This lab investigates data structure alignment and how it may affect performance. It also investigates how data locality may affect performance.

1. Get copies of the sample program `align.c`. Note that `align.c` needs the auxiliary timing routines `timing.c` and `timing.h`. This program used to rely on `#pragma pack` but this feature is no longer supported by `gcc`. Therefore, to compile the program with packed structures and run, do

```
gcc -g -fpack-struct align.c timing.c -o align
./align
```

The results should be recorded in `align_off.log`. To compile the program with alignment and run, do

```
gcc -g align.c timing.c -o align
./align
```

The results should be recorded in `align_on.log`. You should notice that datasize is less in the case of packed (but unaligned) structures, but there is a performance hit in Net Memory Bandwidth.

2. Get copies of the sample program `pager02.c` and the `Makefile`. Note that `pager02.c` needs the auxiliary timing routines `timing.c` and `timing.h`. This program is designed to test paging and TLB cache performance. Compile and link the program.

3. The program reserves a large matrix whose size you are asked to choose (anywhere from 1024 x 1024 to 8192 x 8192) although you should probably choose a middle value in this range to start and run the larger values when the machine is not heavily loaded. The matrix will be filled by randomly chosen integers. The program asks whether you want to process the matrix by `rows` or `columns`. You should try each in separate runs. The program then monitors the time needed to sum all entries of the matrix and puts the results in `pager_r.log` or `pager_c.log` depending upon whether you chose row processing or column processing.

**Assignment**

1. Record (and hand in) the values in Net Memory Bandwidth you got in both cases (alignment on and off). Explain the differences which occurred in the program `align` between packing the structure (alignment off) and letting the compiler do the natural alignment for the structure (alignment on). What can you conclude about the importance of aligned data (especially on RISC systems)?

Try at least four different sizes in the `pager02` program from minimum to maximum and record (and hand in) the times you got for both row and column processing.

2. Carefully explain *why* the profiling times for column processing are significantly greater than the profiling times for row processing, in light of the language used to write the program.

3. Are there any other issues which affect the timings besides choosing rows or columns? Explain.