Instituto Politécnico Nacional

Escuela Superior de Cómputo

Práctica 5 - Archivo de Registros Parte 1

Unidad de aprendizaje: Arquitectura de Computadoras

Grupo: 3CV1

Alumno(a):
Ramos Diaz Enrique

Profesor(a):
Vega García Nayeli

23 de marzo 2020

1. Código de implementación

```cpp
#include <cstdlib>
#include <time.h>
#include <cstring>
#include <bitset>
#include <iostream>
using namespace std;

class ArchivoRegistros {
   private:
      int read_data1;
      int read_data2;
      int banco[16];
   public:
      ArchivoRegistros();
      void set();
      void get();
      void operacionSincrona(int write_data, int write_reg, int read_reg1,
       ↪  int shamt, int clr, int wr, int she, int dir);
      void operacionAsincrona(int clr);
      void operacionAsincrona(int clr, int read_reg1, int read_reg2);
};

ArchivoRegistros::ArchivoRegistros() { }

void ArchivoRegistros::set() {
   for(int i = 0; i < 16; i++) {
      banco[i] = rand() % 65535;
   }
}

void ArchivoRegistros::get() {
    for (int i = 0; i < 16; i++) {
       cout << bitset<16>(banco[i]) << endl;
    }
}

// Reset, Banco[0,1, ... , 15] = 0
void ArchivoRegistros::operacionAsincrona(int clr) {
   if (clr == 1) {
      fill(banco, banco + 16, 0);
   }
}
```

```
42
43 // readData1 = Banco[read_reg1]
44 // readData2 = Banco[read_reg2]
45 void ArchivoRegistros::operacionAsincrona(int clr, int read_reg1, int
    ↪  read_reg2) {
46    cout << "Read_Data1: ";
47    read_data1 = banco[read_reg1];
48    cout << bitset<16>(read_data1) << endl;
49
50    cout << "Read_Data2: ";
51    read_data2 = banco[read_reg2];
52    cout << bitset<16>(read_data2) << endl;
53 }
54
55 void ArchivoRegistros::operacionSincrona(int write_data, int write_reg, int
    ↪  read_reg1, int shamt, int clr, int wr, int she, int dir) {
56    if (clr == 1) {
57       // Reset, Banco[0,1, ... , 15] = 0
58       operacionAsincrona(clr);
59    }
60    else if (clr == 0 && wr == 0 && she == 0 && dir == 0) {
61       // Banco = Banco
62       memcpy(banco, banco, 16);
63    }
64    else if (clr == 0 && wr == 1 && she == 0) {
65       // Banco[write_reg] = write_data
66       banco[write_reg] = write_data;
67    }
68    else if (clr == 0 && wr == 1 && she == 1 && dir == 0) {
69       // Banco[write_reg] = Banco[read_reg1] >> shamt
70       banco[write_reg] = banco[read_reg1] >> shamt;
71    }
72    else if (clr == 0 && wr == 1 && she == 1 && dir == 1) {
73       // Banco[write_reg] = Banco[read_reg1] << shamt
74       banco[write_reg] = banco[read_reg1] << shamt;
75    }
76    else {
77       cout << "Combinacion de parametros incorrecta" << endl;
78    }
79 }
80
81 int main() {
82    srand(time(NULL));
83    ArchivoRegistros ar;
```

