SYSC 4001 Assignment 3

Chukwuemeka Igwe (101219454).

Agrim Kasaju (10128010)

Part 3 – Concepts Part 3 – Concepts [2 marks]
Answer the following questions. Justify your answers. Show all your work.

1. Page Faults

Given reference string:
201,302,203,404,302,201,205,206,302,201,302,203,207,206,203,302,201,302,203,206

(i) **3 Frames Allocated**

(a) **LFU Algorithm**:

- The LFU algorithm replaces the least frequently used page in memory when a page fault occurs. Each time a page is referenced, we update its usage count. When a page fault happens, the page with the lowest usage count is replaced.
- **Number of page faults**: **14 faults**.

[201]	**page fault = 1**
[201, 302]	**page fault = 2**
[201, 302, 203]	**page fault = 3**
[302, 203, 404]	**page fault = 4**
[302, 203, 404]	**page fault = 4**
[302, 404, 201]	**page fault = 5**
[302, 201, 205]	**page fault = 6**
[302, 205, 206]	**page fault = 7**
[302, 205, 206]	**page fault = 7**
[302, 206, 201]	**page fault = 8**
[302, 206, 201]	**page fault = 8**
[302, 201, 203]	**page fault = 9**
[302, 203, 207]	**page fault = 10**
[302, 207, 206]	**page fault = 11**
[302, 206, 203]	**page fault = 12**
[302, 206, 203]	**page fault = 12**
[302, 203, 201]	**page fault = 13**
[302, 203, 201]	**page fault = 13**
[302, 203, 201]	**page fault = 13**
[302, 203, 206]	**page fault = 14**

(b) **FIFO Algorithm (First In, First Out):**

- In FIFO, pages are replaced in the order they were loaded into memory. The page that was loaded first is replaced when a page fault occurs.

- **Number of page faults: 16 faults.**

[201] **page fault = 1**
[201, 302] **page fault = 2**
[201, 302, 203] **page fault = 3**
[302, 203, 404] **page fault = 4**
[302, 203, 404] **page fault = 4**
[203, 404, 201] **page fault = 5**
[404, 201, 205] **page fault = 6**
[201, 205, 206] **page fault = 7**
[205, 206, 302] **page fault = 8**
[206, 302, 201] **page fault = 9**
[206, 302, 201] **page fault = 9**
[302, 201, 203] **page fault = 10**
[201, 203, 207] **page fault = 11**
[203, 207, 206] **page fault = 12**
[203, 207, 206] **page fault = 12**
[207, 206, 302] **page fault = 13**
[206, 302, 201] **page fault = 14**
[206, 302, 201] **page fault = 14**
[302, 201, 203] **page fault = 15**
[201, 203, 206] **page fault = 16**

(c) **Optimal Algorithm**:

- The Optimal algorithm replaces the page that will not be used for the longest time in the future.
- **Number of page faults: 11 faults.**

[201] **page fault = 1**
[201, 302] **page fault = 2**
[201, 302, 203] **page fault = 3**
[201, 302, 404] **page fault = 4**
[201, 302, 404] **page fault = 4**
[201, 302, 404] **page fault = 4**
[201, 302, 404] **page fault = 4**
[201, 302, 205] **page fault = 5**
[201, 302, 206] **page fault = 6**
[201, 302, 206] **page fault = 6**
[201, 302, 206] **page fault = 6**
[201, 302, 206] **page fault = 6**
[201, 302, 206] **page fault = 6**
[302, 206, 203] **page fault = 7**
[206, 203, 207] **page fault = 8**
[206, 203, 207] **page fault = 8**
[206, 203, 207] **page fault = 8**
[206, 203, 302] **page fault = 9**
[203, 302, 201] **page fault = 10**

[203, 302, 201] **page fault = 10**
[203, 302, 201] **page fault = 10**
[203, 302, 201] **page fault = 10**
[203, 302, 206] **page fault= 11**

(ii) 5 Frames Allocated

(a) LFU Algorithm:

Number of page faults: 8 faults.

