Numericals

Consider the following table of segment

Segment	Base	Limit
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

What are the physical address of the following logical address?

(i) 0430

Seg no = 0

offset = 430

table is base = 219, size = 600

As offset < Size

430 < 600 (True)

Physical address = Base + offset

= 219 + 430

= 649

(ii) 110

Seg no = 1

Seg offset = 10

Base = 9300 } from segment table
Limit = 14

10 < 14 True

PA = 9300 + 10

PA = 9310

(3) 9500

s = Seg no = 9
d = offset = 500

from Segment table :-

Base = 90
Limit = 100

500 < 100 false

Hence, reference of Physical address of logical address (9500) will result an error.

Numericals

Important formulas

∴ Physical address Space = Size of main memory

∴ Size of main memory = Total no of frames * Page size

∴ no of pages = Logical address space = 2^{m-n} / Page size

∴ no of frames = Physical address space / Page size

∴ Size of page table = no of entries in PT * Page table entry size

∴ no of entries in page table = no of pages

∴ entry size = Size of page entry

∴ Page size = 2^n = n-bit address

∴ no of bits used to represent page no = m-n

max length of page table = no of pages

PA = (Frame #) X no of bytes/page) + displacement

∴ page table entry size = No of bits in the frame number + no of bits used for optional fields if any.

1 words = 1 bytes

1 MB = 2^{20}

1 GB = 2^{30}

1 kB = $2^{10} \times 2^2 = 2^{12}$

1k = 2^{10}

Example: Consider a system in which logical address equal to 27 bits and the physical address space requires 21 bits. The page size 4k words. Calculate the number of pages and no of frames?

=> if logical address requires 32-bits then the total size = 2^{32}

=> if physical address requires 21-bits then physical address space = 2^{21}

Page size = 2^{12} (4k implies $2^2 \times 2^{10}$)

∴ no of pages = $\frac{\text{Logical address space}}{\text{Page size}}$

$= \frac{2^{27}}{2^{12}}$

$= 2^{15} = 2^{10} \times 2^5 = 32k$

∴ no of frames = $\frac{\text{Physical address space}}{\text{frame size}}$

$= \frac{2^{21}}{2^{12}}$

$= 2^9 = 512$ frames

2) Consider a system with logical address of 32-bits and physical address space 64M words. The page size is 4k. Page table entry size = 2 bytes? what is the approximate size of the page table?

=> if logical address requires 32 bits, then the total size of logical address space = 2^{32}

=> Page size = 4k = 2^{12}

=> no of Pages = Logical address Space / page size

$= \frac{2^{32}}{2^{12}} = 2^{20}$

=> Size of page table in bytes = $2^{20} \times 2$ bytes

= 2MB

3) Consider a System having a page table with 4k entries? Logical address of 29-bits? How many bits physical address requires if the system of 512 frames?

=> No of pages: no of entries in a Page table

$4k = 2^2 \times 2^{10} = 2^{12}$

=> Logical address have 29 bits, so Size of logical memory = 2^{29}

=> Page Size: $\frac{2^{29}}{2^{12}} = 2^{17} = 2^7 2^{10}$

= 128K

Note: Pg size = frm size

no of frames = PAS / frame size

$512 = \frac{PAS}{2^{17}}$

$2^9 \times 2^{17} = PAS$

$PAS = 2^{26}$

So, 26 bits are required to represent physical address

Example how to convert logical address into PA.

Suppose LA = 1024 bytes = 2^{10} — m+n

Page size = 128 byte = 2^7 — n

LAS m

3	7
m-n	n

Suppose CPU generate LA is 698, now we convert logical address into its binary

Binary will be of m size

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
0	1	1	1	0	1	0	0

Now Page no 2+1=3 offset 10 converted into decimal

100 $2^6+2^5+2^4+2^2 = 116$ offset

Page no 4

Practice Exercise

Q.9.1) Name two differences b/w logical address and physical address?

Solved on previous page.

Q 9.2) Why are page sizes always powers of 2?

- Paging is implemented by breaking up an address into a page and offset number.
- It is most efficient to break the address into x page bits and y offset bits, rather than doing by arithmetic operations.
- Since each bit is represents a power of 2, Splitting an address bits results in page size that is power of 2.

