Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №7**

По дисциплине: «Аппаратное обеспечение интеллектуальных систем»

Тема: «**Кэш-память**»

**Выполнил:**

Студент 2 курса

Группы ИИ-23

Макаревич Н. Р.

**Проверил:**

Михно Е.В.

Брест 2024

**Цель работы:** Изучить основные типы кэш-памяти, ознакомиться с алгоритмами размещения и восстановления данных, реализовать программную модель кэш-памяти.

**Задание:**

1. Изучить теоретический материал об основных типах кэш-памяти и алгоритмах ее функционирования.
2. Написать программу на ЯВУ, моделирующую кэш-память согласно [варианту](#_Варианты:).
3. Реализовать поиск данных по заданному физическому адресу в кэш-памяти с определением кэш-попадания либо кэш-промаха и возвратом необходимых данных.

Варианты заданий приведены в следующей таблице:

|  |  |  |
| --- | --- | --- |
| № п.п. | Тип кэш-памяти | Размерность кэш-памяти  (количество кэш-строк на количество блоков (количество наборов)) |
| 3. | множественный ассоциативный кэш | 8 х 8 (4 набора) |

**Ход работы**

**Код программы:**

int Cache::find(const std::bitset<64>& address) {

std::cout << "Address: " << address << "\n";

unsigned long set = ((address >> wordSize) & std::bitset<64>(0b11)).to\_ulong();

std::cout << "Set: " << set << "\n";

std::bitset<64> tag = address >> 7;

std::cout << "Tag: " << tag << "\n";

bool isFound = false;

for (int i = 0; i < setSize; i++) {

if (tags[set \* setSize + i] == tag) {

unsigned long word = (address & std::bitset<64>(0b111)).to\_ulong();

std::cout << "Ofset: " << word << "\n";

if (dataMatrix[set \* setSize + i][word] == NULL)

dataMatrix[set \* setSize + i][word] = \*getValueAtAddress(address);

isFound = true;

updateLRU(set, i);

std::cout << "Hit\n";

return dataMatrix[set \* setSize + i][word];

}

}

if (!isFound) {

unsigned long word = (address & std::bitset<64>(0b111)).to\_ulong();

std::cout << "Ofset: " << word << "\n";

int\* ptr = getValueAtAddress(address);

int data = \*ptr;

saveAddress(tag, set, word, data);

std::cout << "Miss\n";

return data;

}

}

void Cache::saveAddress(std::bitset<64>& tag, unsigned long set, unsigned long word, int data) {

int ind = LRU[set][0];

tags[set \* setSize + ind] = tag;

dataMatrix[set \* setSize + ind] = std::vector<int>(blocks, NULL);

dataMatrix[set \* setSize + ind][word] = data;

updateLRU(set, ind);

}

void Cache::updateLRU(int set, int ind) {

LRU[set].push\_back(ind);

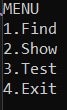
for (int i = 0; i < LRU[set].size() - 1; i++) {

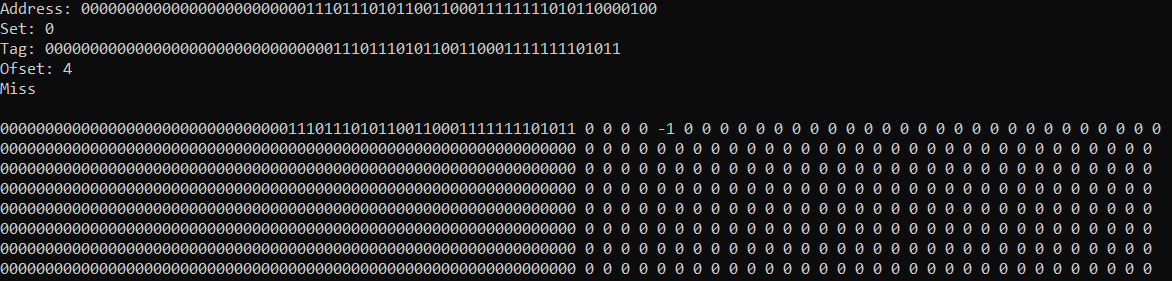
if(LRU[set][i] == ind)

LRU[set].erase(LRU[set].begin() + i);

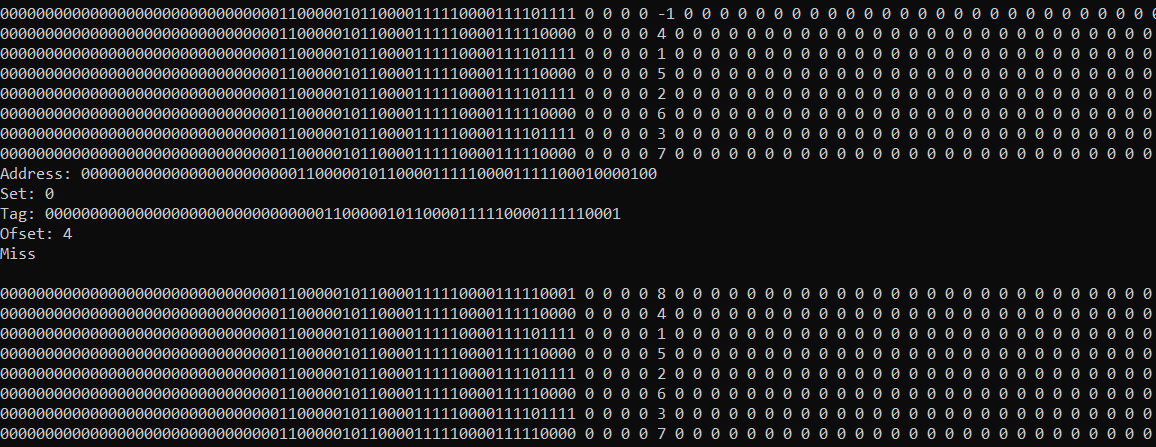
}

}Меню:

****

**Добавление в Кэш:**

**Замена значения в строке:**

****

**Вывод:** изучил основные принципы реализации кэш-памяти