ANSWERS OF SELECTED EXERCISES

Memory Management Strategies

8.1 Explain the difference between internal and external fragmentation.

Answer:
The main deference is the **allocation** operation

- **External fragmentation:** memory space is NOT allocated and unused
- **Internal Fragmentation:** memory space is allocated and unused

* * * *

8.16 Given five memory partitions of 100 KB, 500 KB, 200 KB, 300 KB, and 600 KB (in order), how would each of the first-fit, best-fit, and worst-fit algorithms place processes of 212 KB, 417 KB, 112 KB, and 426 KB (in order)?Which algorithm makes the most efficient use of memory?

Answer:

Let p1, p2, p3 & p4 are the names of the processes

a. *First-fit:*

- **P1>>>** 100, **500**, 200, 300, 600
- **P2>>>** 100, 288, 200, 300, **600**
- **P3>>>** 100, **288**, 200, 300, 183
- 100, 116, 200, 300, 183 <<<<< **final set of hole**
- P4 (426K) must wait

b. *Best-fit:*

- **P1>>>** 100, 500, 200, **300**, 600
- **P2>>>** 100, **500**, 200, 88, 600
- **P3>>>** 100, 83, **200**, 88, 600
- **P4>>>** 100, 83, 88, 88, **600**
- 100, 83, 88, 88, 174 <<<<< **final set of hole**

c. *Worst-fit:*

- **P1>>>** 100, 500, 200, 300, **600**
- **P2>>>** 100, **500**, 200, 300, 388
- **P3>>>** 100, 83, 200, 300, **388**
- 100, 83, 200, 300, 276 <<<<< **final set of hole**
- P4 (426K) must wait

In this example, Best-fit turns out to be the best ***because there is no wait processes.***

* * * *

8.14 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 the logical address?
b. How many bits are there in the physical address?

Answer:

Method1:

a) m =???

Size of logical address space $= 2^m =$ # of pages × page size

$2^m = 64 \times 1024$

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

$2^m = 2^{16}$ »» **m=16 bit**

Method2:

m =???

\# of pages $= 2^{m-n}$

n =???

Page size$=2^n$

$1024=2^n$

$2^{10}=2^n$ »» n=10 bit

Again: # of pages $= 2^{m-n}$

$64=2^{m-10}$

$2^6=2^{m-10}$

6= m-10 »» **m=16 bit**

b)

Let **(x)** is number of bits in the physical address

x =???

Size of physical address space $= 2^x$

Size of physical address space = # of frames × frame size

(frame size = page size)

Size of physical address space = 32 × 1024

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

$2^x = 2^{15}$

»» number of required bits in the physical address=**x =15 bit**

* * * *

8.22 Consider a logical address space of 32 pages of 1024 words per page, mapped onto a physical memory of 16 frames.
a. How many bits are required in the logical address?
b. How many bits are required in the physical address?

Answer:

a) m =???

Size of logical address space $= 2^m$ = # of pages × page size
$= 32 \times 1024 = 2^{15}$ »» m=15 bit

b) Size of physical address space = # of frames × frame size
(frame size = page size)

Size of physical address space $= 16 \times 1024 = 2^{14}$
»» number of required bits in the physical address =14 bit

* * * *

8.19 Assuming a 1-KB page size , What are the *page numbers* and *offsets* for the following address references (provided as decimal numbers)

a. 2375
b. 19366
c. 30000
d. 256
e. 16385

Answer:
Page size $=2^n$ =1024 B= 2^{10} B
# of bits in offset part (n) =10

Solution steps :

1. Convert logical address: Decimal → Binary
2. Split binary address to 2 parts (page # , Offset), offset : *n* digits
3. Convert offset & page# : Binary→ Decimal

Logical address (decimal)	***Logical address (binary)***	***Page # (6 bits) (binary)***	***Offset (10 bits) (binary)***	***Page # (decimal)***	***Offset (decimal)***
2375	0000 1001 0100 0111	0000 10	01 0100 0111	2	327
19366	0100 1011 1010 0110	0100 10	11 1010 0110	18	934
30000	0111 0101 0011 0000	0111 01	01 0011 0000	29	304
256	0000 0001 0000 0000	0000 00	01 0000 0000	0	256
16385	0100 0000 0000 0001	0100 00	00 0000 0001	16	1

* * * *

8.10 Consider a paging system with the page table stored in memory.
a. If a memory reference takes 200 nanoseconds, how long does a paged memory reference take?
b. If we add associative registers, and 75 percent of all page-table references are found in the associative registers, what is the effective memory reference time? (Assume

that finding a page-table entry in the associative registers takes zero time, if the entry is there.)