9.4) Consider a logical address Space of 64 pages of 1024 words each, mapped onto a physical memory of 32 frames.

a) How many bits are there in logical address?

b) How many bits are there in the physical address?

(a)

⇒ no of pages = 64 = 2^6

⇒ size of each page = 1024 w = 2^{10}

⇒ So, logical address space will be

∴ logical address space = page size × no of pages

= $2^{10} \times 2^6 = 2^{16}$

⇒ Hence, 16-bits are there in LA.

(b)

no of frames = 32 = 2^5 — given

∴ size of page = size of frame

size of frame = 2^{10}

∴ Physical address space = frame size × no of frames

= $2^{10} \times 2^5 = 2^{15}$

⇒ Hence, 15-bits are required to represent PAS.

9.8) The BTV OS has 21-bit v
, yet on certain embedded de
ces, it has only a 16-bit Phys
addresses. it also has 2-kB Pz
How many entries are there in ea
of the following?
(a) A Conventional, Single
- level page table
b) An inverted page table

=> 21-bit for logical address ——— given
=> 16 bit for physical address ——— giv
=> Page size = 2kB ——— given
= 2×10^{10}
= 2^{11} B

(a) no of entries ?

=> if 21-bit of logical address are given then the total size = 2^{21} bytes
∴ no of entries = no of pages
∴ no of entries = $\frac{LAS}{\text{Page size}}$
no of entries = $2^{21}/2^{11}$ = 2^{10}

(b) if 16 bit for physical address space then total size will be = 2^{16}
∴ no of entries in inverted table = no of frames
∴ no of frames = $\frac{PAS}{\text{frame size}}$
∴ frame size = Page size
no of frames = $\frac{2^{16}}{2^{11}} = 2^5$
Hence 2^5 entries will be in the inverted page table.

c) maximum amount of physical memory in the BTV OS?

Physical memory = no of entries in Conventional table * page size
= $2^{10} * 2^{11}$
= 2^{21} bytes

(9.9) Consider a logical address space of 256 pages with a 4-kB page-size, mapped into a

physical memory of 64-frames

(a) How many bits are required in the logical address?

(b) How many bits are required in the physical address?

(a)

⇒ no of pages = 256 ——— given

⇒ Page size = 4KB = $4 \times 10^{12} = 2^2 \times 2^{10} = 2^{12}$

⇒ no of frames = 64 ——— given
$= 2^6$

∴ LAS = no of pages × page size
$= 2^8 \times 2^{12}$
$= 2^{20}$

Hence, 20-bits are required.

b) ∴ PAS = no of frames × frame size

∴ frame size = page size

frame size = 2^{12}

PAS = $2^{12} \times 2^6$
$= 2^{18}$

Hence, 18-bits are required.

9.10) Consider a Computer system with a 32-bit logical address and 4kB page size. The system supports upto 512MB of physical memory. How many entries are there in each of the following?

a) Conventional page table, Single-level page table

b) An inverted page table

for LAS 32-bits are given so total size = 2^{32}

⇒ Page size given = 4KB = $2^2 \times 2^{10} = 2^{12}$

⇒ Physical memory = 512 MB
$= 512 * 2^{20}$ ∴ $1MB = 2^{20}$
$= 2^9 * 2^{20}$
$= 2^{29}B$

a) ⇒ no of entries = no of pages

⇒ no of pages = $\frac{LAS}{Page\ size} = \frac{2^{32}}{2^{12}}$

⇒ no of pages = 2^{20}.

Hence, 20-bits are required.

(b)

no of entries in inverted table = no of frames

∴ no of frames = PAS / frame size ∴ FZ = PZ

= $\frac{2^{27}}{2^{19}}$ = 128K

9.7) Assuming a 1-KB page size, what are the page no and offsets for the following reference (provided as decimal numbers):

a) 3085
b) 42095
c) 215201
d) 650000
e) 2000001

⇒ Given page size 1KB, it can be write as $= 2^n = 2^n = 1024 = 2^{10}$, so, the no of bits in the offset part (n) = 10.

⇒ To Solve this problem, following steps:

1) Convert logical address: Decimal to binary.
2) Split binary address to 2 parts (Page no, offset) : n digits
3) Convert offset and page no, Binary to decimal.