[201]	**page fault = 1**	**frequency: 201=1**
[201, 302]	**page fault = 2**	**frequency: 201=1 302=1**
[201, 302, 203]	**page fault = 3**	**frequency: 201=1 302=1 203=1**
[201, 302, 203, 404]	**page fault = 4**	**frequency: 201=1 302=1 203=1 404=1**
[201, 302, 203, 404]	**page fault = 4**	**frequency: 201=1 302=2 203=1 404=1**
[201, 302, 203, 404]	**page fault = 4**	**frequency: 201=2 302=2 203=1 404=1**
[201, 302, 203, 404, 205]	**page fault = 5**	**frequency: 201=2 302=2 203=1 404=1 205=1**
[201, 302, 404, 205, 206]	**page fault = 6**	**frequency: 201=2 302=2 404=1 205=1 206=1**
[201, 302, 404, 205, 206]	**page fault = 6**	**frequency: 201=2 302=3 404=1 205=1 206=1**
[201, 302, 404, 205, 206]	**page fault = 6**	**frequency: 201=3 302=3 404=1 205=1 206=1**
[201, 302, 404, 205, 206]	**page fault = 6**	**frequency: 201=3 302=4 404=1 205=1 206=1**
[201, 302, 205, 206, 203]	**page fault = 7**	**frequency: 201=3 302=4 205=1 206=1 203=1**
[201, 302, 206, 203, 207]	**page fault = 8**	**frequency: 201=3 302=4 206=1 203=1 207=1**
[201, 302, 206, 203, 207]	**page fault = 8**	**frequency: 201=3 302=4 206=2 203=1 207=1**
[201, 302, 206, 203, 207]	**page fault = 8**	**frequency: 201=3 302=4 206=2 203=2 207=1**
[201, 302, 206, 203, 207]	**page fault = 8**	**frequency: 201=3 302=5 206=2 203=2 207=1**
[201, 302, 206, 203, 207]	**page fault = 8**	**frequency: 201=4 302=5 206=2 203=2 207=1**
[201, 302, 206, 203, 207]	**page fault = 8**	**frequency: 201=4 302=6 206=2 203=2 207=1**
[201, 302, 206, 203, 207]	**page fault = 8**	**frequency: 201=4 302=6 206=2 203=3 207=1**
[201, 302, 206, 203, 207]	**page fault = 8**	**frequency: 201=4 302=6 206=1 203=3 207=1**

(b) FIFO Algorithm:

Number of page faults: 10 faults.

[201]	**page fault = 1**
[201, 302]	**page fault = 2**
[201, 302, 203]	**page fault = 3**
[201, 302, 203, 404]	**page fault = 4**
[201, 302, 203, 404]	**page fault = 4**
[201, 302, 203, 404]	**page fault = 4**
[201, 302, 203, 404, 205]	**page fault = 5**
[302, 203, 404, 205, 206]	**page fault = 6**
[302, 203, 404, 205, 206]	**page fault = 6**
[203, 404, 205, 206, 201]	**page fault = 7**
[404, 205, 206, 201, 302]	**page fault = 8**
[205, 206, 201, 302, 203]	**page fault = 9**
[206, 201, 302, 203, 207]	**page fault = 10**

(c) Optimal Algorithm:
Number of page faults: 7 faults.

[201]	**page fault = 1**
[201, 302]	**page fault = 2**
[201, 302, 203]	**page fault = 3**
[201, 302, 203, 404]	**page fault = 4**
[201, 302, 203, 404]	**page fault = 4**
[201, 302, 203, 404]	**page fault = 4**
[201, 302, 203, 404, 205]	**page fault = 5**
[201, 302, 203, 205, 206]	**page fault = 6**
[201, 302, 203, 205, 206]	**page fault = 6**
[201, 302, 203, 205, 206]	**page fault = 6**
[201, 302, 203, 205, 206]	**page fault = 6**
[201, 302, 203, 205, 206]	**page fault = 6**
[201, 302, 203, 206, 207]	**page fault = 7**
[201, 302, 203, 206, 207]	**page fault = 7**
[201, 302, 203, 206, 207]	**page fault = 7**
[201, 302, 203, 206, 207]	**page fault = 7**
[201, 302, 203, 206, 207]	**page fault = 7**
[201, 302, 203, 206, 207]	**page fault = 7**
[201, 302, 203, 206, 207]	**page fault = 7**
[201, 302, 203, 206, 207]	**page fault = 7**

2.

