Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №4**

По дисциплине «Аппаратное обеспечение интеллектуальных систем»

Тема: «Кэш-память»

**Выполнил:**

Студент 2 курса

Группы ИИ-21

Кирилович А. А.

**Проверил:**

Михно Е.В.

Брест 2023

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

**Ход работы:**

**Вариант 4**

|  |  |
| --- | --- |
| Тип кэш-памяти | Размерность кэш-памяти |
| полностью ассоциативный кэш | 16 х 4 |

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

**Результат программы:**

import logging

from math import log2, ceil

from random import randint

logging.basicConfig(level=logging.INFO, format='%(message)s')

def get\_tag\_offset(address, block\_size):

    address = "00" + bin(address)[2:]

    offset\_len = ceil(log2(block\_size))

    return int(address[:-offset\_len], 2), int(address[-offset\_len:], 2)

def create\_ram(size):

    return [0] \* size

def ram\_read(ram, address):

    logging.info(f'Reading from RAM: address={address}')

    return ram[address]

def ram\_write(ram, address, data):

    logging.info(f'Writing to RAM: address={address}, data={data}')

    ram[address] = data

def create\_cache\_line(block\_size):

    return {'is\_empty': True, 'tag': None, 'data': [None] \* block\_size}

def cache\_line\_read(cache\_line, offset):

    return cache\_line['data'][offset]

def cache\_line\_write(cache\_line, tag, offset, data):

    cache\_line['is\_empty'] = False

    cache\_line['tag'] = tag

    cache\_line['data'][offset] = data

def shift(cache\_lines, index):

    cache\_lines[index:] = cache\_lines[index + 1:] + [cache\_lines[index]]

def cache\_write(cache\_lines, address, block\_size, ram, data):

    tag, offset = get\_tag\_offset(address, block\_size)

    for cache\_line in cache\_lines:

        if cache\_line['is\_empty']:

            cache\_line\_write(cache\_line, tag, offset, data)

            logging.info(f'Writing to Cache: address={address}, tag={tag}, offset={offset},data={data}')

            return

    shift(cache\_lines, 0)

    cache\_line\_write(cache\_lines[-1], tag, offset, data)

    logging.info(f'Writing to Cache: address={address}, tag={tag}, offset={offset},data={data}')

def cache\_access(cache\_lines, address, block\_size, ram):

    tag, offset = get\_tag\_offset(address, block\_size)

    for i, cache\_line in enumerate(cache\_lines):

        if cache\_line['data'][offset] is not None and cache\_line['tag'] == tag:

            data = cache\_line\_read(cache\_line, offset)

            shift(cache\_lines, i)

            cache\_line\_write(cache\_lines[-1], tag, offset, data)

            logging.info(f'Cache Hit: address={address}, data={data}')

            return data

    data = ram\_read(ram, address)

    cache\_line\_write(cache\_lines[-1], tag, offset, data)

    cache\_write(cache\_lines, address, block\_size, ram, data)

    logging.info(f'Cache Miss: address={address}, data={data}')

    return data

def show\_cache(cache\_lines):

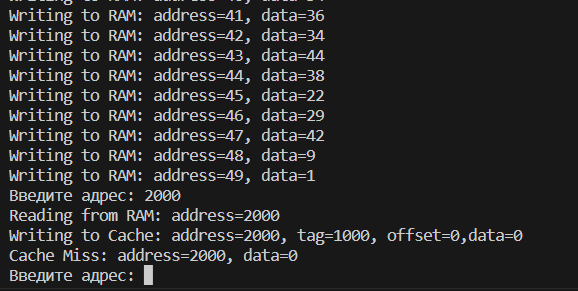
    for i, cache\_line in enumerate(cache\_lines):

        print(f"{i}, tag={cache\_line['tag']}, {cache\_line['data'][0]}\t{cache\_line['data'][1]}")

def main():

    ram = create\_ram(2048)

    cache\_lines = [create\_cache\_line(4) for \_ in range(16)]

    for i in range(50):

        ram\_write(ram, i, randint(0, 50))

    while True:

        address = input("Введите адрес: ")

        try:

            address = int(address)

        except ValueError:

            show\_cache(cache\_lines)

        else:

            cache\_access(cache\_lines, address, 2, ram)

if \_\_name\_\_ == '\_\_main\_\_':

    main()

**Вывод:** Изучил основные типы кэш-памяти, ознакомился с алгоритмами размещения и восстановления данных, реализовал программную модель кэш-памяти.