Northeastern University
College of Engineering
Department of Electrical & Computer Engineering

EECE7376: Operating Systems: Interface and Implementation

Homework 3

Instructions

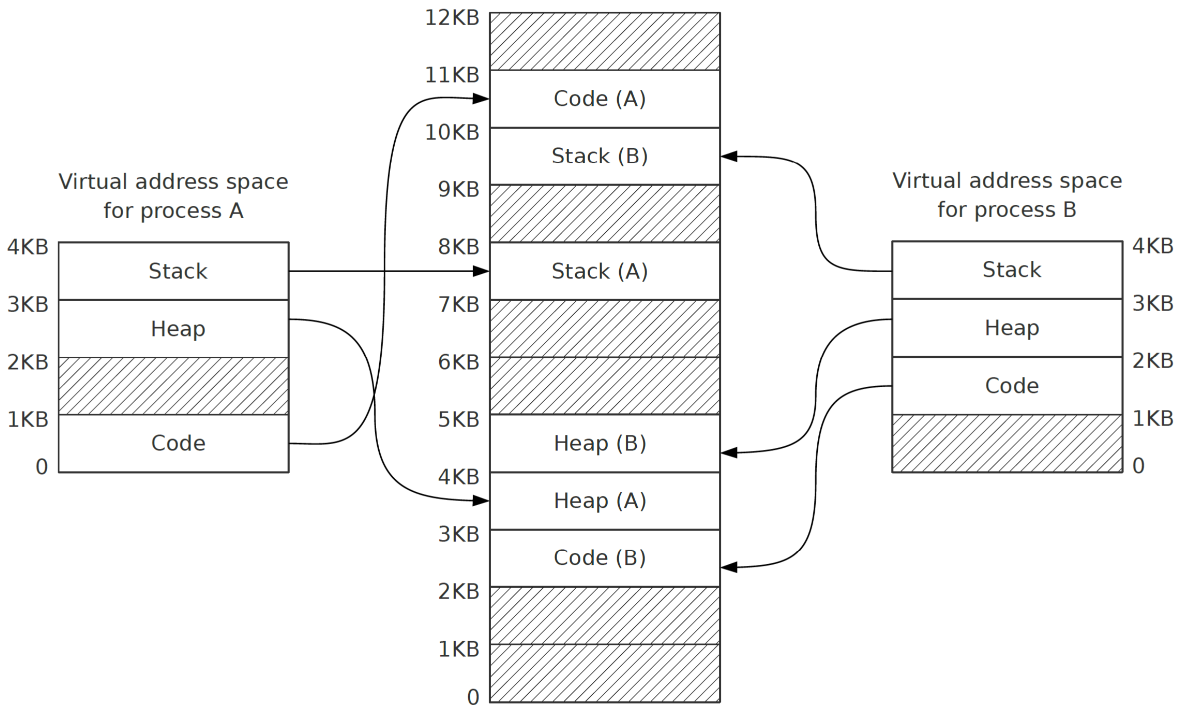
- For programming Problems:
    1. Your code must be well commented by explaining what the lines of your program do. Have at least one comment for every 4 lines of code.
    2. At the beginning of your source code files write your full name, students ID, and any special compiling/running instruction (if any).
    3. Before submitting the source code file(s), your code must compile and tested with a standard GCC compiler (available in the CoE Linux server).
    4. Do not submit any compiled object or executable files.
- Submit the following to the homework assignment page on Canvas:
    1. Your homework report developed by a word processor (e.g., Microsoft Word) and submitted as one PDF file. The report cover page must include your full name, student ID, course/section/semester information. For answers that require drawing and if it is difficult on you to use a drawing application, which is preferred, you can neatly hand draw "only" these diagrams, scan them, and insert the scanned images in your report document. The report includes the following (depending on the assignment contents):
        a. Answers to the non-programming Problems that show all the details of the steps you follow to reach these answers.
        b. A summary of your approach to solve the programming Problems.
        c. The screen shots of the sample run(s) of your program(s)
    2. Your well-commented programs source code files (i.e., the .cc or .cpp files).

Do NOT submit any files (e.g., the PDF report file and the source code files) as a compressed (zipped) package. Rather, upload each file individually.

