1. [30 points] Consider the following synchronization problem. In this problem, there is one dispatcher who decides the access ordering among ***n*** tasks ($\tau_1, \tau_2, \ldots, \tau_n$). The dispatcher does so by randomly generating a total-ordering sequence Who[1..*n*], in which Who[*i*] = *j* means the *i*-th task entering the critical section is τ_j. In addition, we provide another array Order[1..n] to perform the inverse function. Namely, Order[*i*]= *j* means τ_i is the *j*-th task in the ordering. Naturally, Order[Who[*i*]]=*i* and Who[Order[*j*]] = *j*, $\forall$ *i*, *j*. Task τ_i may enter the critical section if each task τ_j whose Order[*j*] < Order[*i*] has exited the critical section. If a task tried to enter the critical section before its predecessor(s) enter the critical section, it waits. We also provide an array of status flag, done[1..*n*] (initialized to all false), to indicate the execution status of each task. After a task τ_i finishes its execution, it sets its corresponding flag done[*i*] to true. We also make the following assumptions:

- Only one task is allowed at a time in the critical section.
- The dispatcher and each task only execute once.
- The dispatcher finishes its execution before any task starts.
- The dispatcher calls **Gen_Seq**(Order,Who) to set up the values in Order[1..*n*] and Who[1..*n*] arrays.

Based on the above description, the declaration of variables and the pseudo code of the dispatcher and each task are given below:

```
semaphore mutex, first-delay, second-delay;
int first-count, second-count, Order[1..n], Who[1..n];
boolean done[1..n];

Dispatcher:
        Gen_Seq(Order,Who);

Task τi :
        wait( mutex);
        while (my_trun(i) ≠ true) do
            /* my_turn(i) uses Who[], Order[], and done[] arrays to check */
            /* whether Task τi is the next one to enter the critical session or not. */
        begin
            first-count := first-count +1;
            if second-count > 0
```


(背面仍有題目,請繼續作答)

```
            then signal( second-delay)
            else signal( mutex);
        wait( first-delay);
        first-count := first-count - 1;
        second-count := second-count +1;
        if first-count > 0
            then signal( first-delay)
            else signal( second-delay);
        wait( second-delay);
        second-count := second-count - 1;
    end;
    S;   /* enter critical session */
    done[i]=true;
    exit_routine(); /* releases semaphores */
```

For clearness and grading, please **highlight** or underline your answer of each subproblem in your answer sheet. (請在作答卷中明顯地標示每一小題的答案，以利評分作業)

(a) [6 points] Please give the initial values of three semaphores mutex, first-delay, and second-delay.

(b) [4 points] Please give the initial values of first-count and second-count.

(c) [15 points] Please complete the code of task τ_i by writing the pseudo code of exit_routine().

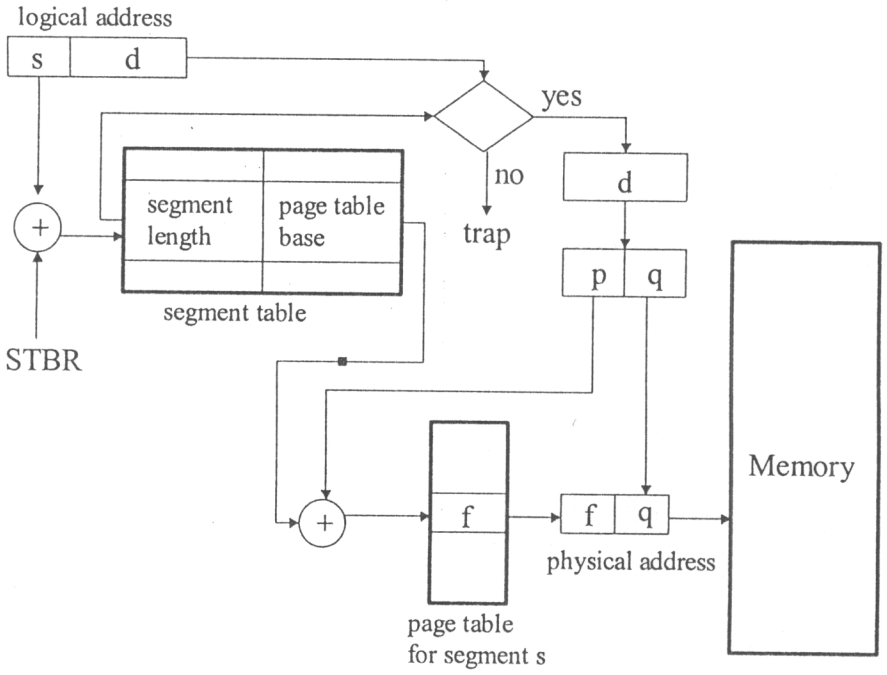
(d) [5 points] Suppose we implement my_turn(i) as shown in the following code.

```
my_turn(i) {
    for k = 1 to Order[i]-1 do
        if !done[Who[k]]
        then return(false);
    end for;
    return(true);
}
```

