OS ①

Numericals

① CPU scheduling.

→ Scheduling Algorithms.

* FCFS
* SJF
* Round Robin
* SRTF
* Priority scheduling

} pg # 51-54 // OS book notes.

② Memory Management

→ Numericals.

③ Virtual Memory

→ Page Replacement Algorithms

* FIFO
* LRU
* Optimal
* Second Chance.

→ Numericals.

④ Deadlock.

→ Resource allocation graph

→ Banker's Algorithm.

⑤ Mass storage structure

→ Disk scheduling algorithm

* FCFS
* SSTF
* SCAN
* C-SCAN
* C-LOOK

* Effective Access Time :

$EAT = \alpha \times (ma\ hit) + fail \times (ma\ fail)$

α → hit ratio

eg $\alpha = 80\%$, $ma = 100ns$ → double

$EAT = (0.80 \times 100) + (1-0.80)(200) = 120\ ns.$

eg $\alpha = 99\%$, $ma = 100ns$

$EAT = (0.99 \times 100) + (1-0.99)(200) = 101\ ns.$

* Page-table entries :

eg : logical addr space = 32 bits

page size = $4kb = (4 \times 1024) = 4096$

$= 2^{12}$

Page table $= 2^{32}/2^{12} = 2^{20}$ entries.

* EAT examples.

① There is an 80% hit ratio of desired page. If it takes 20ns to search in TLB and 100ns to access memory. Find EAT.

to access desired byte

$EAT = 0.80(20+100) + (1-0.80)(20+100+100)$

$= 140\ ns.$

(access pg table & frame no)

Virtual Memory :-

* EAT for a demand-paged memory.

$EAT = (1-p) \times ma + p \times \text{page fault time}$

where

p = probability of page fault.

eg $ma = 200\ ns$, Avg page fault = 8 ms.
If one access out of 1000 causes a page fault then $p = 1/1000 = 0.001$.

$1\ ms = 1 \times 10^6\ ns$.

$EAT = (1-0.001)\,200 + 0.001(8 \times 10^6)$
$= 8199.8\ ns$.

* Proportional Allocation:

$$a_i = \frac{s_i}{S} \times m$$

where $S = \Sigma s_i$ & m = total no of frames

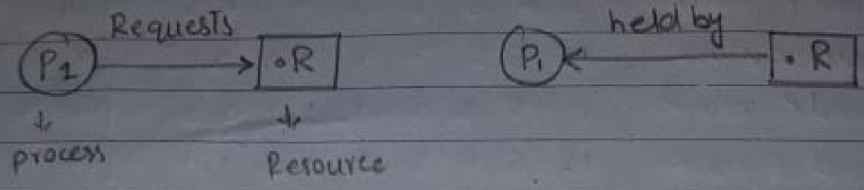
eg 62 frames, $P_1 = 10$ pages, $P_2 = 127$ pages

$$a_1 = \frac{10}{10+127} \times 62 \cong 4$$

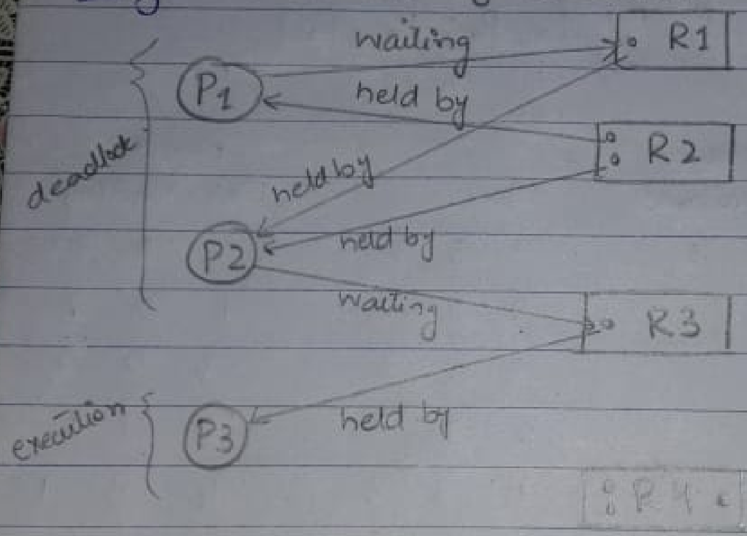
$$a_2 = \frac{127}{10+127} \times 62 \cong 57.$$

Deadlocks:

* Resource allocation graphs:-



eg: Slide 17 of Deadlock.



* Banker's Algorithm / Safety Algorithm:

Q1

	Allocation	Max	work = Available	Need Max - Allocated
P_0	2 0 2 1	9 5 5 5	6 3 5 4	7 5 3 4
P_1	0 1 1 1	2 2 3 3		2 1 2 2
P_2	4 1 0 2	7 5 4 4		3 4 4 2
P_3	1 0 0 1	3 3 3 2		2 3 3 1
P_4	1 1 0 0	5 2 2 1		4 1 2 1
P_5	1 0 1 1	4 4 4 4		3 4 3 3

	Condition Check	Work = Work + Allocation.
P_0	Need > Work *	
P_1	Need < work	6 3 5 4 + 0 1 1 1 = [6 4 6 5]
P_2	Need < work	6 4 6 5 + 4 1 0 2 = [10 5 6 7]
P_3	Need < work	10 5 6 7 + 1 0 0 1 = [11 5 6 8]
P_4	Need < work	11 5 6 8 + 1 1 0 0 = [12 6 6 8]
P_5	Need < work	12 6 6 8 + 1 0 1 1 = [13 6 7 9]
P_0	Need < Work	13 6 7 9 + 2 0 2 1 = [15 6 9 10]

Safe sequence - { P_1, P_2, P_3, P_4, P_5, P_0 }.

