

Reporte de practica 5

González Pardo Adrian

Marzo 2020

1. Código C++

```
1  /*
2   * Alumno: Gonzalez Pardo Adrian
3   * Grupo: 3CV8
4   * Practica 5
5   * Desarrollado para Linux por el define de colores
6   */
7  #include <bits/stdc++.h>
8  using namespace std;
9
10 #define KNRM "\x1B[0m"
11 #define KRED "\x1B[31m"
12 #define KGRN "\x1B[32m"
13 #define KYEL "\x1B[33m"
14 #define KBLU "\x1B[34m"
15 #define KMAG "\x1B[35m"
16 #define KCYN "\x1B[36m"
17 #define KWHT "\x1B[37m"
18 #define BRED "\x1B[91m"
19 #define BGRN "\x1B[92m"
20 #define BYEL "\x1B[93m"
21 #define BBLU "\x1B[94m"
22 #define BMAG "\x1B[95m"
23 #define BCYN "\x1B[96m"
24 #define BWHT "\x1B[97m"
25
26 class registros{
27     private:
28         int banco[16], writeData, writeReg, readData1, readData2, shamt;
29         bool WR, SHE, DIR, CLR;
30     public:
31         /* Constructor */
32         registros(){
33             srand(time(NULL));
34         }
35
36         /* Getters and Setters */
37         int getWriteReg(){return this->writeReg;}
```

```

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

```

```

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

```

```

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

```

```

191 r.operacionSincrona(72,2,0,0,0,1,0,0,0);
192 r.get();
193
194 cout<<KGRN<<"\n\t\t(Operacion 4)\n\tBANCO [3]=123 <==> operacionSincrona
    (123,3,0,0,0,1,0,0,0)\n";
195 r.operacionSincrona(123,3,0,0,0,1,0,0,0);
196 r.get();
197
198 cout<<KYEL<<"\n\t\t(Operacion 5)\n\tBANCO [4]=53 <==> operacionSincrona
    (53,4,0,0,0,1,0,0,0)\n";
199 r.operacionSincrona(53,4,0,0,0,1,0,0,0);
200 r.get();
201
202 cout<<BMAG<<"\n\t\t(Operacion 6)\n\tREAD BANCO [1] & BANCO [2]\n";
203 r.operacionAsincrona(0,1,2);
204
205 cout<<KMAG<<"\n\t\t(Operacion 7)\n\tREAD BANCO [3] & BANCO [4]\n";
206 r.operacionAsincrona(0,3,4);
207
208 cout<<BBLU<<"\n\t\t(Operacion 8)\n\tBANCO [2]=BANCO [1] <<3 <==>
    operacionSincrona(0,2,1,0,3,1,1,1,0)\n";
209 r.operacionSincrona(0,2,1,0,3,1,1,1,0);
210 r.get();
211
212 cout<<KYEL<<"\n\t\t(Operacion 9)\n\tBANCO [4]=BANCO [3] >>5 <==>
    operacionSincrona(0,4,3,0,5,1,1,0,0)\n";
213 r.operacionSincrona(0,4,3,0,5,1,1,0,0);
214 r.get();
215
216 cout<<KGRN<<"\n\t\t(Operacion 10)\n\tREAD BANCO [1] & BANCO [2]\n";
217 r.operacionAsincrona(0,1,2);
218
219 cout<<KYEL<<"\n\t\t(Operacion 11)\n\tREAD BANCO [3] & BANCO [4]\n";
220 r.operacionAsincrona(0,3,4);
221
222 cout<<KBLU<<"\n\t\t(Operacion 12)\n\tget()\n";
223 r.get();
224
225 cout<<BCYN<<"\n\t\t(Operacion 13)\n\toperacionAsincrona(1) <==> RESET\n"
    ;
226 r.operacionAsincrona(1);
227
228 cout<<KNRM<<endl;
229 return 0;
230 }

```

Código fuente de archivo de registros

2. Captura de simulaciones

```
Inicializacion
Registro[0] :=> -12667
Registro[1] :=> -3213
Registro[2] :=> 15915
Registro[3] :=> -26708
Registro[4] :=> 8324
Registro[5] :=> -1511
Registro[6] :=> -31770
Registro[7] :=> -16905
Registro[8] :=> -1648
Registro[9] :=> 7047
Registro[10] :=> 21355
Registro[11] :=> -8001..
Registro[12] :=> -7134
Registro[13] :=> 32000
Registro[14] :=> -16773
Registro[15] :=> -18843
```

Figura 0: Inicialización del programa con números random

```
(Operacion 1)
operacionAsincrona(1) <==> RESET
Registro[0] :=> 0
Registro[1] :=> 0
Registro[2] :=> 0
Registro[3] :=> 0
Registro[4] :=> 0
Registro[5] :=> 0
Registro[6] :=> 0
Registro[7] :=> 0
Registro[8] :=> 0
Registro[9] :=> 0
Registro[10] :=> 0
Registro[11] :=> 0
Registro[12] :=> 0
Registro[13] :=> 0
Registro[14] :=> 0
Registro[15] :=> 0
```

Figura 1: Operación 1 Reset


```

(Operacion 4)
BANCO[3]=123 <==> operacionSincrona(123,3,0,0,0,1,0,0,0)
Registro[0] :=> 0
Registro[1] :=> 89
Registro[2] :=> 72
Registro[3] :=> 123
Registro[4] :=> 0
Registro[5] :=> 0
Registro[6] :=> 0
Registro[7] :=> 0
Registro[8] :=> 0
Registro[9] :=> 0
Registro[10] :=> 0
Registro[11] :=> 0
Registro[12] :=> 0
Registro[13] :=> 0
Registro[14] :=> 0
Registro[15] :=> 0

