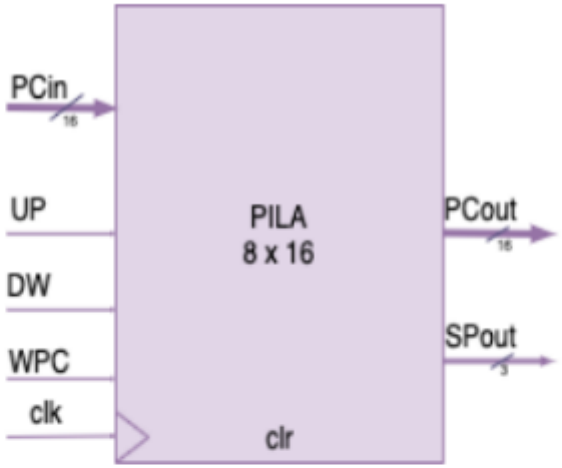
Práctica 9

Pila Hardware 1

1. Programar una clase en C++ que pemita demostrar el funcionamiento del la Pila en Hardware



La clase Pila deberá tener como atributos a cada uno de los buses de entrada y salida de la entidad vista en clase, así como un arreglo de 8 enteros, que serán los contadores de programa y un entero independiente que servirá como el apuntados a la pila. Considere el tamaño de los buses para elegir el tipo de variable correcta. La clase Pila deberá tener los siguientes métodos además de su constructor:

- Método *set()*: Inicializará a los contadores con números aleatorios, así como al apuntado de pila
- Método *get():* Permitirá obtener el contenido de todos los contadores de programa, mediante su direccionamiento a través del apuntador de pila.
- Método *operacion (PCin, UP, DW, WPC, clr)*: Realizará la operación correspondiente a una combinación de parámetros válida con respecto a la tabla de comportamiento. Este método no imprimirá nada en pantalla y debe considerar el caso en el que se reciba una combinación de parámetros incorrecta.
- Método *operacion ():* Realizará la lectura del contador de programa al cual esté apuntando el stack pointer.
- Los únicos métodos que mostrarán un resultado en la pantalla son *get()* y *operación()*.

clr	clk	WPC	UP	DW	Operación
1	x	x	x	x	SP = 0 PILA (0, 1, …, 7) = 0
0	↑	0	0	0	SP = SP PILA(SP) ++
0	↑	1	0	0	SP = SP PILA(SP) = PCin
0	↑	1	1	0	SP ++ PILA(SP) = PCin
0	↑	0	0	1	SP - - PILA(SP) ++
x	x	x	x	x	PCout = PILA(SP)

2. Una vez implementada la clase pila, deberá hacer las llamadas a los métodos correspondientes, de tal manera que permitan "ejecutar " las siguientes instrucciones

1. LI R6, #87
2. LI R8, #90
3. B 34
4. ADD R8, R2, R3
5. SUB R1, R2, R3
6. CALL 0x61
7. LI R6, #87
8. LI R8, #90
9. CALL 100
10. ADD R8, R2, R3
11. SUB R1, R2, R3
12. LI R6, #87
13. RET
14. SUB R1, R2, R3
15. LI R6, #87
16. RET
17. B 300
18. CALL 889
19. ADD R8, R2, R3
20. SUB R1, R2, R3
21. LI R6, #87
22. RET
23. RET
24. Ejecute el método get()

Dado que el conteo de la pila de instrucciones comienza en cero la instrucción 1 aquí mostrada pasará a ser la operación 0 en el conteo de pila, la operación 2 pasará a ser la operación 1, y así sucesivamente hasta completar las instrucciones aquí mostradas.

2.1. LI R6, #87

```
D:\ESCOM\ARQUITECTURA\Practica9>pila.exe

SP: 0
Pila[0]: 0
```

Observamos que el StackPointer(SP) se encuentra en 0, esto es debido a que previo a ejecutar la primer instrucción se a ejecutado un CLR en la pila para comprender mejor su funcionamiento.

Notamos que, el contador de programa apunta a la dirección de SP, por lo cual podemos decir que esta "ejecutando" la instrucción 0.