(a) 3085 as Decimal

- Binary form = 011000000 1101
- page no = 011 = 3
- page offset = 0000001101 = 13

b) 42095 as Decimal

- Binary form = 10100 1000 1101111
- page no = 101001 = 41
- page offset = 0001101111 = 111

c) 215201

- Binary form = 110100 1000 0100001
- page no = 1101001 0 = 210
- page offset = 0010100001 = 161

(d) 650000 as decimal

- Binary form: 10011110101100010000
- page no: 1001111010 = 634
- page offset: 1100010000 = 784

(e) 2000001 as decimal

- Binary form: 11110100001001000001
- page no: 1111010000 = 1953
- page offset: 0010000001 = 129

Ex(5a) Consider a system, with byte-addressable memory, 32-bit logical addresses, 4kB page size and page table entries of 4 bytes each. find the page table size in MB?

∴ Page table size = no of page table entries × entry size

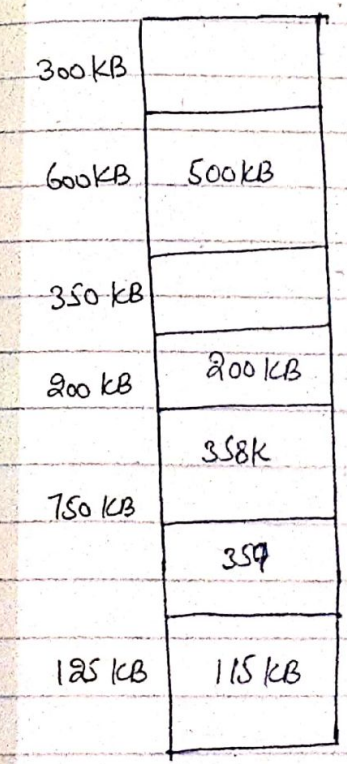
∴ = no of pages * 4 bytes

$= \frac{2^{32}}{2^{12}} \times 4$ bytes

= 4MB

#96) Given a Six memory partitions 300kB, 600kB, 350kB, 200kB, 750 kB, & 125 kB (in order), how would the first fit, best fit, and worst fit algorithm places processes of size 115 kB, 500 kB, 358 kB, 200kB and 351 kB (in order)?

(1) Best fit:



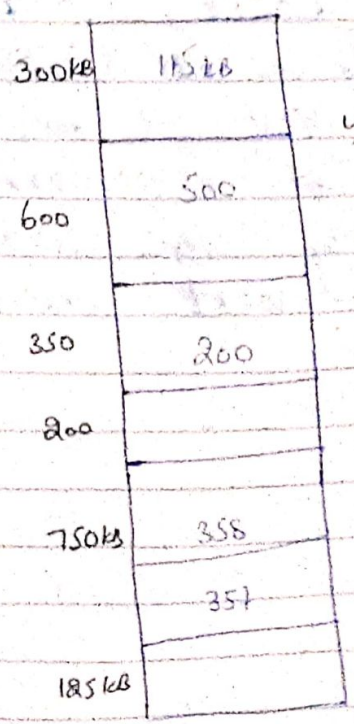
Wastage of memory

Wastage: (600-500) + (750-(358+351)) + (125-115) + 850 + 300

= 100 + 35 + 10 + 350 + 300

= 795 kB

2) First fit:

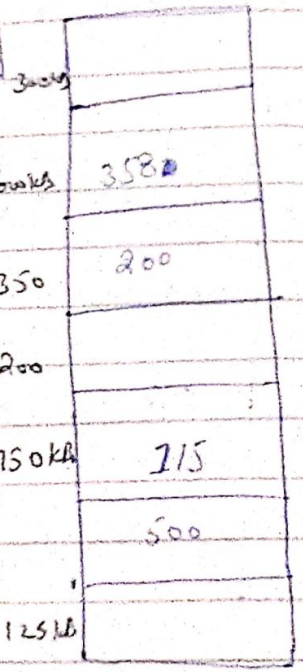


Wastage of memory

= (300 - 115) + (600 - 500) + (350 - 200) + 200 + (750 - (358 + 357)) + 125

= 185 kB + 100 kB + 150 kB + 35 kB + 125 kB

= 715 kB



3) Worst fit

351 kB need to wait

Wastage:

300 + (600 - 358) + (350 - 200) + 200 + (750 - (115 + 500)) + 125

= 300 + 242 + 150 + 200 + 85 = 977 kB

Extra: Given 5 memory partition of 100kB, 500kB, 200kB, 300 kB, 600 kB (in order). How would each of first fit, best fit, worst fit & Next fit, place process of 212 kB, 417kB, 1124 kB, 426 kB (in order). which algorithm makes efficient use of memory.