Note: You can submit multiple attempts for this homework, however, only what you submit in the last attempt will be graded. This means all required files must be included in this last submission attempt.

Problem 1 (25 Points)

In a memory segmentation operating system, consider a machine with a 12KB physical memory space running an OS that uses a 4KB virtual memory space for its processes, where a 1KB means 1024 bytes. The machine is executing two processes (***A*** and ***B***), where each process has 3 allocated segments with the following mappings:



a) Show the state of a basic segment table containing the information represented in the diagram above. The table should include the following columns (write all numbers in decimal):
   - *Segment*. Each of the above segments are representing by an index from 0 to 5 where 0 is for ***A***'s Code segment and 5 for ***B***'s Stack segment (i.e., indices are sorted by process (first ***A***, then ***B***), and then by base virtual address (lower virtual address first)).
   - *Process*. Process identifier (***A*** or ***B***).
   - *Virtual address*. Base address in virtual address space.
   - *Size*. The segment size in KB.
   - *Physical address*. Base address in physical address space.

b) Assume the following virtual memory access addresses in decimal:

    i. Process A, address 2148
    ii. Process A, address 1324
    iii. Process A, address 4095
    iv. Process B, address 512
    v. Process B, address 2048
    vi. Process B, address 3272

   Which of the above are valid virtual memory access addresses? and for those valid accesses, find the following:

   - The corresponding *Segment Index* (from the table you formed above).
   - The *Offset* within the segment in decimal.
   - The *Physical* address in decimal.

Problem 2 (25 Points)

Assume the following simple memory segmentation operating system:

- The OS supports two segments for a process:
  - Segment ***A*** that starts from the beginning of the address space and is positive growing for code and a heap, and
  - Segment ***B*** that starts from the end of the address space and is negative growing for a stack.
- The Virtual Address space size is 128 bytes.
- Physical memory size is 512 bytes.
- Segment register information:
  - Segment ***A*** base (grows positive) = 0
  - Segment ***A*** limit = 20
  - Segment ***B*** base (grows negative) = 511
  - Segment ***B*** limit = 20

Which of the following are valid virtual memory access addresses? and for those valid accesses, what are the corresponding physical addresses? Explain your answers.

a) 29
b) 123
c) 16
d) 90
e) 10

All numbers in this problem are decimal numbers.

Problem 3 (25 Points)

Consider a system using paging as the memory virtualization technique, with 256-byte pages, virtual memory spaces formed of 8 pages, a physical memory space formed of 4 frames, and the following content for the page table of process ***A***:

Page	**Frame**
0	1
2	0
5	3

a) Draw a diagram representing the above page table for process ***A***. Draw the virtual address space on the left, and the physical on the right. Assign numbers to both virtual pages and physical frames and draw an arrow connecting each page with its associated frame.
b) How many bits are used to represent the Virtual Address? How many for a Physical Address? How many for the page offset? Explain your answer.
c) Translate the following decimal numbers, which represent virtual addresses, into physical addresses (if such addresses exist): 418, 0, 581, 460. Show all the translation steps.

Problem 4 (25 Points)

A page-based virtual memory is using **a linear page table** where its:

- Virtual address space size = 128 bytes
- Physical memory size = 1024 bytes
- Page size = 16 bytes

The leftmost bit in each PTE is the VALID bit. If this bit is 1, the rest of the entry is the PFN. If the bit is 0, the page is not valid. Assume the shown contents of the page table (in **hexadecimal**).

VPN	PTE
0	`0x80000034`
1	`0x00000000`
2	`0x00000000`
3	`0x00000000`
4	`0x8000001e`
5	`0x80000017`
6	`0x80000011`
7	`0x8000002e`

a) In front of each of the following virtual addresses, write its corresponding physical address as a **3-digit hexadecimal** number in the format 0xabc. For an invalid virtual address, write FAULT for its physical address.

Virtual address	Physical Address
`0x34`	
`0x44`	
`0x57`	
`0x18`	
`0x06`	

b) Explain how a **multi-level page table** (by utilizing a page directory) helps in reducing the memory needed compared to a **linear page table**.