[팀프로젝트 2]

캐시 메모리 구현

과목 : 컴퓨터 구조

담당교수 : 김종완

제출일 : 2019.12.2(월)

컴퓨터공학과 20172029 김은영

컴퓨터공학과 20174706 김혜민

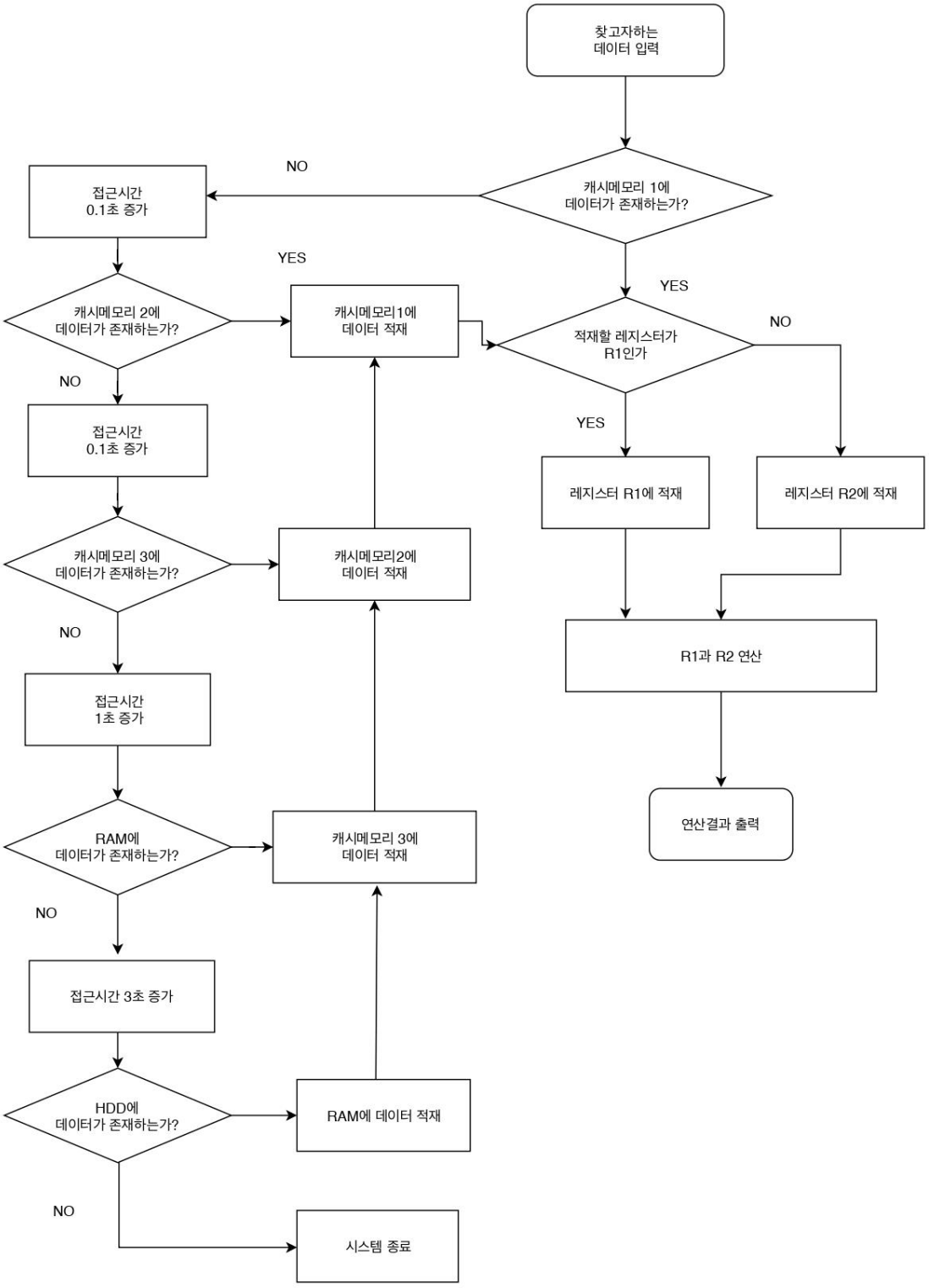
융합보안공학과 20171942 신예린

IT학부 20161140 정희재

IT학부 20161158 홍민영

〈목차〉

1. 프로그램 전체 개괄 순서도



2. 프로그램 기능 설명

-캐시 1개일 때 FIFO 교체정책

읽어오고 싶은 데이터가 캐시에 있는지 확인하고, 캐시에 존재하면 캐시에서 레지스터로 바로 적재한다. 소요되는 접근 시간은 0.1초이다. 캐시 적중을 의미한다.

캐시에 데이터가 없으면 램에 데이터가 있는지 확인하고, 램에 존재하면 램에서 캐시로 적재하고, 캐시에서 레지스터로 적재한다. 접근 소요시간은 0.1 + 1.0 = 1.1초이다. 캐시 미스 및 메모리 적중을 의미한다.

램에도 데이터가 없으면 하드디스크에 입력값이 있는지 확인하고, 하드디스크에 존재할 경우 램으로 적재, 램에서 캐시로 적재, 캐시에서 레지스터로 적재한다. 접근 소요시간은 1.1 + 3.0 = 4.1초이다. 캐시 미스 및 메모리 미스를 의미한다.

적재할 때 캐시 및 메모리의 용량을 초과하면 선입선출에 맞게 가장 먼저 적재된 데이터를 교체한다. 읽어온 두 데이터로 덧셈 연산을 한다.

-캐시 3개일 때 FIFO 교체정책

캐시를 L1, L2, L3 세 레벨로 나누어 각각 용량 차이 및 접근 속도 차이를 구현한다.

L1 접근 소요 시간은 0.1초, L2 접근 소요 시간은 0.2초, L3 접근 소요 시간은 0.3초, RAM 접근 소요시간은 1.3초, 하드디스크 접근 소요 시간은 4.3초이다.

-팀만의 특이점

교체방식을 무작위 교체방식으로 선정하여 캐시 및 메모리에 남은 용량이 없을 경우 무작위로 값을 교체한다. 또한 덧셈 외의 뺄셈, 곱셈, 나눗셈의 사칙연산을 구현하였다.

3. 성능 비교 결과

성능을 비교하기 위해 키보드로 원하는 데이터 입력을 받는 코드에서 random()으로 값을 받았고, 각 시나리오별로 일정 횟수 반복을 거쳤다.

1)캐시 1개일 때 선입선출 교체방식 성능 측정 결과

전체 메모리 참조 횟수	캐시 메모리1 적중횟수	캐시 메모리1 hit율	캐시 메모리1 miss율
1000	4	0.4	99.6
3000	5	0.166667	99.8333
5000	5	0.1	99.9
8000	9	0.1125	99.8875
10000	11	0.11	99.89

2)캐시 3개일 때 선입선출 교체방식 성능 측정 결과

전체 메모리 참조횟수	캐시1 적중 횟수	캐시1 hit율	캐시1 miss율	캐시2 적중횟수	캐시2 hit율	캐시2 miss율	캐시3 적중횟수	캐시3 hit율	캐시3 miss율
1000	1	0.1	99.9	3	0.3	99.7	27	2.7	97.3
3000	4	0.133	99.866	6	0.2	99.8	102	3.4	96.6
5000	2	0.04	99.96	15	0.3	99.7	195	3.9	96.1
8000	6	0.075	99.925	18	0.225	99.775	304	3.8	96.2
10000	9	0.09	99.91	30	0.3	99.7	364	3.64	96.36

캐시를 3개 두었을 경우 캐시가 1 개 일 때보다 하드디스크를 참조할 확률이 낮으며, 캐시가 1개 있는 것 보다 더 많은 값을 빠르게 참조할 수 있으므로 데이터에 접근하는데 소요하는 시간도 감소한다.

3)캐시 3개일 때 무작위 교체정책을 사용하였을 때 성능 측정 결과

전체 메모리 참조횟수	캐시1 적중횟수	캐시1 hit율	캐시1 miss율	캐시2 적중횟수	캐시2 hit율	캐시2 miss율	캐시3 적중횟수	캐시3 hit율	캐시3 miss율
1000	190	19	81	431	43.1	56.9	242	24.2	75.8
3000	638	21.26	78.733	689	22.96	77.033	619	20.63	79.366
5000	1635	32.7	67.3	316	6.32	93.68	230	4.6	95.4
8000	2685	33.56	66.437	307	3.837	96.162	170	2.125	97.875
10000	3353	33.53	66.47	217	2.17	97.83	179	1.79	98.21

의외로 선입선출 교체방식 보다 무작위 교체정책을 사용하였을 때 캐시 적중률이 높았다. 데이터 입력을 random()으로 받았기 때문에 실 사용에서도 동일한 성능을 보장할 수는 없지만, 같은 테스트 조건에서는 무작위 교체방식이 캐시 적중횟수가 더 높았다. 또한 선입선출 방식은 캐시 1보다는 캐시 2나 캐시3의 적중률이 높았는데, 무작위 교체방식은 캐시 1의 적중률 또한 매우 높았고, 전체 데이터 참조 횟수가 증가할 수록 캐시1 적중횟수와 히트율이 증가하는 것을 볼 수 있었다. 다만 무작위 교체방식은 새로운 데이터를 메모리 상 어느 위치에 적재할 지 계산 하는데 비용이 소비가 된다는 단점이 있다.