2.2. LI R6, #87

```
D:\ESCOM\ARQUITECTURA\Practica9>pila.exe

SP: 0
Pila[0]: 1
```

Ahora el contador de programa se ha aumentado en uno, mientras que el StackPointer ha mantenido su valor en 0. En este caso podemos decir que se está ejecutando la instrucción 1.

2.3. B 34

```
D:\ESCOM\ARQUITECTURA\Practica9>pila.exe

SP: 0
Pila[0]: 2
```

El contador de programa se ha aumentado una vez más, mientras que StackPointer permanece igual. En este caso se ejecuta la instrucción 2.

2.4. ADD R8, R2, R3

```
D:\ESCOM\ARQUITECTURA\Practica9>pila.exe

SP: 0
Pila[0]: 34
```

Dado que la instrucción anterior era B, el contador de programa ha saltado hasta la instrucción 34, no obstante, en la ejecución de este programa tomaremos las instrucciones subsecuentes a B 34, como si estas estuvieran seguidas por ellas, es decir que en este caso la instrucción 4 pasaría a ser la 34.

2.5. SUB R1, R2, R3

```
D:\ESCOM\ARQUITECTURA\Practica9>pila.exe
SP: 0
Pila[0]: 35
```

En este caso, como se menciono en el punto anterior simularemos que la instrucción 5 es la que ocupa el lugar 35, por lo cual decimos que ha sido ejecutada la operación SUB.

2.6. CALL 0x61

```
D:\ESCOM\ARQUITECTURA\Practica9>pila.exe
SP: 0
Pila[0]: 36
SP: 1
Pila[1]: 97
```

Observamos que esta instrucción es la 6, por lo cual ocupara el lugar 36, dado que se trata de una instrucción CALL veremos que se aumenta SP, además el contador de programa (Pila) en la localidad SP es igual al valor hexadecimal que se introdujo, o bien Pila[SP] = PCin.

2.7. LI R6, #87

```
D:\ESCOM\ARQUITECTURA\Practica9>pila.exe
SP: 1
Pila[1]: 97
```

Ejecutamos la operación 7 como si se tratara de la que se encuentra en la posición 97. Observamos que, dado que se trata de una instrucción LI, SP no se mueve.

2.8. LI R8, #90

```
D:\ESCOM\ARQUITECTURA\Practica9>pila.exe
SP: 1
Pila[1]: 98
```

Ejecutamos la operación 8 simulando que ocupa el lugar 98. Dado que se trata de un LI el SP no se ve afectado.

2.9. CALL 100

```
D:\ESCOM\ARQUITECTURA\Practica9>pila.exe
SP: 1
Pila[1]: 99
SP: 2
Pila[2]: 100
```

Ejecutamos la instrucción 9, pero dado que se trata de una instrucción CALL, notamos que se aumentará el SP y se saltará hasta la instrucción 100.

2.10 ADD R8, R2, R3

```
D:\ESCOM\ARQUITECTURA\Practica9>pila.exe
SP: 2
Pila[2]: 100
```

Ejecutamos la instrucción 10 simulando que ocupa el lugar de la instrucción 100. Dado que se trata de una instrucción ADD, SP no se ve afectado.

2.11. SUB R1, R2, R3

```
D:\ESCOM\ARQUITECTURA\Practica9>pila.exe
SP: 2
Pila[2]: 101
```

Ejecutamos la instrucción 11 simulando que ocupa el lugar de la instrucción 101. Dado que se trata de una instrucción SUB, SP no se ve afectado.

2.12. LI R6, #87

```
D:\ESCOM\ARQUITECTURA\Practica9>pila.exe
SP: 2
Pila[2]: 102
```

Ejecutamos la instrucción 12 simulando que se trata de la instrucción 102. Al ser una instrucción LI, SP no se modifica.

2.13. RET

```
D:\ESCOM\ARQUITECTURA\Practica9>pila.exe
SP: 2
Pila[2]: 103
SP: 1
Pila[1]: 100
```

Ejecutamos la instrucción 13 simulando ser la 103. Al ser un RET, SP se reduce en una unidad, por lo cual regresará a 1. Además, dado que Pila[SP] = 99 en la ultima iteración de SP = 1, ahora existe el aumento provocado por RET, es decir Pila[SP] = 100.

