Reporte de practica 5

González Pardo Adrian

Marzo 2020

1. Código C++

```
/*
 * Alumno: Gonzalez Pardo Adrian
 * Grupo: 3CV8
 * Practica 5
 */
#include <bits/stdc++.h>
using namespace std;

#define KNRM  "\x1B[0m"
#define KRED  "\x1B[31m"
#define KGRN  "\x1B[32m"
#define KYEL  "\x1B[33m"
#define KBLU  "\x1B[34m"
#define KMAG  "\x1B[35m"
#define KCYN  "\x1B[36m"
#define KWHT  "\x1B[37m"
#define BRED  "\x1B[91m"
#define BGRN  "\x1B[92m"
#define BYEL  "\x1B[93m"
#define BBLU  "\x1B[94m"
#define BMAG  "\x1B[95m"
#define BCYN  "\x1B[96m"
#define BWHT  "\x1B[97m"

class registros{
  private:
    int banco[16],writeData,writeReg,readData1,readData2,shamt;
    bool WR,SHE,DIR,CLR;
  public:
    /* Constructor */
    registros(){
      srand(time(NULL));
    }

    /* Getters and Setters */
    int getWriteReg(){return this->writeReg;}

```

```
38      int getWriteData(){return this->writeData;}
39
40      int getReadData1(){return this->readData1;}
41
42      int getReadData2(){return this->readData2;}
43
44      int getShamt(){return this->shamt;}
45
46      bool isWR(){return this->WR;}
47
48      bool isSHE(){return this->SHE;}
49
50      bool isDIR(){return this->DIR;}
51
52      bool isCLR(){return this->CLR;}
53
54      void setWriteReg(int writeReg){this->writeReg=writeReg;}
55
56      void setWriteData(int writeData){this->writeData=writeData;}
57
58      void setReadData1(int readData1){this->readData1=readData1;}
59
60      void setReadData2(int readData2){this->readData2=readData2;}
61
62      void setShamt(int shamt){this->shamt=shamt;}
63
64      void setWR(bool WR){this->WR=WR;}
65
66      void setSHE(bool SHE){this->SHE=SHE;}
67
68      void setDIR(bool DIR){this->DIR=DIR;}
69
70      void setCLR(bool CLR){this->CLR=CLR;}
71
72      /* Set banco with random number between -32768 to 32767 */
73      void set(){
74        for(int i=0; i<16; i++) {
75          /* Limite positivo 32767
76           * Limite negativo -32768 */
77          *(banco+i)=(rand()%65536)-32768;
78        }
79      }
80
81
82      /* Validation function */
83      bool isInReg(int reg){
84        return (reg<0 || reg>16);
85      }
86
87      bool isReg(int reg){
88        return (reg>0&&reg<16);
89      }
90
```

```
91      /* Get banco data */
92      void get(){
93        for(int i=0; i<16; i++){
94          cout<< "Registro["<<i<<"] :=> "<<banco[i]<<"\n";
95        }
96      }
97
98
99      /* Operaciones del banco de registros */
100     void operacionSincrona(int writeData,int writeReg,
101         int readReg1,int readReg2,int shamt,
102         bool WR,bool SHE, bool DIR,
103         bool CLR){
104       setWR(WR);
105       setSHE(SHE);
106       setDIR(DIR);
107       setCLR(CLR);
108       setWriteReg(writeReg);
109       setWriteData(writeData);
110       setShamt(shamt);
111       if(getWriteReg()<0 || getWriteReg()>16){
112         cout<<BRED<<"Fuera del limite de registros\n"<<KNRM;
113         return;
114       }
115
116       if(getWriteData()<-32768 || getWriteData()>32767){
117         cout<<BRED<<"Valor mayor a un Slit16\n"<<KNRM;
118         return;
119       }
120
121       if(isInReg(readReg1)){
122         cout<<BRED<<"Reg1 fuera del limite de registros\n"<<KNRM;
123         return;
124       }
125
126       if(isInReg(readReg2)){
127         cout<<BRED<<"Reg2 fuera del limite de registros\n"<<KNRM;
128         return;
129       }
130
131       if(isInReg(getShamt())){