4. 시나리오별 코드 및 주석

〈1. 선입선출 교체& 캐시 메모리1개 코드 및 주석〉

```

#include <iostream>
using namespace std;

class Cache { //**BasicCache** 2개의 정수를 검색하고 덧셈하는 프로그램, 캐시1개

private:
	int HDD[5000],  //하드 디스크(5000*4바이트): 유일한 값 5000개 저장
		RAM[500] = { 0 },  //램(500*4바이트)
		L1[5] = { 0 },  //캐시1(5*4바이트)
		R1 = 0, //레지스터1(1*4바이트)
		R2 = 0, //레지스터2(1*4바이트)
		h = 0, r = 0, l1 = 0, l2 = 0, l3 = 0; //배열들 시작 인덱스
	bool check; //접근 성공 실패를 나타냄
	bool isRegister1 = 1; //레지스터1이면 1, 레지스터2면 0
	double hit1 = 0, hit2 = 0, hit3 = 0; //캐시1,2,3 적중 횟수
	double acc_time = 0; //접근 시간

public:
	Cache() {
		for (int i = 0; i < 5000; i++) HDD[i] = i + 1;//HDD의 내용 1~5000
	}

	bool check_cache1(int data) // 캐시1(L1)에 입력받은 데이터가 있는지 검사
	{
		for (int i = 0; i < 5; i++) {
			if (L1[i] == data) {
				check = true;
				cout << "!!Cache1 Hit!!" << endl;
				hit1++;
				break;
			}
			else check = false;
		}
		acc_time += 0.1;
		return check;
	}

	bool check_ram(int data) // 램에 입력받은 데이터가 있는지 검사
	{
		for (int i = 0; i < 500; i++) {
			if (RAM[i] == data) {
				check = true;
				cout << "data in RAM!" << endl;
				break;
			}
			else check = false;
		}
		acc_time += 1;
		return check;
	}

	bool check_hdd(int data) // 하드디스크에 입력받은 데이터가 있는지 검사
	{
		for (int i = 0; i < 5000; i++) {
			if (HDD[i] == data) {
				check = true;
				cout << "Data in Hdd!" << endl;
				break;
			}
		}
		acc_time += 3;
		return check;
	}
```

```
65
66          //********레지스터에 넣을 데이터 찾기********
67
68          void in_cache1(int data, bool isRegister1)
69          {
70                  if (check_cache1(data)) { //캐시에 데이터가 있으면 가져와서 레지스터1에 저장
71                          if (isRegister1)
72                                  R1 = data;
73                          else
74                                  R2 = data;
75                          //캐시 적중
76                  }
77                  else in_ram(data, isRegister1); //캐시에 데이터가 없으면 RAM을 검사하여 RAM에 저장하는 함수 호출
78          }
79
80          void in_ram(int data, bool isRegister1)
81          {
82                  if (check_ram(data)) { //램을 검사해서 데이터가 있으면 가져와서 캐시에 저장 -> 레지스터에 저장
83                          if (l1 >= 5)
84                                  l1 = 0;
85                          L1[l1] = data;
86
87
88                          if (isRegister1)
89                                  R1 = data;
90                          else
91                                  R2 = data;
92
93                          l1++;
94                  }
95                  else in_hdd(data, isRegister1); //램에 데이터가 없으면 하드디스크를 검사하여 램에 저장하는 함수 호출
96          }
97
98          void in_hdd(int data, bool isRegister1)
99          {
100                 if (check_hdd(data)) { //하드디스크를 검사해서 데이터가 있으면 가져와서 램에 저장 -> 캐쉬에 저장 -> 레지스터에 저장
101                         if (r >= 500) {
102                                 r = 0;
103
104                                 if (l1 >= 5) { // l1 인덱스 초과
105                                         l1 = 0;
106                                 }
107                         }
108                         else if (r < 500) {
109                                 if (l1 >= 5) { // l1 인덱스 초과
110                                         l1 = 0;
111                                 }
112                         }
113
114                         RAM[r] = data;
115                         L1[l1] = data;
116
117                         if (isRegister1)
118                                 R1 = data;
119                         else
120                                 R2 = data;
121
122                         r++; l1++;
123                         // 캐시실패, 메모리 실패
124                 }
125                 else error(); //하드디스크에도 원하는 데이터가 없으면 오류 문구 출력
126         }
127
128
129         void error()
130         {
131                 cout << "**접근 실패! 데이터 존재하지 않음!**" << endl;
132                 exit(0);//종료
```

```
133         }
134
135         void show_hit(double count)
136         {
137                 cout << "전체 메모리의 참조 횟수: " << count << endl;
138                 cout << "캐시 메모리의 적중 횟수: " << hit1 << endl;
139                 cout << "캐시 메모리 Hit률: " << (hit1 / count) * 100 << "%" << endl; // 히트률 = (적중 횟수)/(참조 횟수)
140                 cout << "캐시 메모리 Miss률: " << 100 - (hit1 / count) * 100 << "%" << endl << endl;
141
142         }
143
144         void show()
145         {
146                 /*cout << "<하드디스크>";
147                 for (int i = 0; i < 5000; i++)//HDD 값 출력
148                 {
149                         if (i % 16 == 0) cout << endl;
150                         cout << HDD[i] << " ";
151                 }*/
152                 cout << endl;
153                 cout << "<메인메모리>" << endl;
154                 for (int i = 0; i < 500; i++) cout << RAM[i] << " "; //RAM의 내용 출력
155
156                 cout << endl;
157                 cout << "<캐시메모리>" << endl;
158                 for (int i = 0; i < 5; i++) cout << L1[i] << " "; //Cache1의 내용 출력
159
160                 cout << endl;
161                 cout << "<레지스터1>" << endl; //연산 결과값 출력
162                 cout << R1 << endl;
163
164                 cout << "<레지스터2>" << endl; //연산 결과값 출력
165                 cout << R2 << endl;
166
167                 cout << "데이터 접근 시간: " << acc_time << "초" << endl;
168         }
169
170         void show_add_result() {
171                 cout << "두 정수의 덧셈 결과는 " << R1 + R2 << " 이다." << endl;
172         }
173 };
174
175 int main()
176 {
177         Cache c1;
178         int input1, input2, count = 0;
179
180         while (true) {
181                 cout << "읽고싶은 첫번째 데이터를 입력하시오(1이상의 정수)" << endl;
182                 cin >> input1;
183
184                 cout << endl;
185                 count++;
186
187                 c1.in_cache1(input1, 1);
188                 c1.show();
189                 c1.show_hit(count);
190
191                 cout << endl;
192
193
194                 cout << "읽고싶은 두번째 데이터를 입력하시오(1이상의 정수)" << endl;
195                 cin >> input2;
196
197                 cout << endl;
198                 count++;
199
200                 c1.in_cache1(input2, 0);
201                 c1.show();
202                 c1.show_hit(count);
203
204                 cout << endl;
205
206                 c1.show_add_result();
207
208                 cout << endl;
209         }
210
211         system("pause");
212         return 0;
213 }
```

