**COP4634 – Systems & Networks I**

**Fall 2017 – November 17, 2017**

**Project 3 – Large Arrays**

**Page Fault Problem**

**Ali Al-Senan**

**Benjamin Linam**

**Introduction**

We are given an array that is 20,480 rows by 4,096 columns and told to implement an LRU page replacement algorithm to evict pages from memory for two processes which access memory: *ReadRow* and *ReadColumn.* We are asked to determine the total number of page faults for each process. We are then asked to assume the following:

1. Each process is given 10 frames of virtual memory by the system.
2. The two-dimensional array is a global data element.
3. 2 of the 10 frames are used for code and stack together.
4. An LRU page replacement algorithm is applied to evict pages from memory.

**Diagram Analysis**

Since we have been given 10 frames of virtual memory and each frame is 4,096 bytes so it will contain exactly one row of data from the array.

|  |  |  |
| --- | --- | --- |
|  | Virtual Memory (After the first 8 Page Faults) | |
| *ReadRow* | *ReadColumn* |
| Frame 1 | Row 0 – Reads whole row | Row0 – Reads one element of row |
| Frame 2 | Row 1 – Reads whole row | Row1 - Reads one element of row |
| Frame 3 | Row 2 – Reads whole row | Row2 - Reads one element of row |
| Frame 4 | Row 3 – Reads whole row | Row3 - Reads one element of row |
| Frame 5 | Row 4 – Reads whole row | Row4 - Reads one element of row |
| Frame 6 | Row 5 – Reads whole row | Row5 - Reads one element of row |
| Frame 7 | Row 6 – Reads whole row | Row6 - Reads one element of row |
| Frame 8 | Row 7 – Reads whole row | Row7 - Reads one element of row |
| Frame 9 | Reserved for Code and Stack | Reserved for Code and Stack |
| Frame 10 |

The LRU algorithm (Least Recently Used) dictates that the least recently used frame is replaced by new data. For *ReadColumn*, each row is accessed sequentially, so a single frame will never stay in memory long enough to be available for access again. *ReadRow* will access a whole row of data so once it has read every element, it will never need access to that same frame of data.

|  |  |  |
| --- | --- | --- |
|  | Virtual Memory (After 16 page faults) | |
| *ReadRow* | *ReadColumn* |
| Frame 1 | Row 8 – Reads whole row | Row 8 – Reads one element of row |
| Frame 2 | Row 9 – Reads whole row | Row 9 - Reads one element of row |
| Frame 3 | Row 10 – Reads whole row | Row 10 - Reads one element of row |
| Frame 4 | Row 11 – Reads whole row | Row 11 - Reads one element of row |
| Frame 5 | Row 12 – Reads whole row | Row 12 - Reads one element of row |
| Frame 6 | Row 13 – Reads whole row | Row 13 - Reads one element of row |
| Frame 7 | Row 14 – Reads whole row | Row 14 - Reads one element of row |
| Frame 8 | Row 15 – Reads whole row | Row 15 - Reads one element of row |
| Frame 9 | Reserved for Code and Stack | Reserved for Code and Stack |
| Frame 10 |

For *ReadRow*, each row of the array will be stored and each element of the row will be accessed. Virtual Memory will be updated for the number of times that there are rows: 20,480 and since the program never accesses the same row twice, we will have 20,480 page faults.

For *ReadColumn*, each row of the array will be stored; however, only one element of any given row will be accessed sequentially. This means that Virtual Memory will be updated for the number of times that there are rows (20,480) times the number of columns (4,096). In total there will be 83,886,080 page faults.