correct stmt: MMU

Select all the correct statements about MMU and it's functionality (on a non-demand paged system)

- ☐ MMU is inside the processor
- ☐ Logical to physical address translations in MMU are done in hardware, automatically
- ☐ The Operating system sets up relevant CPU registers to enable proper MMU translations
- ☐ Illegal memory access is detected in hardware by MMU and a trap is raised
- ☐ MMU is a separate chip outside the processor
- ☐ Logical to physical address translations in MMU are done with specific machine instructions
- ☐ The operating system interacts with MMU for every single address translation
- ☐ Illegal memory access is detected by operating system

fifo balady's anomaly

For the reference string
3 4 3 5 2

using FIFO replacement policy for pages,

consider the number of page faults for 2, 3 and 4 page frames.
Select the correct statement.

- ○ Do not exhibit Balady's anomaly
- ○ Exhibit Balady's anomaly between 2 and 3 frames
- ○ Exhibit Balady's anomaly between 3 and 4 frames

LRU balady's anomaly

For the reference string
3 4 3 5 2

using LRU replacement policy for pages,

consider the number of page faults for 2, 3 and 4 page frames.
Select the most correct statement.

○

This example does not exhibit Balady's anomaly

○

Exhibit Balady's anomaly between 2 and 3 frames

○

Exhibit Balady's anomaly between 3 and 4 frames

○

LRU will never exhibit Balady's anomaly

buddy allocated-1

Suppose a kernel uses a buddy allocator. The smallest chunk that can be allocated is of size 32 bytes. One bit is used to track each such chunk, where 1 means allocated and 0 means free. The chunk looks like this as of now:

10011010

Now, there is a request for a chunk of 50 bytes.

After this allocation, the bitmap, indicating the status of the buddy allocator will be

Answer

FIFO page faults

Consider the reference string
6 4 2 0 1 2 6 9 2 0 5

If the number of page frames is 3, then total number of page faults (including initial), using FIFO replacement is:

Answer

impossible-sequence-of-events-1

Mark whether the given sequence of events is possible or not-possible. Also, select the reason for your answer.

For each sequence it's a not-possible sequence if some important event is not mentioned in the sequence.

Assume that the kernel code is non-interruptible and uniprocessor system.

```
Process P1, user code executing
Timer interrupt
Context changes to kernel context
Generic interrupt handler runs
Generic interrupt handler calls Scheduler
Scheduler selects P2 for execution
After scheduler, Process P2 user code executing
```

This sequence of events is:{#1}

Because

{#2}

process state correct stmt

Select all the correct statements about process states.

Note that in this question you lose marks for every incorrect choice that you make, proportional to actual number of incorrect choices.

- [ ] Process state is stored in the PCB
- [ ] Process state can be implemented as just a number
- [ ] The scheduler can change state of a process from RUNNALBE to RUNNING
- [ ] A process becomes ZOMBIE when it calls exit()
- [ ] Process state is changed only by interrupt handlers
- [ ] Process state is stored in the processor
- [ ] Process state is implemented as a string
- [ ] The scheduler can change state of a process from RUNNALBE to RUNNING and vice-versa
- [ ] A process becomes ZOMBIE when another process bites into it's memory

process-state-change - 1

Mark statements True/False w.r.t. change of states of a process. Note that a statement is true only if the claim and argument both are true.

Reference: The process state diagram (and your understanding of how kernel code works). Note - the diagram does not show zombie state!