〈2. 선입선출 교체방식& 캐시 메모리 3개 코드 및 주석〉

```cpp
#include <iostream>
using namespace std;

class Cache { //**BasicCache** 2개의 정수를 검색하고 덧셈하는 프로그램

private:
	int HDD[5000],  //하드 디스크(5000*4바이트): 유일한 값 5000개 저장
		RAM[500] = { 0 },  //램(500*4바이트)
		L1[5] = { 0 },  //캐시1(5*4바이트)
		L2[20] = { 0 },  //캐시2(20*4바이트)
		L3[200] = { 0 },  //캐시3(200*4바이트)
		R1 = 0, //레지스터1(1*4바이트)
		R2 = 0, //레지스터2(1*4바이트)
		h = 0, r = 0, l1 = 0, l2 = 0, l3 = 0; //배열들 시작 인덱스
	bool check; //접근 성공 실패를 나타냄
	bool isRegister1 = 1; //레지스터1이면 1, 레지스터2면 0
	double hit1 = 0, hit2 = 0, hit3 = 0; //캐시1,2,3 적중 횟수
	double acc_time = 0; //접근 시간

public:
	Cache() {
		for (int i = 0; i < 5000; i++) HDD[i] = i + 1;//HDD의 내용 1~5000
	}

	bool check_cache1(int data) // 캐시1(L1)에 입력받은 데이터가 있는지 검사
	{
		for (int i = 0; i < 5; i++) {
			if (L1[i] == data) {
				check = true;
				cout << "!!Cache1 Hit!!" << endl;
				hit1++;
				break;
			}
			else check = false;
		}
		acc_time += 0.1;
		return check;
	}

	bool check_cache2(int data) // 캐시2(L2)에 입력받은 데이터가 있는지 검사
	{
		for (int i = 0; i < 20; i++) {
			if (L2[i] == data) {
				check = true;
				cout << "!!Cache2 Hit!!" << endl;
				hit2++;
				break;
			}
			else check = false;
		}
		acc_time += 0.1;
		return check;
	}

	bool check_cache3(int data) // 캐시3(L3)에 입력받은 데이터가 있는지 검사
	{
		for (int i = 0; i < 200; i++) {
			if (L3[i] == data) {
				check = true;
				cout << "!!Cache3 Hit!!" << endl;
				hit3++;
				break;
			}
			else check = false;
		}
		acc_time += 0.1;
		return check;
	}

	bool check_ram(int data) // 램에 입력받은 데이터가 있는지 검사
	{
		for (int i = 0; i < 500; i++) {
			if (RAM[i] == data) {
```

```
                        check = true;
                        cout << "data in RAM!" << endl;
                        break;
                }
                else check = false;
        }
        acc_time += 1;
        return check;
}

bool check_hdd(int data) // 하드디스크에 입력받은 데이터가 있는지 검사
{
        for (int i = 0; i < 5000; i++) {
                if (HDD[i] == data) {
                        check = true;
                        cout << "Data in Hdd!" << endl;
                        break;
                }
        }
        acc_time += 3;
        return check;
}


//********레지스터에 넣을 데이터 찾기********

void in_cache1(int data, bool isRegister1)
{
        if (check_cache1(data)) { //캐시1에 데이터가 있으면 가져와서 레지스터에 저장
                if (isRegister1)
                        R1 = data;
                else
                        R2 = data;
                //캐시 적중
        }
        else in_cache2(data, isRegister1); //캐시1에 데이터가 없으면 캐시2를 검사하여 캐시에 저장하는 함수 호출
}

void in_cache2(int data, bool isRegister1)
{
        if (check_cache2(data)) { //캐시2에 데이터가 있으면 가져와서 레지스터에 저장
                if (l1 >= 5)
                        l1 = 0;
                L1[l1] = data;

                if (isRegister1)
                        R1 = data;
                else
                        R2 = data;

                l1++;
                //캐시 적중
        }
        else in_cache3(data, isRegister1); //캐시2에 데이터가 없으면 캐시3를 검사하여 캐시에 저장하는 함수 호출
}


void in_cache3(int data, bool isRegister1)
{
        if (check_cache3(data)) { //캐시3에 데이터가 있으면 가져와서 레지스터에 저장
                if (l1 >= 5)
                        l1 = 0;
                L1[l1] = data;

                if (l2 >= 20)
                        l2 = 0;
                L2[l2] = data;

                if (isRegister1)
                        R1 = data;
```

```
                else
                        R2 = data;

                l1++; l2++;
                //캐시 적중
        }
        else in_ram(data, isRegister1); //캐시3에 데이터가 없으면 RAM을 검사하여 RAM에 저장하는 함수 호출
}

void in_ram(int data, bool isRegister1)
{
        if (check_ram(data)) { //램을 검사해서 데이터가 있으면 가져와서 캐시에 저장 -> 레지스터에 저장
                if (l1 >= 5)
                        l1 = 0;
                L1[l1] = data;

                if (l2 >= 20)
                        l2 = 0;
                L2[l2] = data;

                if (l3 >= 200)
                        l3 = 0;
                L3[l3] = data;

                if (isRegister1)
                        R1 = data;
                else
                        R2 = data;

                l1++; l2++; l3++;
        }
        else in_hdd(data, isRegister1); //램에 데이터가 없으면 하드디스크를 검사하여 램에 저장하는 함수 호출
}

void in_hdd(int data, bool isRegister1)
{
        if (check_hdd(data)) { //하드디스크를 검사해서 데이터가 있으면 가져와서 램에 저장 -> 캐쉬에 저장 -> 레지스터에 저장
                if (r >= 500) {
                        r = 0;

                        if (l1 >= 5) { // l1 인덱스 초과
                                l1 = 0;

                                if (l2 >= 20) { //l2 인덱스 초과
                                        l2 = 0;

                                        if (l3 >= 200) // l3 인덱스 초과
                                                l3 = 0;
                                }
                                else if (l2 < 20) { //l2 인덱스 초과 아님
                                        if (l3 >= 200) // l3 인덱스 초과
                                                l3 = 0;
                                }
                        }

                        else if (l1 < 5) { // l1 인덱스 초과 아님
                                if (l2 >= 20) { // l2 인덱스 초과
                                        l2 = 0;

                                        if (l3 >= 200) //l3 인덱스 초과
                                                l3 = 0;
                                }
                                else if (l2 < 20) { //l2 인덱스 초과 아님
                                        if (l3 >= 200) //l3 인덱스 초과
                                                l3 = 0;
                                }
                        }
                }

                else if (r < 500) {
                        if (l1 >= 5) { // l1 인덱스 초과
                                l1 = 0;

                                if (l2 >= 20) { //l2 인덱스 초과
                                        l2 = 0;

                                        if (l3 >= 200) // l3 인덱스 초과
                                                l3 = 0;
                                }
                                else if (l2 < 19) { //l2 인덱스 초과 아님
                                        if (l3 >= 200) // l3 인덱스 초과
                                                l3 = 0;
                                }
                        }
```

```
227
228                             else if (l1 < 5) { // l1 인덱스 초과 아님
229                                     if (l2 >= 20) { // l2 인덱스 초과
230                                             l2 = 0;
231
232                                             if (l3 >= 200) //l3 인덱스 초과
233                                                     l3 = 0;
234                                     }
235                                     else if (l2 < 20) { //l2 인덱스 초과 아님
236                                             if (l3 >= 200) //l3 인덱스 초과
237                                                     l3 = 0;
238                                     }
239                             }
240                     }
241
242                     RAM[r] = data;
243                     L1[l1] = data;
244                     L2[l2] = data;
245                     L3[l3] = data;
246
247                     if (isRegister1)
248                             R1 = data;
249                     else
250                             R2 = data;
251
252                     r++; l1++; l2++; l3++;
253                     // 캐시실패, 메모리 실패
254             }
255             else error(); //하드디스크에도 원하는 데이터가 없으면 오류 문구 출력
256     }
257
258
259     void error()
260     {
261             cout << "**접근 실패! 데이터 존재하지 않음!**" << endl;
262             exit(0);//종료
263     }
264
265     void show_hit(double count)
266     {
267             cout << "전체 메모리의 참조 횟수: " << count << endl;
268             cout << "캐시 메모리1의 적중 횟수: " << hit1 << endl;
269             cout << "캐시 메모리1 Hit률: " << (hit1 / count) * 100 << "%" << endl;
270             // 히트률 = (적중 횟수)/(참조 횟수)
271             cout << "캐시 메모리1 Miss률: " << 100 - (hit1 / count) * 100 << "%" << endl << endl;
272
273             cout << "캐시 메모리2의 적중 횟수: " << hit2 << endl;
274             cout << "캐시 메모리2 Hit률: " << (hit2 / count) * 100 << "%" << endl;
275              // 히트률 = (적중 횟수)/(참조 횟수)
276             cout << "캐시 메모리1 Miss률: " << 100 - (hit2 / count) * 100 << "%" << endl << endl;
277
278             cout << "캐시 메모리3의 적중 횟수: " << hit3 << endl;
279             cout << "캐시 메모리3 Hit률: " << (hit3 / count) * 100 << "%" << endl;
280             // 히트률 = (적중 횟수)/(참조 횟수)
281             cout << "캐시 메모리1 Miss률: " << 100 - (hit3 / count) * 100 << "%" << endl << endl;
282     }
283
284     void show()
285     {
286             /*cout << "<하드디스크>";
287             for (int i = 0; i < 5000; i++)//HDD 값 출력
288             {
289                     if (i % 16 == 0) cout << endl;
290                     cout << HDD[i] << " ";
291             }*/
292             cout << endl;
293             cout << "<메인메모리>" << endl;
294             for (int i = 0; i < 500; i++) cout << RAM[i] << " "; //RAM의 내용 출력
295
296             cout << endl;
297             cout << "<캐시메모리3>" << endl;
298             for (int i = 0; i < 200; i++) cout << L3[i] << " "; //Cache3의 내용 출력
299
300             cout << endl;
301             cout << "<캐시메모리2>" << endl;
```

```
302                 for (int i = 0; i < 20; i++) cout << L2[i] << " "; //Cache2의 내용 출력
303
304                 cout << endl;
305                 cout << "<캐시메모리1>" << endl;
306                 for (int i = 0; i < 5; i++) cout << L1[i] << " "; //Cache1의 내용 출력
307
308                 cout << endl;
309                 cout << "<레지스터1>" << endl; //연산 결과값 출력
310                 cout << R1 << endl;
311
312                 cout << "<레지스터2>" << endl; //연산 결과값 출력
313                 cout << R2 << endl;
314
315                 cout << "데이터 접근 시간: " << acc_time << "초" << endl;
316         }
317
318         void show_add_result() {
319                 cout << "두 정수의 덧셈 결과는 " << R1 + R2 << " 이다." << endl;
320         }
321 };
322
323 int main()
324 {
325         Cache c1;
326         int input1, input2, count = 0;
327
328         while (true) {
329                 cout << "원고싶은 첫번째 데이터를 입력하시오(1이상의 정수)" << endl;
330                 cin >> input1;
331
332                 cout << endl;
333                 count++;
334
335                 c1.in_cache1(input1, 1);
336                 c1.show();
337                 c1.show_hit(count);
338
339                 cout << endl;
340
341
342                 cout << "원고싶은 두번째 데이터를 입력하시오(1이상의 정수)" << endl;
343                 cin >> input2;
344
345                 cout << endl;
346                 count++;
347
348                 c1.in_cache1(input2, 0);
349                 c1.show();
350                 c1.show_hit(count);
351
352                 cout << endl;
353
354                 c1.show_add_result();
355
356                 cout << endl;
357         }
358
359         system("pause");
360         return 0;
361 }
```

