Part A. 作業系統 (50%)

1. (10%) In UNIX, a parent process can create a child process to execute a designated program. Write a Unix program in C language (Algorithm is not acceptable.) which creates a child process to perform the designated system command “/bin/ls.” The parent process will simply create the child process, wait for the child to complete executing the command “/bin/ls.” If the child process completes the task successfully, the parent will print on the console “/bin/ls is executed successfully.”

2. (9%) As a multiple-process solution to the critical-section problem, the MX algorithm is proposed. Determine the correctness of the MX algorithm. If it is correct, prove it. Otherwise, change the least number of lines to correct it. You must identify the lines that you changed. Failing to comply with the rules, no points will be given.

```c
var mutex: array[0..n-1] of boolean;
order: array[0..n-1] of integer;
```

These data structures are initialized to false and 0, respectively. We also define the following notation:

\[(a, b) < (c, d) \text{ if } a < c \text{ or if } a = c \text{ and } b < d.\]

\[
\text{max}(a[0], \ldots, a[n-1]) \text{ is a number } k \text{ such that } k \geq a[i], \text{ for } i = 0, \ldots, n-1.
\]

The structure of process Pi is shown below:

**MX algorithm**

```
L1. repeat
L2.   mutex[i] := true;
L3.   order[i] := max(order[0], order[1], \ldots, order[n-1])+1;
L4.   mutex[i] := false;
L5.   for j:= 0 to n-1
L6.     do begin
L7.       while order[j] != 0
L8.         and (order[j], j) < (order[i], i) do no-op;
L9.       end
L10.  critical section
L11.  order[i] := 0;
L12.  remainder section
L13.  until false
```

3. (6%) The semaphore operations can be defined as

```c
wait(S) : S.value := S.value - 1;
```
if S.value < 0
    then begin
        add this process to S.L;
        block;
    end;

signal(S):  S.value := S.value + 1;
if S.value <= 0
    then begin
        remove a process P from S.L;
    end;

Consider the case that these operations are implemented in a UNIX environment. a) Does these operations work well if they are implemented as subroutines of the application layer in a uniprocessor environment? b) Do these operations work well if they are implemented as system calls in a uniprocessor environment? c) Do these operations work well if they are implemented as system calls in a multiprocessor environment? Do not guess the answer. You will receive +2 points for a correct answer, 0 point for not answering, or -2 penalty points for an incorrect answer.

4. (6%) List 3 different implementations of the page table for a paging memory management system. Discuss the relative merits of them.

5. (8%) Consider the following page reference string:
   1, 2, 3, 4, 2, 1, 2, 5.
   Show the contents of the frames after each reference for the following replacement algorithms, assuming that there are 3 frames of memory with the system and all frames are initially empty.
   (a) FIFO algorithm
   (b) Second-chance algorithm

6. (5%) Consider a system where free space is kept in a free-space list.
   Suppose that the pointer to the free-space list is lost. Can the system reconstruct the free-space list? If your answer is "yes", how to reconstruct the free-space list? If your answer is "no", explain why not.

7. (6%) Explain the following terms for a file system:
   (a) Sequential access method
   (b) Direct access method
   (c) Indexed sequential access method
Part B. 計算機組織 (50%)

1. (4%) What are the purposes of the Booth’s algorithm and the non-restoring division algorithm?

2. (5%) Define zero, de-normalized number, floating point number, infinity, and NaN (Not a Number) in IEEE 754 single precision format by giving range of their exponents and significants, respectively. Write your answer in the following format.

<table>
<thead>
<tr>
<th>zero</th>
<th>de-normalized number</th>
<th>floating point number</th>
<th>infinity</th>
<th>NaN</th>
</tr>
</thead>
<tbody>
<tr>
<td>exponent</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>significant</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3. (8%) Two different compilers are being tested for an 800 MHz machine with three different classes of instructions: Class A, Class B, and Class C, which require three, two, and one cycles, respectively. Both compilers are used to produce code for a large piece of software.

The first compiler’s code uses 10 million Class A instructions, 10 million Class B instructions, and 50 million Class C instructions.

The second compiler’s code uses 10 million Class A instructions, 10 million Class B instructions, and 100 million Class C instructions.

Which sequence will be faster according to MIPS?
Which sequence will be faster according to execution time?

4. (8%) The selection of a set of instructions that the machine can understand demands a delicate balance among the number of instructions needed to execute a program, the number of clock cycles needed by an instruction, and the speed of the clock. Four design principles guide the authors of instruction sets in making that delicate balance: simplicity favors regularity, smaller is faster, good design demands good compromises, and make the common case fast. Briefly explain the four design principles.

5. (6%) Describe the three types of pipeline hazards and the solutions for each type.

6. (5%) Illustrate the following ways for data transfer in an I/O system:
   (a) Polling
   (b) Interrupt-Driven I/O
   (c) I/O using DMA
7. (6%) Describe how a memory hierarchy deals with the following four questions:
   (a) Where can a block be placed?
   (b) How is a block found?
   (c) What block is replaced on a miss?
   (d) How are writes handled?

8. (8%) Given the datapath for the single-cycle implementation of a computer and the definition of its instructions:

   add $rd, $rs, $rt  #$rd = $rs + $rt
   lw $rt, addr($rs)  #$rt = Memory[$rs + addr]
   sw $rt, addr($rs)  #Memory[$rs + addr] = $rt
   beq $rs, $rt, addr  #if ($rs = $rt) go to PC + 4 + 4 × addr

Assume that the instructions are fixed length and the operation time for the major functional units in this implementation are as follows:

- Memory units: 2 ns
- ALU and adders: 2 ns
- Register file (read or write): 1 ns
- Multiplexors, control unit, PC accesses, sign extension unit, and wires: no delay

Compute the time required for each instruction and explain why.