132         cout<<BRED<<"Shamt fuera del limite de registros\n"<<KNRM;
133         return;
134       }
135
136       setReadData1(banco[readReg1]);
137       setReadData2(banco[readReg2]);
138       if(isCLR()){
139         operacionAsincrona(isCLR());
140         return;
141       }else if(isWR() && !isSHE()){
142         *(banco+getWriteReg())=getWriteData();
143         return;
```

```
144       }else if(isWR() && isSHE() && !isDIR()){
145         *(banco+getWriteReg())=(*(banco+readReg1)>>getShamt()) & 0
    x0000ffff;
146         return;
147       }else if(isWR() && isSHE() && isDIR()){
148         *(banco+getWriteReg())=(*(banco+readReg1)<<getShamt()) & 0
    x0000ffff;
149         return;
150       }
151     }
152
153     /* Operacion que manda a 0 todo el banco de registros */
154     void operacionAsincrona(bool CLR){
155       for(int i=0; i<16; i++){
156         *(banco+i)=0;
157       }
158       get();
159     }
160
161     /* Operacion que muestra Registros */
162     void operacionAsincrona(bool CLR, int readReg1, int readReg2){
163       setCLR(CLR);
164       if(isCLR()){
165         operacionAsincrona(isCLR());
166       }
167       if(isReg(readReg1)){
168         cout<<BGRN<<"Registro["<<readReg1<<"] :=> "<<banco[readReg1]<<"\n"
    <<KNRM;
169       }
170       if(isReg(readReg2)){
171         cout<<BYEL<<"Registro["<<readReg2<<"] :=> "<<banco[readReg2]<<"\n"
    <<KNRM;
172       }
173
174     }
175 };
176
177 int main(void) {
178   registros r;
179   cout<<BBLU<<"Inicializacion\n";
180   r.set();
181   r.get();
182   cout<<BCYN<<"\n\t\t(Operacion 1)\n\toperacionAsincrona(1) <==> RESET\n";
183   r.operacionAsincrona(1);
184
185   cout<<KGRN<<"\n\t\t(Operacion 2)\n\tBANCO[1]=89 <==> operacionSincrona
     (89,1,0,0,0,1,0,0,0)\n";
186   r.operacionSincrona(89,1,0,0,0,1,0,0,0);
187   r.get();
188
189   cout<<KCYN<<"\n\t\t(Operacion 3)\n\tBANCO[2]=72 <==> operacionSincrona
     (72,2,0,0,0,1,0,0,0)\n";
190   r.operacionSincrona(72,2,0,0,0,1,0,0,0);
```

```
191   r.get();
192
193   cout<<KGRN<<"\n\t\t(Operacion 4)\n\tBANCO[3]=123 <==> operacionSincrona
        (123,3,0,0,0,1,0,0,0)\n";
194   r.operacionSincrona(123,3,0,0,0,1,0,0,0);
195   r.get();
196
197   cout<<KYEL<<"\n\t\t(Operacion 5)\n\tBANCO[4]=53 <==> operacionSincrona
        (53,4,0,0,0,1,0,0,0)\n";
198   r.operacionSincrona(53,4,0,0,0,1,0,0,0);
199   r.get();
200
201   cout<<BMAG<<"\n\t\t(Operacion 6)\n\tREAD BANCO[1] & BANCO[2]\n";
202   r.operacionAsincrona(0,1,2);
203
204   cout<<KMAG<<"\n\t\t(Operacion 7)\n\tREAD BANCO[3] & BANCO[4]\n";
205   r.operacionAsincrona(0,3,4);
206
207   cout<<BBLU<<"\n\t\t(Operacion 8)\n\tBANCO[2]=BANCO[1]<<3 <==>
        operacionSincrona(0,2,1,0,3,1,1,1,0)\n";
208   r.operacionSincrona(0,2,1,0,3,1,1,1,0);
209   r.get();
210
211   cout<<KYEL<<"\n\t\t(Operacion 9)\n\tBANCO[4]=BANCO[3]>>5 <==>
        operacionSincrona(0,4,3,0,5,1,1,0,0)\n";
212   r.operacionSincrona(0,4,3,0,5,1,1,0,0);
213   r.get();
214
215   cout<<KGRN<<"\n\t\t(Operacion 10)\n\tREAD BANCO[1] & BANCO[2]\n";
216   r.operacionAsincrona(0,1,2);
217
218   cout<<KYEL<<"\n\t\t(Operacion 11)\n\tREAD BANCO[3] & BANCO[4]\n";
219   r.operacionAsincrona(0,3,4);
220
221   cout<<KBLU<<"\n\t\t(Operacion 12)\n\tget()\n";
222   r.get();
223
224   cout<<BCYN<<"\n\t\t(Operacion 13)\n\toperacionAsincrona(1) <==> RESET\n"
        ;
225   r.operacionAsincrona(1);
226
227   cout<<KNRM<<endl;
228   return 0;
229 }
```