〈3. 팀의 특이점 코드 및 주석 (**무작위 교체방식& 사칙연산 수행**)〉

```
#include <iostream>
#include <ctime>
#include <cstdlib>
using namespace std;

class Cache { //**BasicCache** 2개의 정수를 검색하고 사칙연산하는 프로그램.
 //빈 공간이 없을 때 데이터를 무작위로 선정하고 해당 위치에 저장한다. 그리고 덧셈, 뺄셈, 곱셈, 나눗셈 사칙연산을 추가했다.
private:
	int HDD[5000],  //하드 디스크(5000*4바이트): 유일한 값 5000개 저장
		RAM[500] = { 0 },  //램(500*4바이트)
		L1[5] = { 0 },  //캐시1(5*4바이트)
		L2[20] = { 0 },  //캐시2(20*4바이트)
		L3[200] = { 0 },  //캐시3(200*4바이트)
		R1 = 0, //레지스터1(1*4바이트)
		R2 = 0, //레지스터2(1*4바이트)
		h = 0, r = 0, l1 = 0, l2 = 0, l3 = 0; //배열들 시작 인덱스
	bool check; //접근 성공 실패를 나타냄
	bool isRegister1 = 1; //레지스터1이면 1, 레지스터2면 0
	double hit1 = 0, hit2 = 0, hit3 = 0; //캐시1,2,3 적중 횟수
	double acc_time = 0; //접근 시간

public:
	Cache() {
		for (int i = 0; i < 5000; i++) HDD[i] = i + 1;//HDD의 내용 1~5000
	}
	void reset_acc_time() {
		acc_time = 0;
	}
	bool check_cache1(int data) // 캐시1(L1)에 입력받은 데이터가 있는지 검사
	{
		for (int i = 0; i < 5; i++) {
			if (L1[i] == data) {
				check = true;
				cout << "!!Cache1 Hit!!" << endl;
				hit1++;
				break;
			}
			else check = false;
		}
		acc_time += 0.1;
		return check;
	}

	bool check_cache2(int data) // 캐시2(L2)에 입력받은 데이터가 있는지 검사
	{
		for (int i = 0; i < 20; i++) {
			if (L2[i] == data) {
				check = true;
				cout << "!!Cache2 Hit!!" << endl;
				hit2++;
				break;
			}
			else check = false;
		}
		acc_time += 0.1;
		return check;
	}

	bool check_cache3(int data) // 캐시3(L3)에 입력받은 데이터가 있는지 검사
	{
		for (int i = 0; i < 200; i++) {
			if (L3[i] == data) {
				check = true;
				cout << "!!Cache3 Hit!!" << endl;
				hit3++;
				break;
			}
			else check = false;
		}
		acc_time += 0.1;
		return check;
	}
```

```
73
74      bool check_ram(int data) // 램에 입력받은 데이터가 있는지 검사
75      {
76              for (int i = 0; i < 500; i++) {
77                      if (RAM[i] == data) {
78                              check = true;
79                              cout << "data in RAM!" << endl;
80                              break;
81                      }
82                      else check = false;
83              }
84              acc_time += 1;
85              return check;
86      }
87
88      bool check_hdd(int data) // 하드디스크에 입력받은 데이터가 있는지 검사
89      {
90              for (int i = 0; i < 5000; i++) {
91                      if (HDD[i] == data) {
92                              check = true;
93                              cout << "Data in Hdd!" << endl;
94                              break;
95                      }
96              }
97              acc_time += 3;
98              return check;
99      }
100
101
102     //********레지스터에 넣을 데이터 찾기********
103
104     void in_cache1(int data, bool isRegister1)
105     {
106             if (check_cache1(data)) { //캐시1에 데이터가 있으면 가져와서 레지스터에 저장
107                     if (isRegister1)
108                             R1 = data;
109                     else
110                             R2 = data;
111                     //캐시 적중
112             }
113             else in_cache2(data, isRegister1); //캐시1에 데이터가 없으면 캐시2를 검사하여 캐시에 저장하는 함수 호출
114     }
115
116     void in_cache2(int data, bool isRegister1)
117     {
118             if (check_cache2(data)) { //캐시2에 데이터가 있으면 가져와서 레지스터에 저장
119                     if (l1 >= 5) { // l1 인덱스 초과
120                             l1 = rand() % 5;
121                             L1[l1] = data;
122                             l1 += 5; // if 조건에 만족해야 난수를 생성하기 때문에 조건에 맞추기 위해 알맞은 값을 더한다.
123                     }
124                     else {
125                             L1[l1] = data;
126                             l1++;
127                     }
128
129                     if (isRegister1)
130                             R1 = data;
131                     else
132                             R2 = data;
133                     //캐시 적중
134             }
135             else in_cache3(data, isRegister1); //캐시2에 데이터가 없으면 캐시3를 검사하여 캐시에 저장하는 함수 호출
136     }
```

```
140     void in_cache3(int data, bool isRegister1)
141     {
142         if (check_cache3(data)) { //캐시3에 데이터가 있으면 가져와서 레지스터에 저장
143             if (l1 >= 5) { // l1 인덱스 초과
144                 l1 = rand() % 5;
145                 L1[l1] = data;
146                 l1 += 5; // if 조건에 만족해야 난수를 생성하기 때문에 조건에 맞추기 위해 알맞은 값을 더한다.
147             }
148             else {
149                 L1[l1] = data;
150                 l1++;
151             }
152 
153             if (l2 >= 20) { // l2 인덱스 초과
154                 l2 = rand() % 20;
155                 L2[l2] = data;
156                 l2 += 20;
157             }
158             else {
159                 L2[l2] = data;
160                 l2++;
161             }
162 
163             if (isRegister1)
164                 R1 = data;
165             else
166                 R2 = data;
167             //캐시 적중
168         }
169         else in_ram(data, isRegister1); //캐시3에 데이터가 없으면 램을 검사하여 캐시에 적재하는 함수 호출
170     }
171 
```

```
172     void in_ram(int data, bool isRegister1)
173     {
174         srand(time(0));
175 
176         if (check_ram(data)) { //램을 검사해서 데이터가 있으면 가져와서 캐시에 저장 -> 레지스터에 저장
177             if (l1 >= 5) { // l1 인덱스 초과
178                 l1 = rand() % 5;
179                 L1[l1] = data;
180                 l1 += 5; // if 조건에 만족해야 난수를 생성하기 때문에 조건에 맞추기 위해 알맞은 값을 더한다.
181             }
182             else {
183                 L1[l1] = data;
184                 l1++;
185             }
186 
187             if (l2 >= 20) { // l2 인덱스 초과
188                 l2 = rand() % 20;
189                 L2[l2] = data;
190                 l2 += 20;
191             }
192             else {
193                 L2[l2] = data;
194                 l2++;
195             }
196 
197             if (l3 >= 200) { // l3 인덱스 초과
198                 l3 = rand() % 200;
199                 L3[l3] = data;
200                 l3 += 200;
201             }
202             else {
203                 L3[l3] = data;
204                 l3++;
205             }
206 
207             if (isRegister1)
208                 R1 = data;
209            else
210                R2 = data;
211        }
212        else in_hdd(data, isRegister1); //램에 데이터가 없으면 하드디스크를 검사하여 램에 저장하는 함수 호출
213    }
```

```
218
219         void in_hdd(int data, bool isRegister1)
220         {
221                 srand(time(0));
222
223                 if (check_hdd(data)) {
224                    //하드디스크를 검사해서 데이터가 있으면 가져와서 램에 저장 -> 캐쉬에 저장 -> 레지스터에 저장
225                         if (r >= 500) { //ram 인덱스 초과
226                                 r = rand() % 500;
227                                 RAM[r] = data;
228                                 r += 500;
229                    // if 조건에 만족해야 난수를 생성하기 때문에 조건에 맞추기 위해 말맞은 값을 더한다.
230
231                                 if (l1 >= 5) { // l1 인덱스 초과
232                                         l1 = rand() % 5;
233                                         L1[l1] = data;
234                                         l1 += 5;
235
236                                         if (l2 >= 20) { //l2 인덱스 초과
237                                                 l2 = rand() % 20;
238                                                 L2[l2] = data;
239                                                 l2 += 20;
240
241                                                 if (l3 >= 200) { // l3 인덱스 초과
242                                                         l3 = rand() % 200;
243                                                         L3[l3] = data;
244                                                         l3 += 200;
245                                                 }
246                                                 else { // l3 인덱스 초과 아님
247                                                         L3[l3] = data;
248                                                         l3++;
249                                                 }
250
251                                         }
252                                         else { //l2 인덱스 초과 아님
253                                                 L2[l2] = data;
254                                                 l2++;
255
256                                                 if (l3 >= 200) {// l3 인덱스 초과
257                                                         l3 = rand() % 200;
258                                                         L3[l3] = data;
259                                                         l3 += 200;
260                                                 }
261                                                 else { // l3 인덱스 초과 아님
262                                                         L3[l3] = data;
263                                                         l3++;
264                                                 }
265                                         }
266                                 }
267
268                                 else { // l1 인덱스 초과 아님
269                                         L1[l1] = data;
270                                         l1++;
271
272                                         if (l2 >= 20) { // l2 인덱스 초과
273                                                 l2 = rand() % 20;
274                                                 L2[l2] = data;
275                                                 l2 += 20;
276
```

```
277                                             if (l3 >= 200) {//l3 인덱스 초과
278                                                     l3 = rand() % 200;
279                                                     L3[l3] = data;
280                                                     l3 += 200;
281                                             }
282                                             else { // l3 인덱스 초과 아님
283                                                     L3[l3] = data;
284                                                     l3++;
285                                             }
286                                     }
287                                     else { //l2 인덱스 초과 아님
288                                             L2[l2] = data;
289                                             l2++;
290
291                                             if (l3 >= 200) { //l3 인덱스 초과
292                                                     l3 = rand() % 200;
293                                                     L3[l3] = data;
294                                                     l3 += 200;
295                                             }
296                                             else {// l3 인덱스 초과 아님
297                                                     L3[l3] = data;
298                                                     l3++;
299                                             }
300                                     }
301                             }
302                     }
303                     else { //ram 인덱스 초과 아님
304                             RAM[r] = data;
305                             r++;
306
307                             if (l1 >= 5) { // l1 인덱스 초과
308                                     l1 = rand() % 5;
309                                     L1[l1] = data;
310                                     l1 += 5;
311
312                                     if (l2 >= 20) { //l2 인덱스 초과
313                                             l2 = rand() % 20;
314                                             L2[l2] = data;
315                                             l2 += 20;
316
317                                             if (l3 >= 200) { // l3 인덱스 초과
318                                                     l3 = rand() % 200;
319                                                     L3[l3] = data;
320                                                     l3 += 200;
321                                             }
322                                             else {// l3 인덱스 초과 아님
323                                                     L3[l3] = data;
324                                                     l3++;
325                                             }
326                                     }
327                                     else { //l2 인덱스 초과 아님
328                                             L2[l2] = data;
329                                             l2++;
330
331                                             if (l3 >= 200) {// l3 인덱스 초과
332                                                     l3 = rand() % 200;
333                                                     L3[l3] = data;
334                                                     l3 += 200;
335                                             }
336                                             else {// l3 인덱스 초과 아님
337                                                     L3[l3] = data;
338                                                     l3++;
339                                             }
340                                     }
341                             }
342
343                             else { // l1 인덱스 초과 아님
344                                     L1[l1] = data;
345                                     l1++;
346
347                                     if (l2 >= 20) { // l2 인덱스 초과
348                                             l2 = rand() % 20;
349                                             L2[l2] = data;
350                                             l2 += 20;
351
```

