Paging and Address Translation

Introduction
This shows the function used to map a logical address to a physical address for some paging schemes. Throughout this handout, an address in virtual memory is a pair \((\text{logical\_page}, \text{offset})\) where \text{logical\_page} is the page number within the logical address space and \text{offset} the offset into that page. Also, \text{page\_size} is the size of the page (which is a multiple of 2). We will assume the entire program is in memory, so no error handling is given; were this assumption false, the situation where the requested address were not in memory would need to be handled (by generating a page fault and loading the necessary page):

Paging Address Translation by Direct Mapping
This method stores the page table in main memory and the address of this table in the process control block, in a register called the page table base register. Let the page table base register be called \text{pt\_base\_register}, and let memory represent the main store of the computer. Then:

\[
\text{function NL\_map((logical\_page, offset)):] physical\_address};
\]
\[
\text{begin}
\]
\[
\quad \text{NL\_map := memory[pt\_base\_register + logical\_page] * page\_size + offset;}
\]
\[
\text{end (* NL\_map *)}
\]

In pictures, here is what is going on:

Paging Address Translation by Associative Mapping
In this algorithm, \text{assoc\_page\_table} represents an associative memory. This function can check a type of memory called "associative memory" (or "cache" or "lookaside memory") which stores both a frame number and a page number. The search is done in parallel, and is much faster than a linear (or binary) search. The function returns the frame number associated with its argument:
function NL_map((logical_page, offset)): physical_address;
begin
    NL_map := assoc_page_table(logical_page) * 
              page_size + offset;
end (* NL_map *)

Paging Address Translation with Combined Associative and Direct Mapping

This combines the above two methods. The array page_table is a small associative store that can hold only a few page numbers; there is also a page table kept in memory. For this method, we shall assume that if there is no entry for logical_page in the associative memory, assoc_page_table returns –1. Taking everything else as in the previous two sections:

function NL_map((logical_page, offset)): physical_address;
var  frame_number: integer;
begin
    frame_number := assoc_page_table(logical_page);
    if frame_number = -1 then(* not in associative memory *)
        NL_map := memory[pt_base_register + logical_page] 
                  * page_size + offset;
    else
        NL_map := frame_number * page_size + offset;
end (* NL_map *)

This is the most common method, and is used in modern computers with paging.