(a) Without TLB: 250 ns (page table lookup) +250 ns (memory access) =500 ns

(b) Effective Memory Access Time (EMAT):

Given:

TLB hit time: 30ns

TLB miss time: 30ns

Memory reference time: 250ns

TLB hit rate (p): 80% or 0.8

TLB miss rate: 20% or 0.2 (1-p)

EMAT = (TLB hit rate) × (TLB hit time + memory access time) + (TLB miss rate) ×

(TLB miss time + page table lookup time + memory access time)

EMAT = 0.8 × (30ns + 250ns) + 0.2 × (30ns + 250ns + 250ns)

EMAT = 0.8 × (280ns) + 0.2 × (530ns)

EMAT = 224ns + 106ns = 330ns

(c) Impact of Adding TLB: Adding a TLB reduces the number of page table lookups on hits. If the TLB hit rate is high (e.g., 80%), the EMAT improves significantly, as fewer memory accesses are

required. However, performance can be worse with TLB if the overhead of TLB management such as misses or context switches is greater than the time saved, especially when memory access patterns are random, or tasks change often.

3. **Given**:

Logical address space = 32 pages, each 2 KB.
Physical memory = 1 MB.
a. What is the format of the processor's logical address?

Number of bits for page number: log_2(32) = 5 bits.

Number of bits for offset log_2(2048) = 11 bits.

Logical Address Format: 5+11=16bits.

b. What is the length and width of the page table (disregarding all the control bits)?

Page length: 32

Page width: 4 bytes.

The physical memory space has a size of 1-Mbyte = 2^{20}

Therefore, we require 20 bits for the representation of the page number and a page offset of 11 so we get the difference between the total page number and offset

20-11(offset)=9 this is the width of the page.

c. What is the effect on the page table if the physical memory space is reduced by half? Assume that the number of page entries and page size stay the same.

Ans: When the physical memory space is halved from 1 MB to 512 KB, the page table is not affected because the number of pages (32) and page size (2 KB) remain the same. However, the number of bits needed to represent a page frame decreases:

The original 1 MB memory required 20 bits to address each frame.

The reduced 512 KB memory requires 19 bits, as 2^19=512KB.

Thus, while the page table size remains unchanged, the physical frame representation shrinks to 19 bits due to the reduced physical memory size.

4. [0.5 marks] Explain, in detail, what a write operation must do. Consider the case of a file opened for APPEND, in a file system using a hierarchical directory structure.

Ans: When a file is opened in **APPEND** mode, the operating system finds the file by navigating the hierarchical directory structure and retrieves its metadata, such as size and location. The file pointer is set to the end, ensuring new data is added sequentially without overwriting existing content. If the new data exceeds the current file size, additional disk blocks are allocated, and the metadata is updated to reflect the changes. The data is written efficiently, often using buffer caching for performance. The file system ensures consistency by updating allocation structures and metadata while preventing conflicts through locking mechanisms for concurrent writes. Upon completion, the system verifies integrity and returns control to the process or reports an error if the operation fails.

5. File System
Given:

- Block size = 8KB.
- Pointer size = 4 bytes
- Direct pointers = 12.
- Total size using direct pointers = 12 x 8 KB = 96 KB
- Total size using indirect block: 8 KB/4=2048 blocks.
- Total size using single indirect pointer: 2048 x 8 KB=16,384 KB=16 MB
- **Number of pointers in a double indirect block** = 2048 pointers
-
- **Total size using a double indirect pointer** = 2048 x 16 MB
- = 32,768 MB
- =32 GB
- **Number of pointers in a double indirect block** = 2048 pointers
- **Total size using a double indirect pointer** = 2048 x 32 GB
- = 65,536 GB
- =64 TB
- **Converting all units to TB**
- 96 KB ≈ 0.000093 TB
- 16 MB ≈ 0.000015 TB
- 32 GB ≈ 0.031 TB
- Total maximum file size = 0.000093 + 0.000015 + 0.031 + 64
- = 64.031 TB

(b) Extending Maximum File Size:

To extend the maximum file size:

- Use multiple inodes linked together.
- Extend metadata structures to allow more levels of indirect pointers.