19th May 2021

EOPSY LAB 4

Krzysztof Piotrowski

Introduction

Our Task is to map any 8 pages of physical memory to the first 8 pages of virtual memory and reads form one virtual memory address on each of the virtual pages. For this we will use Memory Management simulator. Simulator consists of 64 virtual pages and 32 physical pages. Each page is defined as 16384B by default.

Configuration

Two files need to be configured: memory.conf and commands. Here are its content defined for our task:

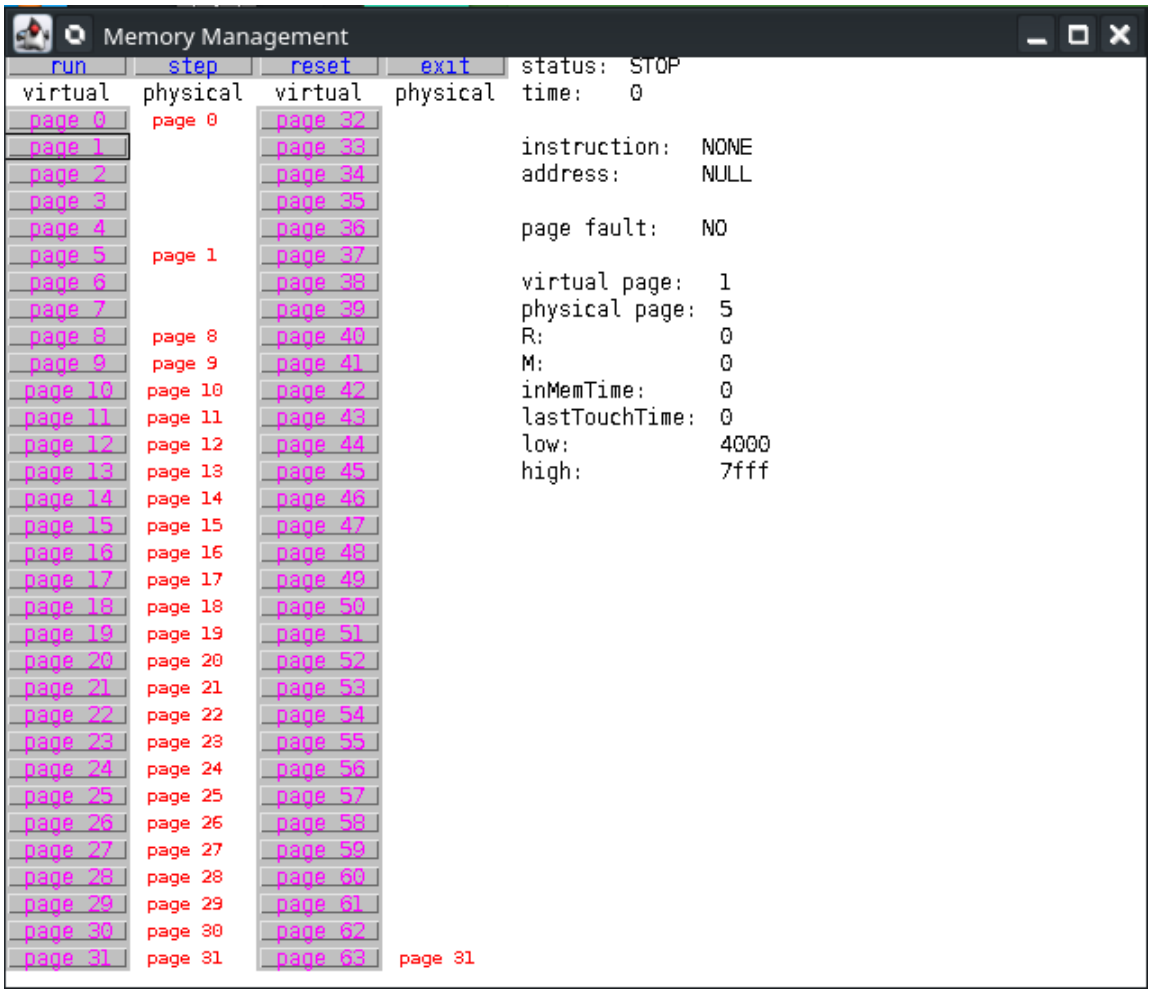
memset 0 0 0 0 0 0	READ hex 00000
memset 1 5 0 0 0 0	READ hex 04000
memset 2 10 0 0 0 0	READ hex 08000
memset 3 15 0 0 0 0	READ hex 0C000
memset 4 20 0 0 0 0	READ hex 10000
memset 5 25 0 0 0 0	READ hex 14000
memset 6 30 0 0 0 0	.
memset 7 31 0 0 0 0	.
	.
enable_logging true	READ hex 40000
log_file tracefile	READ hex 44000
pagesize 16384	READ hex 48000
addressradix 16	READ hex 4C000
numpages 64	READ hex 50000
	READ hex 54000
	READ hex 58000
	READ hex 5C000
	READ hex F4000
	READ hex F8000
	READ hex FC000

As it can be seen above I randomly assign physical memory pages to virtual memory pages with the use of command memset, where first column stand for virtual page and second for physical page. Rest of columns does not matter in our exercise.

On the right size I defined commands for reading each of 64 virtual memory pages. Values go from 0 to FC000 (hex) and change by 4000 (hex) as it is size of our default page 16384 B in hexadecimal form.