Q2

	Allocation	Work = Available	Max	Need = Max - Allocated
P_0	0 1 0	3 3 2	7 5 3	7 4 3
P_1	2 0 0		3 2 2	1 2 2
P_2	3 0 2		9 0 2	6 0 0
P_3	2 1 1		2 2 2	0 1 1
P_4	0 0 2		4 3 3	4 3 1

	Condition Check	Work = Work + allocation.
P_0	Need > Work*	
P_1	Need < Work	3 3 2 + 2 0 0 = [5 3 2]
P_2	Need > Work*	
P_3	Need < Work	5 3 2 + 2 1 1 = [7 4 3]
P_4	Need < Work	7 4 3 + 0 0 2 = [7 4 5]
P_0	Need < Work	7 4 5 + 0 1 0 = [7 5 5]
P_2	Need < Work	7 5 5 + 3 0 0 = [10 5 5]

Safe Sequence = { P_1, P_3, P_4, P_0, P_2 }.

Disk scheduling algorithms:
(From slides).

* Calculate logical & physical address bits

eg 1: Logical address space of 64 pages of 1024 words mapped to 32 frames.

($64 = 2^6$, $1024 = 2^{10}$, $32 = 2^5$)

Logical address bits:

$= 64 \cdot 1024$

$= 2^6 \cdot 2^{10}$

$= 16$ bits

Physical address bits

$= 32 \cdot 1024$

$= 2^5 \cdot 2^{10}$

$= 15$ bits.

eg 2: logical address space of 32 pages with 2048 words mapped to 16 frames.

logical address bits:

$= 32 \cdot 2048$

$= 2^5 \cdot 2^{11}$

$= 16$ bits.

Physical address bits:

$= 16 \cdot 2048$

$= 2^4 \cdot 2^{11}$

$= 2^{15}$

$= 15$ bits.

* Calculate page no. & offset of 1kb page size for following address references

(a) 2375

$$\begin{array}{r} 2 \rightarrow \text{pg no\#} \\ 1024 \overline{)\,2375} \\ 2048 \\ \hline 327 \rightarrow d\ (\text{offset}) \end{array}$$

(b) 19366
pg # 18
d = 934

(c) 256
pg # 0
offset # 256

(d) 3000
pg # 29
offset # 304

(e) 16385
pg # 16
offset # 1.

{ page size in any hardware
= 512 bytes ⟷ 1 giga byte }

* Page s = Process / Page size
* d = Process % page size.

* Internal Fragmentation:

Internal Fragmentation = Allocated − Requested.

eg Pg size = 2048 bytes , Process size = 72766 bytes
No. of pages = 35 pages + 1086 bytes.

Internal fragmentation = 2048 − 1086 = 962 bytes.

② If hit ratio = 98%

$EAT = (0.98)(20+100) + (1-0.98)(20+100+100)$

$= 122$ ns.

* Two-level paging algorithm.

eg 32-bit logical address space

page size = 4KB $= 2^{12}$

Page no # 20 bits

page offset = 12 bits.

page number		page offset
P_1	P_2	d
10	10	12

32-bits

Memory Management ended!

* Calculate logical & physical address bits

eg 1: Logical address space of $\overset{2^6}{64}$ pages of $\overset{2^{10}}{1024}$ words mapped to $\underset{2^5}{32}$ frames.

Logical address bits:

$= 64 \cdot 1024$

$= 2^6 \cdot 2^{10}$

$= 16$ bits

Physical address bits

$= 32 \cdot 1024$

$= 2^5 \cdot 2^{10}$

$= 15$ bits.

eg 2: logical address space of 32 pages with 2048 words mapped to 16 frames.

logical address bits:

$= 32 \cdot 2048$

$= 2^5 \cdot 2^{11}$

$= 16$ bits.

Physical address bits:

$= 16 \cdot 2048$

$= 2^4 \cdot 2^{11}$

$= 2^{15}$

$= 15$ bits.

Best Fit : (accomodate all processes) .

212 Kb → 300 Kb
417 Kb → 500 Kb
112 Kb → 200 Kb
426 Kb → 600 Kb

Worst Fit :

212 Kb → 600 Kb
417 Kb → 500 Kb
112 Kb → 300 Kb
426 Kb → wait

* Address translation :

eg :

Segment	Base	limit/ length	logical Addr	Base+limit
0	219	600	430	819
1	2300	14	10	2314
2	90	100	500	190
3	1327	580	400	1907
4	1952	96	112	2048

Physical Address / Base + logical .

430 + 219 = 649
10 + 2300 = 2310
500 + 90 = 590 (trap)(out of limit)
400 + 1327 = 1727
112 + 1952 = 2064 (trap)(out of limit).

Memory Management :=

* Swap in time = size of process / transfer rate

eg: 100 MB process swapping to hard disk with transfer rate of 50MB/sec.

swap in time = $100/50$ = 2 sec.

* Swap out time = swap in time.

* Total context switch time = swap in + swap out.

* Dynamic storage - allocation :
  1. First fit (First hole, big enough)
  2. Best fit (Smallest hole; big enough)
  3. Worst fit (Largest hole)

eg:

Memory Partitions: 100kb, 500kb, 200kb, 300 kb, 600 kb

Processes: 212 Kb, 417 Kb, 112 kb, 426 Kb.

First Fit

212 Kb → 500kb

417 Kb → 600kb

112 Kb → 200 kb

426 Kb → Wait (b/c no partition available)