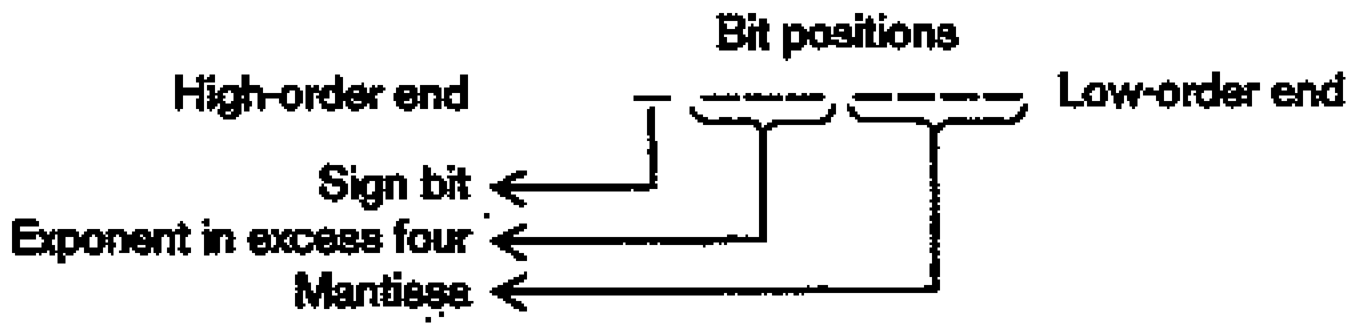
1. (7%) The three bit patterns 01101000, 10000010, and 00000010 are representations of the same value in two's complement, excess, and the eight-bit floating-point format described in the following, but not necessarily in that order. What is the common value, and which pattern is in which notation?




2. (8%) Using the above eight-bit floating-point format, what would be the result (in decimal) of computing the sum $\frac{1}{8}+\frac{1}{8}+\frac{1}{8}+2\frac{1}{2}$ from left to right? How about from right to left?

3. (15%) Given a machine language described in the following. Each machine instruction is two bytes long. The first four bits consist of the op-code; the last 12 bits make up the operand field. The table that follows lists the instructions in hexadecimal notation together with a short description of each. The letters R, S, and T are used in place of hexadecimal digits in those fields representing a register identifier that varies depending on the particular application of the instruction. The letters X and Y are used in lieu of hexadecimal digits in variable fields not representing a register.

Suppose the following program, written in the above-described machine language, is stored in main memory beginning at address 30 (hexadecimal). What task will the program perform when executed?

2003
2101
2200
2310
1400
3410
5221
5331
3239
333B
B248
B038
C000

| Op-code | Operand | Description |
|---|---|---|
| 1 | RXY | LOAD the register R with the bit pattern found in the memory cell whose address is XY. *Example:* 14A3 would cause the contents of the memory cell located at address A3 to be placed in register 4. |
| 2 | RXY | LOAD the register R with the bit pattern XY. *Example:* 20A3 would cause the value A3 to be placed in register 0. |
| 3 | RXY | STORE the bit pattern found in register R in the memory cell whose address is XY. *Example:* 35B1 would cause the contents of register 5 to be placed in the memory cell whose address is B1. |
| 4 | 0RS | MOVE the bit pattern found in register R to register S. *Example:* 40A4 would cause the contents of register A to be copied into register 4. |
| 5 | RST | ADD the bit patterns in registers S and T as though they were two's complement representations and leave the result in register R. *Example:* 5726 would cause the binary values in registers 2 and 6 to be added and the sum placed in register 7. |
| 6 | RST | ADD the bit patterns in registers S and T as though they represented values in floating-point notation and leave the floating-point result in register R. *Example:* 634E would cause the values in registers 4 and E to be added as floating-point values and the result to be placed in register 3. |
| 7 | RST | OR the bit patterns in registers S and T and place the result in register R. *Example:* 7CB4 would cause the result of ORing the contents of registers B and 4 to be placed in register C. |
| 8 | RST | AND the bit patterns in register S and T and place the result in register R. *Example:* 8045 would cause the result of ANDing the contents of registers 4 and 5 to be placed in register 0. |
| 9 | RST | EXCLUSIVE OR the bit patterns in registers S and T and place the result in register R. *Example:* 95F3 would cause the result of EXCLUSIVE ORing the contents of registers F and 3 to be placed in register 5. |
| A | R0X | ROTATE the bit pattern in register R one bit to the right X times. Each time place the bit that started at the low-order end at the high-order end. *Example:* A403 would cause the contents of register 4 to be rotated 3 bits to the right in a circular fashion. |
| B | RXY | JUMP to the instruction located in the memory cell at address XY if the bit pattern in register R is equal to the bit pattern in register number 0. Otherwise, continue with the normal sequence of execution. *Example:* B43C would first compare the contents of register 4 with the contents of register 0. If the two were equal, the execution sequence would be altered so that the next instruction executed would be the one located at memory address 3C. Otherwise, program execution would continue in its normal sequence. |
| C | 000 | HALT execution. *Example:* C000 would cause program execution to stop. |

4. (10%) 是非題
   - (a) DMA is the ability of CPU to access memory directly.
   - (b) In a real-time system, the dispatcher divides CPU time into time slices for multiple programs in the memory.
   - (c) C is an object-oriented programming language.
   - (d) Prolog is a declarative programming language.
   - (e) Relational model is the most popular database today.

