CSCE 351 Operating System Kernels

Fall 2001
Steve Goddard

Homework 0, August 28
(Total of 100 points)

Evaluation of prerequisite knowledge

Due: 12:30pm September 4

The prerequisites for CSCE 351 are CSCE 230, CSCE 230L or CSCE 231, and CSCE 310. This purpose of this assignment is to evaluate your understanding of the prerequisite knowledge. You may consult any textbook or other resource to complete the assignment, but do the work on your own. This is not a group project.

CSCE 230 and 230L or 231

1) How is the decimal number 49 represented in

   A) Binary?

   B) Octal?

   C) Hexadecimal?

   D) Base 7?

   Try to do this one without a calculator :)

2) Why should the value of the program counter be updated *before* an instruction is executed? (*Hint*: What instructions change the value of the program counter as part of their execution? Would it make a difference if the program counter were updated *after* these instructions had been "performed?")

3) What addressing mode would you use if you wanted to implement a stack in assembly language? Why?

4) Suppose procedure A calls procedure B, and procedure B calls procedure C. Assume that register R5 is used for linking; that is, register R5 contains the address by which any called procedure returns to its calling procedure. What must procedure B do so that it can send a return address to C without losing the return address it needs to return to A? Give the assembly instructions by which B could do this.

5) Consider the following state of a MIPS R2000 machine:

```
Register 5 contains 1000
Register 6 contains 2000
Register 7 contains 3000
Memory location 1000 contains 2000
Memory location 2000 contains 3000
Memory location 3000 contains 1000
```

All numbers are in octal (base 8) notation.

After the following three R2000 assembly instructions are executed from the above initial state, what is the state of the machine? That is, show the contents of regisers 5, 6, and 7 as well as memory locations 1000, 2000, and 3000.

```
ADDI  $5, $6, 1000
LW    $7, 0($5)
LW    $5, 0($5)
ADDI  $6, $7, 1000
ADD   $7, $6, $5
```

Keep all numbers in octal (base 8) notation.

CSCE 310 (and 235 by transitivity)

Consider the following definitions for functions that operate on a list *L*.

A) End(*L*). Return the position following position *n* in an *n*-element list *L*.

B) INSERT(*x, p, L*). Insert *x* at position *p* in list *L*, moving elements at *p* and following position to the next higher position. That is, if L is $a_1, a_2, \ldots a_n$, then *L* becomes $a_1, a_2, \ldots, a_{p-1}, x, a_p, \ldots, a_n$. If *p* is END(*L*), then *L* becomes $a_1, a_2, \ldots, a_n, x$. If list *L* has no position *p*, the result is undefined.

C) RETRIEVE(*p,L*). This function returns the element at position *p* on list *L*. The result is undefined if *p* = END(*L*) or if *L* has no position *p*.

D) DELETE(*p,L*). Delete the element at position *p* of list *L*. If *L* is $a_1, a_2, \ldots a_n$, then *L* becomes $a_1, a_2, \ldots, a_{p-1}, a_{p+1}, \ldots, a_n$. The result is undefined if *p* = END(*L*) or if *L* has no position *p*.

E) NEXT(*p,L*) and PREVIOUS(*p,L*) return the positions following and preceding position *p* on list *L*. If *p* is the last position on *L*, then NEXT(*p,L*) = END(*L*). NEXT is undefined if p is END(*L*). PREVIOUS is undefined if *p* is 1. Both functions are undefined if L has no position *p*.

F) FIRST(*L*). This function returns the first position on list *L*. If *L* is empty the position return is END(L).

G) PRINTLIST(L). Print the elements of L in the order of occurrence.

6) Let list L be *y,z,b,c,d,f,z,b,c,a*. Element *y* is at position 1, and element *a* is at position 10. What is the output of the following statements:

```
INSERT(q,5,L);
INSERT(r,5,L);
INSERT(b,4,L);
DELETE(6,L);
PRINTLIST(L);
```

7) The following procedure was intended to remove all occurrences of element x from list L. Does it work? If not, explain why it doesn't always work and suggest a way to repair the procedure so it performs its intended task.

```
procedure delete( x: elmenttype; L: List);
      var
            p: position;
      begin
            p := FIRST(L);
            while p != END(L) do begin
                  if RETRIEVE(p,L) = x then
                        DELETE(p,L);
                  p := NEXT(p,L);
            end while
      end procedure; {delete}
```

Consider the following definitions for functions that operate on a stack *S*.

A) MAKENULL(S). Make stack *S* be an empty stack.

B) EMPTY(*S*). Return true if *S* is an empty stack; return false otherwise.

C) TOP(*S*). Return the element at the top of stack *S*. The element is *not* deleted from the stack.

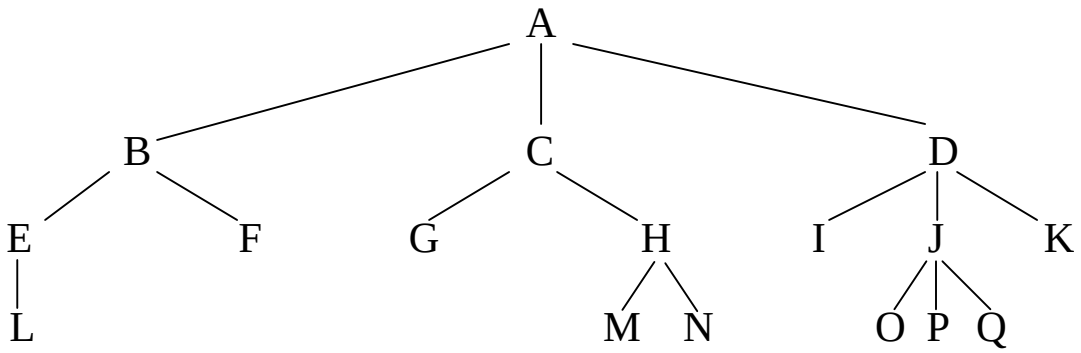
D) POP(*S*). Delete the top element of stack *S*.

E) PUSH(*x*, *S*). Insert *x* at the top of stack *S*. The old top element becomes next-to-top, and so on.

8) Show the stack created by the following commands

```
MAKENULL(S);
PUSH(p,S);
PUSH(u,S);
PUSH(s,S);
PUSH(h,S);
POP(S);
PUSH(TOP(S),S);
PUSH(s,S);
PUSH(TOP(S),S);
PUSH(u,S);
PUSH(p,S);
```

Consider the following tree:



9) Answer the following questions about the tree.

A) Which nodes are leaves?

B) Which node is the root?

C) What is the parent of node B?

D) Which nodes are children of B?

E) Which nodes are ancestors of G?

F) Which nodes are descendants of G?

G) What are the siblings of G and E?

H) Which nodes are to the left and to the right of J?

I) What is the depth of node M?

J) What is the height of node M?

10) List the nodes of the tree in

a) preorder,

b) postorder, and

c) inorder.