Take Home Final Examination

Assigned: December 10, 2020 Due: 11:59pm December 16, 2020

Instructions:

1. Fill in your name and answers and leave the file in your cs444/final\_exam/ folder.
2. The work on this examination is your own and you are expected to adhere to the UMass-Boston honor system.
3. I give partial credit. Show all work.

1. (3 points) In hw3, how does the aswswtch function start the new process with the correct eflags?  
     
   it pushes the eflag using pushfl then we put in the new entry

And popfl to restore the new flags

1. (3 points) ) Linux uses a 3-level page table. Why is it better than a 2-level one?  
     
     
   3-level page table works once we get larger VA spaces. We then begin to partition the whole VA space into 3 levels.

This extra level is another way of having a faster

Lookup   
  
(3 points) Any shortcomings?  
  
Though it can provide a faster lookup in some cases it, it has to compare and make a memory reference which could make it slower

1. (3 points) What is the difference between swapping and paging?

Swapping is simple strategy in which it puts the process entirely in the hard disk if it’s not being used. While Paging puts the memory segment or page within the hard disk. This allows the ability for flexibility of VA is in the physical address

1. (3 points)What is the major application of non-preemptive schedulers?

Processes don’t have a set amount of time to work on something, this works we

1. (6 points) Name 2 approaches to fix a software deadlock situation?

1.\_\_\_\_\_\_\_\_\_Killing Process\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
 2.\_\_\_\_\_\_\_\_\_\_Rollback\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. (9 points) In hw4, after a page fault has occurred, the register contents are shown as follows:

EAX=00057565 EBX=00070a70 EBP=00070a48

EDX=00053120 ECX=00064a10 ESP=00070a48

ESI=00070a70 EDI=00000007 EIP=00060eda

EFLAGS=10206 (IF=1 SF=0 ZF=0 CF=0 OF=0)

CS=0010 DS=0018 SS=0018 ES=0018 FS= 0018 GS=0018

CR0=80050033 CR2=c0053120 CR3=51000

IDTR=c0056060 (limit 7ff)

GDTR=c0056870 (limit 4f)

i) Where did the page fault occur?   
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_0x0053120\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
 ii) Did it occur during an interrupt service routine?   
  
\_\_\_\_\_\_\_\_\_\_YES IF=1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

iii) Where were the Page Directory Elements?

\_\_\_\_\_\_\_\_0x51000\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**7. (20 points) Banker’s Algorithm**.

The total number of available resources A, B, C, and D are 3, 5, 7, and 7, respectively. Consider the snapshot of the system in Table 1.

**Current Allocation Maximum Required**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **A** | **B** | **C** | **D** |  |  | **A** | **B** | **C** | **D** |
| P1 | 1 | 1 | 1 | 2 |  | P1 | 3 | 2 | 2 | 3 |
| P2 | 1 | 0 | 1 | 2 |  | P2 | 1 | 1 | 2 | 3 |
| P3 | 0 | 1 | 2 | 1 |  | p3 | 2 | 2 | 4 | 2 |
| P4 | 1 | 2 | 2 | 1 |  | p4 | 2 | 3 | 3 | 2 |

Answer the following questions using the Banker’s algorithm:

1. (5 points) How many A, B, C, and D are still available?

0 left in A, 1 left in B, 1 left in C, 1 left in D

1. (5 points) What is the content of the Remaining Needs matrix?

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **A** | **B** | **C** | **D** |  |
| P1 | 2 | 1 | 1 | 1 |  |
| P2 | 0 | 1 | 1 | 1 |  |
| P3 | 2 | 1 | 2 | 1 |  |
| P4 | 1 | 1 | 1 | 1 |  |

1. (5 points) Does the snapshot in Table 1 represent a safe state? Show your reasoning.

There is (0,1,1,1) free space left, and this would satisfy the need from P2,

After P2 is done it frees up (1,1,2,3). After that it the only thing that’s avail next would be P4, and there would be (0,0,1,2) space left. After P4 is done it frees up (2,3,4,4) which cumulates with the existing space making it. Then it chooses P3 which takes (2,1,2,1) – (2,3,4,4) free space = (0,2,2,3) then it frees the space giving back (0,2,2,3) + (2,2,4,2) = (2,4,6,5). Then going to p4 and eventually freeing (3,5,7,7)

1. (5 points) If a request for process P1 is to add (0, 1, 1, 0) to its Current Allocation, can this request be granted immediately? Show your reasoning.