Answer:
a. memory reference time= 200+200= 400 ns
(200 ns to access the page table in RAM and 200 ns to access the word in memory)

b.
Case (1) : ***page entry found in associative registers (part1)***
Memory access time = 0+200=200 ns
(0 ns to access the page table in associative registers and 200 ns to access the word in memory)

Case (2) : ***page entry NOT found in associative registers (part1) but found in page table in RAM***
Memory access time = 0+200+200=400 ns
(0 ns to access the page table in associative registers (part1) ,200 ns to access the page table(part2) in RAM and 200 ns to access the word in memory)

>>> Effective access time =∑ [probability of the case × access time of this case**]**
Effective access time = [0.75 × 200]+ [0.25 × 400]= 250 ns.

* * * *

8.18 Consider a computer system with a 32-bit logical address and 4-KB page size . The system supports up to 512MB of physical memory. How many entries are there in each of the following:
a. A conventional single-level page table

Answer:
a) # of pages= # of entries =????
Size of logical address space $= 2^m$ = # of pages × page size
»» 2^{32} = # of pages × 2^{12}
# of pages $=2^{32} / 2^{12}= 2^{20}$ pages

* * * * *

8.21 Consider the following segment table:

Segment	Base	Length	
0	219	600	
1	2300	14	
2	90	100	>> there is a typing error in this row in book
3	1327	580	
4	1952	96	

What are the physical addresses for the following logical addresses?
a. 0,430
b. 1,10
c. 2,500
d. 3,400
e. 4,112

Answer:
a. (430 < 600)*....True* >>>>> physical address= 219 + 430 = 649
b. (10 < 14))*....True* >>>>> physical address= 2300 + 10 = 2310
c. (500 < 100))*....False* >>>>> **illegal reference**, trap to operating system
d. (400 < 580))*....True* >>>>> physical address= 1327 + 400 = 1727
e. (112 < 96) *....False* >>>>> **illegal reference**, trap to operating system

* * * *

8.17 Describe a mechanism by which one segment could belong to the address space of two different processes.

Answer: Since segment tables are a collection of base–limit registers, segments can be shared when entries in the segment table of two different jobs point to the same physical location. *The two segment tables must have identical base and limit values.*

* * * *

((أخيتي.....لا تنسينا من صالح دعائك))

8.2 Compare the main memory organization schemes of contiguous-memory allocation, pure segmentation, and pure paging with respect to the following issues:
a. external fragmentation
b. internal fragmentation
c. ability to share code across processes

Answer:

<table>
<tr><th>Method</th><th>External fragmentation</th><th>Internal fragmentation</th><th>ability to share code</th></tr>
<tr><td>contiguous-memory allocation (variable size method)</td><td>There is external fragmentation (as address spaces are allocated contiguously and holes develop as finished processes release its space and new processes are allocated and the size of the new process is almost smaller than the old one)</td><td>There is no internal fragmentation</td><td rowspan="2">It does not allow processes to share code.</td></tr>
<tr><td>contiguous-memory allocation (fixed size partition method)</td><td>There is no external fragmentation</td><td>There is internal fragmentation (the size of the partition is almost bigger than the process’s size)</td></tr>
<tr><td>pure paging</td><td>There is no external fragmentation</td><td>There is internal fragmentation (it appears in the last frame because the process size almost not a multiplex of page size)</td><td>Able to share code between processes</td></tr>
<tr><td>pure segmentation</td><td>There is external fragmentation (fragmentation would occur as segments of finished processes are replaced by segments of new processes.
and the size of the new process is almost smaller than the old one)</td><td>There is no internal fragmentation</td><td>Able to share code between processes</td></tr>
</table>

* * * *

Good Luck

Reem Al_Salih

>>How to convert from logical address to physical address in paging???

Physical address = base address + offset
= (frame # * frame size) + offset

IMPORTANT NOTE:

Paging arithmetic laws:

Page size = frame size

Logical address space (/size) = 2^m

Physical address space (/size) = 2^x (*where **x** is the number of bits in physical address*)

Logical address space (/size) = # of pages × page size

Physical address space (/size) = # of frames × frame size

Page size= frame size= 2^n

# of pages= 2^{m-n}

# of entries (records)in page table = # of pages

>>How to convert from logical address to physical address in segmentation???

IF (offset < limit) then
    Physical address= base address+ offset
Else
    Trap (address error)

Effective access time =
∑ [probability of the case × access time of this case]

Where (∑ probability of all cases =100%)

((أخيتي.....لا تنسينا من صالح دعائك))