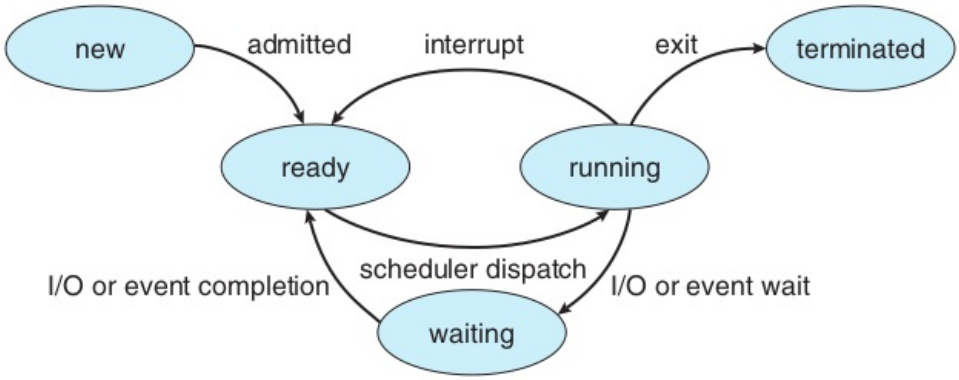

Figure 3.2 Diagram of process state.

correct stmt-thread models

Select all the correct statements w.r.t user and kernel threads

- ☐ many-one model can be implemented even if there are no kernel threads
- ☐ all three models, that is many-one, one-one, many-many , require a user level thread library
- ☐ one-one model increases kernel's scheduling load
- ☐ many-one model gives no speedup on multicore processors
- ☐ A process blocks in many-one model even if a single thread makes a blocking system call
- ☐ one-one model can be implemented even if there are no kernel threads
- ☐ A process may not block in many-one model, if a thread makes a blocking system call

Thread

If one thread opens a file with read privileges then

- ○ other threads in the another process can also read from that file

○

other threads in the same process can also read from that file

○

any other thread cannot read from that file

○

none of these

correct stmt - signal handling

Select all the correct statements about signals

☐

Signals are delivered to a process by kernel

☐

A signal handler can be invoked asynchronously or synchronously depending on signal type

☐

The signal handler code runs in user mode of CPU

☐

SIGKILL definitely kills a process because it can't be caught or ignored, and it's default action terminates the process

☐

Signals are delivered to a process by another process

☐

The signal handler code runs in kernel mode of CPU

☐

SIGKILL definitely kills a process because it's code runs in kernel mode of CPU

☐

Signal handlers once replaced can't be restored

functionality/use with function/variable in xv6 code.

Map the functionality/use with function/variable in xv6 code.

Setup kernel part of a page table, mapping kernel code, data, read-only data, I/O space, devices

Answer 1 kmap[]

return a free page, if available; 0, otherwise

Answer 2 kmap[]

Create page table entries for a given range of virtual and physical addresses; including page directory entries if needed

Answer 3 kmap[]

Return address of page table entry in a given page directory, for a given virtual address; creates page table if

necessary

Answer 4 kmap[]

Array listing the kernel memory mappings, to be used by setupkvm()

Answer 5 kmap[]

Setup kernel part of a page table, and switch to that page table

Answer 6 kmap[]

After VM implementation

After virtual memory is implemented

(select T/F for each of the following)One Program's size can be larger than physical memory size

code,segmentation status,xv6

For each function/code-point, select the status of segmentation setup in xv6

bootasm.S

Answer 1 gdt setup with 3 entries, at start32 symbol of bootasm.S

bootmain()

Answer 2 gdt setup with 3 entries, at start32 symbol of bootasm.S

entry.S

Answer 3 gdt setup with 3 entries, at start32 symbol of bootasm.S

kvmalloc() in main()

Answer 4 gdt setup with 3 entries, at start32 symbol of bootasm.S

after seginit() in main()

Answer 5 gdt setup with 3 entries, at start32 symbol of bootasm.S

after startothers() in main()

Answer 6 gdt setup with 3 entries, at start32 symbol of bootasm.S

xv6 memory management T/F

W.r.t. Memory management in xv6,

xv6 uses physical memory upto 224 MB onlyMark statements True or False

correct stmt: xv6 interrupt handler

Select the correct statements about interrupt handling in xv6 code

☐

All the 256 entries in the IDT are filled

☐

Each entry in IDT essentially gives the values of CS and EIP to be used in handling that interrupt

☐

xv6 uses the 64th entry in IDT for system calls

☐

On any interrupt/syscall/exception the control first jumps in vectors.S

☐

Before going to alltraps, the kernel stack contains upto 5 entries.