No, because once we give it’s request the next state will be unsafe state. Due to the fact that if it were to request another request, it would have nothing to free.

**8. (20 points) Memory Management**

PTEs for a process on an x86 system with trivial segments (VA=LA) look like this:

virtual page number(vpn) page table entry (PTE)

0 0x00245005

1 0x00982005

2 0x00832007

3 0x10403000

1. (15 points) When running the following program, write down the PTEs after each program step. Assume that the jmp instruction does not cause page faults, but decoding and/or execution of an instruction at a certain PC location may cause page faults. #n denotes literal value n. R1 is a register and (R1) means the content of R1 if R1 contains an address.

Address Instruction

0x0000 0050 jmp 0x00001050 /\* jump to address 0x1050 \*/

Page fault occurred(Y/N): \_\_\_\_n\_\_\_

PTE0: \_\_\_0x00245025\_\_\_\_

PTE1: \_\_\_0x00982025\_\_\_\_\_\_\_

…..

0x0000 1050 movl $0x00002050, R1 /\* set register R1= 0x2050\*/

Page fault occurred(Y/N): \_\_\_N\_\_\_\_

PTE0: \_\_\_\_\_\_0x00245025\_

PTE1: \_\_\_\_\_\_0x00982025\_\_\_\_\_\_\_\_

PTE2: \_\_\_\_\_\_0x83200027\_\_\_\_\_

0x 0000 1054 jmp 0x00002054 /\* jump to address 0x2054

….

