Best Fit:

- Chooses the block that is closest in size to the requested process
- Best use of memory.
- Slow because traverse whole list than allocate.
- It'll choose closest blocks, but it can make small wholes in each block, ...
- Less space but more time (times space tradeoff)

$P_1 = 18B$



Paging:

- Another memory management technique.
- A logical concept.
- A process is divided into small chunks called pages.
- Chunks of memory are called frames.

Process P_1

Page no		
0	0	1 ← Bytes.
1	2	3

Main memory

frame no		
0	0	1
1	2	3
2	4	5
3	6	7
4	8	9
5	10	11
6	12	13
7	14	15

Process size = 4B

Page size = 2B

No. of pages = $\frac{4B}{2B}$ = 2B.

2 Bytes in each page

MM. Size = 16B

Frame size = 2B.

No. of frames = 16/2

8 B

8 frames

Page size & frame size should be same

Let assume, page 0 is in fram 3.

& page 1 is in fram 5

When CPU need any data of process, how do it know that where it is in main memory.

Date

- A mapping function is used to convert the address of that required data by CPU, that is actually page's data into frame's address of main memory.
- MMU does the mapping function. It uses page table.
- Page table contains frame numbers that where that page is actually present in the main memory

Page table of a Process Pr1

Page no	Frame no
P0	F3
P1	F5

P1

Each process will have its own page table.

CPU works with logical address, logical address is made up of page number and page offset

Logical address		
	page no.	page offset

no. of pages ~~size~~ 2 / 2 numbers 0, 1 can be rep. in 1 bit.

Page no => 0, 1 => 1 Bit

Page size contain 2 Bytes mean 0, 1.

Physical address is for main memory.

logical addresses are converted into physical addresses

4

Physical address.

frame no frame offset (size).

Our MM is of 16 Bytes, 16 can be represented in binary in 4 bits.

Physical address is of 4 bits.

Frame size is 2 Bytes, 2 can be represented in binary in 1 bit.

frame offset 1 bit.

3 bit for frame no. 1 bit for offset.

as in our example CPU asks for ~~Byte~~ 1 Byte

its logical address => | 0 | 1 |

PN PO.

In Page table, Page 0 has Frame 3.

The mapping function → converts it into physical address →

| 0 | 1 | 1 | 1 |

FN FO

return 7 Byte from MM.

Date

Frames.

- Frames are physical concept.
- In RAM.
- Small chunks are called frames in RAM.
- We can't see pages but we can see frames.

Mapping.

- Converting virtual or logical address of pages by CPU into physical address of frames in MM.
- Performed by MMU.
- If an error occurs at any physical address, it retrieves back to CPU.

Paging Protection:

- Paging process is protected by adding valid/invalid bit
- Consider -> 14bit address space.

$$2^{14} = 16383 \text{ bytes.}$$

- address limit is set to 10468.

MY SCHOOL

Date

- If five process P_0, P_1, P_2, P_3, P_4 are defined within this address space (i.e. their address will be less then 10468) then it is valid bit.
- P5 has started before 10468, it alone is considered
- Remaining processes are invalid.
- Pages are internally fragmented in this way.
- The bit is in Page table.

Advantage & Disadvantages of Paging.

① No external fragmentation.
② Simple memory management algo.
③ Easy swapping (equal sized pages & frames)

① Internal fragmentation
② Page table consumes memory.

The wasted space in memory for each process is due to internal fragmentation consisting of only a fraction of the last page of a process.

Date

MY SCHOOL

Fragmentation:

As process are loaded & removed from memory, the available memory space is broken into little pieces. hence, it is ~~f~~ not utilized or wasted. This is fragmentation in memory.

External fragmentation:

When there is enough memory space to satisfy a request but the available spaces are not contiguous; storage is fragmented into a large number of small holes.

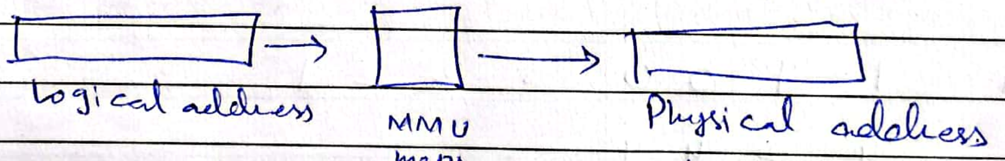
Internal fragmentation:

If memory is divided into fixed patitions & process are loaded in the partitions, the process sizes are smaller than the partition, there is wasted space internal to the partition due to the fact that the block of data loaded is smaller then the partition.

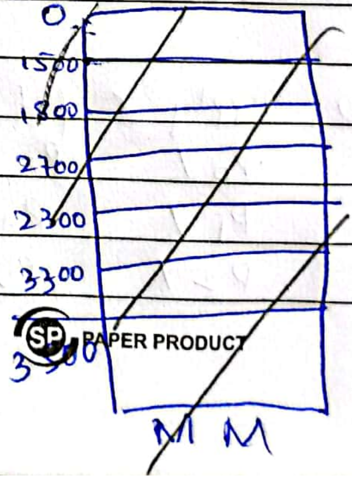
MEMORY MANAGEMENT (CONTINUED..) Date: 1 - 2 - 24

SEGMENTATION:

- A user program/process can be subdivided into segments, in which the program/process & its associated data are divided into segments.
- Segments of a process ~~can~~ are not required to be of same length but they must be ~~l~~less than a limited length.
- As in paging, a logical address using segmentation makes a physical address consist of segment number & an offset.
- Segmentation is more confine with the user's view.
- A segment will consist of full module. E.g. if there is an Addition module & we've done paging, it'll be divided into fixed pages, let's say 2, when ~~user~~ CPU needs it, it will get one page/frame at a time & to execute addition fully it has to get second page frame as well.
- The segment will solve this problem & hence, segments can be of unequal ~~length~~ sizes.
- Segments will also be kept in main memory.
- Now, again CPU will ask for data in logical address.
- MMU will translate logical address to physical address.