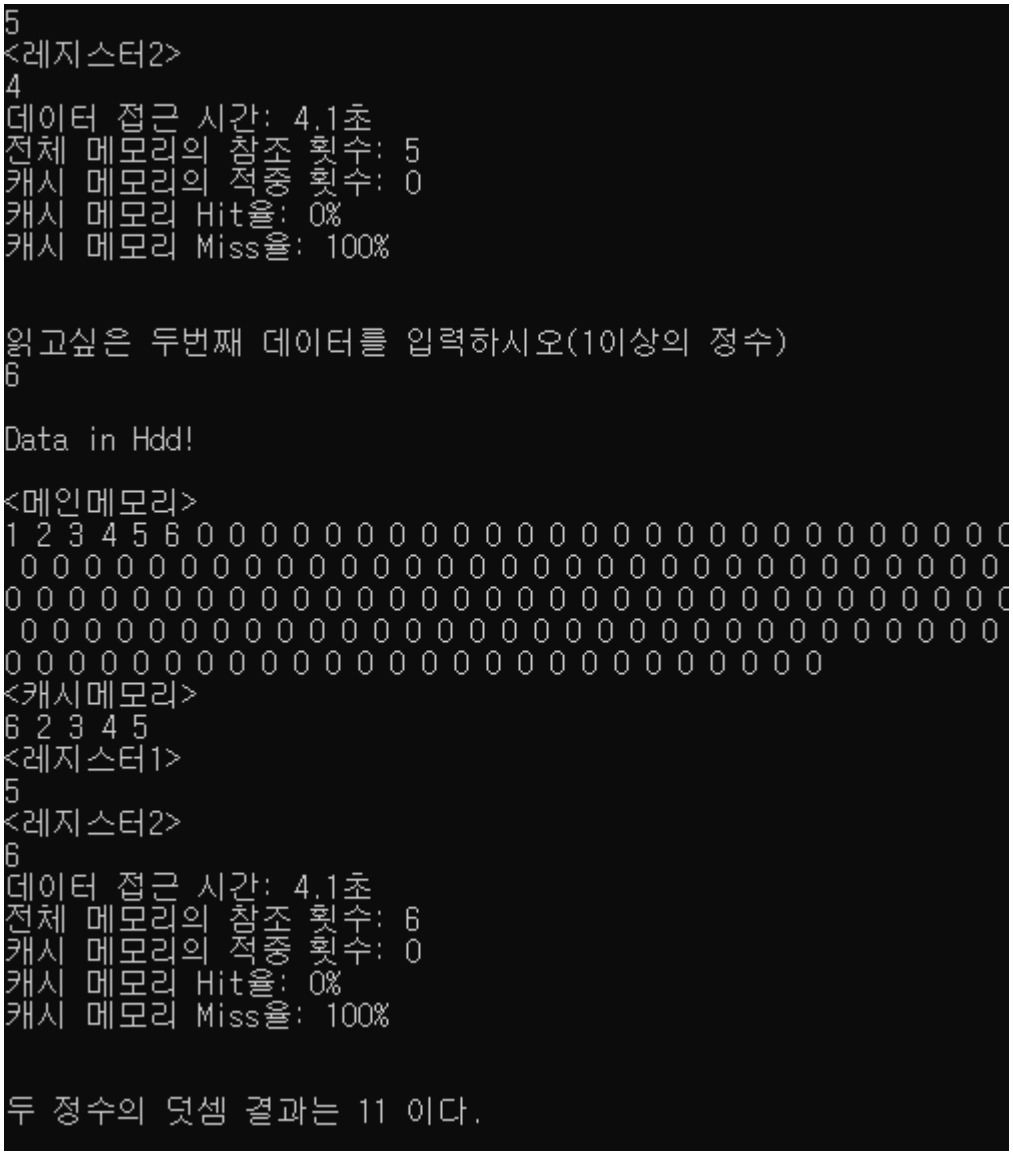
```
352                                             l3++;
353                                     }
354                             }
355                             else { //l2 인덱스 초과 아님
356                                     L2[l2] = data;
357                                     l2++;
358
359                                     if (l3 >= 200) { //l3 인덱스 초과
360                                             l3 = rand() % 200;
361                                             L3[l3] = data;
362                                             l3 += 200;
363                                     }
364                                     else {// l3 인덱스 초과 아님
365                                             L3[l3] = data;
366                                             l3++;
367                                     }
368                             }
369                     }
370             }
371
372             if (isRegister1)
373                     R1 = data;
374             else
375                     R2 = data;
376             // 캐시실패, 메모리 실패
377     }
378     else error(); //하드디스크에도 원하는 데이터가 없으면 오류 문구 출력
379 }
380
381
382 void error()
383 {
384     cout << "**접근 실패! 데이터 존재하지 않음!**" << endl;
385     exit(0);//종료
386 }
387
388 void show_hit(double count)
389 {
390     cout << "전체 메모리의 참조 횟수: " << count << endl;
391     cout << "캐시 메모리1의 적중 횟수: " << hit1 << endl;
392     cout << "캐시 메모리1 Hit률: " << (hit1 / count) * 100 << "%" << endl;
393     // 히트률 = (적중 횟수)/(참조 횟수)
394     cout << "캐시 메모리1 Miss률: " << 100 - (hit1 / count) * 100 << "%" << endl << endl;
395
396     cout << "캐시 메모리2의 적중 횟수: " << hit2 << endl;
397     cout << "캐시 메모리2 Hit률: " << (hit2 / count) * 100 << "%" << endl;
398      // 히트률 = (적중 횟수)/(참조 횟수)
399     cout << "캐시 메모리1 Miss률: " << 100 - (hit2 / count) * 100 << "%" << endl << endl;
400
401     cout << "캐시 메모리3의 적중 횟수: " << hit3 << endl;
402     cout << "캐시 메모리3 Hit률: " << (hit3 / count) * 100 << "%" << endl;
403     // 히트률 = (적중 횟수)/(참조 횟수)
404     cout << "캐시 메모리1 Miss률: " << 100 - (hit3 / count) * 100 << "%" << endl << endl;
405 }
406
407 void show()
408 {
409     /*cout << "<하드디스크>";
410     for (int i = 0; i < 5000; i++)//HDD 값 출력
411     {
412             if (i % 16 == 0) cout << endl;
413             cout << HDD[i] << " ";
414     }*/
415     cout << endl;
416     cout << "<메인메모리>" << endl;
417     for (int i = 0; i < 500; i++) cout << RAM[i] << " "; //RAM의 내용 출력
418
419     cout << endl;
420     cout << "<캐시메모리3>" << endl;
421     for (int i = 0; i < 200; i++) cout << L3[i] << " "; //Cache3의 내용 출력
422
423     cout << endl;
424     cout << "<캐시메모리2>" << endl;
```

```
424                 cout << "<캐시메모리2>" << endl;
425                 for (int i = 0; i < 20; i++) cout << L2[i] << " "; //Cache2의 내용 출력
426
427                 cout << endl;
428                 cout << "<캐시메모리1>" << endl;
429                 for (int i = 0; i < 5; i++) cout << L1[i] << " "; //Cache1의 내용 출력
430
431                 cout << endl;
432                 cout << "<레지스터1>" << endl; //연산 결과값 출력
433                 cout << R1 << endl;
434
435                 cout << "<레지스터2>" << endl; //연산 결과값 출력
436                 cout << R2 << endl;
437
438                 cout << "데이터 접근 시간: " << acc_time << "초" << endl;
439         }
440
441         void show_calculation_result(int cal) {
442                 if (cal == 1)
443                         cout << "두 정수의 덧셈 결과는 " << R1 + R2 << " 이다." << endl;
444                 else if (cal == 2)
445                         cout << "두 정수의 뺄셈 결과는 " << R1 - R2 << " 이다." << endl;
446                 else if (cal == 3)
447                         cout << "두 정수의 곱셈 결과는 " << R1 * R2 << " 이다." << endl;
448                 else if (cal == 4)
449                         cout << "두 정수의 나눗셈 결과는 " << static_cast<double>(R1) / R2 << " 이다." << endl;
450         }
451 };
452
453 int main()
454 {
455         Cache c1;
456         int input1, input2, count = 0, calculation;
457
458         while (true) {
459                 cout << "읽고싶은 첫번째 데이터를 입력하시오(10이상의 정수)" << endl;
460                 cin >> input1;
461
462                 cout << endl;
463                 count++;
464                 c1.reset_acc_time();
465                 c1.in_cache1(input1, 1);
466                 c1.show();
467                 c1.show_hit(count);
468
469                 cout << endl;
470
471
472                 cout << "읽고싶은 두번째 데이터를 입력하시오(10이상의 정수)" << endl;
473                 cin >> input2;
474
475                 cout << endl;
476                 count++;
477                 c1.reset_acc_time();
478                 c1.in_cache1(input2, 0);
479                 c1.show();
480                 c1.show_hit(count);
481
482                 cout << endl;
483
484                 cout << "연산 하고자 하는 사칙 연산의 숫자를 입력하시오. (덧셈 : 1, 뺄셈 : 2, 곱셈 : 3, 나눗셈 : 4)" << endl;
485                 cin >> calculation;
486
487                 c1.show_calculation_result(calculation);
488
489                 cout << endl;
490         }
491
492         system("pause");
493         return 0;
494 }
```