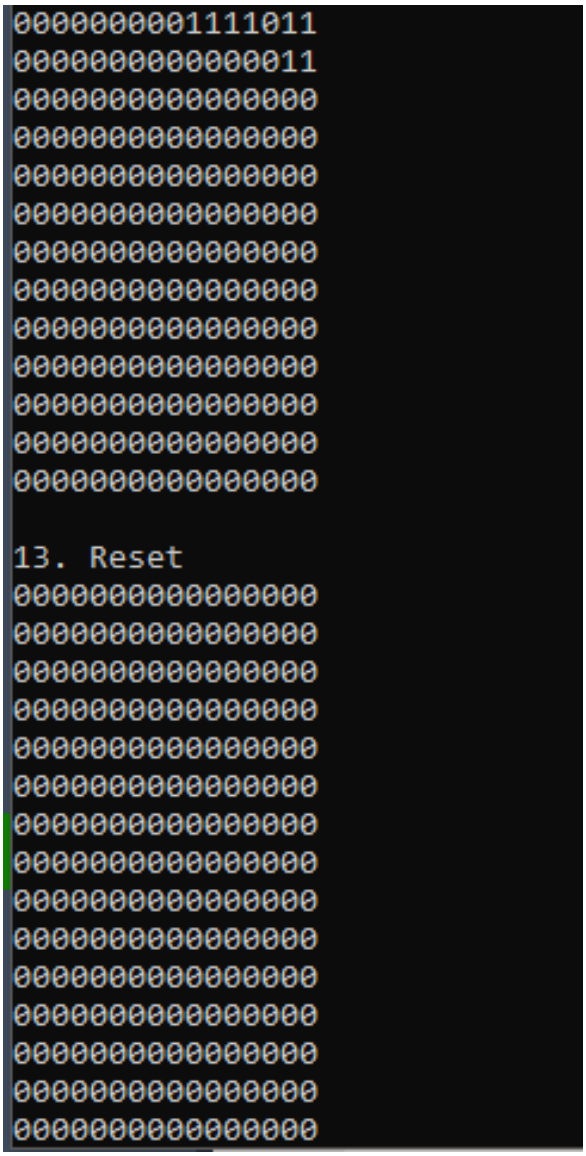
```
84
85      cout << "\n1. Reset" << endl;
86      ar.operacionAsincrona(1); // Reset
87      ar.get();
88
89      cout << "\n2. Banco[1] = 89" << endl;
90      ar.operacionSincrona(0b1011001, 0b1, 0b0, 0b0, 0, 1, 0, 0);
91      //ar.operacionSincrona(89, 1, 0, 0, 0, 1, 0, 0);
92      //ar.operacionAsincrona(1, 1, 1);
93
94      cout << "\n3. Banco[2] = 72" << endl;
95      ar.operacionSincrona(0b1001000, 0b10, 0b0, 0b0, 0, 1, 0, 0);
96      //ar.operacionSincrona(72, 2, 0, 0, 0, 1, 0, 0);
97      //ar.operacionAsincrona(1, 2, 2);
98
99      cout << "\n4. Banco[3] = 123" << endl;
100     ar.operacionSincrona(0b1111011, 0b11, 0b0, 0b0, 0, 1, 0, 0);
101     //ar.operacionSincrona(123, 3, 0, 0, 0, 1, 0, 0);
102     //ar.operacionAsincrona(1, 3, 3);
103
104     cout << "\n5. Banco[4] = 53" << endl;
105     ar.operacionSincrona(0b110101, 0b100, 0b0, 0b0, 0, 1, 0, 0);
106     //ar.operacionSincrona(53, 4, 0, 0, 0, 1, 0, 0);
107     //ar.operacionAsincrona(1, 4, 4);
108
109     cout << "\n6. Leer Banco[1] y Banco[2]" << endl;
110     ar.operacionAsincrona(1, 0b1, 0b10);
111     //ar.operacionAsincrona(1, 1, 2)
112
113     cout << "\n7. Leer Banco[3] y Banco[4]" << endl;
114     ar.operacionAsincrona(1, 0b11, 0b100);
115     //ar.operacionAsincrona(1, 3, 4)
116
117     cout << "\n8. Banco[2] = Banco[1] << 3" << endl;
118     ar.operacionSincrona(0b0, 0b10, 0b1, 0b11, 0, 1, 1, 1);
119     //ar.operacionSincrona(0, 2, 1, 3, 0, 1, 1, 1);
120     //ar.operacionAsincrona(1, 1, 2);
121
122     cout << "\n9. Banco[4] = Banco[3] >> 5" << endl;
123     ar.operacionSincrona(0b0, 0b100, 0b11, 0b101, 0, 1, 1, 0);
124     //ar.operacionSincrona(0, 4, 3, 5, 0, 1, 1, 0);
125     //ar.operacionAsincrona(1, 3, 4);
126
127     cout << "\n10. Leer Banco[1] y Banco[2]" << endl;
```

```
128     ar.operacionAsincrona(1, 0b1, 0b10);
129     //ar.operacionAsincrona(1, 1, 2);
130 
131     cout << "\n11. Leer Banco[3] y Banco[4]" << endl;
132     ar.operacionAsincrona(1, 0b11, 0b100);
133     //ar.operacionAsincrona(1, 3, 4);
134 
135     cout << "\n12. Get()" << endl;
136     ar.get();
137 
138     cout << "\n13. Reset" << endl;
139     ar.operacionAsincrona(1); // Reset
140     ar.get();
141 
142     cout << endl;
143     return 0;
144 }
```

2. Pruebas

```
1. Reset
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000

2. Banco[1] = 89

3. Banco[2] = 72

4. Banco[3] = 123

5. Banco[4] = 53

6. Leer Banco[1] y Banco[2]
Read_Data1: 0000000001011001
Read_Data2: 0000000001001000
```

```
7. Leer Banco[3] y Banco[4]
Read_Data1: 0000000001111011
Read_Data2: 0000000000110101

8. Banco[2] = Banco[1] << 3

9. Banco[4] = Banco[3] >> 5

10. Leer Banco[1] y Banco[2]
Read_Data1: 0000000001011001
Read_Data2: 0000001011001000

11. Leer Banco[3] y Banco[4]
Read_Data1: 0000000001111011
Read_Data2: 0000000000000011

12. Get()
0000000000000000
0000000001011001
0000001011001000
0000000001111011
0000000000000011
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
```


0000000001111011
0000000000000011
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
13. Reset
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000