# SOFTENG 370 Assignment 3

Name: Aiden Burgess

UPI: abur970

## Question 1

Number of ptrs/block = 4KB/4 = 1024 ptrs/block

Assume that we can use blocks defined in previous questions

### a)

16 * 4KB = 65,536 B

### b)

65,536 + 1024 * 4KB = 4,259,840 B

### c)

4,259,840 + 1024^2 * 4KB = 4,299,227,136 B

### d)

4,299,227,136 + 1024^3 * 4KB = 4,402,345,738,240 B

### e)

5 levels (1024^5 * 4KB = 1024^6 * 4B)

### f)

Direct block read 4,050-> 4,096

Direct block read 4,096-> 4,150

Read: 2

Write: 0

### g)

Read in single indirect block

Block read 4,259,820 -> 4,259,840

Read in double indirect block

Read in 2nd layer double indirect block

Read 4,259,840 -> 4,259,920

Read: 5

Write: 0

### h)

Read in double indirect block @ 4,259,840

Read 2nd layer double indirect block @ 4,259,840

Read 4,263,900 -> 4,263,936

Read 2nd layer double indirect block @ 4,263,936

Read 4,263,936-> 4,264,000

Read: 5

Write: 0

### i)

Read in single indirect block

Write 4,259,820 -> 4,259,840

Read in double indirect block @ 4,259,840

Read 2nd layer double indirect block @ 4,259,840

Write @ 4,259,840 -> 4,259,920

Write on disk inode update

Read: 3

Write: 3

### j)

Read in single indirect block

Write 4,259,820 -> 4,259,840

Read in double indirect block @ 4,259,840

Assign and Write 2nd layer double indirect block @ 4,259,840 to 1st layer

Assign and Write 2nd layer double indirect block @ 4,259,840 new block

Write 4,259,820 -> 4,259,840

Write on disk inode update

Read: 2

Write: 4

### k)

1. The inode itself is a block on disk, so if we did not have this assumption, then there would at least be an extra read for each  operation.
2. If the blocks are already in memory at the start, then there is no need to read them in from disk in the operations.
3. File access time writing would add at least one extra write to each operation, and this is not integral to the main task of reading/writing from/to the disk
4. This assumption is made due to j), which requires new blocks to be assigned. Without this assumption, there would be an extra read.
5. This adds an extra write to the operations which modify the file.
6. This assumption clarifies assumption (5), as we only need one write per inode update.
7. Some file systems use clusters of blocks, which may affect our read calculations. As this cluster information is not provided, it is best to make this assumption.

# Question 2

Current Intel and AMD CPUs are compatible with both 32 and 64 bit code.

Moving to 64 bits increases the address space, and data values are also 64 bits each. Programs can compute larger calculations in the same amount of time as more information is held in each data value.

# Question 3

$$\#Frames = \frac{16GB}{8KB}$$

$$\#Frames = 2,097,152$$

$$\#Free\_frames = 1,048,576$$

### Space requirements for bitmap

1 bit for each frame, so 2,097,152 bits or 256KB

### Space requirements for linked list

64 bits per free frame, so

$$Space = 8B \times 1,048,576 = 8MB$$

### Extents version

Extent-based systems have collections of contiguous allocation. As there is only ever 1 frame in a row, the extent based system would look the same as the linked list system.

However, there would probably be a higher overhead with the extent-based system as it has to store starting VCN, starting LCN and cluster count for each free frame, which is probably higher than the 64 bits used per frame with the linked list system.

# Question 4

$$2^{32}\ bits\ per\ processor$$

Each process can potentially use the entire virtual address space, which is 512MB per processor and 512GB overall

This is not realistic as it is unlikely all process would use the entire address space at the same time.

# Question 5

$$EAT = 2\beta - \alpha\beta + \epsilon + page\_fault\_time$$

$$EAT = 2 \times 50ns - 0.99 \times 50ns + 1ns + 0.0001 \times 5ms$$

$$EAT = 551.5ns$$

# Question 6

$$\#Entries\_in\_table = \frac{2^{22} bits}{2^6 bytes} = 2^{16}$$

# Question 7

*Parameters:*

- Normal instruction: 1ns
- Page fault: 2,000,000ns
- Program runtime: 60s
- # Page faults: 20,000

$$20,000 \times 2ms + NormalInstructionTime = 60s$$

$$40s + NormalInstructionTime = 60s$$

$$NormalInstructionTime = 20s$$

Assume 10,000 page faults as we double the amount of memory

$$\#Normal_instr.1ns + 20,000.2ms = 60s$$

$$Normal_instr.1ns + 20,000.2ms = 60s$$

$$Runtime = 20s + 10,000 \times 2ms$$

$$Runtime = 40s$$

# Question 8

## FIFO

| | 1 | 2 | 3 | 4 | 5 | 4 | 3 | 2 | 1 | 6 | 7 | 1 | 2 | 3 | 7 | 5 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | = | = | = | 5 | = | = | = | = | = | = | = | 2 | = | = | = | = |
| 0 | = | 2 | = | = | = | = | = | = | 1 | = | = | = | = | 3 | = | = | = |
| 0 | = | = | 3 | = | = | = | = | = | = | 6 | = | = | = | = | = | 5 | = |
| 0 | = | = | = | 4 | = | = | = | = | = | = | 7 | = | = | = | = | = | 1 |

Number of page faults: 12

## LRU

| | 1 | 2 | 3 | 4 | 5 | 4 | 3 | 2 | 1 | 6 | 7 | 1 | 2 | 3 | 7 | 5 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | = | = | = | 5 | = | = | = | 1 | = | = | = | = | = | = | 5 | = |
| 0 | = | 2 | = | = | = | = | = | = | = | = | = | = | = | = | = | = | 1 |
| 0 | = | = | 3 | = | = | = | = | = | = | = | 7 | = | = | = | = | = | = |
| 0 | = | = | = | 4 | = | = | = | = | = | 6 | = | = | = | 3 | = | = | = |

Number of page faults: 11

## LFU

| | 1 | 2 | 3 | 4 | 5 | 4 | 3 | 2 | 1 | 6 | 7 | 1 | 2 | 3 | 7 | 5 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | = | = | = | 5 | = | = | = | 1 | 6 | 7 | 1 | = | = | 7 | 5 | 1 |
| 0 | = | 2 | = | = | = | = | = | = | = | = | = | = | = | = | = | = | = |
| 0 | = | = | 3 | = | = | = | = | = | = | = | = | = | = | = | = | = | = |
| 0 | = | = | = | 4 | = | = | = | = | = | = | = | = | = | = | = | = | = |

Number of page faults: 12

**Optimal**

| | 1 | 2 | 3 | 4 | 5 | 4 | 3 | 2 | 1 | 6 | 7 | 1 | 2 | 3 | 7 | 5 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | = | = | = | 5 | = | = | = | = | 6 | 7 | = | = | = | = | = | = |
| 0 | = | 2 | = | = | = | = | = | = | = | = | = | = | = | = | = | 5 | = |
| 0 | = | = | 3 | = | = | = | = | = | = | = | = | = | = | = | = | = | = |
| 0 | = | = | = | 4 | = | = | = | = | 1 | = | = | = | = | = | = | = | = |

Number of page faults: 9