Assignment 1

Due September 30^{th} 2009

Continuous and discrete simulation. Modelling a system.

The problem

Two very long processes are currently in execution; they will not terminate in the near future and no other processes are going to appear.

```
A.0   double A[N][N] , B[N][N] , C[N][N] ;
A.1   ... // reads A and B
A.2   for( i = 0 ; i ≤ N ; i++ )
A.3       for( j = 0 ; j ≤ N ; j++ )
A.4           C[i][j] = 0 ;
A.5   for( i = 0 ; i ≤ N ; i++ )
A.6       for( j = 0 ; j ≤ N ; j++ )
A.7           for( k = 0 ; k ≤ N ; k++ )
A.8               C[i][j] += A[i][k] * B[k][j] ;
```

```
B.0     mf = open( ”order” , O_RDONLY , mode ) ;
B.1     inp = open( ”input” , O_RDONLY , mode ) ;
B.2     out = open( ”output” , O_WRONLY , mode ) ;
B.3     for( ;; ) {
B.4         n = read( mf , &offset , sizeof(off_t) ) ;
B.5         if( n != sizeof(off_t) ) break ;
B.6         read( inp , buf , BLOCK ) ;
B.7         lseek( out , n , 0 ) ;
B.8         write( out , buf , BLOCK ) ;
          }
```

Note that only parts of the code of the processes are shown. In order to compile and run they will need a few declarations (and **include**s).

Meaningful parameters

The units used are: 1*ns* as a unit of time and 1B as a unit of memory,

A few parameters will be needed. While a sample value is associated with each of them, your simulator must treat them as parameters.

S:

T_i: CPU time needed to handle an interrupt (without a context switch). Equal to 200 *ns*.

T_c: context switch time. Equal to 200 *ns* for each half (store old context or load new context)

D: time needed by the CPU to hand a request to the disk controller (always successful). Equal to 80 *ns*.

Z: the Scheduler takes 40 ns to decide what to do.

N: is a parameter in process **A** and is a value large enough to have $\mathrm{N}^2/8 >$

memory size.

Processes

A

This process computes the product of 2 matrices. We do not need to know what the result is; all we care is how its execution is performed. The value of **N** is a parameter.

The bulk (and only interesting part) of the execution of **A** is loop A.7. Each iteration of it does the following:

1. Fetch **A[i][k]**. Triggers a page fault every $S/8$ times, in which case the OS is invoked. Takes 11*ns* when successful.
2. Fetch **B[k][j]**. Triggers an interrupt every time. After the OS does its job, this instruction is restarted and succeeds in 11*ns*.
3. The rest is performed. We assume that the store (C[i][j] =) never causes a page fault (not quite true). Thus, is all takes 36*ns*.
4. The next iteration is started.

The OS

Only small parts of the OS are of interest to us:

Interrupt handler (we consider only two interrupts, ignoring all other) is handled in T_i time and does not perform a context switch. The handler ends with a **goto** to the Scheduler.

Scheduler: there are several possibilities:

Handling a page fault: interrupt + request for page (D) + Scheduler.

Handling a disk interrupt: interrupt + Scheduler.

To be done

CPU utilisation:

Disk utilisation:

These values are functions of S (although they obviously depend on all the other parameters).

The first simulator uses continuous simulation. It must work before you start working on the second simulator which will be event–driven (discrete).

Submission rules

You will deliver your assignment work in person at a time scheduled in advance (a sign–up sheet will be available). The details of the presentation (and the deliverables) will be discussed in due time.