5.실행결과 ScreenShot & 실행 환경 설명

[기본 1. FIFO 교체 방식과 캐시 메모리 1개& 덧셈 연산]

1-5까지의 데이터를 캐시에 적재한 후 FIFO 교체를 보여주기 위해 새로운 데이터를 입력한 실행 화면.

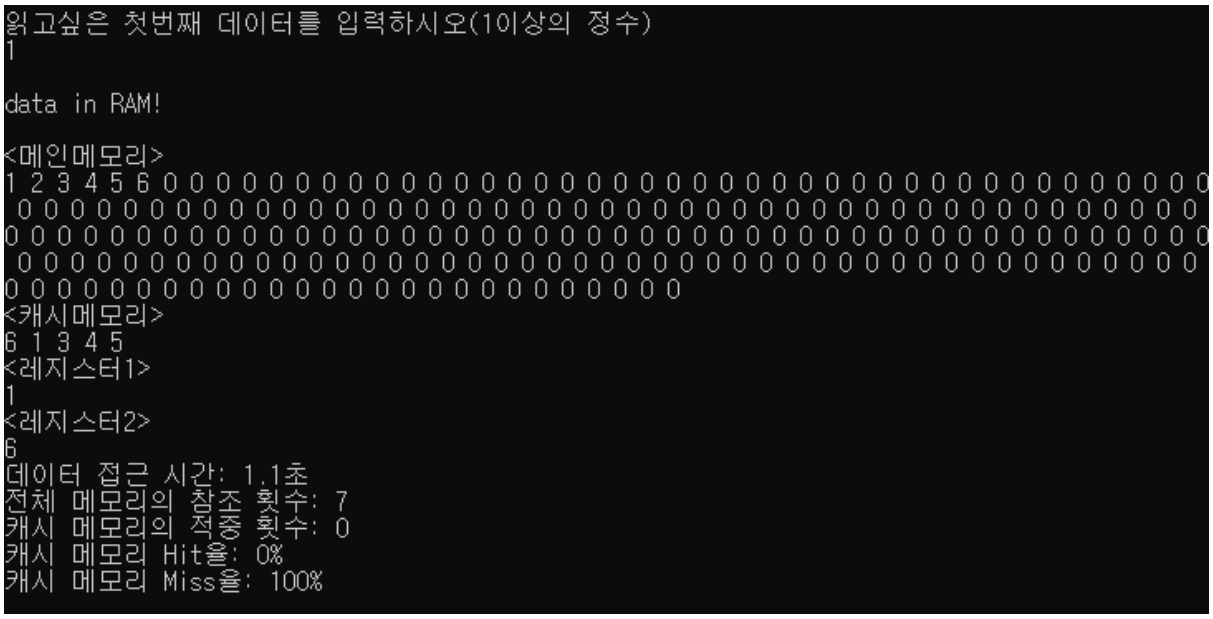


-캐시메모리의 용량의 2배를 입력하여 메인메모리에도 없는 데이터 찾기 위해 HDD 접근 시 4.1초, 캐시메모리에 없어서 메인메모리에 접근 시 1.1초, 캐시메모리에 데이터 존재하여 캐시메모리에 접근 시 0.1초 소요되는 것을 볼 수 있다.

