

# Proyecto final (fase #3)

Equipo #1 :

Abraham Magaña Hernández - 220791217

Ángel Damián Raúl García Guevara - 220791063

Samuel Espinoza Sucilla - 214017739

Seminario de Arquitectura en Computación. López Arce Delgado Jorge Ernesto.  
D13 Martes y Jueves 7:00 AM a 9:00 AM.

Código

# Código de Verilog (Concatenación)

La concatenación está implementada en el Top Level y cumple la función de unir los 5 bits más significativos del sumador del PC con los 28 bits menos significativos de la instrucción.

```
/*Concatenacion T11(  
    .A1(C_concatenacion),  
    .A2(C_PC_Adder_out),  
    .nadd(C_IDEX_shift_left_in)  
);*/  
  
assign C_IDEX_shift_left_in = {C_PC_Adder_out[31:28],C_concatenacion[27:0]};
```

# Código de Verilog (Shift left)

Lo que hace el shift left es concatenar la entrada (Shift\_in) con dos bits en 0. La entrada del shift\_left entra siendo de 26 bits y sale siendo de 28

```
Shift_left_2.v X
C: > Users > PC > Desktop > Fase 3 > Shift_left_2.v
1  `timescale 1ns/1ns
2
3  module Shift_left_2(
4      input [25:0]shift_in,
5      output [27:0]shift_out
6  );
7
8      assign shift_out = {shift_in,2'b00};
9
10 endmodule
11
```

# Código de Verilog (Implementación en la Unidad de control)

La implementación de la operación *Jump dentro de la Unidad de Control*

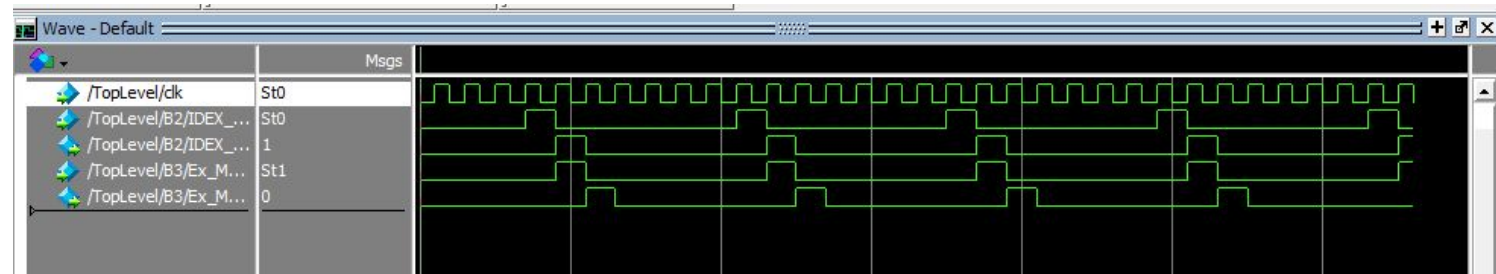
```
Shift_Left_2.v  UnidadDeControl.v X
C:\Users\PC\Desktop\Fase 3 > UnidadDeControl.v
120     AluSrc = 0;
121     RegWrite = 1;
122 end
123 //Jump
124 6'b0000010:
125 begin
126     jump = 1;
127     //RegDst = 0;
128     //Branch = 0;
129     //MemRead = 0;
130     //MemToReg = 0;
131     //AluOp = 0;
132     //MemToWrite = 0;
133     //AluSrc = 0;
134     //RegWrite = 0;
135 end
136 endcase
137 end
138
139 endmodule
140
```

# Simulación Shift\_left

Wave - Default			
		Msgs	
+	/Shift_left_2/shift_in	0000000000000000...	0000000000000000000000000000
	/Shift_left_2/shift_out	0000000000000000...	0000000000000000000000000000

# Simulación de la instrucción Jump

Memory Data - /TopLevel/BR/Reg - Default	
0	8
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20
21	21
22	22
23	23
24	24
25	25
26	26
27	27
28	28
29	29
30	30
31	16



Algoritmo



# Ecuación

El algoritmo se creo a partir de la ecuación del interés compuesto. Cabe destacar que esta misma ecuación puede servir para modelar crecimiento poblacional, de la cual se tomaron datos para crear el algoritmo.

