Student Name: Himanshu Kumar

Student ID: 11910102

Email Address: Himanshuraj0012@gmail.com

GitHub Link: https://github.com/himanshuxx2/Ca3-os-

Code: Question No. 21

Subject: Operating System Project

1. Problem in terms of Operating System:

Virtual Memory

Virtual memory is a memory management technique that can be implemented using both hardware and software. As the name indicates, it adds virtual memory to available memory, so that your system will appear to have more memory than what actually exists. Virtual memory is a layer of memory addresses (virtual addresses) that map to physical addresses. The memory addresses used by a program is the virtual addresses. These virtual address spaces and the assignment of real memory to the virtual memory are managed by the operating system.

Demand Paging

Demand paging is a type of swapping done in virtual memory systems. In demand paging, the data is not copied from the disk to the RAM until they are needed or being demanded by some program. The data will not be copied when the data is already available on the memory. This is otherwise called a lazy evaluation because only the demanded pages of memory are being swapped from the secondary storage (disk space) to the main memory. In contrast during pure swapping, all the memory for a process is swapped from secondary storage to main memory during the process startup.

Page Fault

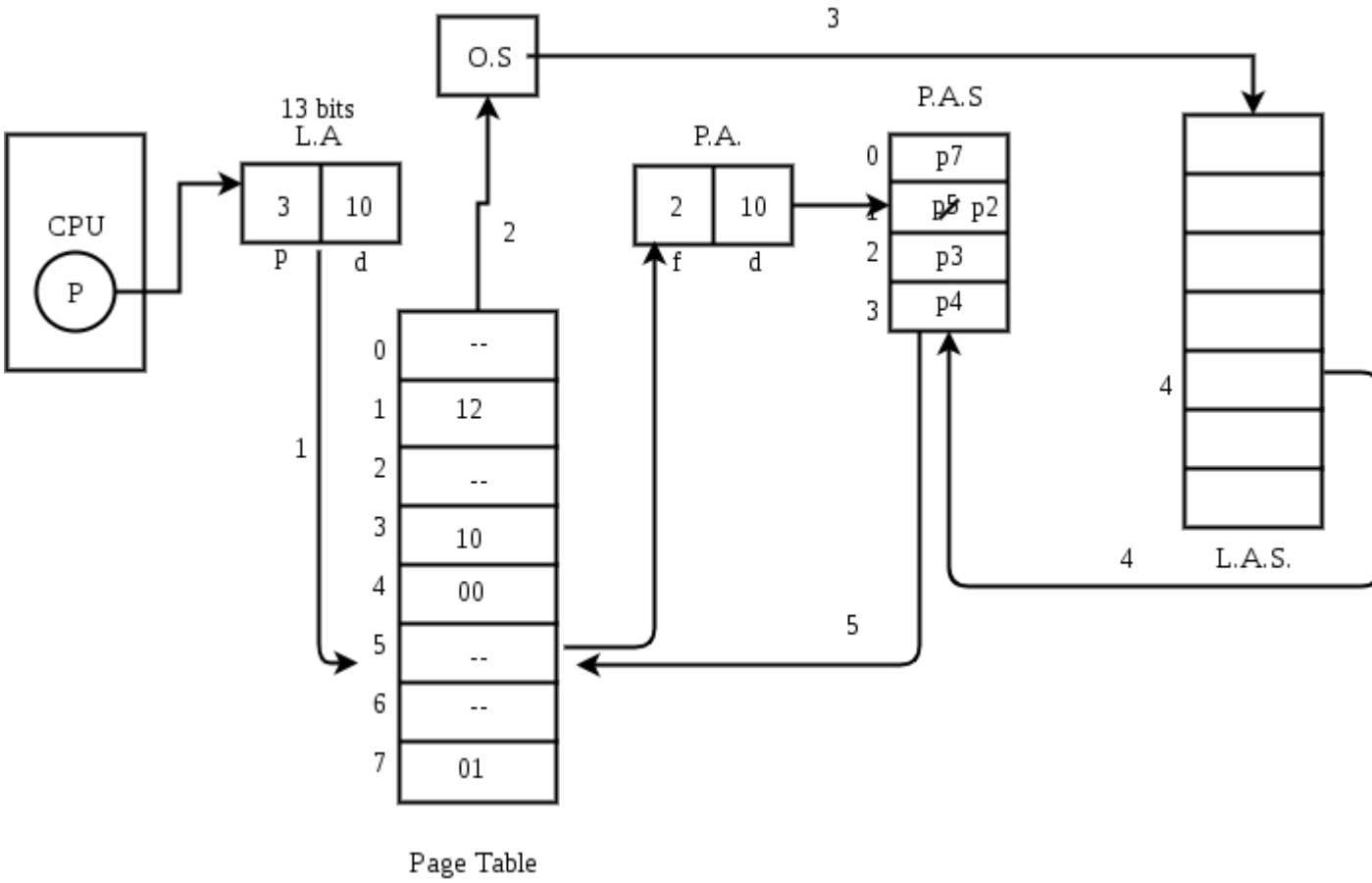
If the referred page is not present in the main memory then there will be a miss and the concept is called Page miss or page fault.The CPU has to access the missed page from the secondary memory. If the number of page fault is very high then the effective access time of the system will become very high.