Código fuente de archivo de registros

2. Captura de simulaciones

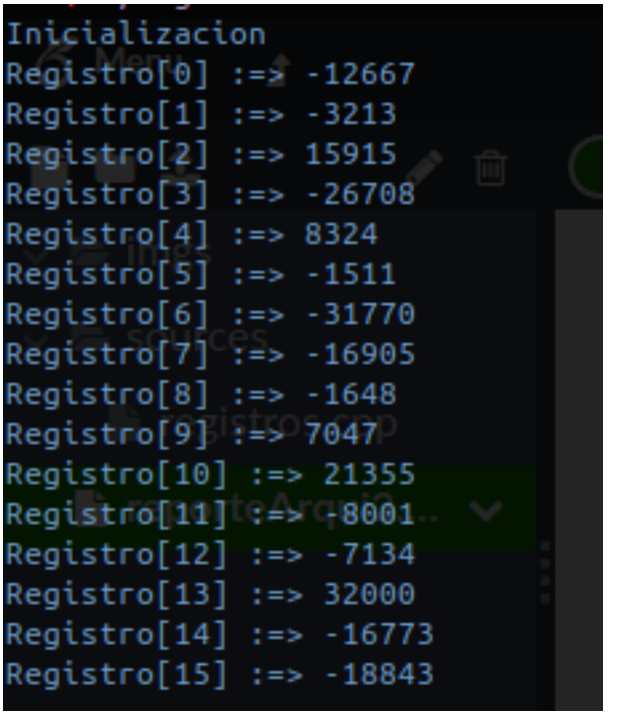

Figura 0: Inicialización del programa con números random

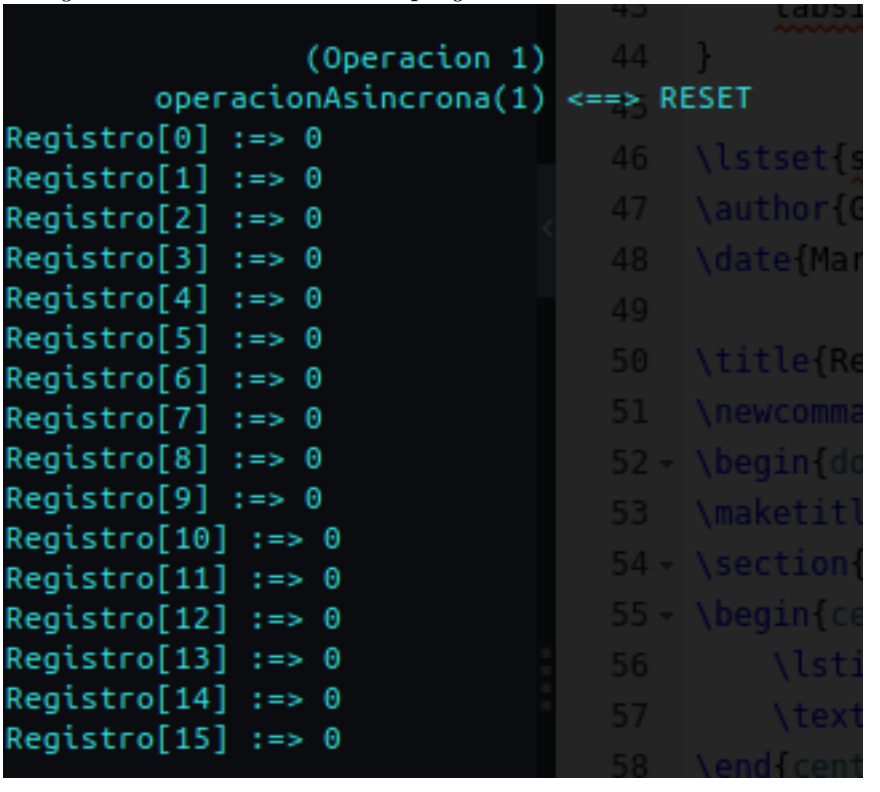

Figura 1: Operación 1 Reset"

