# Project 5 Virtual Memory Manager

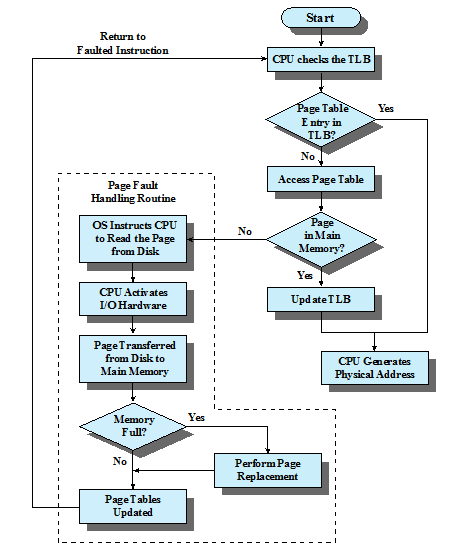
# Put It All Together

This document shows you how to integrate 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 “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.



1. 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);

1. 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() \*/