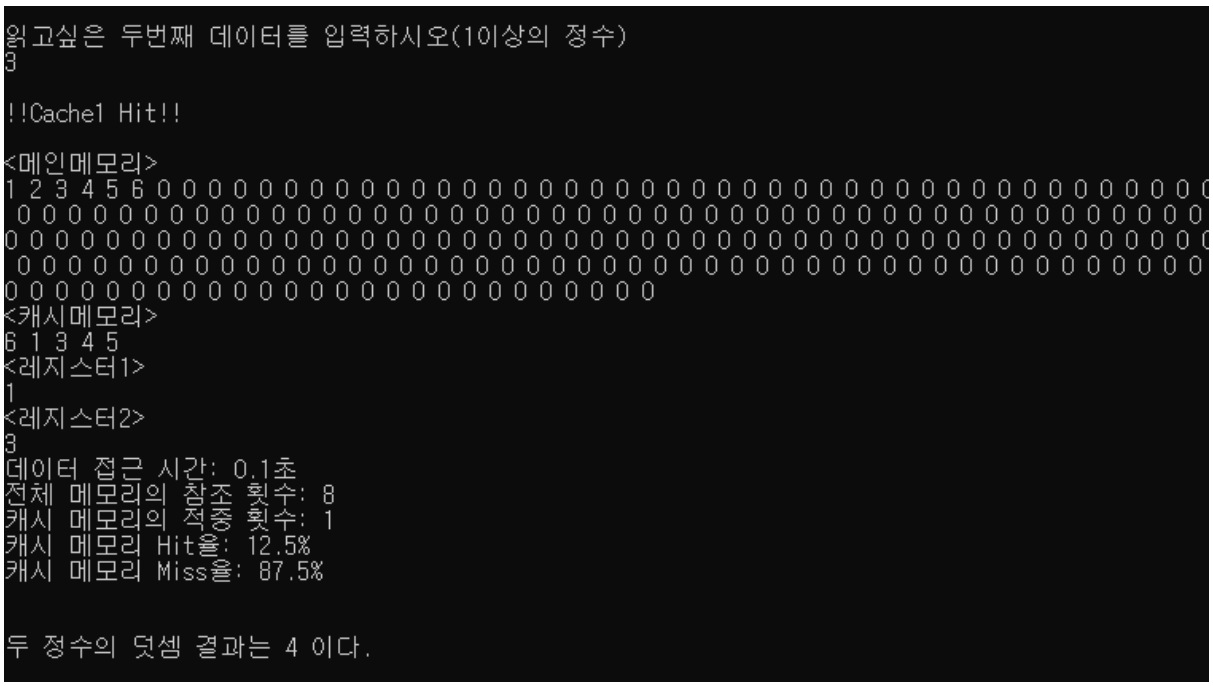
```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
<캐시메모리>
11 7 8 9 10
<레지스터1>
11
<레지스터2>
10
데이터 접근 시간: 4.1초
전체 메모리의 참조 횟수: 11
캐시 메모리의 적중 횟수: 0
캐시 메모리 Hit율: 0%
캐시 메모리 Miss율: 100%


읽고싶은 두번째 데이터를 입력하시오(1이상의 정수)
1

data in RAM!

<메인메모리>
1 2 3 4 5 6 7 8 9 10 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
<캐시메모리>
11 1 8 9 10
<레지스터1>
11
<레지스터2>
1
데이터 접근 시간: 1.1초
전체 메모리의 참조 횟수: 12
캐시 메모리의 적중 횟수: 0
캐시 메모리 Hit율: 0%
캐시 메모리 Miss율: 100%


두 정수의 덧셈 결과는 12 이다.

읽고싶은 첫번째 데이터를 입력하시오(1이상의 정수)
10

!!Cache1 Hit!!

<메인메모리>
1 2 3 4 5 6 7 8 9 10 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
<캐시메모리>
11 1 8 9 10
<레지스터1>
10
<레지스터2>
1
데이터 접근 시간: 0.1초
전체 메모리의 참조 횟수: 13
캐시 메모리의 적중 횟수: 1
캐시 메모리 Hit율: 7.69231%
캐시 메모리 Miss율: 92.3077%
```

-선입선출 교체정책에 의해 캐시 메모리에서 교체되었던 값 입력 시 RAM 접근 시간과

cache miss 확인 실행 화면



-캐시 메모리에 존재하는 숫자 데이터의 접근 시간 및 cache hit 확인 화면



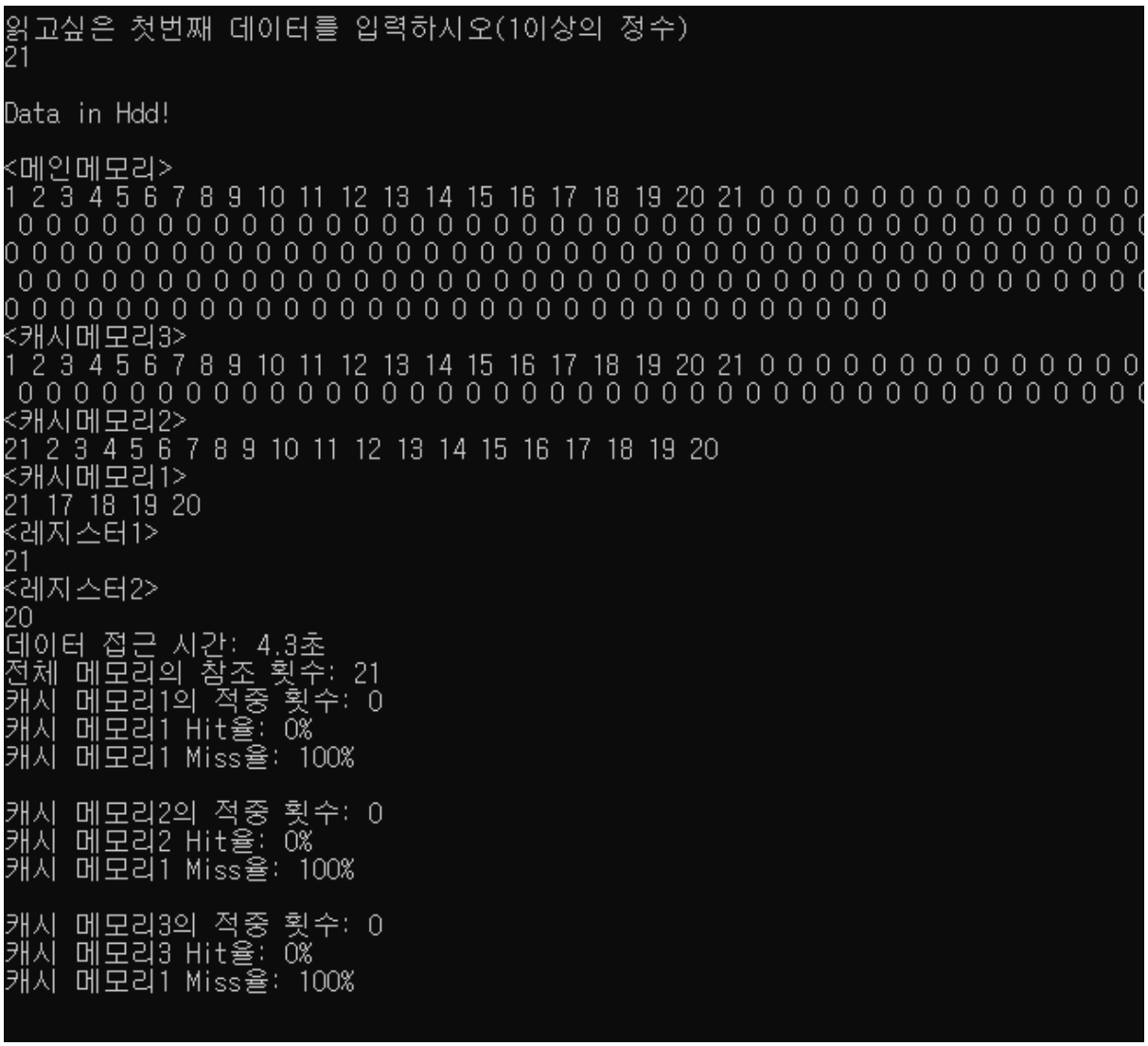
[기본 2. FIFO 교체 방식과 캐시 메모리 3개& 덧셈 연산]

-선입선출 보이기 위해1-21까지의 수를 입력.

캐시 메모리1은 용량을 넘은 6부터 선입선출 교체 방식에 따라 작동된다.

