

# Reporte de practica 9

González Pardo Adrian

Abril 2020

## 1. Código fuente:

```
1  /*
2   * Alumno: Gonzalez Pardo Adrian
3   * Grupo: 3CV8
4   * Practica 9
5   */
6  #include <bits/stdc++.h>
7
8  #define tamPC 8
9
10 using namespace std;
11 class Stack{
12 private:
13     bool UP,DW,WPC,CLR;
14     unsigned short PC[tamPC],SP;
15     unsigned short PCin,PCout;
16 public:
17     Stack(){
18         srand(time(NULL));
19     }
20     bool getUP(){
21         return this->UP;
22     }
23
24     bool getDW(){
25         return this->DW;
26     }
27
28     bool getWPC(){
29         return this->WPC;
30     }
31
32     bool getCLR(){
33         return this->CLR;
34     }
35
36     unsigned short *getPC(){
37         return this->PC;
38     }
39
40     unsigned short getSP(){
41         return this->SP;
42     }
43
44     unsigned short getPCin(){
45         return this->PCin;
46     }
47
48     unsigned short getPCout(){
49         return this->PCout;
50     }
51 }
```

```

52 void setUP(bool UP){
53     this->UP=UP;
54 }
55
56 void setDW(bool DW){
57     this->DW=DW;
58 }
59
60 void setWPC(bool WPC){
61     this->WPC=WPC;
62 }
63
64 void setCLR(bool CLR){
65     this->CLR=CLR;
66 }
67
68 void setSP(unsigned short SP){
69     this->SP=SP;
70 }
71
72 void setPCin(unsigned short PCin){
73     this->PCin=PCin;
74 }
75
76 void setPCout(unsigned short PCout){
77     this->PCout=PCout;
78 }
79
80 void set(){
81     for(int i=0;i<tamPC;i++){
82         *(PC+i)=(unsigned short)(rand()%65535)-16384;
83     }
84     setSP(0);
85 }
86
87 void get(){
88     for(int i=0;i<8;i++){
89         printf("\tPC[%d] = %d\n",i,*(PC+i));
90     }
91 }
92
93 void operacion(unsigned short PCin,bool UP,bool DW,bool WPC, bool CLR){
94     setPCin(PCin);
95     setUP(UP);
96     setDW(DW);
97     setWPC(WPC);
98     setCLR(CLR);
99     if(getCLR()){
100         setSP(0);
101         for(int i=0;i<tamPC;i++){
102             *(PC+i)=0;
103         }
104         PCout=*(PC+getSP());
105         return;
106     }
107     if(!getWPC()&&!getUP()&&!getDW()){
108         setSP(getSP());
109         *(PC+getSP())+=1;
110         PCout=*(PC+getSP());
111         return ;
112     }
113     if(getWPC()&&!getUP()&&!getDW()){
114         setSP(getSP());
115         *(PC+getSP())=PCin;
116         PCout=*(PC+getSP());
117         return;
118     }

```

```

119     if(getWPC() && getUP() && !getDW()){
120         if(getSP()+1 < tamPC){
121             setSP(getSP()+1);
122         }else{
123             setSP(0);
124         }
125         *(PC+getSP())=PCin;
126         PCout=*(PC+getSP());
127         return;
128     }
129     if(!getWPC() && !getUP() && getDW()){
130         if(getSP() > 0){
131             setSP(getSP()-1);
132         }else{
133             setSP(tamPC-1);
134         }
135         *(PC+getSP())+=1;
136         PCout=*(PC+getSP());
137         return;
138     }
139 }
140
141
142 void operacion(){
143     printf("PC[%d] = %d\n", getSP(), *(PC+getSP()));
144 }
145 };
146 int main(int argc, char *argv[]) {
147     Stack st;
148     printf("Set()\n");
149     st.set();
150     st.get();
151     printf("CLR\n");
152     st.operacion(0,0,0,0,1);
153     st.get();
154     st.operacion();
155     printf("Instruccion\n");
156     printf("LI R6, #87 \t");
157     st.operacion(0,0,0,0,0);
158     st.operacion();
159     printf("LI R8, #90 \t");
160     st.operacion(0,0,0,0,0);
161     st.operacion();
162     printf("B 34 \t");
163     st.operacion(34,0,0,1,0);
164     st.operacion();
165     printf("ADD R8, R2, R3 \t");
166     st.operacion(0,0,0,0,0);
167     st.operacion();
168     printf("SUB R1, R2, R3 \t");
169     st.operacion(0,0,0,0,0);
170     st.operacion();
171     printf("CALL 0x61 \t");
172     st.operacion(61,1,0,1,0);
173     st.operacion();
174     printf("LI R6, #87 \t");
175     st.operacion(0,0,0,0,0);
176     st.operacion();
177     printf("LI R8, #90 \t");
178     st.operacion(0,0,0,0,0);
179     st.operacion();
180     printf("CALL 100 \t");
181     st.operacion(100,1,0,1,0);
182     st.operacion();
183     printf("ADD R8, R2, R3 \t");
184     st.operacion(0,0,0,0,0);
185     st.operacion();