0x 0000 2050 .byte 0 /\* data memory address 0x2050 =0 \*/

0x 0000 2054 movb $1, (R1) /\* move 1 to the address pointed to by R1 \*/

Page fault occurred(Y/N): \_\_\_\_N\_\_\_

PTE0: \_\_\_\_\_\_0x00245025\_\_\_\_\_\_\_\_\_\_\_\_

PTE1: \_\_\_\_\_\_0x00982025\_\_\_\_\_\_\_\_\_\_\_\_

PTE2: \_\_\_\_\_\_0x83200067\_\_\_\_\_\_\_\_\_\_\_\_

0x 0000 2058 jmp 0x00003054 /\* jump to address 0x3054\*/

Page fault occurred(Y/N): \_\_\_N\_\_\_\_

PTE0: \_\_\_\_\_\_\_0x00245025\_\_\_\_\_\_\_

PTE1: \_\_\_\_0x00982025\_\_\_\_\_\_\_\_\_

…… PTE2: \_\_\_\_0x83200067\_\_\_\_\_\_\_\_\_\_\_\_\_\_

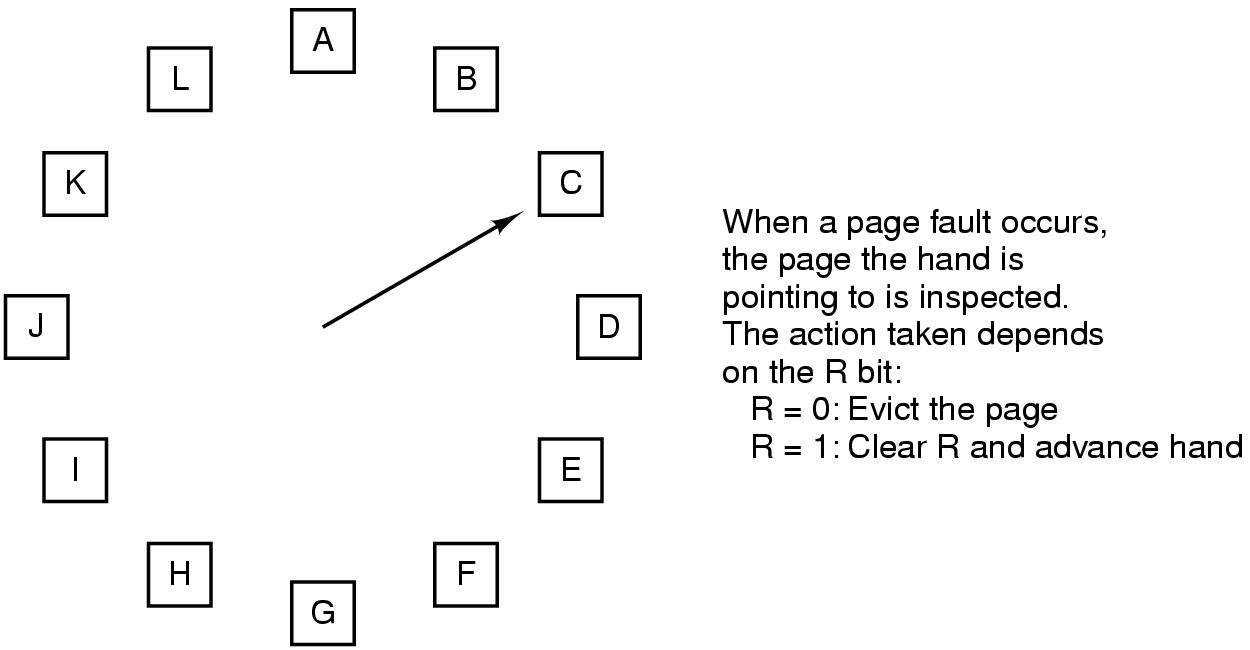
0x 0000 3054 movl $0x1234, R1

b. (5 points) Describe how paging works when the instruction at 0x0000 3054 is executed.

It should cause a page fault because the CPU will have to lookup the memory address 3054 where the opcode is located. By doing that the VA 3054 needs to be valid, and since that bit is 0 a page fault will occur and it will have to make the page available.

**9. (30 points) One-hand Clock Page Replacement Algorithm**

The operating system page daemon wakes up every 250ms and uses a clock algorithm to look for pages that are candidates for reclaiming when memory is low. Write a C program clock\_replacement (…) to implement the clock page replacement algorithm. This program will be called by the timer ISR.



pte\_ptr

int pte[12] your reader’s attention with a great quote from the document or use this space to emphasize a key point. To place this text box anywhere on the page, just drag it.]

pte#111 your reader’s attention with a great quote from the document or use this space to emphasize a key point. To place this text box anywhere on the page, just drag it.]

pte#3 ptepte#3 your reader’s attention with a great quote from the document or use this space to emphasize a key point. To place this text box anywhere on the page, just drag it.]

pte#2 your reader’s attention with a great quote from the document or use this space to emphasize a key point. To place this text box anywhere on the page, just drag it.]

pte#0 your reader’s attention with a great quote from the document or use this space to emphasize a key point. To place this text box anywhere on the page, just drag it.]

Use a pointer to inspect the page table entries. If the R-bit (same as the A-bit in Intel processor 0x20) is a 0, call the page\_reclaim( ..) function. If the A-bit is 1, change the bit to a 0. Increment the pointer and examine the next page table entry’s A-bit. Repeat the test until a page that can be evicted. This has to be done in a circular buffer fashion. Use the following definitions and declarations:

#define TABLE\_SIZE 12

int pte[TABLE\_SIZE];

int \*pte\_ptr = &pte[0];

int \* clock\_replacement( int \* ptr); /\* ptr is pointing to the present page being inspected \*/

/\* the return ptr will be pointing to the next page to be inspected \*/

void irq0inthandc(void)

{

…..

pte\_ptr = clock\_replacement(pte\_ptr);

..

}

Please Note: You do not need to compile the code. All data types have to be defined.

#DEFINE R\_BIT 0x40

int \* clock\_replacement(int \* ptr) {

static ptr\_index =0;

// 1. R bit = 0 page is evicted? page is reclaimed

// 2. R bit = 1 go to next page

while (!(ptr & R\_BIT == ptr))

//doo work

ptr = ptr ^ R\_BIT;

ptr\_index = (ptr\_index + 1 ) % TABLE\_SIZE

ptr = pte[ptr\_index];

page\_reclaim(checking\_ptr);

return checking\_ptr;