Part I. Operating Systems (50%)

(1) What is a semaphore? What are the classic definitions for P and V? Why can the mutual exclusion problems be solved by using 'semaphore'? In implementation, explain how to implement the semaphore using a slightly modified P/V definition in order to overcome the need of busy-waiting (which is required for software solutions to the mutual exclusion problems)? (15%)

(2) What is the general interrupt mechanism in interrupt-based systems? (namely, interrupt occurrence, interrupt handling, the general interrupt-driven structure of the system, etc.) (10%)

(3) Answer each of the following true/false questions. Give your explanations to the questions to which your answers are 'false'. (15%)

(a) In a system using virtual memory technique to manage its memory, if the space of its secondary storage is unlimited then the user logical address space can be unlimited.

(b) The busy-wait problem will not exist if we use high-level synchronization construct 'monitor' to solve the mutual exclusion problems.

(c) In resource allocation of a multipresocessing system, the deadlock may not occur even there exists a circular wait among processes.

(d) In UNIX file system, when a file is opened and read (or written), the updated current file position (offset) is stored in an incore i-node list such that this value can be accessed easily when next reading (writting) is required.

(4) Explain the concept of threads (lightweight processes) in terms of interprocess communication. (5%)

(5) What is an i-node in Unix system? What information does an i-node contain? (5%)

Part II. Assemblers and Compilers (50%)

(1) What is recursive-descent parsing? Consider the following BNF grammar

<S> ::= c <A> d
<A> ::= a <B>
<B> ::= b | f <B>

where S, A, B are nonterminal symbols and a, b, c, d, f are terminal symbols.

Based on this grammar, construct a recursive-descent parser. (15%)

(2) What is an activation record? When is it allocated and deallocated? In Pascal language, what information does it contain? (15%)

(3) Answer the following questions (true or false) and give your reasons to the questions to which your solutions are 'false'. (10%)

(a) For a Pascal program, type-checking is done both at compile-time and at run-time.

(b) Lex is a lexical analyzer and Yacc is an syntax analyzer.

(c) A compiler at code optimization phase cannot detect any types of errors in source code.

(4) What advantages can be obtained for a one-pass assembler? In implementation, what difficulties may be encountered (met)? What are the approaches to overcome these difficulties? (10%) (Assume the assembler will generate object code, the operands of the instructions in the assembly language are single symbols, and literals are not allowed in the language.)