

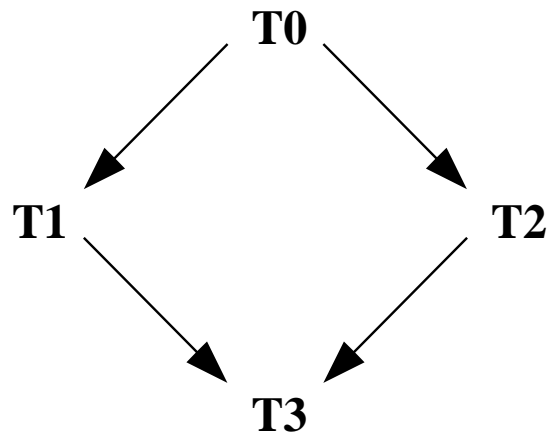
## Homework 6

Reading: Tanenbaum, Section 10.4, Begin Chapter 4

Due: Wed, Apr. 19, 2006

**Problem 1** (1 Points)

The task graph below shows the desired execution order of 4 tasks (T0, T1, T2, T3). An arrow from  $T_i$  to  $T_j$  indicates that task  $i$  must be executed before task  $j$ . For example, T0 must execute before T1, and T2. T1 and T2 can execute in parallel. And T3 can not execute until T1 and T2 are done.



What forms of task graphs CAN NOT be implemented using just the `parbegin` and `sequence (... ; ... ;)` control constructs? Give the simplest example of such a graph.

**Problem 2** (4 Points)

When a process wants a buffer in a uniprocessor OS, it normally first checks its *locked* flag. If the flag is 0, the process sets the flag and uses the buffer. Any other process wanting the buffer while the *locked* flag is set usually sets the buffers *wanted* flag and blocks (places itself on a sleep queue). When the buffer is free, the first process clears *locked* and moves all processes in the sleep queue to the run queue.

- Why does this approach work for a uniprocessor?
- What problem(s) does this approach have in a multiprocessor system; i.e., a system that has multiple CPUs?
- What would be a better alternative in a multiprocessor system?

**Problem 3** (0 Points)

Consider a dynamically partitioned memory that has two holes with sizes: 1300 KB and 1200 KB. Assume that memory requests arrive in the following order: A) 1000 KB, B) 1100 KB, and C) 250 KB. Compare the results of the four allocation algorithms first-fit, best-fit, next-fit, and worst-fit.

- a) Use a table to show the evolution of free space as each memory request is handled. The table should have five columns (memory request, and free space sizes for the four algorithms) and four rows (initial free space, and each memory request).
- b) For the above workload, rank the four algorithms by memory utilization. By speed.
- c) Do the results for the above workload correspond to expectations for general workloads? Explain.

**Problem 4** (0 Points)

Consider a buddy system and the address 011011110000.

- a) If the block size associated with this address is 8 bytes, what is the binary address of the buddy?
- b) What is the largest block size  $N$  such that the above address still has a buddy? Explain.

**Problem 5** (4 Points)

Consider a buddy system and the address 100100001000. Assume the largest block has  $2^U$  bytes and the smallest block has  $2^L$  bytes.

- a) If the block size is 8 bytes, what is the binary address of the buddy?
- b) What is the largest block size  $N$  such that the above address still has a buddy? Explain.
- c) Let  $b_k(x)$  be the buddy of address  $x$  with block size  $2^k$ . Write an expression (NOT an algorithm) for  $b_k(x)$ . Explain why the form of the expression is correct.
- d) Demonstrate that your expression in Part c is correct.

**Problem 6** (0 Points)

If page table entries are 4 bytes each and the page size is 8 KB, how many levels of page tables would be required to map a 32-bit address space if the top level page table fits into a single page?

**Problem 7** (2 Points)

Suppose that a logical address space has eight pages where each page is 1,024 bytes and this logical address space is mapped onto a physical memory of 32 page frames.

- a) How many bits are there in the logical address?
- b) How many bits are there in the physical address?
- c) Describe the structure of the page table for this system.