МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

**ИНТЕЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ**

ОТЧЁТ

По лабораторной работе № \_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Выполнил:

Студент группы ИИ-22

Копанчук Евгений Романович

Проверил\_\_:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Брест – 2023

**Ход работы**

TASK

|  |
| --- |
| Реализовать поиск данных по заданному физическому адресу в кэш-памяти с определением кэш-попадания либо кэш-промаха и возвратом необходимых данных. |

|  |  |  |
| --- | --- | --- |
| № п.п. | Тип кэш-памяти | Размерность кэш-памяти  (количество кэш-строк на количество блоков (количество наборов)) |
| 9 | множественный ассоциативный кэш | 32 х 2 (8 наборов) |

PROGRAM

|  |
| --- |
| string int2bitstr(int data);  int bitstr2int(string bits);  int search(int item, int arr[], int size);  class Cash {  private:  int n\_block\_;  int n\_;  int m\_;  int\*\*\* mem\_;  int\*\*\* flag\_;  public:  Cash();  ~Cash();  int GET\_TIME();  void STATUS();  void INSERT(int data);  int SEARCH(int data);  };  auto start = std::chrono::steady\_clock::now();  // BITS  string int2bitstr(int data) { return bitset<15>(data).to\_string(); }  int bitstr2int(string bits) { return (int)bitset<15>(bits).to\_ulong(); }  // ARRAY  int search(int item, int arr[], int size) {  for (int i = 0; i < size; i++)  if (arr[i] == item)  return i;  return -1;  }  int search\_min(int arr[], int size) {  int min\_idx = 0;  for (int i = 1; i < size; i++)  if (arr[i] < arr[min\_idx])  min\_idx = i;  return min\_idx;  }  // CASH  Cash::Cash() {  n\_block\_ = 8;  n\_ = 2;  m\_ = 32;  mem\_ = new int\*\* [n\_block\_];  flag\_ = new int\*\* [n\_block\_];  for (int i = 0; i < n\_block\_; i++) {  mem\_[i] = new int\* [n\_];  flag\_[i] = new int\* [n\_];  for (int j = 0; j < n\_; j++) {  mem\_[i][j] = new int [m\_];  flag\_[i][j] = new int[m\_];  for (int k = 0; k < m\_; k++) {  mem\_[i][j][k] = -1;  flag\_[i][j][k] = -1;  } } } }  Cash::~Cash() {  for (int i = 0; i < n\_block\_; i++) {  for (int j = 0; j < n\_; j++) {  delete[] mem\_[i][j];  delete[] flag\_[i][j];  }  delete[] mem\_[i];  delete[] flag\_[i];  }  delete[] mem\_;  delete[] flag\_;  }  int Cash::GET\_TIME() {  auto now = chrono::steady\_clock::now();  return (int)chrono::duration\_cast<chrono::milliseconds>(now - start).count();  }  void Cash::STATUS() {  for (int i = 0; i < 6 + m\_ \* 4; i++) cout << '=';  cout << endl << setw(4) << "BLK" << setw(2) << ' ';  for (int i = 0; i < m\_; i++) cout << setw(4) << i; cout << endl;  for (int i = 0; i < 6 + m\_ \* 4; i++) cout << '=';  for (int i = 0; i < n\_block\_; i++) {  cout << endl << setw(3) << i << ':' << setw(2) << '|';  for (int j = 0; j < n\_; j++) {  if (j != 0) cout << "\n" << setw(6) << '|';  for (int k = 0; k < m\_; k++)  if (mem\_[i][j][k] != -1) cout << setw(4) << mem\_[i][j][k];  else cout << setw(4) << ' ';  }  }  cout << endl;  for (int i = 0; i < 6 + m\_ \* 4; i++) cout << '=';  }  void Cash::INSERT(int data) {  if (SEARCH(data) != -1)  return;  string bits = int2bitstr(data);  int tag = bitstr2int(bits.substr(0, 7));  int blk = bitstr2int(bits.substr(7, 3));  int pad = bitstr2int(bits.substr(10, 5));  // Ищем пустое место  for (int i = 0; i < n\_; i++)  if (mem\_[blk][i][pad] == -1) {  mem\_[blk][i][pad] = tag;  flag\_[blk][i][pad] = GET\_TIME();  return;  }  // Если не нашли, ищем самые старые данные;  int str = 0;  for (int i = 1; i < n\_; i++)  if (flag\_[blk][i][pad] < flag\_[blk][str][pad]) str = i;  mem\_[blk][str][pad] = tag;  flag\_[blk][str][pad] = GET\_TIME();  }  int Cash::SEARCH(int data) {  string bits = int2bitstr(data);  int tag = bitstr2int(bits.substr(0, 7));  int blk = bitstr2int(bits.substr(7, 3));  int pad = bitstr2int(bits.substr(10, 5));  for (int i = 0; i < n\_; i++)  if (mem\_[blk][i][pad] == tag)  return i;  return -1;  } |

OUTPUT

|  |
| --- |
|  |