2.14. SUB R1, R2, R3

```
D:\ESCOM\ARQUITECTURA\Practica9>pila.exe
SP: 1
Pila[1]: 100
```

Ejecutamos la instrucción 14 como si se tratara de la 100, esto con el fin de dar continuidad al programa. Dado que se trata de una instrucción SUB, no se afecta al SP.

2.15. LI R6, #87

```
D:\ESCOM\ARQUITECTURA\Practica9>pila.exe
SP: 1
Pila[1]: 101
```

Ejecutamos la instrucción 15 como si se tratara de la 101. Dado que se trata de una instrucción LI, el SP no se ve afectado.

2.16. RET

```
D:\ESCOM\ARQUITECTURA\Practica9>pila.exe
SP: 1
Pila[1]: 102
SP: 0
Pila[0]: 37
```

Ejecutamos la instrucción 16 como si se tratara de la 102. Dado que se trata de una instrucción RET, observamos que SP se redujo a 0, mientras que Pila[0] se aumentó en 1 desde la última iteración.

2.17. B 300

```
D:\ESCOM\ARQUITECTURA\Practica9>pila.exe
SP: 0
Pila[0]: 37
SP: 0
Pila[0]: 300
```

Ejecutamos la instrucción 17 como si se tratara de la 37, además notamos que, dado que se trata de una instrucción B, el contador del programa PC[SP] = 300.

2.18 CALL 889

```
D:\ESCOM\ARQUITECTURA\Practica9>pila.exe
SP: 0
Pila[0]: 300
SP: 1
Pila[1]: 889
```

Ejecutamos la instrucción 18 como si de la 300 se tratara. Al ser una instrucción CALL, el SP aumenta en 1, mientras que Pila[1] = 889.

2.19. ADD R8, R2, R3

```
D:\ESCOM\ARQUITECTURA\Practica9>pila.exe
SP: 1
Pila[1]: 889
```

Ejecutamos la instrucción 19 como si se tratara de la 889. Dado que es una instrucción ADD, el SP no cambia.

2.20. SUB R1, R2, R3

```
D:\ESCOM\ARQUITECTURA\Practica9>pila.exe
SP: 1
Pila[1]: 890
```

Ejecutamos la instrucción 20 como si fuera la 890 Dado que se trata de una instrucción SUB, el SP no se ve afectado.

2.21. LI R6, #87

```
D:\ESCOM\ARQUITECTURA\Practica9>pila.exe
SP: 1
Pila[1]: 891
```

Ejecutamos la instrucción 21 como si se tratara de la 891. Dado que se trata de una instrucción LI, el SP no se ve afectado.

2.22 RET

```
D:\ESCOM\ARQUITECTURA\Practica9>pila.exe
SP: 1
Pila[1]: 892
SP: 0
Pila[0]: 301
```

Ejecutamos la instrucción 22 como si se tratara de la 892. Dado que se trata de un RET, el SP se reduce a 0, por lo cual SP[0] sufre un aumento, es decir que de 300 pasa a 301.

2.23 RET

```
D:\ESCOM\ARQUITECTURA\Practica9>pila.exe
SP: 0
Pila[0]: 301
SP: 7
Pila[7]: 1
```

Ejecutamos la instrucción 23, dado que esta instrucción es un RET el SP se reduce en 1, sin embargo, notamos que al estar en 0 existe un desborde, por lo cual en lugar de reducir a SP=-1, esta toma el valor de 7, el cual es el ultimo elemento de la pila. Además Pila[7] se incremente en 1, por lo cual obtenemos a Pila[7] = 1.

2.24 get

```
D:\ESCOM\ARQUITECTURA\Practica9>pila.exe
Pila[0]: 301
Pila[1]: 892
Pila[2]: 103
Pila[3]: 0
Pila[4]: 0
Pila[5]: 0
Pila[6]: 0
Pila[7]: 1
```

Observamos que las localidades del contador del programa se han quedado en su ultimo valor de modificación, además, tal como se esperaba, el desborde ha afectado el ultimo elemento de la pila.