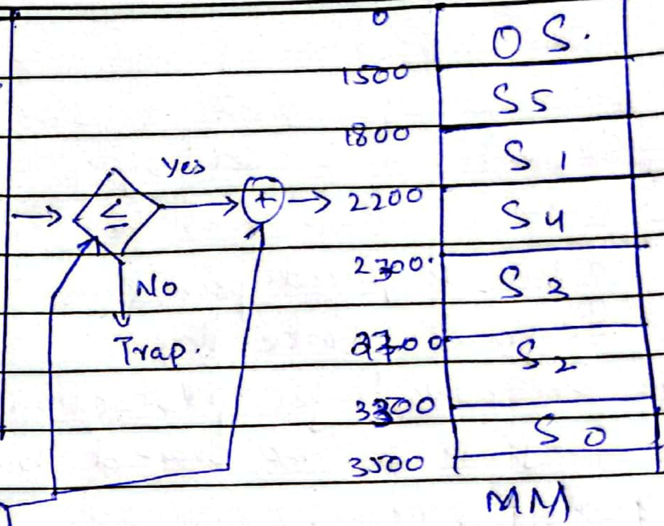
- It uses segment table maps to map/translate LA to PA
- In segmentation, a program may occupy more than one partition & they are not mandatory to be contigous.
- When a process is brought in, all of its segments are loaded into available regions of memory & a segment table is setup.



Date:

Seg no	Base Address	Size
0	3300	200
1	1800	400
2	2700	600
3	2300	400
4	2200	100
5	1500	300

Segment table



CPU generates logical address.

s	d

segment no

segment size/ offset.

s	d
1	400

$d \leq size \Rightarrow 400 \leq 400$ ✓

$d + base \Rightarrow 400 + 1800 = 2200$

Let

0001	10000

Example.

- Seg. no is 1, in segment table, it shows base address 1800, But there is S_1 from 1800 – 2200, so we have to make sure how much CPU needs. → d comes in
- Check the d value corresponding to segment no. in table. which shows the size of segment in MM.

  make sure $\boxed{d \leq size.}$
- if condition meets, add d in base address & read data from base address till the calculated value. Else we'll generate an error as it is invalid.

Let

3	200

base address = 2300.

$d = 200 \Rightarrow d \leq size \Rightarrow 200 \leq 400$ ✓.

$d + base \Rightarrow 200 + 2300 \Rightarrow 2500$

read data from 2300 to 2500 from MM.

SP PAPER PRODUCT

Date:

- In this type, memory addresses used are not contiguous.
- Each memory segment is associated with a specific length of segment & a set of permissions.
- When a process tries to access memory, first check whether it has required permission & is within specified length.

Segment Protection,

- One advantage, protection is associated within the segments.
- Segments can be shared by referencing multiple processes.
- Two process, sharing same segment should have same segment name & address in their own segment tables.

Advantages.

1. No internal fragmentation
2. Segment tables consumes less memory than page tables
3. Sharing data among processes.
4. Protection.

Disadvantages.

1. Costly memory managmt technique.
2. ~~Its~~ External fragmentation -> as processes are loaded in MM & removed ~~to~~ freeing up MM, the available space is broken into little pieces, causing external fragmentation.

Combining Segmentation & Paging

- User's address space is broken down into variable size segments.
- Each segment is further divided into fixed sized pages, which are equal to a frame in MM.

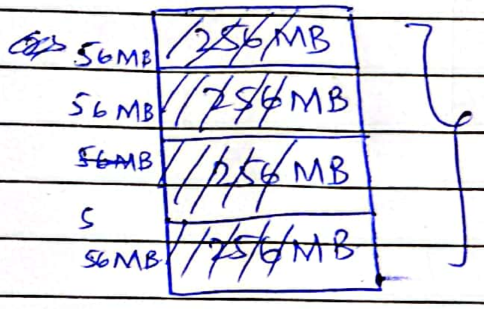
-> Segmentation is visible to programmer.

-> Paging is transparent to programmer.

Date:

Fragmentation question:

Consider that 1024MB memory with 256 MB of 4 allocated memory processes. Assume that each memory process has allocated with 200MB space. What will be the free/fragmented space?

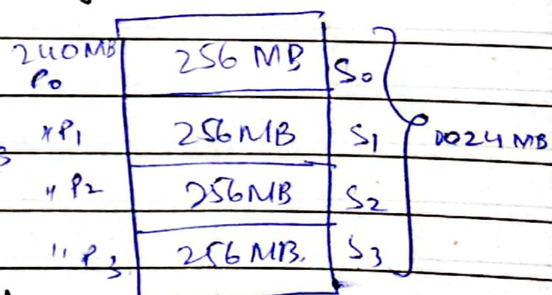


56+56+56+56 = 224MB free/fragmented.

Question

(a) Consider a memory size 1024MB, assume that each memory block or segment has been assigned with 256MB of each memory block/space. Suppose that P_0, P_1, P_2 and P_3 has occupied 240 MB 4 are assigned in each memory segment. What will be the internally fragmented space.

64MB space, 16MB from each = 16MB × 4 = 64MB is internally fragmented.



(b) If P_0 4 P_2 has completed their execution then compute the externally fragmented space?

P_0 4 P_2 completed, freed up 240×2 = 480MB space, it is externally fragmented space.

(c) Will P_1 4 P_3 be executed or not?

Non-contiguous memory (No).