Given memory partitions are:-

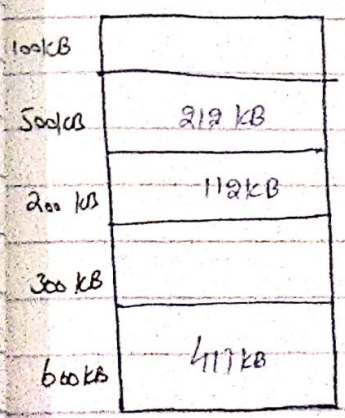
100kB, 500kB, 200kB, 300kB, 600kB

Given processes are:-

212 kB, 417kB, 1124kB, 426 kB

Solution:

(1) First fit



=> 426kB cannot get memory

Total wastage

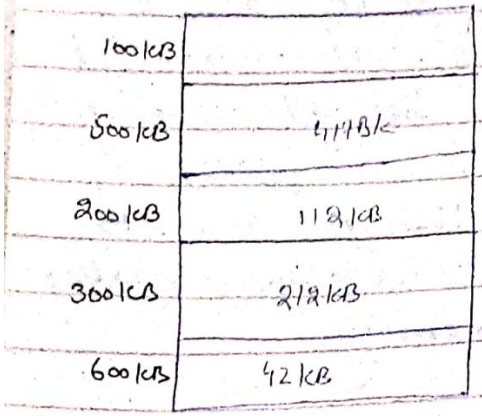
=> 1700 - (212 + 112 + 417)

= Total memory - usage memory

= 1700 - 741

= 959 kB

2) Best fit :-

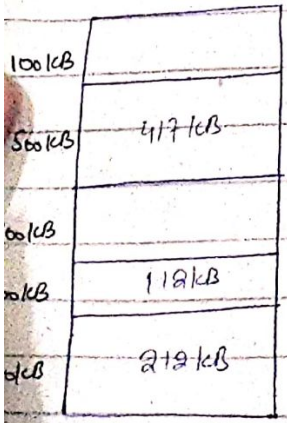


Totalal wastage:

= 1700 − (417 + 112 + 212 + 42) =

= 1700 − 1167 = 533 KB

3) Worst fit:



=> 426 KB can not get memory.

Wastage:

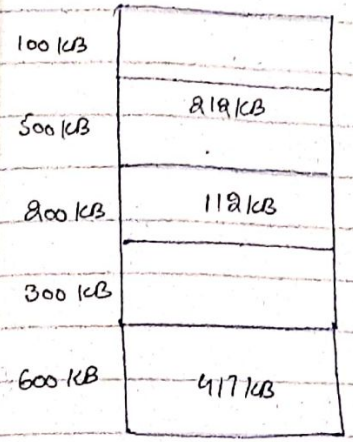
= 1700 − (417 + 112 + 212)

= 1700 − 954

= 959 KB

∴ fixed partitioning leaving memory of 600 can not be assigned to 426.

4) Next fit:



=> 426 KB cannot get memory.

Wastage :-

=> 1700 − (212 + 112 + 417)

= 959 KB

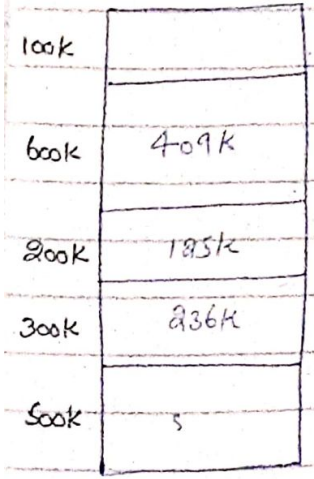
So, best-fit algorithm is best for this process.

Past Papers :-

Given memory partitions of 100K, 600K, 200K, 300K, and 500K (in order) how would each of the first fit, best fit, and worst fit algorithms place process of 409K, 236K, 125K and 514 K (in orders)? Also define them.