```

```

186 printf("SUB R1, R2, R3 \t");
187 st.operacion(0,0,0,0,0);
188 st.operacion();
189 printf("LI R6, #87 \t");
190 st.operacion(0,0,0,0,0);
191 st.operacion();
192 printf("RET \t");
193 st.operacion(0,0,1,0,0);
194 st.operacion();
195 printf("SUB R1, R2, R3 \t");
196 st.operacion(0,0,0,0,0);
197 st.operacion();
198 printf("LI R6, #87 \t");
199 st.operacion(0,0,0,0,0);
200 st.operacion();
201 printf("RET \t");
202 st.operacion(0,0,1,0,0);
203 st.operacion();
204 printf("B 300 \t");
205 st.operacion(300,0,0,1,0);
206 st.operacion();
207 printf("CALL 889 \t");
208 st.operacion(889,1,0,1,0);
209 st.operacion();
210 printf("ADD R8, R2, R3 \t");
211 st.operacion(0,0,0,0,0);
212 st.operacion();
213 printf("SUB R1, R2, R3 \t");
214 st.operacion(0,0,0,0,0);
215 st.operacion();
216 printf("LI R6, #87 \t");
217 st.operacion(0,0,0,0,0);
218 st.operacion();
219 printf("RET \t");
220 st.operacion(0,0,1,0,0);
221 st.operacion();
222 printf("RET \t");
223 st.operacion(0,0,1,0,0);
224 st.operacion();
225 printf("Get\n");
226 st.get();
227 return 0;
228 }

```

## 2. Simulación (Screenshots)

```

Set()
PC[0] = 37034
PC[1] = 53250
PC[2] = 64874
PC[3] = 7656
PC[4] = 5140
PC[5] = 25729
PC[6] = 490
PC[7] = 30714

CLR
PC[0] = 0
PC[1] = 0
PC[2] = 0
PC[3] = 0
PC[4] = 0
PC[5] = 0
PC[6] = 0
PC[7] = 0

```

Figura 0: con instrucción en la captura de pantalla

```
PC[0] = 0
Instruccion
LI R6, #87      PC[0] = 1
```

Figura 1: con instrucción en la captura de pantalla

```
LI R8, #90      PC[0] = 2
```

Figura 2: con instrucción en la captura de pantalla

```
B 34            PC[0] = 34
```

Figura 3: con instrucción en la captura de pantalla

```
ADD R8, R2, R3  PC[0] = 35
```

Figura 4: con instrucción en la captura de pantalla

```
SUB R1, R2, R3  PC[0] = 36
```

Figura 5: con instrucción en la captura de pantalla

```
CALL 0x61       PC[1] = 61
```

Figura 6: con instrucción en la captura de pantalla

```
LI R6, #87      PC[1] = 62
```

Figura 7: con instrucción en la captura de pantalla

```
LI R8, #90      PC[1] = 63
```

Figura 8: con instrucción en la captura de pantalla

```
CALL 100        PC[2] = 100
```

Figura 9: con instrucción en la captura de pantalla

```
ADD R8, R2, R3  PC[2] = 101
```

Figura 10: con instrucción en la captura de pantalla

```
SUB R1, R2, R3  PC[2] = 102
```

Figura 11: con instrucción en la captura de pantalla

```
LI R6, #87      PC[2] = 103
```

Figura 12: con instrucción en la captura de pantalla

```
RET             PC[1] = 64
```

Figura 13: con instrucción en la captura de pantalla

```
SUB R1, R2, R3  PC[1] = 65
```

Figura 14: con instrucción en la captura de pantalla

```
LI R6, #87      PC[1] = 66
```

Figura 15: con instrucción en la captura de pantalla

```
RET             PC[0] = 37
```

Figura 16: con instrucción en la captura de pantalla

```
B 300           PC[0] = 300
```

Figura 17: con instrucción en la captura de pantalla

```
CALL 889        PC[1] = 889
```

Figura 18: con instrucción en la captura de pantalla

```
ADD R8, R2, R3  PC[1] = 890
```

Figura 19: con instrucción en la captura de pantalla

```
SUB R1, R2, R3  PC[1] = 891
```

Figura 20: con instrucción en la captura de pantalla

```
LI R6, #87      PC[1] = 892
```

Figura 21: con instrucción en la captura de pantalla

```
RET             PC[0] = 301
```

Figura 22: con instrucción en la captura de pantalla

```
RET             PC[7] = 1
```

Figura 23: con instrucción en la captura de pantalla

```
Get
PC[0] = 301
PC[1] = 892
PC[2] = 103
PC[3] = 0
PC[4] = 0
PC[5] = 0
PC[6] = 0
PC[7] = 1
```

*Figura 24: con instrucción en la captura de pantalla*