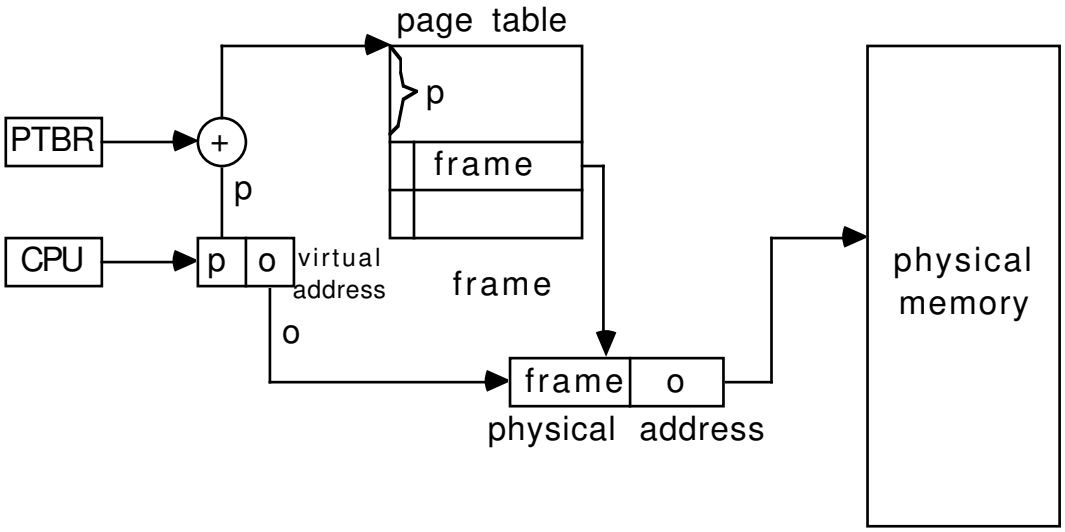
false, the situation where the requested address were not in memory would need to be handled (by generating a page fault and loading the necessary page):

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 *pt_base_register*, and let memory represent the main store of the computer. Then:

```
function NL_map((logical_page, offset)): physical_address;
begin
      NL_map := memory[pt_base_register + logical_page] * page_size + offset;
end (* NL_map *)
```

In pictures, here is what is going on:



Paging Address Translation by Associative Mapping

In this algorithm, *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_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.