Consider the situation in which several tasks execute my_trun() as show above at the same time. Is the above code correct in terms of multiple access of done[] array? If you think there is no problem when several tasks access done[] array at the same time, please say "no problem" in your answer sheet;

Otherwise, if you think there might be some "mutual exclusion" problem for my_turn(), please say "not correct" in your answer sheet.

2. [20 points] Consider a segmentation-with-paging address translation scheme shown in the figure below. This scheme provides each process with eight segments, each with a maximum size of 1 Mbytes (2^{20} bytes). This scheme is to be implemented on an architecture that supports a maximum physical memory size of 64 Mbytes (2^{26} bytes). The architecture provides a single base register (i.e. STBR) from which to start the address translation process. In the architecture, pages of size 4 Kbytes (2^{12} bytes) are to be used. Note that one word in the memory has four bytes and a logical/physical address specifies one word in the memory. The segment table and page table are not necessarily implemented in main memory, hence are not constrained by word-wide (i.e. 4-byte) access. Please compute the width (i.e. bit length) of variables s, d, p, f, and q in the figure. For clearness and grading, please **highlight** or underline your answer of each variable in your answer sheet. (請在作答卷中明顯地標示每一變數的答案，以利評分作業)



(背面仍有題目,請繼續作答)

3. Assume the following operand notation: (*24%*)

Rn refers to register n in register mode.

#x refers to the number x in immediate mode.

(x) refers to the number x used in direct mode, i.e. as an address.

x(Rn) refers to register n and immediate x in displacement mode.

@(x) refers to the number x used in indirect mode, i.e. as an address of an address.

@x(Rn) refers to register Rn and number x used in displacement deferred mode, i.e. the value obtained from displacement mode access is used as an address for the desired value.

Given the following "memory map" with values given in Hex, and that the register R1 contains 4. Please use the **little Endian** format to show the contents of the two-word register R7 after each instruction in the a)~h) instructions does ?

Byte Address	Content
18	00
17	03
16	04
15	0A
14	80
13	00
12	00
11	00
10	04
F	62
E	56
D	11
C	10
B	00
A	00
9	00
8	04
7	00
6	00
5	00
4	08
3	00
2	00
1	01
0	00

a) **LW** R7, #16
b) **LW** R7, (16)
c) **LW** R7, @(16)
d) **LBU** R7, #14
e) **LB** R7, (14)
f) **LH** R7, 4(R1)
g) **LHU** R7, @4(R1)
h) **LW** R7, (12)

Note:

Literals in the instructions are in decimal.
LW: load word
LH: load half word
LHU: load half word unsigned
LB: load byte
LBU: load byte unsigned

4. Between 1982 and 1992 the CPU performance of the x86 family of microprocessors improved by a factor of 25. Note that this does not count improvements in the memory hierarchy and I/O. (*26%*)
   a) What is the average yearly CPU performance increase? (*10%*)
   b) Assuming that during that time the CPU clock rate increased from 5 Mhz to 50 Mhz, but that no new instructions were introduced, to what other factor would you attribute the rest of the performance increase? (*6%*)
   c) What yearly percentage increase would you attribute to each of the two factors in part b) ? (*10%*)