Lab 4: Memory Management

Objectives

- Gain a practical understanding of Memory Management
- Practice virtual to physical memory addressing
- Explore tradeoffs with varying parameters for linear page tables

You may complete this lab individually or with one colleague from this class using pair programming rules (see the course Canvas page).

1 Paging and Address Spaces

A certain computer provides its users with a virtual address space of 2^{32} bytes. The computer has 2^{22} bytes of physical memory. **(1) How many physical page frames does this computer have in physical memory?** The virtual memory is implemented by paging with virtual addresses formatted as follows: the left 12 bits are used for the Virtual Page Number and the right 20 bits are used for the OFFSET. **(2) How many virtual pages can a virtual address space in this system have? How large is each page (express your answer in kB, MB, or GB, whichever is most appropriate)?** The system doesn't use TLB. A user process generates the virtual address 0x0012D687. **(3) Write a detailed explanation for how the system establishes the corresponding physical location.**

2 Address Translation

The page table shown in the table below is for a system with 16-bit virtual and physical addresses and with 4,096-byte pages.

Physical Frame	Valid Bit
9	1
1	1
14	1
10	1
-	0
13	1
8	1
15	1
-	0
0	1
5	1
4	1
-	0
-	0

3	1
2	1

(4) Convert the following virtual addresses (in hexadecimal) to the equivalent physical address (in hexadecimal).

- 0xE12C
- 0x3A9D
- 0xA9D9
- 0x7001
- 0xACA1

(5) Using the above addresses as a guide, provide an example of a logical address (in hexadecimal) that results in a page fault.

3 Page Table Simulation

In this lab, you will use a simple program, which is known as `paging-linear-translate.py`, to understand how simple virtual-to-physical address translation works with linear page tables. To run the program, remember to either type just the name of the program (`./paging-linear-translate.py`) or possibly this (`python paging-linear-translate.py`). Answer the questions in the text.

- Let's use the simulator to study how linear page tables change size given different parameters. Compute the size of linear page tables as different parameters change. Some suggested inputs are below; by using the -v flag, you can see how many page-table entries are filled. First, to understand how linear page table size changes as the address space grows:

  ```
  paging-linear-translate.py -P 1k -a 1m -p 512m -v -n 0
  paging-linear-translate.py -P 1k -a 2m -p 512m -v -n 0
  paging-linear-translate.py -P 1k -a 4m -p 512m -v -n 0
  ```

  Then, to understand how linear page table size changes as page size grows:

  ```
  paging-linear-translate.py -P 1k -a 1m -p 512m -v -n 0
  paging-linear-translate.py -P 2k -a 1m -p 512m -v -n 0
  paging-linear-translate.py -P 4k -a 1m -p 512m -v -n 0
  ```

  Before running any of these, try to think about the expected trends. **(6) How should page-table size change as the address space grows? (7) As the page size grows? (8) Why**

Lab 4: Memory Management

shouldn't we just use really big pages in general?

- Now let's do some translations. Start with some small examples and change the number of pages that are allocated to the address space with the -u flag. For example:

```
paging-linear-translate.py -P 1k -a 16k -p 32k -v -u 0
paging-linear-translate.py -P 1k -a 16k -p 32k -v -u 25
paging-linear-translate.py -P 1k -a 16k -p 32k -v -u 50
paging-linear-translate.py -P 1k -a 16k -p 32k -v -u 75
paging-linear-translate.py -P 1k -a 16k -p 32k -v -u 100
```

(9) What happens as you increase the percentage of pages that are allocated in each address space?

- Now let's try some different random seeds and some different (and sometimes quite crazy) address-space parameters for variety:

```
paging-linear-translate.py -P 8 -a 32 -p 1024 -v -s 1
paging-linear-translate.py -P 8k -a 32k -p 1m -v -s 2
paging-linear-translate.py -P 1m -a 256m -p 512m -v -s 3
```

(10) Which of these parameter combinations are unrealistic? Why?

- Use the program to try out some other problems. (11) Can you find the limits of where the program doesn't work anymore? For example, what happens if the address-space size is bigger than physical memory?

Submission

Submit `handin.txt`, with your 11 answers to the questions above to Canvas.