CSS-430 : Operating Systems : HW04-Questions

**Assignment Text**

Complete the following problems from the OSC book, 10th edition:

* Problem 5.2
* Problem 5.11
* Problem 6.7
* Problem 6.10
* Problem 6.17

5.2

Question: Explain the difference between preemptive and nonpreemptive scheduling.

5.11

Question: Of these two types of programs:

a. I/O-bound

b. CPU-bound

which is more likely to have voluntary context switches, and which is more likely to have nonvoluntary context switches? Explain your answer.

6.7

Question: The pseudocode of Figure 6.15 illustrates the basic push() and pop() operations of an array-based stack. Assuming that this algorithm could be used in a concurrent environment, answer the following questions:

1. What data have a race condition?
2. b. How could the race condition be fixed?

push(item) {

if (top < SIZE) {

stack[top] = item;

top++;

} else

ERROR

}

pop() {

if (!is empty()) {

top--;

return stack[top];

} else

ERROR

}

is.empty() {

if (top == 0)

return true;

else

return false;

}

6.10

Question: The compare and swap() instruction can be used to design lock-free data structures such as stacks, queues, and lists. The program example shown in Figure 6.17 presents a possible solution to a lock-free stack using CAS instructions, where the stack is represented as a linked list of Node elements with top representing the top of the stack. Is this implementation free from race conditions?

typedef struct node {

  value\_t data;

  struct node \*next;

} Node;

Node \*top; // top of stack

void push(value\_t item) {

  Node \*old\_node;

  Node \*new\_node;

  new\_node = malloc(sizeof(Node));

  new\_node->data = item;

  do {

   old\_node = top; new\_node->next = old\_node;

  } while (compare\_and\_swap(top, old\_node, new\_node) != old node);

}

value\_t pop() {

  Node \*old\_node;

  Node \*new\_node;

  do {

    old\_node = top;

    if (old\_node == NULL)

      return NULL;

    new\_node = old\_node->next;

  }

  while (compare\_and\_swap(top, old\_node, new\_node) != old\_node);

  return old\_node->data;

}

6.17

Question: Explain why interrupts are not appropriate for implementing synchronization primitives in multiprocessor systems.