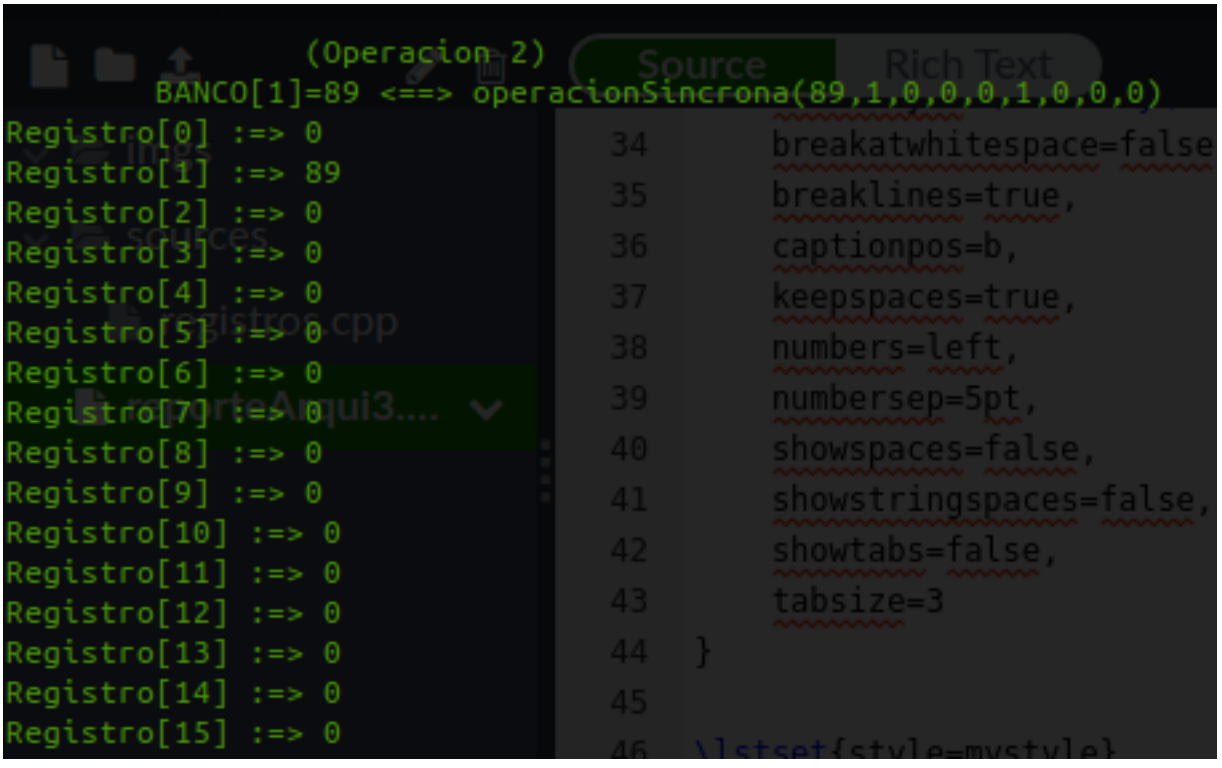

Figura 2: Operación 2 "Banco[1]=89"