- Given processes :-
  409K, 236K, 125K, 514K
- Given memory partitions :-
  100K, 600K, 200K, 300K, 500K

(i) first-fit:



=> 514k need to wait will not get memory.

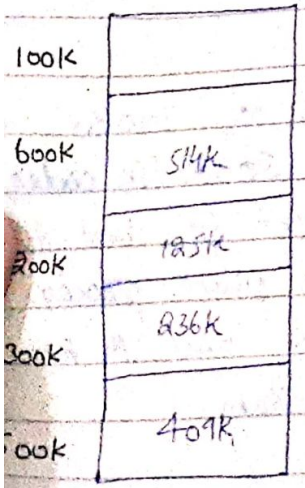
Wastage of memory:

= 1700 - (409+185+236)

= 1700 - (770)

= 930k

(ii) Best-fit:



Wastage of memory:

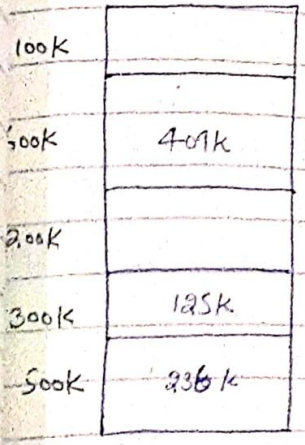
=

= 1700 - (514+185+236+409)

= 1700 - (1284)

= 416k

(iii) Worst-fit:



=> 514k need to wait, will not get memory.

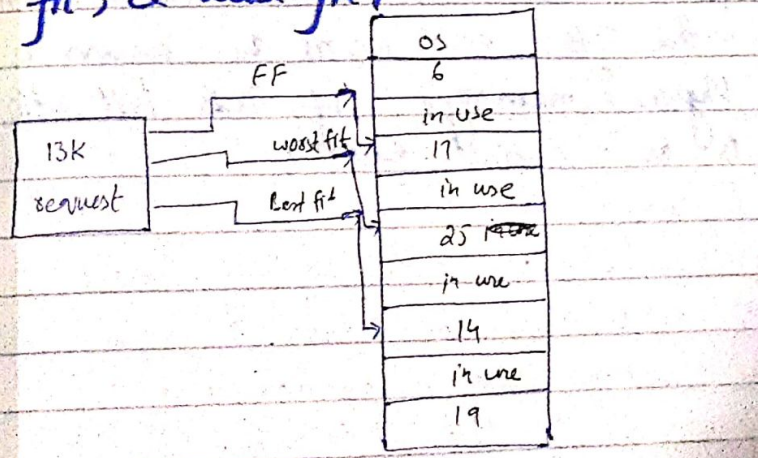
Wastage of memory:

0 = 1700 - (409+125+236)

= 1700 - (770)

= 930k

#Suppose that we have free segment with sizes: 6, 17, 25, 14 and 19. Place a program with size 13kB in the free segment using first fit, best-fit, & worst-fit?



Numericals

ToPic: TLB

Formula:

=> Hit ratio of * [Access time of TLB + Access time of main memory] + miss ratio of TLB * [Access time of TLB + (L+1) * Access time of main memory]

where L = no of levels of page table

=> valid formula if there is no page fault.

Example: Consider a single level paging scheme with a TLB. Assume no page fault occurs. it takes 20ns to search the TLB and 100ns to access the Physical memory? if TLB hit ratio is 80%. find EAT?

Given-

- Number of levels of page table = 1
- TLB access time = 20 sec
- main memory access time = 100ns
- TLB hit ratio = 80% = 0.8

Calculating TLB miss ratio:

TLB miss ratio = 1 - TLB hit ratio

= 1 - 0.8

= 0.2

Calculating effective access time:-

EAT = 0.8 * [20ns + 100ns] + 0.2 * [20ns + (1+1) * 100ns]

= 0.8 * 120ns + 0.2 * 220ns

= 96ns + 44ns = 140ns

What are Pros and Cons of paging?

Pros	Cons
o No external fragmentation o Efficient use of memory. o Swapping of pages are easy as of same size. o Allocating memory to pages is easy and cheap.	o Longer memory access time. o internal fragmentation to every last page of the process. o Large memory space is required as page table are stored in main memory as well. o Size of page is crucial (not too small and not too large)