☐

The trapframe pointer in struct proc, points to a location on kernel stack

☐

The function trap() is the called irrespective of hardware interrupt/system-call/exception

☐

The CS and EIP are changed only after pushing user code's SS,ESP on stack

☐

xv6 uses the 0x64th entry in IDT for system calls

☐

On any interrupt/syscall/exception the control first jumps in trapasm.S

☐

The trapframe pointer in struct proc, points to a location on user stack

☐

The function trap() is the called only in case of hardware interrupt

☐

The CS and EIP are changed only immediately on a hardware interrupt

free pages VA range

The complete range of virtual addresses (after main() in main.c is over), from which the free pages used by kalloc() and kfree() is derived,are:

○

end, P2V(PHYSTOP)

○

end, PHYSTOP

○

P2V(end), P2V(PHYSTOP)

○

P2V(end), PHYSTOP

○

end, 4MB

○

end, (4MB + PHYSTOP)

○

end, P2V(4MB + PHYSTOP)

kalloc, kfree data structure

The data structure used in kalloc() and kfree() in xv6 is

○

Singly linked NULL terminated list

○

Singly linked circular list

○

Double linked NULL terminated list

○

Doubly linked circular list

Global/local arguments

Choice of the global or local replacement strategy is a subjective choice for kernel programmers. There are advantages and disadvantages on either side. Out of the following statements, that advocate either global or local replacement strategy, select those statements that have a logically CONSISTENT argument. (That is any statement that is logically correct about either global or local replacement)

Named vs unnamed pipe

Mark the statements about named and un-named pipes as True or False

T/F about mmap()

Mark the statements as True or False, w.r.t. mmap()

T/F argument passing xv6

Mark the statements as True or False, w.r.t. passing of arguments to system calls in xv6 code.

T/F multiple about Thrashing

Mark the statements as True or False, w.r.t. thrashing

xv6 process state changes

W.r.t. xv6 code, match the state of a process with a code that sets the state

EMBRYO
Answer 1 exit(), called by process itself
UNUSED
Answer 2 exit(), called by process itself
SLEEPING

Answer 3 exit(), called by process itself

RUNNABLE

Answer 4 exit(), called by process itself

RUNNING

Answer 5 exit(), called by process itself

ZOMBIE

Answer 6 exit(), called by process itself

xv6 VM functions

Match the description of a memory management function with the name of the function that provides it, in xv6

Create a copy of the page table of a process

Answer 1 switchuvm()

Mark the page as in-accessible

Answer 2 switchuvm()

setup the kernel part in the page table

Answer 3 switchuvm()

Load contents from ELF into existing pages

Answer 4 switchuvm()

Load contents from ELF into pages after allocating the pages first

Answer 5 switchuvm()

Copy the code pages of a process

Answer 6 switchuvm()

Setup and load the user page table for initcode process

Answer 7 switchuvm()

Switch to kernel page table

Answer 8 switchuvm()

Switch to user page table

Answer 9 switchuvm()

Major & Minor page faults

Select all correct statements w.r.t. Major and Minor page faults on Linux

- ☐ Minor page fault may occur because the page was a shared memory page
- ☐ Minor page fault may occur because of a page fault during fork(), on code of an already running process
- ☐ Minor page fault may occur because the page was freed, but still tagged and available in the free page list
- ☐ Major page faults are likely to occur in more numbers at the beginning of the process
- ☐ Thrashing is possible only due to major page faults
- ☐ Minor page faults are an improvement of the page buffering techniques

POSIX & SYSV SHM

Select the correct points of comparison between POSIX and System V shared memory.