Simulation

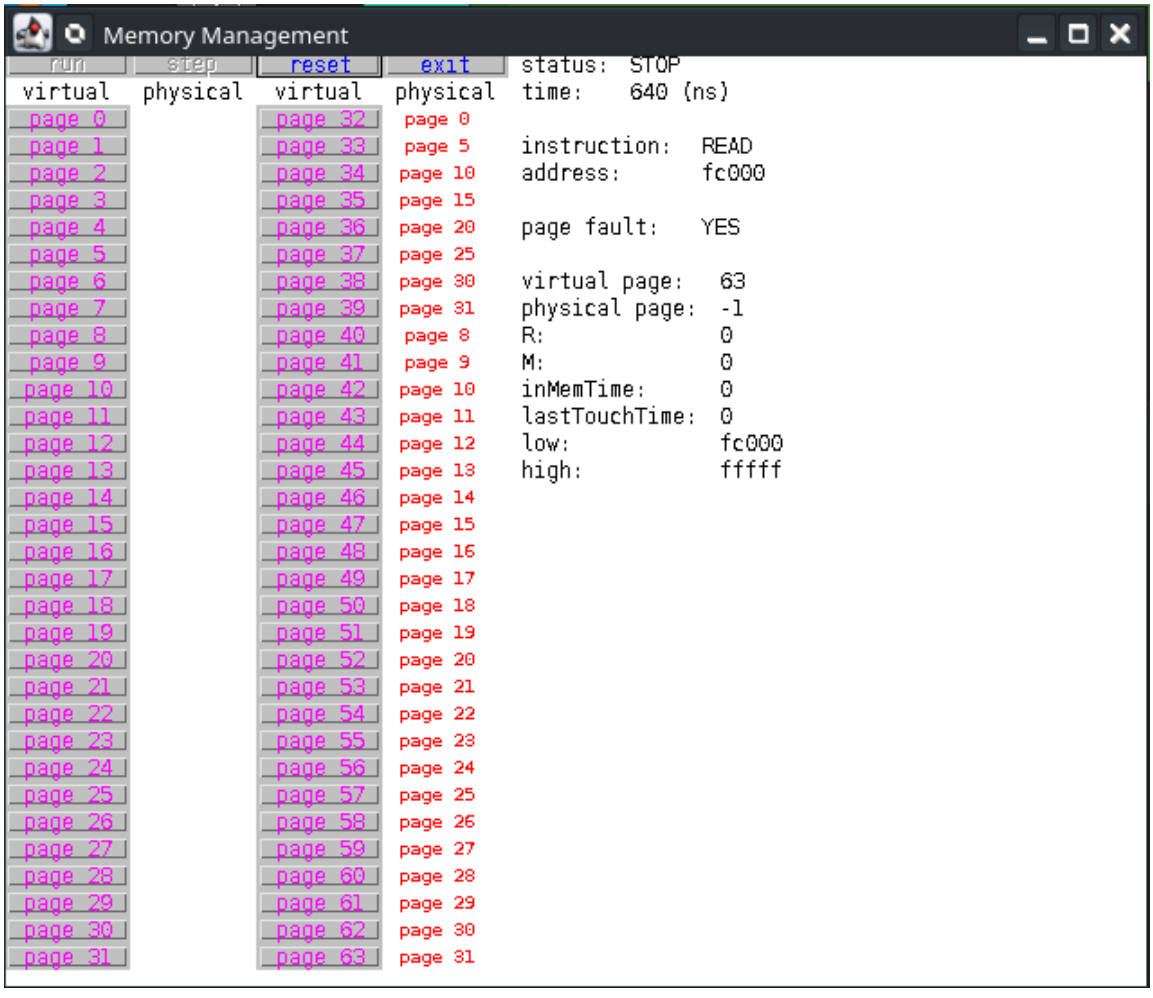
When program was ran, the following view was seen:



At first glance It looks like there is some trouble with our page assigment. However, when we look deeply into virtual description we can see assigned physical page (In that case 1st virtual page has assigned 5th physcial page).

Before starting simulation I can already predict that „page fault” will occur after 31st page. We have 64 virtual pages and 32 physical pages, When program will try to reference virtual page without physical page mapping, the page replacement algorithm will start its work.

READ 0 ... okay
READ 4000 ... okay
READ 8000 ... okay
READ c000 ... okay
.
.
READ 70000 ... okay
READ 74000 ... okay
READ 78000 ... okay
READ 7c000 ... okay
READ 80000 ... page fault
READ 84000 ... page fault
READ 88000 ... page fault
READ 8c000 ... page fault
.
.
READ f0000 ... page fault
READ f4000 ... page fault
READ f8000 ... page fault
READ fc000 ... page fault



As it can be seen above we see from which page we got page fault (from tracefile) and the way algorithm assign physical pages (program screen). According to my prediction page fault begins from 80000 (hex) address which stands for 32 page and goes up to the last virtual page. Starting from page 32 and going in order, algorithm assign following physical pages: 0, 5, 10, 15 etc. These are pages that we assign to first 8 virtual pages. Thanks to that infromation we know if there is need for page mapping, algorithm assign physical pages from the oldest virtual page. It means that page replacement algorithm base on **First in First Out** algorithm. System is tracking every page and when there is need for page replacement it takes first one from the queue which in our case is the oldest one.