2. Description:

Demand Page Working:

1.)If CPU try to refer a page that is currently not available in the main memory, it generates an interrupt indicating memory access fault.
2.)The OS puts the interrupted process in a blocking state. For the execution to proceed the OS must bring the required page into the memory.
3.)The OS will search for the required page in the logical address space.
4.)The required page will be brought from logical address space to physical address space. The page replacement algorithms are used for the decision making of replacing the page in physical address space.
5.)The page table will updated accordingly.The signal will be sent to the CPU to continue the program execution and it will place the process back into ready state.

\



4. Purpose of use:

According to the concept of Virtual Memory, in order to execute some process, only a part of the process needs to be present in the main memory which means that only a few pages will only be present in the main memory at any time.However, deciding, which pages need to be

kept in the main memory and which need to be kept in the secondary memory, is going to be difficult because we cannot say in advance that a process will require a particular page at particular time.Therefore, to overcome this problem, there is a concept called Demand Paging is introduced. It suggests keeping all pages of the frames in the secondary memory until they are required. In other words, it says that do not load any page in the main memory until it is required.Whenever any page is referred for the first time in the main memory, then that page will be found in the secondary memory.

5. Code snippet:

```
#include <stdio.h>
#include <stdlib.h>
double page_fault_rate();
void userInput(void);

double service_page_fault_empty;
double service_page_fault_modified;
double mem_access_time;
double times_page_modified;
double effective_access_time;
double pageFaultRate;
double service_page_fault_empty_ns;
double service_page_fault_modified_ns;
double times_page_modified_per;


void main(){

  userInput();
}
void userInput(){


  printf("\nEnter service Page Fault [not Modified]:");
  scanf("%lf",&service_page_fault_empty);
  printf("Enter Service Page Fault [Modified]:");
  scanf("%lf",&service_page_fault_modified);
  printf("Enter Memory Access Time[in nanoseconds]:");
  scanf("%lf",&mem_access_time);
  printf("Enter Percentage of time the page to be replaced:");
  scanf("%lf",&times_page_modified);
  printf("Enter Effective Access time[0-200]:");
  scanf("%lf",&effective_access_time);

  service_page_fault_empty_ns = (service_page_fault_empty*1000000);
  service_page_fault_modified_ns = (service_page_fault_modified*1000000);
  times_page_modified_per = (times_page_modified/100);
  printf("\n***Page Fault rate calculated\n***");
    pageFaultRate =
page_fault_rate(service_page_fault_empty_ns,service_page_fault_modified_ns,mem_access_time,times
_page_modified_per,effective_access_time);
  printf("\nMaximum Acceptable Page Fault rate = %.2e[exponential notation]",pageFaultRate);


}

double page_fault_rate(double servicePageFaultEmpty,double servicePageFaultMod,double
memAccess,double timesPages,double effAccess){
```

```
  double assume,serve;
  double numErator,denOminator;
  double pageFault;
   assume = (1- timesPages)*servicePageFaultEmpty;
   serve = timesPages*servicePageFaultMod;
   numErator = effAccess - memAccess;
     denOminator = (assume+serve);

  pageFault = numErator/denOminator;
  return pageFault;


}
```

Output:

Enter service Page Fault [not Modified]:20
Enter Service Page Fault [Modified]:8
Enter Memory Access Time[in nanoseconds]:20
Enter Percentage of time the page to be replaced:25
Enter Effective Access time[0-200]:200
*****Page Fault rate calculated*****
Maximum Acceptable Page Fault rate = 1.06e-05[exponential notation]

6. Description (Example):

EAT = (1-p)*(100) + (p) * (**100**+(1-0.7) * (8msec) + (0.7)*(20msec))
= 100 -100p + 100p + (2.4e6) *p + (14e6)*p

200 = 100 + (16.4e6)*p
p = 100/16.4e6

Time complexity = O(1).

I have made 5 revisions of solution on GitHub.

GitHub Link: https://github.com/himanshuxx2/Ca3-os-