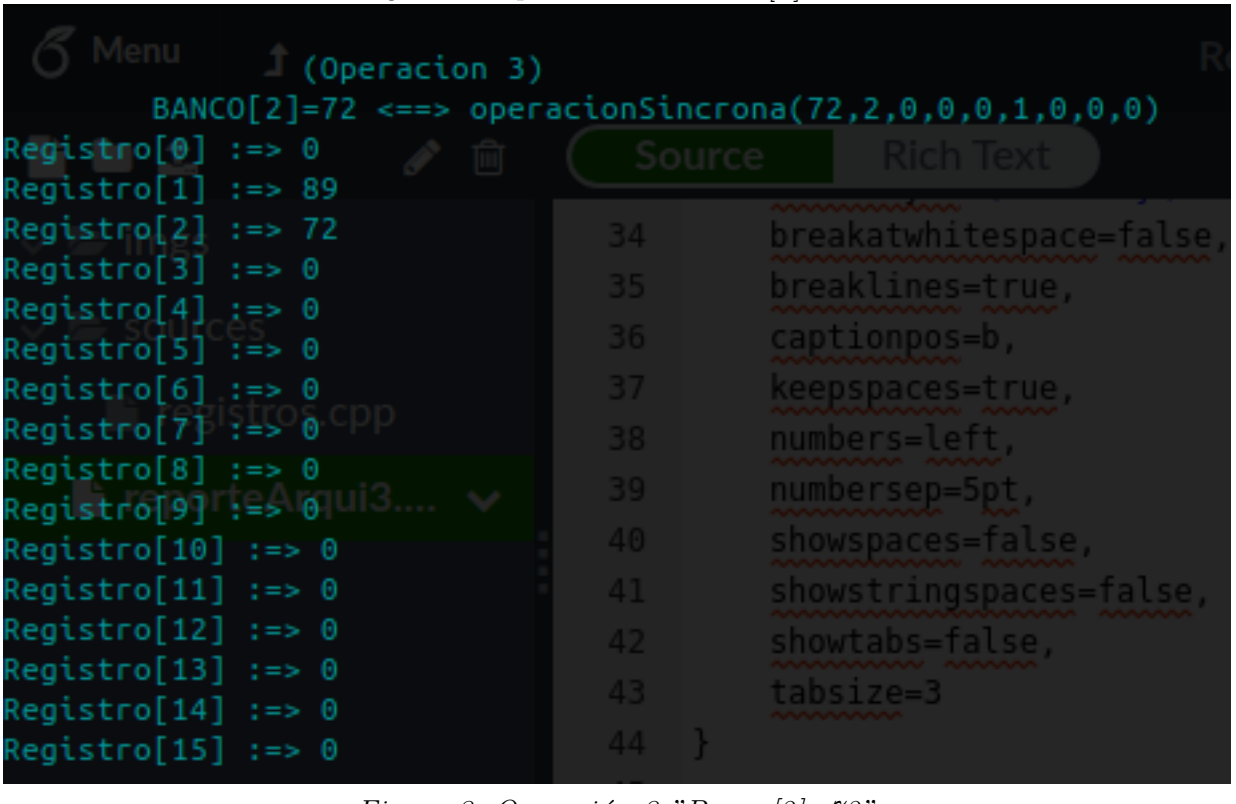

Figura 3: Operación 3 "Banco[2]=72"

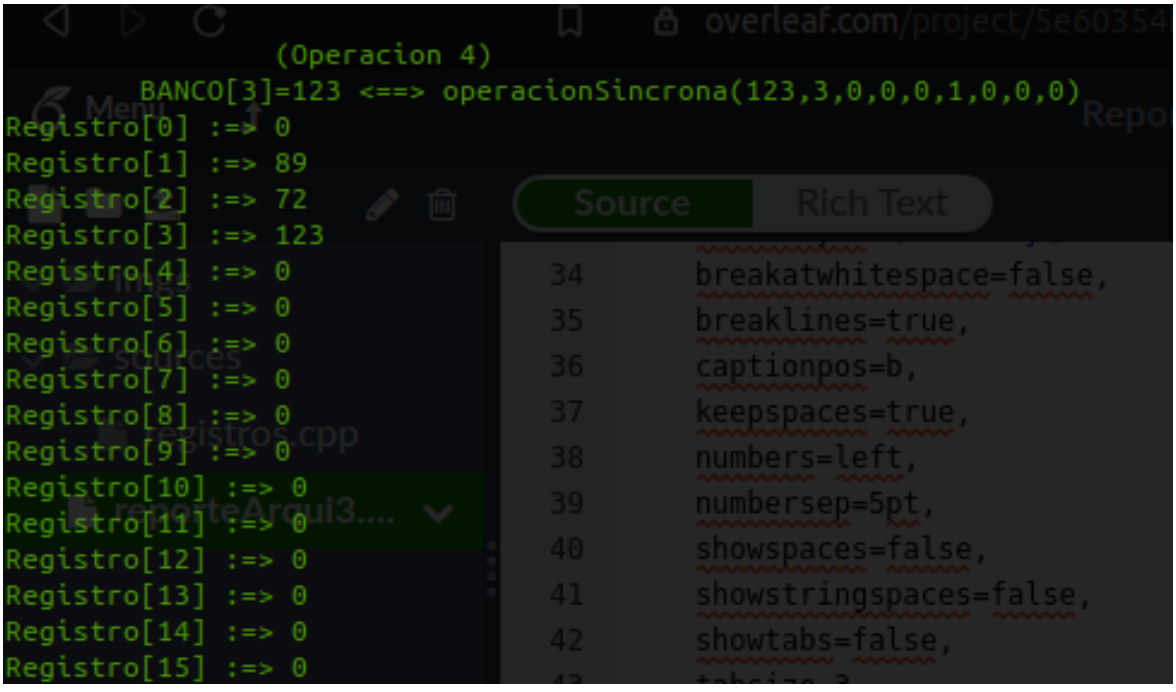

Figura 4: Operación 4 "Banco[3]=123"

