Illinois Institute of Technology **Homework #10**
CS 450 - Operating Systems **November 23, 2002**

Homework # 10 – DUE: 11:59pm December 6, 2002
NO EXTENSIONS WILL BE GIVEN

1. **Overview**

   This is a much shorter assignment than your previous two. Your task is to implement two functions that you have already used heavily, `prepage` and `start_cost` from the `MEMORY` module of *OSP*.

2. **Getting Set Up**

   To start out with, make a directory called `as10` and copy the following files into your directory.

   ```
   ~osp/asg10.sparc64/Makefile
   ~osp/asg10.sparc64/memory.c
   ~osp/asg10.sparc64/dialog.c
   ~osp/asg10.sparc64/par.low
   ~osp/asg10.sparg64/par.high
   ```

3. **What To Do**

   In order to not overburden you with this assignment, there are only two functions to implment, and we're going to give you a pretty good outline of what to do. Those functions, as previously stated are `prepage` and `start_cost`, they will link in with the rest of the `MEMORY` module from *OSP*. This gives you a little less flexibility in your design, but makes the assignment easier. For most people you should not have to implement any more functions than these two.

4. **Grading**

   For your program, credit will be given for *style* and *correctness*. Elements important to good style include:

   - discussion - see below,
   - documentation - especially function headers,
   - modularity - a function should not (usually) extend beyond a single page, and
   - readability - the various functions should be visually separated.

   At the end of your program, create a section called "Implementation and Design Analysis" where you comment about how you implemented your program.

   The correctness of your implementation reflects the ability of your program to run to completion without errors on both parameter files.

5. **Handing In Your Assignment**

   There are two simulation runs that are used with this set of programs. When you submit your program using the `hand_in` script, you should use these files (in this order):

   ```
   ~osp/asg10.sparc64/par.high
   ~osp/asg10.sparc64/par.low
   ```

Make sure that your *name* and *login name* are included in your source file. Also include the descriptions mentioned previously as a block of comments. Use the `hand_in` procedure to post your work to the submissions directory by 11:59:59pm of the due date. Make sure to save copies of your program for your own reference. If we need to get more information from you, we will contact you.

NOTICE:

Unauthorized collaboration or assistance is prohibited. If you have a question, ask the teaching assistants. Do not copy code from a neighbor. Students who participate in copying will receive an 'E' grade for the course. The same penalty applies for students who fail to adequately protect their accounts.

Appendix

Helper Functions

Because there is a lot of functions and variables that provided for you that are not in the book, they will be documented here.

`PAGE_TBL *get_page_tbl(PCB *pcb)`

This is a macro that expands to `pcb->page_tbl`; It canbe used to make your code cleaner.

`WORKING_SET *find_working_set(PAGE_TBL *page_tbl_ptr)`

Returns a pointer to working set of pages for this page table.

`FRACTION`

A constant declared to show how many pages to prepage. This 10% of the degree of prepaging as specified in your simulation parameters.

`void swap_in_working_set(WORKING_SET *working_set, PCB *pcb)` Swaps the working set of pages, specified by the pointer argument `working_set`, for `pcb` into memory. You will call this from `prepage`.

`void initial_swap_in(int num_of_pages, PCB *pcb)` Is called when a process has no working set. Swaps in up to `num\_of\_pages` into main memory. This is also called from `prepage`.

`void set_working_set(PAGE_TBL *page_tbl)` Sets the next available working set from the working set pool to this page table. This is called, along with `initial_swap_in` from `prepage` when there is no working set for the process.

Function Descriptions

This section provides a virtual step by step of how to implement the functions. If you read this, and understand this, you should be good.

`int start_cost(PCB *pcb)`

This first thing you need is pointer to the `PAGE_TBL` and `WORKING_SET` for the process. To get these use the `get_page_tbl` macro and `find_working_set` function as detailed above.

Then, you must check to see if the process is already partially loaded. This is indicated by the situation where the working set is not `NULL` and the top element of the working set (if your `WORKING_SET*` was `wsptr` then the top element is `wsptr->top`) is not `NIL`.

If this is the case, you will want to prepage at most `(wsptr->top + 1) * FRACTION` pages. Then your job is to iterate over the pages and working set, if a page is not loaded, load it into memory and increment the counter of the actual number of pages to swap in. You can tell if `page[i]` is loaded already or not by checking `page_tbl_ptr->page_entr[ws_ptr->stack[i]].valid`. Now that you know how many pages you actually have to swap in (which should at most be the max number of pages to load - number of currently loaded pages) return the actual cost to load those pages. This should take the form of `COST_OF_PAGE_TRANSFER * pages_to_swap_in`.

In the case where there is not a working set, we only want to swap in a small fraction. To accomplish this we need to figure out how much to load. We take the minimum of `pcb->size` and `SET_SIZE*PAGE_SIZE` and multiply this by `FRACTION` to get what percentage of the overall program size to load. The number of pages is then the ceiling of `swap_in_size/PAGE_SIZE`.

From here we can iterate over the page entries of the page table for the pcb up to number of pages to swap in. If the entry is not valid, we increment the counter `pages_to_swap_in`. Then we return `COST_OF_PAGE_TRANSFER * pages_to_swap_in`.

`void prepage(PCB *pcb)`

This is actually easier than the start cost, but much of the logic is the same. As before, obtain the `PAGE_TBL` and `WORKING_SET` for the pcb. If there is a working set, swap it in by calling `swap_in_working_set(working_set, pcb)`. If there isn't the number of pages should be equal to the maximum number of pages you calculated before in `start_cost`. Then call `initial_swap_in(num_of_pages,pcb)` and `set_working_set(page_tbl_ptr)`. That's all there is to it.