- La ecuación es la siguiente:

$$P_n = P_{n-1}(1 + c) + S$$

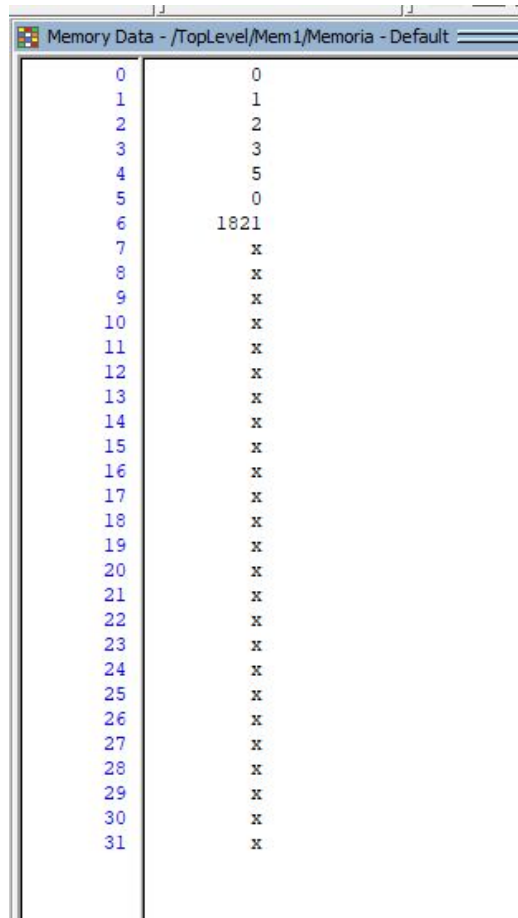
Dónde:

- $P_{n-1} = 1$
- $c = 1$
- $S = 3$

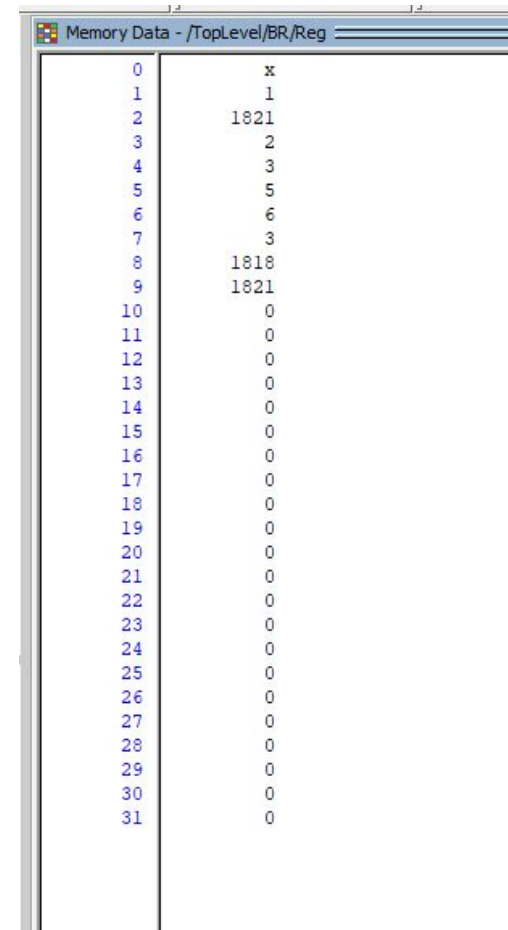
# Tabla de datos

Memoria de Instrucciones (Dirección)	Numero De Instrucción	Instrucción	Banco de registros		Memoria de datos	
			Dirección	Data	Dirección	Data
0	0	Lw \$1, \$2, #0	\$0	0	\$0	X
4	1	Lw \$1, \$3, #1	\$1	1	\$1	1 (P0)
8	2	Lw \$1, \$4, #2	\$2	X	\$2	2 (C)
12	3	Lw \$1, \$5, #3	\$3	X	\$3	3 (S)
16	4	Lw \$1, \$6, #4	\$4	X	\$4	5 (n)
20	5	Addi \$3, \$7, #1	\$5	X	\$5	0 (Cont)
24	6	Beq \$5, \$6, #6	\$6	X	\$6	X (Pn)
28	7	Mul \$2, \$7, \$8	\$7	X	\$7	X
32	8	Add \$4, \$8, \$9	\$8	X	\$8	X
36	9	Addi \$9, \$2, #0	\$9	X	\$9	X
40	10	Addi \$6, \$6, #1	...	...	...	...
44	11	J #6				
48	12	Sw \$1, \$9, #5				