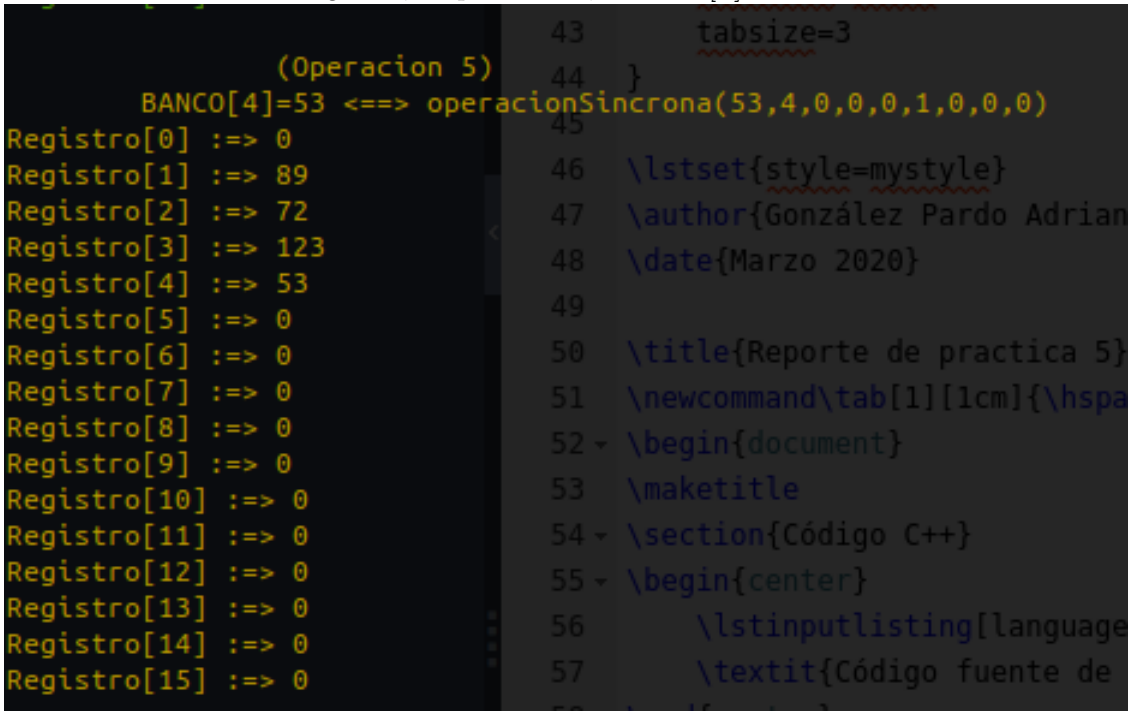

Figura 5: Operación 5 "Banco[4]=53"

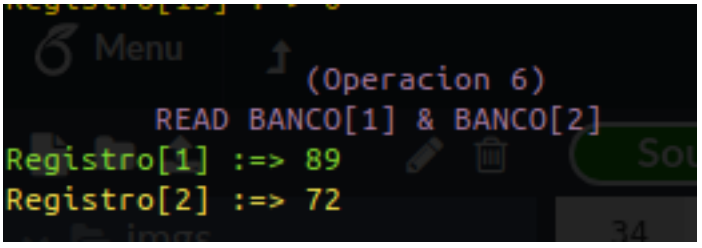

Figura 6: Operación 6 READ Banco[1] & Banco[2]"

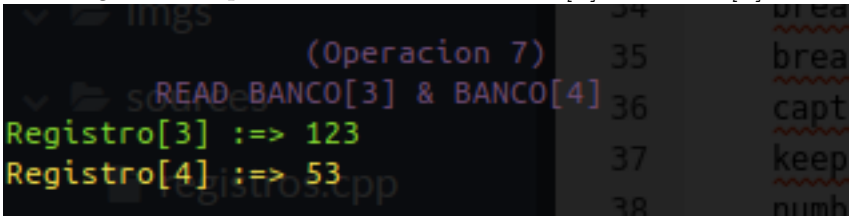

Figura 7: Operación 7 READ Banco[3] & Banco[4]"

Figura 8: Operación 8 "Banco[2]=Banco[1]<<3"

