CIT 5950 Recitation 6 - Scheduling & Virtual Memory

Welcome back to recitation! We're glad that you're here :)

Exercise 1 - Scheduling

Consider the following set of tasks/processes:

Name	Arrival Time	Running Time
Bert	2	11
Ernie	0	8
Oscar	12	20
Grover	7	15
Elmo	10	4

a) Using the **Round Robin** scheduling algorithm and a time slice of **8**, what is the finishing time for each?

0: Ernie arrives, starts executing
2: Bert arrives; queue is Ernie □ Bert
7: Grover arrives; queue is Ernie □ Bert □ Grover
8: **Ernie finishes**; queue is Bert □ Grover; Bert starts executing
10: Elmo arrives; queue is Bert □ Grover □ Elmo
12: Oscar arrives; queue is Bert □ Grover □ Elmo □ Oscar
16: Bert's time slice expires (3 left); queue is Grover □ Elmo □ Oscar □ Bert; Grover starts executing
24: Grover's time slice expires (7 left); queue is Elmo □ Oscar □ Bert □ Grover; Elmo starts executing
28: **Elmo finishes**; queue is Oscar □ Bert □ Grover; Oscar starts executing
36: Oscar's time slice expires (12 left); queue is Bert □ Grover □ Oscar; Bert starts executing
39: **Bert finishes**; queue is Grover □ Oscar; Grover starts executing
46: **Grover finishes**; Oscar starts executing
54: Oscar's time slice expires (4 left); Oscar keeps executing
58: **Oscar finishes**

Name	Finishing Time
Bert	39
Ernie	8
Oscar	58

Grover	46
Elmo	28

b) What is the **average waiting time?**

Waiting time = finish – running – arrival
Bert: 39 – 11 – 2 = 26
Ernie: 8 – 8 – 0 = 0
Oscar: 58 – 20 – 12 = 26
Grover: 46 – 15 – 7 = 24
Elmo: 28 – 4 – 10 = 14
Average = (26 + 0 + 26 + 24 + 14) / 5 = 90 / 5 = 18

Question #2

Consider a system as follows:

- 32-bit address space
- 16-bit addressable
- 1GB of physical memory
- page sizes of 64kB

a) How many **pages** are there in virtual memory? Express your answer as a power of 2

2^17 or 128k

32-bit address space -> 2^32 addresses

16-bit (2-byte) addressable -> each address is 2 bytes -> 2^33B virtual memory

Each page is 64kB

2^33B (virtual memory size) / 64kB (page size) = 2^33 / 2^16 = 2^17 (or 128k)

b) How many **frames** are there in physical memory?

2^14 or 16k

Frame size = page size

Physical memory is 1GB, so 1GB (total size) / 64kB (frame size) = 2^30 / 2^16 = 2^14 (or 16k)

c) How many **bits** are there in each address' page number?

17

There are 2^17 pages

To represent N pages we need log_2 N bits

So we need log 2^17 = 17 bits

d) Consider the virtual address xABCDEF01. What is its **page number** in hexadecimal?

x1579B

In binary, the virtual address is 1010 1011 1100 1101 1110 1111 0000 0001

There are 2^17 pages so the first 17 bits are the page number.

So the page number is 1010 1011 1100 1101 1.

In hexadecimal, that's x1579B.

Question #3

We are working with a byte-addressable system that has a 16-bit address space, 32kB of physical memory, and page sizes of 8kB. Assume the page table is initially empty, and then a process generates the following sequence of virtual addresses:

x3311	**001**1
x1234	**000**1
x1255	**000**1
x3456	**001**1
xA349	**101**0
x7777	**011**1
xB222	**101**1
x6222	**011**0

a) If virtual address `x5324` is requested next, which page will be evicted if using a First In First Out (FIFO) replacement algorithm? State the page number.

001

The page number is the first three bits, because there are eight virtual pages (16-bit address space = 64k addresses; each holds 1 byte so 64kB total; 8kB per page so 64kB/8kB = 8).

The four virtual addresses above have page numbers 000, 001, 101, and 011, and those take up the four frames (there are four frames because there's 32kB physical memory, and 32kB/8kB = 4).

When we get virtual address x5324, the page number is 010, and this causes an eviction.

The one that's oldest will be evicted, which in this case is the first one to be loaded, which is page number 001.

b) Instead of using FIFO, which page will be evicted if using a Least Recently Used (LRU) replacement algorithm? State the page number.

000

Using LRU, it's page number **000** that has been last used furthest in the past, so it gets evicted.

c) Rather than using FIFO or LRU, imagine that the system could look into the future and see that the next four virtual address requests (after `x5324`) would be as follows:

`x1A23`
`x399A`
`x7282`
`x4A32`

Knowing this information, which page should be evicted when the request for `x5324` generates a page fault?

101

As explained above, when x5324 is requested, the page numbers that are in the page table (i.e., that are mapped to frames) are 000, 001, 101, and 011, and the page number for address x5324 is 010 (the first three bits).

Given the requests indicated above, it would make sense to indicate page number **101** (which contains the addresses xA349 and xB222), since it is not used in any of the subsequent requests, whereas all the other page numbers are.