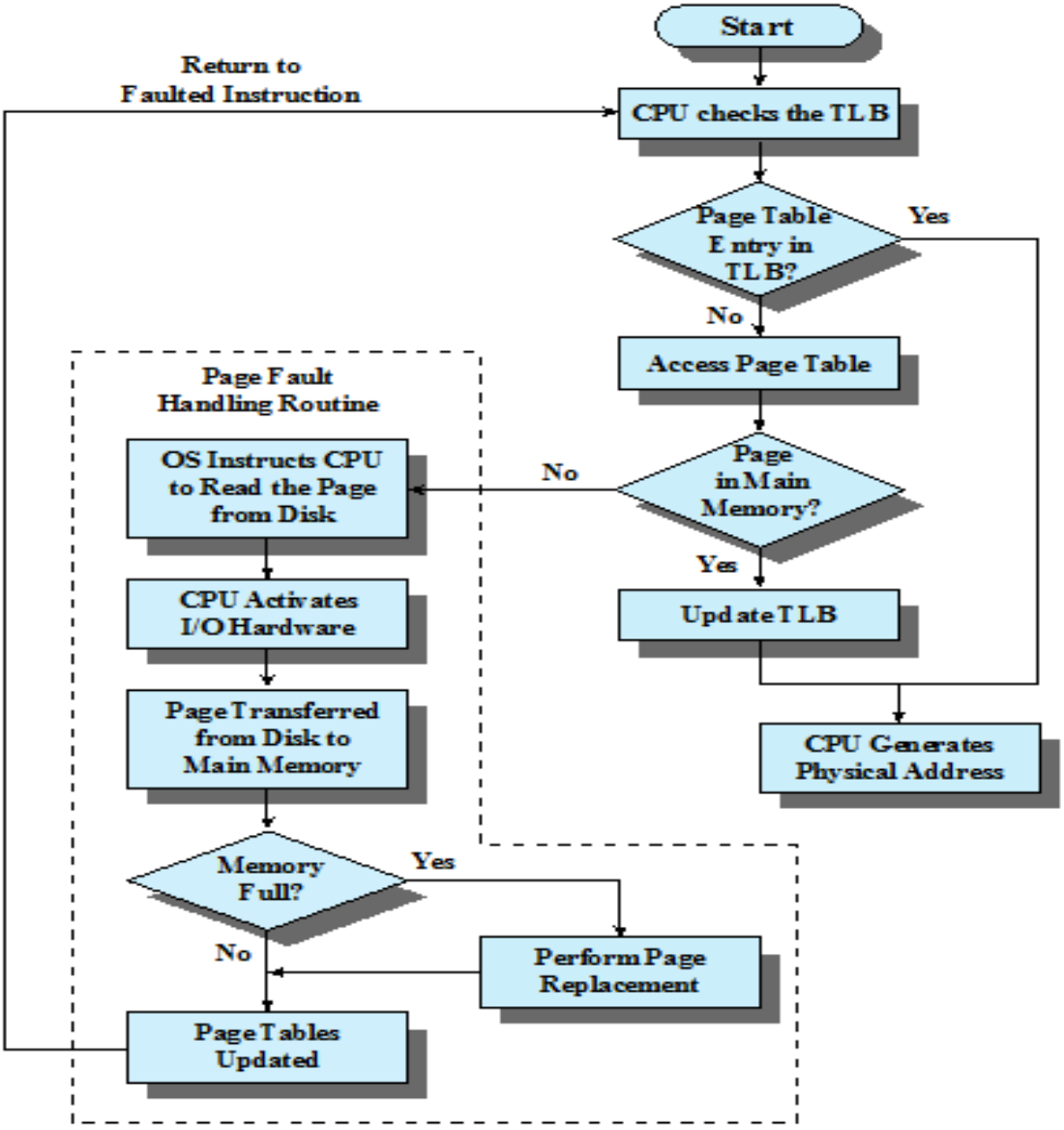
Project 5 Virtual Memory Manager

Put It All Together

This document shows you how to integrated our designed functions together to implemented a simulated virtual memory manager. We also demonstrate below how to invoke the `TLB_replacement_LRU()` function, which can be found in "`Lec16a-Project 5 LRU Implementation Draft.pdf`"

1. The algorithm:

   The implementation of this algorithm can be found on page 3.
   The functions integrated in the implementation are listed on page 2.