- ☐ POSIX shared memory is newer than System V shared memory
- ☐ POSIX shared memory is "thread safe", System V is not
- ☐ POSIX allows giving name to shared memory, System V does not
- ☐ System V is more prevalent than POSIX even today

Use of IPC

Select the most common causes of use of IPC by processes

☐Sharing of information of common interest
☐Breaking up a large task into small tasks and speeding up computation, on multiple core machines
☐More modular code
☐Get the kernel performance statistics
☐More security checks

Second chance

Given below is a sequence of reference bits on pages before the second chance algorithm runs. Before the algorithm runs, the counter is at the page marked (x). Write the sequence of reference bits after the second chance algorithm has executed once. In the answer write PRECISELY one space BETWEEN each number and do not mention (x).

0 0 1(x) 1 0 1 1

Answer

order: creation of init process

Order the following events, in the creation of init() process in xv6:

T/F basics of paging

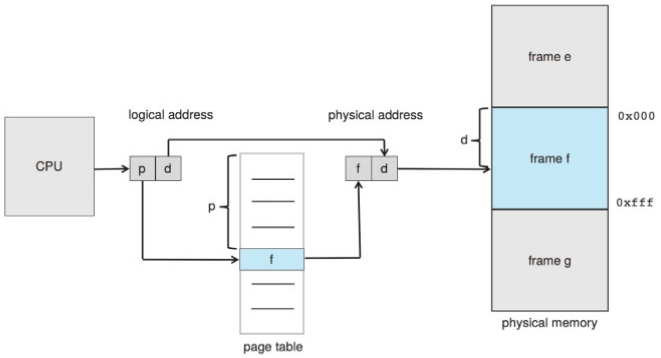

Figure 9.8 Paging hardware.

Mark the statements as True or False, w.r.t. the above diagram (note that the diagram does not cover all details of what actually happens!)

first-best-worst fit(2)

Given that a kernel has 1000 KB of total memory, and holes of sizes (in that order) 300 KB, 200 KB, 100 KB, 250 KB. For each of the requests on the left side, match it with the chunk chosen using the specified algorithm.

Consider each request as first request.

200 KB, first fit
Answer 1 300 KB
150 KB, first fit
Answer 2 300 KB
220 KB, best fit
Answer 3 300 KB
150 KB, best fit
Answer 4 300 KB
50 KB, worst fit
Answer 5 300 KB
100 KB, worst fit

Answer 6 300 KB

correct stmt: linking loading

Select all the correct statements about linking and loading.

- ☐ Continuous memory management schemes can support static linking and static loading. (may be inefficiently)
- ☐ Continuous memory management schemes can support static linking and dynamic loading. (may be inefficiently)
- ☐ Dynamic linking essentially results in relocatable code.
- ☐ Loader is part of the operating system
- ☐ Dynamic linking and loading is not possible without demand paging or demand segmentation.
- ☐ Loader is last stage of the linker program
- ☐ Static linking leads to non-relocatable code
- ☐ Continuous memory management schemes can support dynamic linking and dynamic loading.
- ☐ Dynamic linking is possible with continous memory management, but variable sized partitions only.

number of page table entries

Consider a computer system with a 32-bit logical address and 4- KB page size. The system supports up to 512 MB of physical memory. How many entries are there in each of the following?

Write answer as a decimal number.

A conventional, single-level page table: {#1}
An inverted page table: {#2}

increase cpu utilisation

Consider a demand-paging system with the following time-measured utilizations:
CPU utilization : 20%
Paging disk: 97.7%
Other I/O devices: 5%

For each of the following, indicate whether it will (or is likely to) improve CPU utilization (even if by a small amount). Explain your answers.

a. Install a faster CPU :{#1}
b. Install a bigger paging disk. :{#2}
c. Increase the degree of multiprogramming. :{#3}
d. Decrease the degree of multiprogramming. :{#4}
e. Install more main memory.:{#5}

f. Install a faster hard disk or multiple controllers with multiple hard disks. :{#6}
g. Add prepaging to the page-fetch algorithms. :
{#7}

h. Increase the page size. :{#8}

Submit