5. (15%) 單選題
   - (a) Paging is (1) part of operation system (2) application software (3) utility software (4) shell program (5) part of I/O system.
   - (b) You can run a program larger than 64M on a 64M RAM computer because the operation system (1) has a loader (2) supports dynamic link library (3) supports virtual

memory (4) is on a local area network (5) has object-oriented kernel system.

(c) Which of the following statement is true?

(1) Both C and Pascal are based on regular grammar.

(2) Both C and Pascal are based on context-free grammar.

(3) C is based on context-free grammar and Pascal is based on regular grammar.

(4) C is based on regular grammar and Pascal is based on context-free grammar.

(5) None of the above.

(d) Machine languages are (1) functional (2) object-oriented (3) imperative (4) declarative (5) high level programming languages.

(e) SQL (Structured Query Language) (1) is a navigational language (2) specifies a sequence of raw relational operations (3) is a declarative language (4) is a low level programming language.

6. (4%) Describe the four parts of a typical URL:
http://mozart.aw.com/authors/Shakespeare/Romeo.html
(a) http (b) mozart.aw.com (c) authors/Shakespeare/ (d) Romeo.html

7. (6%) Describe how the bootstrap is executed and where the bootstrap is located.

8. (6 %) Judge the following statements and explain your selection.

(a) [Yes □ No □] The greedy algorithms for finding the shortest path between two nodes in a network require the lengths of all edges to be non-negative. (If true is chosen, answer (b); otherwise, answer (c).)

(b) [Yes □ No □] To meet the non-negative requirement, one can add some large enough value to each possible negative edge. This approach will help the greedy algorithms to find the shortest path between any two nodes.

(c) Using an example to illustrate how Dijkstra's or Floyd's algorithm works in finding the shortest path in presence of negative edges.

9. (8 %) Assume n is a power of two and $t(1)=1$, derive the closed form for

$$t(n) = 2 * t(n/2) + n * lg_2(n).$$

10. (11 %) Figure 1 shows a partially ordered binary tree. The value stored in each node is no less than those of its children nodes. This balanced tree can be implemented efficiently using an array structure called heap, as illustrated in Figure 2.

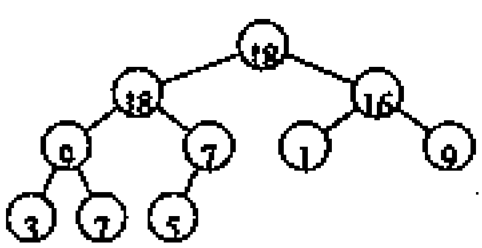



Figure 1: Partially ordered tree with 10 nodes.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| 19 | 18 | 16 | 9 | 7 | 1 | 9 | 3 | 7 | 5 |

Figure 2: Heap for Figure 1.

(a) (3 %) What is the visiting sequence of nodes when the tree in Figure 1 is pre-orderly traversed? What results will be under both in-order and post-order traversals?

(b) (2 %) Which positions could be occupied by the 3rd largest value in some heap of size 10?

(c) (2 %) Which positions could be occupied by the 3rd smallest value in some heap of size 10?

(d) (2 %) If we insert the values, 3, 1, 4, 1, 5, 8, 7, 2, 6, 5, serially into an empty heap, what would be the content of the heap (like Figure 2)?

(e) (2 %) Illustrate how the heap-sort algorithm sorts the heap shown in Figure 2.

11. (10 %) In common programming practices, recursion that enables a function or procedure to call itself can improve the modularity of the design. As an example, the Fibonacci numbers:

$$F_N = \begin{cases} 1 & \text{if } N = 0 \text{ or } 1, \\ F_{N-1} + F_{N-2} & \text{if } N \geq 2. \end{cases}$$

can be programmed as a recursive function, in addition to using loop iterations.

(a) (3 %) Write the function in some programming language of your choice (or in pseudo code) to compute the Fibonacci numbers recursively.

(b) (3 %) Approach the function using loop iterations.

(c) (4 %) State the pros and cons （正面反面） of both approaches.