2. The following functions are integrated to implement the algorithms shown on page 1. Please refer to "Lec14c-Project 5 Data Structures Exercise Handout.pdf" for the prototype design of these functions.

```
/* see function 3 in Lec14c-Project 5 Data Structures Exercise Handout */
int search_TLB(page_t p_num, tlb_t tlb, bool *is_tlb_hit, frame_t *f_num)

/* see function 4 in Lec14c-Project 5 Data Structures Exercise Handout */
int search_page_table(page_t p_num, page_table_t p_table, bool *is_page_fault,
                  frame_t *f_num);

/*
 * see function 5 in Lec14c-Project 5 Data Structures Exercise Handout
 * Attention: The prototype has been changed by adding frame_number as
 * an input parameter.
 */
int page_fault_handler(page_t p_num, frame_t frame_num,
                       physical_memory_t *physical_mem,
                       page_table_t *p_table, tlb_t *tlb);

/* see function 6 in Lec14c-Project 5 Data Structures Exercise Handout */
create_physical_address(frame_t f_num, offset_t off,
                        physical_address_t *physical_addr)

/* see function 7 in Lec14c-Project 5 Data Structures Exercise Handout */
int read_physical_memory (physical_address_t p_addr,
                      physical_memory_t physical_mem,
                      value_t *value)

/* see function 8 in Lec14c-Project 5 Data Structures Exercise Handout */
update_address_value_list(logic_address_t l_addr, physical_address_t p_addr,
                       value_t value, address_value_list_t *addr_value_list);

 /* see function 9 in Lec14c-Project 5 Data Structures Exercise Handout */
output_address_value_list(const char *output_file_name,
                          address_value_list_t addr_value_list)

/* see also Lec16a-Project 5 LRU Implementation Draft.pdf */
int TLB_replacement_LRU(page_t p_num, frame_t f_num, tlb_t *tlb);

/*
 * Get a page number from a logical address.
 * Input: a logical address.
 * Output: a page number
 */
int get_page_num(logic_address_t l_addr, page_t *p_num);

/*
 * Get a page number from a logical address.
 * Input: a logical address.
 * Output: an offset
 */
int get_offset(logic_address_t l_addr, offset_t *off);
```

3. Sample code of the main function:

```
main() {
   /* Variables: page number, frame number and offset */
   page_t page_num;
   frame_t frame_num;
   offset_t offset;

   /* Addresses */
   logic_address_t logic_address;
   physical_address_t physical_address;

   /* The TLB and page table */
   tlb_t sys_tlb;
   page_table_t page_table;

   /* Simulated main memory */
   physical_memory_t physical_memory;

   /* value and address-value list */
   value_t value;
   address_value_list_t address_value_list;

   /* Boolean for TLB hit and page fault */
   bool is_tlb_hit;
   bool is_page_fault;

   /* Input and output file names */
   const char input_file[] = "input_logical_address_file";
   const char output_file[] = "output_physical_address_value";

   /* Initialize the system */
   init_tlb(&sys_tlb);
   init_page_table(&page_table);

   /* Create a logical address list from the file */
   logic_address_loader(logic_address_file_name, logic_address_list);

   for (each logical address in logic_address_list) {
       /* Get a logic address, its page number and offset */
       get_a_logic_address(logic_address_list, logic_address);
       /*
        * The code below demonstrates how to use a pointer to access
        * page_number updated by the get_page_number() function
        */
       get_page_number(logic_address, &page_number);
       get_offset(logic_address, &offset);

       /* Search the TLB */
       search_TLB(page_num, sys_tlb, &is_tlb_hit, &frame_num);

       /* Hit the TLB: the address translation is done. */
       if (is_tlb_hit == TRUE) {
           create_physical_address(frame_num, offset, &physical_address);
       }
```

```
        /* TLB Miss: check page table */
        else {
            search_page_table(page_num, page_table,
                              &is_page_fault, &frame_num);

            /* page is found in the page table */
            if (is_page_fault == FALSE) {
                create_physical_address(frame_num, offset, &physical_address);

                /* Replace the oldest entry in the TLB with this new entry */
                TLB_replacement_LRU(page_num, frame_num, &sys_tlb);
            }

            /* page fault occurs: call fault_fault_handler */
            else {
                /*
                 * Handling a page fault: Load a 256-byte page from backing_store
                  * into the simulated main memory.
                  * Attention: You need to call the page_table_update() and
                  * TLB_replacement_LRU() functions to implement the following
                  * page_fault_handler() function, where both page_table and
                  * sys_tlb must be updated.
                  * The prototype has been changed by adding frame_number as
                  * an input parameter.
                  */
                page_fault_handler(page_num, frame_num, &physical_memory,
                                   &page_table, &sys_tlb);
                create_physical_address(frame_num, offset, &physical_address);
            }
        } /* end of else TLB Miss */

        /* Read one-byte value from the physical memory */
        read_physical_memory(physical_address, &value);

        /* Update the address-value list */
        update_address_value_list(logic_address, physical_address, value
                                  &address_value_list)
    } /* end of for logic_address_list */

    /* Output the address-value list into an output file */
    output_address_value_list(output_file, address_value_list);

} /* end of main() */
```