```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define PAGE_SIZE 256
#define PHYSICAL_MEM_SIZE 1024
#define LOGICAL_MEM_SIZE 2048
#define NUM_FRAMES (PHYSICAL_MEM_SIZE / PAGE_SIZE)
#define NUM_PAGES (LOGICAL_MEM_SIZE / PAGE_SIZE)

int physical_memory[PHYSICAL_MEM_SIZE];
int page_table[NUM_PAGES];

void initialize_page_table() {
  for (int i = 0; i < NUM_PAGES; i++) {
    page_table[i] = -1;
  }
}

int allocate_frame(int page_number) {
  int frame_number = rand() % NUM_FRAMES;
  page_table[page_number] = frame_number;
  return frame_number;
}

int translate_address(int logical_address) {
  int page_number = logical_address / PAGE_SIZE;
  int offset = logical_address % PAGE_SIZE;

  int frame_number = page_table[page_number];
  if (frame_number == -1) {
    frame_number = allocate_frame(page_number);
  }

  return frame_number * PAGE_SIZE + offset;
}

void write_memory(int logical_address, int value) {
  int physical_address = translate_address(logical_address);
  physical_memory[physical_address] = value;
}

int read_memory(int logical_address) {
  int physical_address = translate_address(logical_address);
  return physical_memory[physical_address];
}

int main() {
  srand(time(0));
  initialize_page_table();

  write_memory(500, 42);
  write_memory(1200, 84);

  printf("Value at logical address 500: %d\n", read_memory(500));
  printf("Value at logical address 1200: %d\n", read_memory(1200));

  return 0;
}
```