```

```

34 breakatwhitespace=false,
35 breaklines=true,
36 captionpos=b,
37 keepspaces=true,
38 numbers=left,
39 numbersep=5pt,
40 showspace=false,
41 showstringspaces=false,
42 showtabs=false,

```

Figura 4: Operación 4 "Banco[3]=123"

```

(Operacion 5)
BANCO[4]=53 <==> operacionSincrona(53,4,0,0,0,1,0,0,0)
Registro[0] :=> 0
Registro[1] :=> 89
Registro[2] :=> 72
Registro[3] :=> 123
Registro[4] :=> 53
Registro[5] :=> 0
Registro[6] :=> 0
Registro[7] :=> 0
Registro[8] :=> 0
Registro[9] :=> 0
Registro[10] :=> 0
Registro[11] :=> 0
Registro[12] :=> 0
Registro[13] :=> 0
Registro[14] :=> 0
Registro[15] :=> 0

```

```

43 tabsize=3
44 }
45
46 \lstset{style=mystyle}
47 \author{González Pardo Adrian}
48 \date{Marzo 2020}
49
50 \title{Reporte de practica 5}
51 \newcommand\tab[1][1cm]{\hspace{1cm}}
52 \begin{document}
53 \maketitle
54 \section{Código C++}
55 \begin{center}
56 \lstinputlisting[language=C++]{CodigoFuente/CodigoFuente.cpp}
57 \textit{Código fuente de

```

Figura 5: Operación 5 "Banco[4]=53"


```

Menu
(Operacion 6)
READ BANCO[1] & BANCO[2]
Registro[1] :=> 89
Registro[2] :=> 72

```

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

```

(Operacion 7)
READ BANCO[3] & BANCO[4]
Registro[3] :=> 123
Registro[4] :=> 53

```

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

```

(Operacion 8)
BANCO[2]=BANCO[1]<<3 <==> operacionSincrona(0,2,1,0,3,1,1,1,0)
Registro[0] :=> 0
Registro[1] :=> 89
Registro[2] :=> 712
Registro[3] :=> 123
Registro[4] :=> 53
Registro[5] :=> 0
Registro[6] :=> 0
Registro[7] :=> 0
Registro[8] :=> 0
Registro[9] :=> 0
Registro[10] :=> 0
Registro[11] :=> 0
Registro[12] :=> 0
Registro[13] :=> 0
Registro[14] :=> 0
Registro[15] :=> 0

```

```

38 numbers=left,
39 numbersep=5pt,
40 operacionSincrona(0,2,1,0,3,1,1,1,0)
41 showspaces=false,
42 showstringspaces=false,
43 showtabs=false,
44 tabspace=3
45 }
46 \lstset{style=mystyle}
47 \author{González Pardo Adrian}
48 \date{Marzo 2020}
49
50 \title{Reporte de practica 5}
51 \newcommand\tab[1][1cm]{\hspace*{#1}}
52 \begin{document}

```

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

```

      (Operacion 9)
      BANCO[4]=BANCO[3]>>5 <==> operacionSincrona(0,4,3,0,5,1,1,0,0)
Registro[0] :=> 0
Registro[1] :=> 89
Registro[2] :=> 712
Registro[3] :=> 123
Registro[4] :=> 3
Registro[5] :=> 0
Registro[6] :=> 0
Registro[7] :=> 0
Registro[8] :=> 0
Registro[9] :=> 0
Registro[10] :=> 0
Registro[11] :=> 0
Registro[12] :=> 0
Registro[13] :=> 0
Registro[14] :=> 0
Registro[15] :=> 0
34      breakatwhitespace=false,
35      breaklines=true,
36      captionpos=b,
37      keepspaces=true,
38      numbers=left,
39      numbersep=5pt,
40      showspaces=false,
41      showstringspaces=false,
42      showtabs=false,
43      tabsize=3
44  }
45

```

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

```

      (Operacion 10)
      READ BANCO[1] & BANCO[2]
Registro[1] :=> 89
Registro[2] :=> 712
46  \lstse
47  \autho
48  \date{
49
50  \title

```

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

```

      (Operacion 11)
      READ BANCO[3] & BANCO[4]
Registro[3] :=> 123
Registro[4] :=> 3
over

```

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

```

(Operación 12) Source
get()
Registro[0] :=> 0
Registro[1] :=> 89
Registro[2] :=> 712
Registro[3] :=> 123
Registro[4] :=> 3
Registro[5] :=> 0
Registro[6] :=> 0
Registro[7] :=> 0
Registro[8] :=> 0
Registro[9] :=> 0
Registro[10] :=> 0
Registro[11] :=> 0
Registro[12] :=> 0
Registro[13] :=> 0
Registro[14] :=> 0
Registro[15] :=> 0

34 breaka
35 breakl
36 captio
37 keepspa
38 number:
39 number:
40 showspa
41 showst
42 showtal
43 tabsiz
44 }
45
46 \lstset{st

```

Figura 12: Operación 12 "GET()"

```

(Operación 13)
operacionAsincrona(1) <==> RESET
Registro[0] :=> 0
Registro[1] :=> 0
Registro[2] :=> 0
Registro[3] :=> 0
Registro[4] :=> 0
Registro[5] :=> 0
Registro[6] :=> 0
Registro[7] :=> 0
Registro[8] :=> 0
Registro[9] :=> 0
Registro[10] :=> 0
Registro[11] :=> 0
Registro[12] :=> 0
Registro[13] :=> 0
Registro[14] :=> 0
Registro[15] :=> 0

47 \author
48 \date{Ma
49
50 \title{E
51 \newcom
52 \begin{c
53 \maketit
54 \section
55 \begin{c
56 \lst
57 \tex
58 \end{cer
59 \clearpa
60 \section
61

```

Figura 13: Operación 13 Reset"