



Question 1 - 15 points - answer either Q1A or Q1B, but not both

(A) Given the memory partitions of 600K, 300K, 700K and 200K in order, how would each of the First-Fit, Best-Fit algorithms place processes of 230 K, 525K, 175K and 300K (in order)?

First Fit：

230K to 600K

525K to 700K

175K to 300K

300K process will not be loaded.

Best Fit：

230K to 300K

525K to 600K

175K to 200K

300K to 700K.

(B) Consider a paging system with the page table stored in memory and registers.

If 70% of all page-table references are found in the TLBs, what is the effective access time? Assume that finding a page-table entry in the TLBs takes 25ns and a memory access takes 175ns. Give a literal solution first (formula).

ma = 175 ns , TLB time = 25 ns

e.a.t. = TLB miss time + TLB hit time

= TLB miss rate\*(1ma+1ma) + TLB hit rate\*(1ma+TLB time)

= (1-0.7)\*(175+175) + (0.7)\*(175+25)

= 0.3\*350 + 0.7\*200

= 245 ns

Question 2 - 20 points

1. Define INTERNAL FRAGMENTATION. Give examples of memory allocation schemes(at least 2) that suffer from internal fragmentation.

Internal Fragmentation is when a process is loaded into main memory and part of the allocated space for that process is not being used. This is a waste of space.

Example : Fixed Equal Sized Partitioning ; Fixed Unequal Size Partitioning.

Fixed Equal Sized Partitioning :

100K

If P1 needs 25K of memory => 75K will be wasted

100K

If P2 needs more than 100k of memory => P2 will be kept

100K

in Queue

100K

Fixed Unequal Sized Partitioning :

25K 1

If P1 needs 100K => go to 4

50K 2

250K 3

If P2 needs 50K => go to 2

100K 4

If P3 needs 100K => go to 3 and then 150K is wasted, still

internal fragmentation

1. Describe ASSOCIATIVE MAPPING. Give its advantages and disadvantages.

Page tables are in CPU registers.

e.a.t. = 1 register search + 1ma

Register

PA (Physical Address)

LA (Logical Address)

F#

Key

F#

d

d

P#

CPU

Advantages : It is very fast.

Disadvantage : It is expensive – the time necessary of search the associative registers and takes a lot of CPU registers

Question 3 - 25 points

(A) Consider the 2-dimensional array A: int a[][] = new int[9][9];

Where A[0][0] is at location 89, in a paged system with pages of size 27. As small process is in page 1 for manipulating the matrix; every instruction fetch will be from page 1.

For 3 page frames, how many page faults are generated by the following

array-initialization loop? All frames were initially empty. An LRU algorithm is used:

for(int j=0; j<0; j++)

{

for(int i=0; i<9; i++)

{

A[i][j]=0;

}

}

In the solution of this problem, you should:

-Show the contents of the logical address space, how the array and

instructions are stored

-Give the reference string generated by the CPU during the

initialization

-On the reference string, show the page faults

-For each reference, explain what element is initialized or if the

instruction is fetched

-Explain the final computation of the page fault number

A[9][9]

A[0][0] location 89

Page size = 27

Page 1 : instruction

LRU (Least Recenting Used)

Logical Address Space Frame

Page1

1. 0-26

Page3

change to page 5

1. 27-53

Instruction

Page4

change to page 6

n be change

2) 54-80

3) 81-107 Location starting from 89

Row0 A[0][0] - A[0][8]

Row1 A[1][0] - A[1][8]

4) 108-134 Row2 A[2][0] - A[2][8]

Row3 A[3][0] - A[3][8]

Row4 A[4][0] - A[4][8]

5) 135-161 Row5 A[5][0] - A[5][8]

Row6 A[6][0] - A[6][8]

Row7 A[7][0] - A[7][8]

6) 162-188 Row8 A[8][0] - A[8][8]

Instruction ; A[0][0] PF; Instruction ; A[1][0] No-PF ;

Instruction ; A[2][0] PF; Instruction ; A[3][0] No-PF ; Instruction ; A[4][0] No-PF

Instruction ; A[5][0] PF; Instruction ; A[6][0] No-PF ; Instruction ; A[7][0] No-PF

Instruction ; A[8][0] PF;

4 Page Faults \* 9 Columns = 36 Page Faults

(B) Is this the most efficient initialization? Briefly explain your answer.

No since the array is initializing by column first it will go through many more page faults. A more efficient way is to initialize by row. Since the Array is stored in numerical order, the OS will only have to service a page fault once for every page the Array is stored in. This equates to 4 page faults. For the coloumn to be initialized first, the OS must page fault 36 times to initialize the array.

(c) What is Thrashing?

When resources have become too little or exhausted. A process will ask the Operating System for resources which in turn the OS will take resources from another process and give it to the new process. Then the previous process will also ask for resources and the OS will take resources from another process or the new process. This will continue to happen which leaves the computer gain little or no progress. All of the time is spent allocating resources over and over.

Question 4 - 30 points

Consider the logical address spaces and page tables for P1 and P2. The main memory contains pages of P1 and P2. The main memory has 12 frames. There are free frames.

(A) Draw a picture that will illustrate the content of the main memory (what variable, from P1 and P2, each frame might contain).

0

1 Process 2 (B)

2

3 Process 1 (S)

4 Process 1 (T)

5

6

7 Process 1 (L)

8 Process 2 (U)

9

10 Process 1 (K)

11

(B) Describe the steps that will be followed by the OS and MMU in case that the next instruction to be executed by the CPU for P2 is: load M. A reference to variable M is a legal reference.

NOTE: You can draw a picture to illustrate your answer. Show any table update

Since there are free frames, M will be loaded into the Main Memory Frame (any free frame will do), the Page Table will be updated, M V/I will change from 0 to 1 and the Frame will change from DA to which ever frame has been chosen by OS. And execution will start.

(C) For the logical or physical address, registers of size 16 are used. The offset field takes 6 bits, page or frame number fields take 10 bits. Offset for variable L is 18.

-What will be the logical address generated by the CPU for load L?

-What will be the physical address generated by the CPU for load L?

NOTE: The logical/physical address is the binary representation of 2 fields

(page/frame number and displacement)

Offset size = 6

Offset = 18 (二進位010010)

Page or Frame number size 10

logical address = page number + offset

page number = 1

(page number 1 二進位換算後0000000001)

LA = 0000000001 010010

Offset

Page number

physical address = frame number + offset

frame number = 7

(frame number 7 二進位換算後0000000111)

Offset

Frame umber

PA = 0000000111 010010

Logical Address Space for P1 Logical Address Space for P2

0 O 0 B

1 L 1 M

2 S 2 U

3 N 3 C

4 K

5 T

Page Table P1 Page Table P2

P f V/I p f V/I

0 disk address 0 0 1 1

1 7 1 1 disk address 0

2 3 1 2 8 1

3 disk address 0 3 disk address 0

4 10 1

5 4 1