# Simulación del algoritmo (Archivos de memoria)

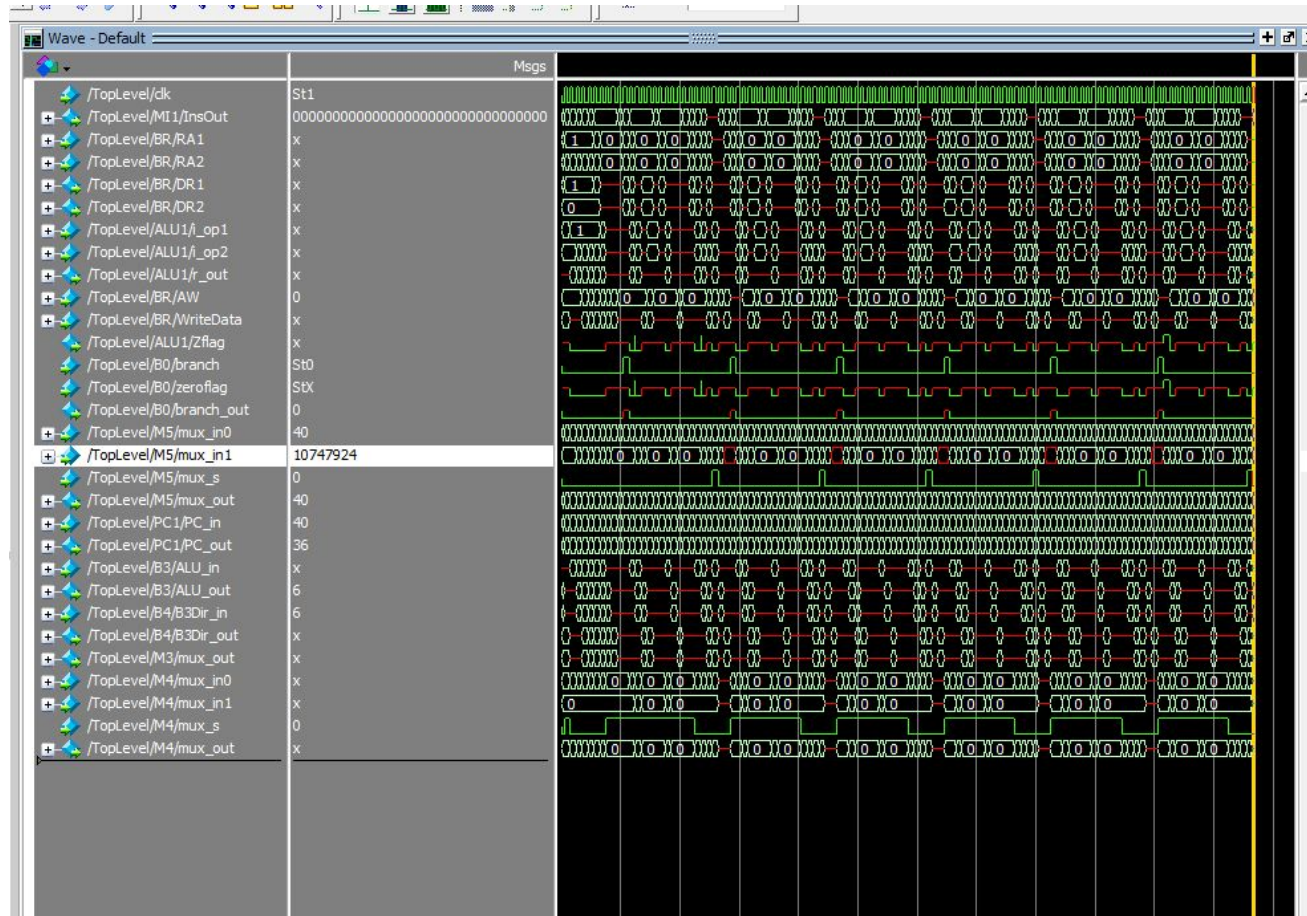


0	0
1	1
2	2
3	3
4	5
5	0
6	1821
7	x
8	x
9	x
10	x
11	x
12	x
13	x
14	x
15	x
16	x
17	x
18	x
19	x
20	x
21	x
22	x
23	x
24	x
25	x
26	x
27	x
28	x
29	x
30	x
31	x



0	x
1	1
2	1821
3	2
4	3
5	5
6	6
7	3
8	1818
9	1821
10	0
11	0
12	0
13	0
14	0
15	0
16	0
17	0
18	0
19	0
20	0
21	0
22	0
23	0
24	0
25	0
26	0
27	0
28	0
29	0
30	0
31	0

# Simulación del algoritmo (Dataflow)



Decoder

# Decoder



Es importante denotar que esta interfaz gráfica recibe los datos en decimal, nos da una pre visualización de la instrucción en su formato de bits, bloquea las instrucciones erróneas, resetea archivos de memoria y también el archivo que contiene las instrucciones escritas en su formato de ensamblador.