캐시 메모리2 역시 용량을 넘은 21 부터 선입선출 교체 방식으로 작동한다.

숫자 두 개 입력시 자동적으로 덧셈 연산 실행.



```
읽고싶은 첫번째 데이터를 입력하시오(1이상의 정수)
21

Data in Hdd!

<메인메모리>
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
<캐시메모리3>
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
<캐시메모리2>
21 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
<캐시메모리1>
21 17 18 19 20
<레지스터1>
21
<레지스터2>
20
데이터 접근 시간: 4.3초
전체 메모리의 참조 횟수: 21
캐시 메모리1의 적중 횟수: 0
캐시 메모리1 Hit율: 0%
캐시 메모리1 Miss율: 100%

캐시 메모리2의 적중 횟수: 0
캐시 메모리2 Hit율: 0%
캐시 메모리1 Miss율: 100%

캐시 메모리3의 적중 횟수: 0
캐시 메모리3 Hit율: 0%
캐시 메모리1 Miss율: 100%
```

-1 입력시 성능과 cache hit율

```
읽고싶은 두번째 데이터를 입력하시오(1이상의 정수)
1

!!Cache3 Hit!!

<메인메모리>
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
<캐시메모리3>
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
<캐시메모리2>
21 1 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
<캐시메모리1>
21 1 18 19 20
<레지스터1>
21
<레지스터2>
1
데이터 접근 시간: 0.3초
전체 메모리의 참조 횟수: 22
캐시 메모리1의 적중 횟수: 0
캐시 메모리1 Hit율: 0%
캐시 메모리1 Miss율: 100%

캐시 메모리2의 적중 횟수: 0
캐시 메모리2 Hit율: 0%
캐시 메모리1 Miss율: 100%

캐시 메모리3의 적중 횟수: 1
캐시 메모리3 Hit율: 4.54545%
캐시 메모리1 Miss율: 95.4545%


두 정수의 덧셈 결과는 22 이다.

읽고싶은 첫번째 데이터를 입력하시오(1이상의 정수)
```

-21 입력시 성능과 cache hit율

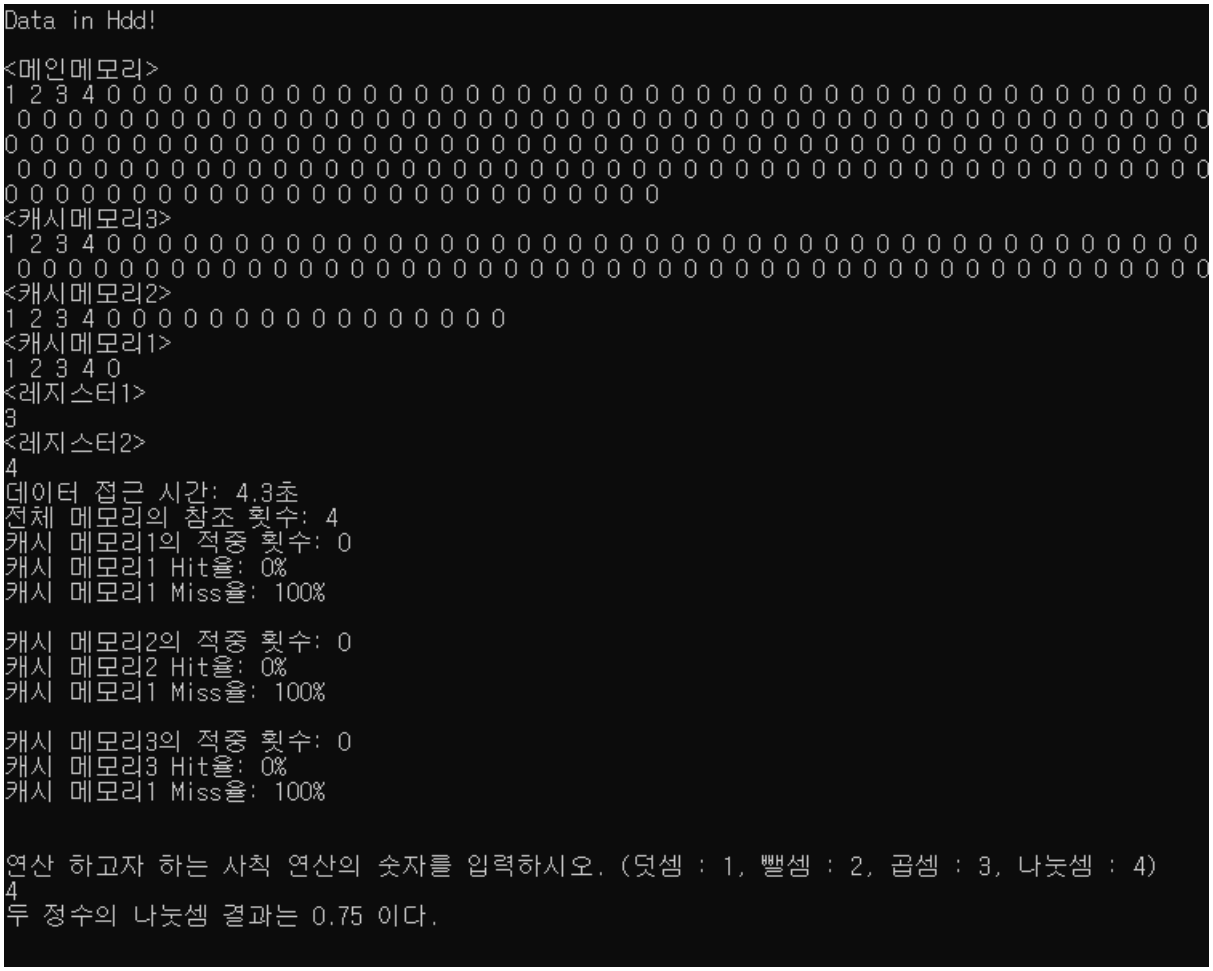
```
읽고싶은 첫번째 데이터를 입력하시오(1이상의 정수)
21

!!Cache1 Hit!!

<메인메모리>
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
<캐시메모리3>
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
<캐시메모리2>
21 1 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
<캐시메모리1>
21 1 18 19 20
<레지스터1>
21
<레지스터2>
1
데이터 접근 시간: 0.1초
전체 메모리의 참조 횟수: 23
캐시 메모리1의 적중 횟수: 1
캐시 메모리1 Hit율: 4.34783%
캐시 메모리1 Miss율: 95.6522%

캐시 메모리2의 적중 횟수: 0
캐시 메모리2 Hit율: 0%
캐시 메모리1 Miss율: 100%

캐시 메모리3의 적중 횟수: 1
캐시 메모리3 Hit율: 4.34783%
캐시 메모리1 Miss율: 95.6522%
```

[특이점 코드: 무작위 교체& 사칙연산]

캐시 메모리의 여분의 용량이 남은 상태에서 입력 받은 두 수는 순차적으로 캐시에 적재된다.

연산은 두 수를 입력받고 사칙연산 중 원하는 연산을 선택하여 실행하는 기능을 추가하였다.

1-4까지의 수를 두 개씩 입력하였고 나눗셈 연산을 실행한 화면이다.



아래는 캐시 메모리가 가득 찬 상태에서 새로 입력받은 수가 무작위 교체되는

코드 실행 화면이다. 1-8까지 두 수 씩 입력과 곱셈 연산을 수행한다.

캐시 메모리1의 경우 5까지 입력 후의 입력된 데이터는 무작위로 교체되는 것을 볼 수 있다.

```
읽고싶은 두번째 데이터를 입력하시오(1이상의 정수)
8

Data in Hdd!

<메인메모리>
1 2 3 4 5 6 7 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
<캐시메모리3>
1 2 3 4 5 6 7 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
<캐시메모리2>
1 2 3 4 5 6 7 8 0 0 0 0 0 0 0 0 0 0 0 0
<캐시메모리1>
1 7 3 8 5
<레지스터1>
7
<레지스터2>
8
데이터 접근 시간: 4.3초
전체 메모리의 참조 횟수: 8
캐시 메모리1의 적중 횟수: 0
캐시 메모리1 Hit율: 0%
캐시 메모리1 Miss율: 100%

캐시 메모리2의 적중 횟수: 0
캐시 메모리2 Hit율: 0%
캐시 메모리1 Miss율: 100%

캐시 메모리3의 적중 횟수: 0
캐시 메모리3 Hit율: 0%
캐시 메모리1 Miss율: 100%


연산 하고자 하는 사칙 연산의 숫자를 입력하시오. (덧셈 : 1, 뺄셈 : 2, 곱셈 : 3, 나눗셈 : 4)
3
두 정수의 곱셈 결과는 56 이다.
```

6. 팀원 인증샷