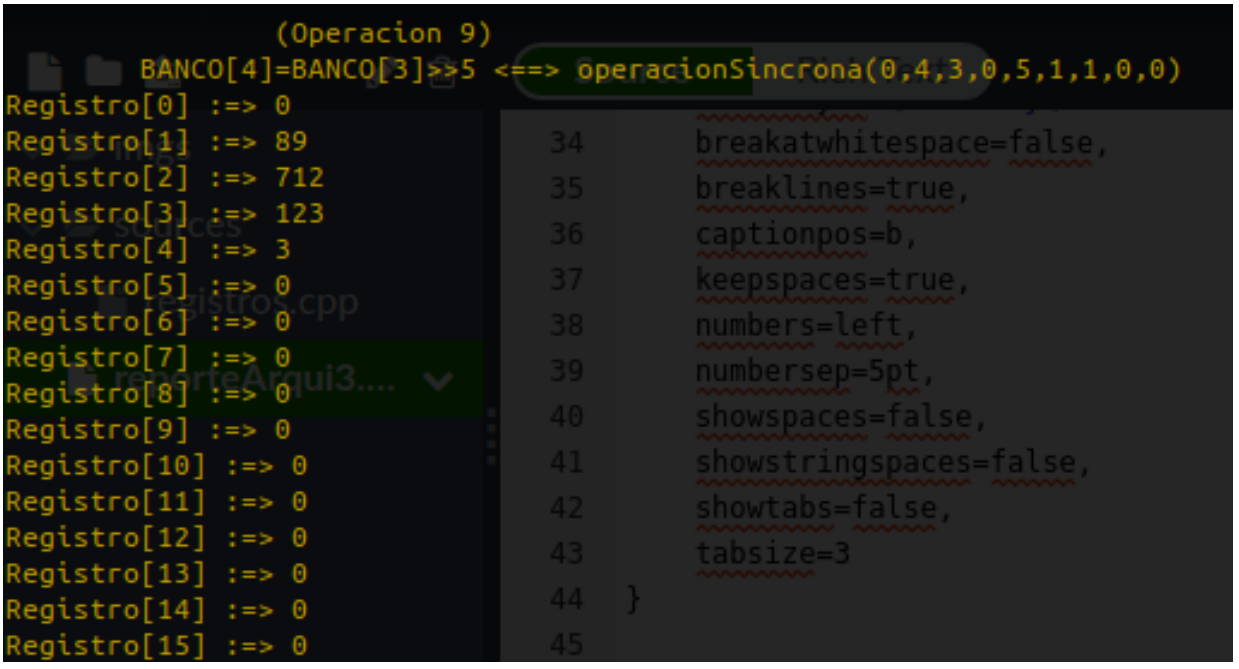

Figura 9: Operación 9 "Banco[4]=Banco[3]>>5"

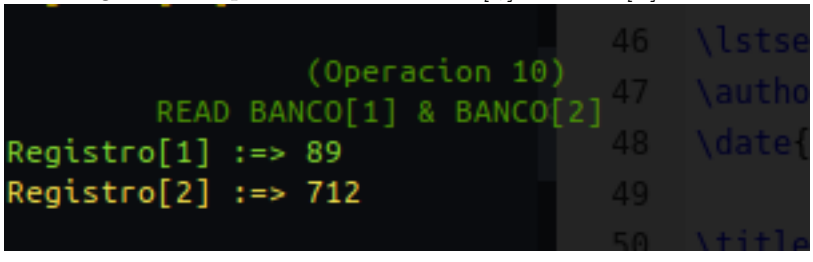

Figura 10: Operación 10 READ Banco[1] & Banco[2]"

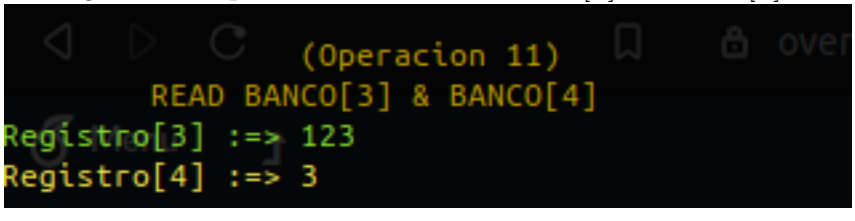

Figura 11: Operación 11 READ Banco[3] & Banco[4]"

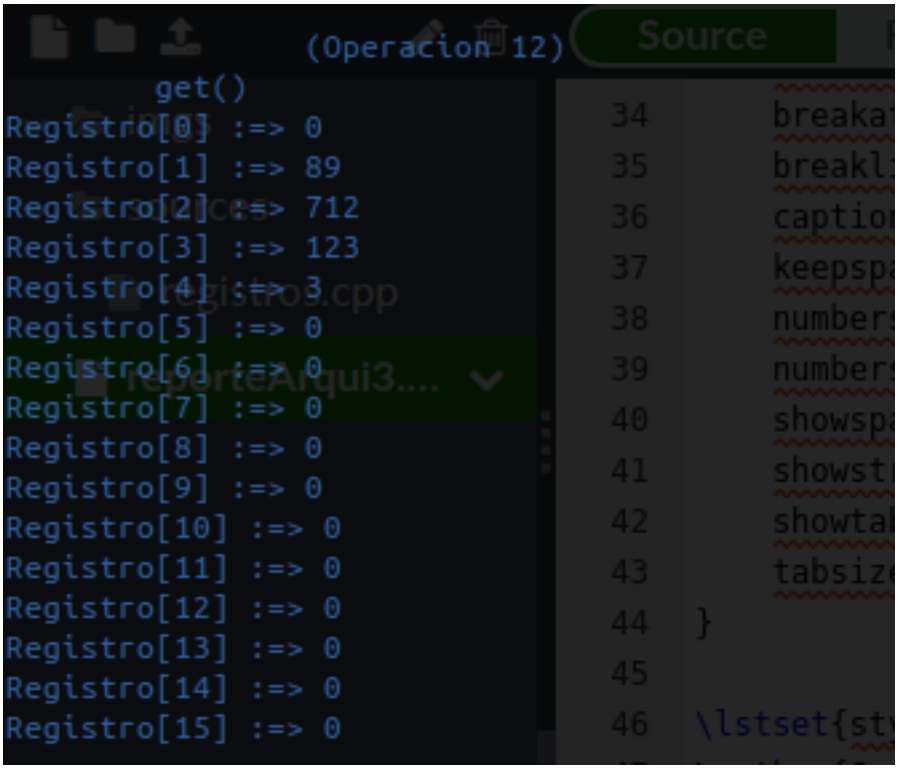

Figura 12: Operación 12 "GET()"

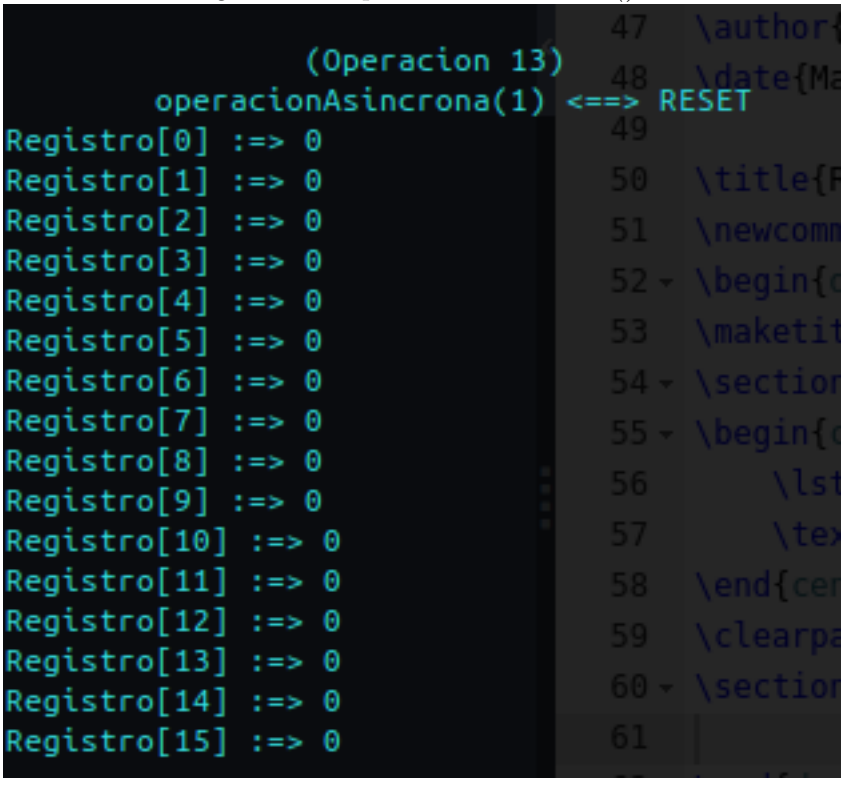

Figura 13: Operación 13 Reset"