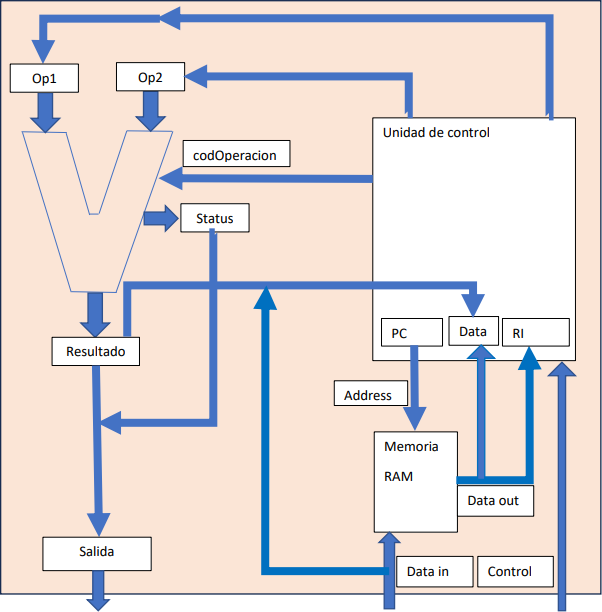
**Practico Diseño de Sistemas Digitales Parte 3**

Alumnos: Chamorro, Juan – Valori, Tomás

Facultad de Ciencia y Tecnología, Universidad Autónoma de Entre Ríos

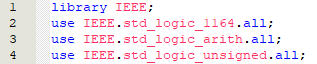
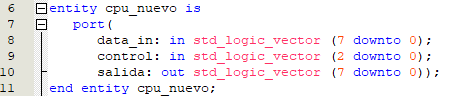
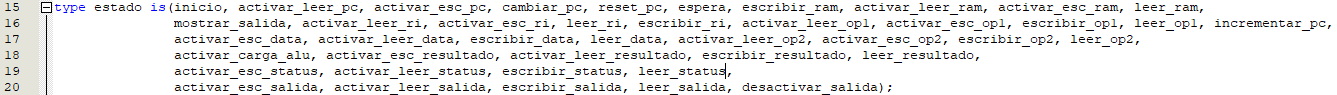
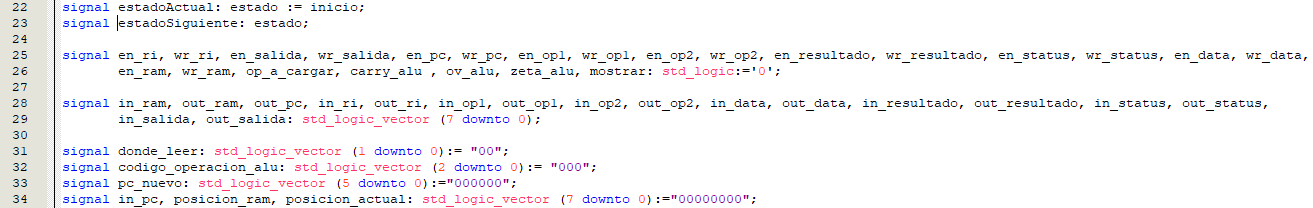
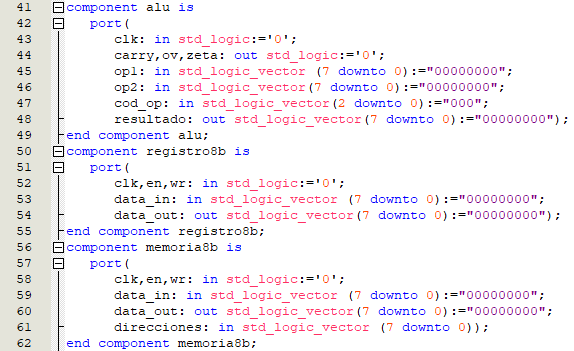
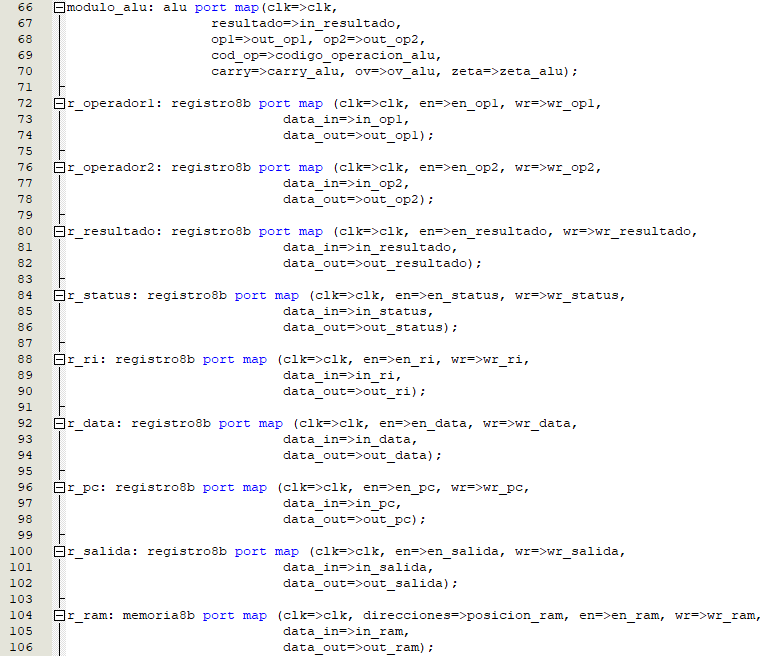
Asignatura: Diseño de Sistemas Digitales  
Profesor: Eduardo Velazquez, Gisela Giménez  
Fecha de vencimiento:

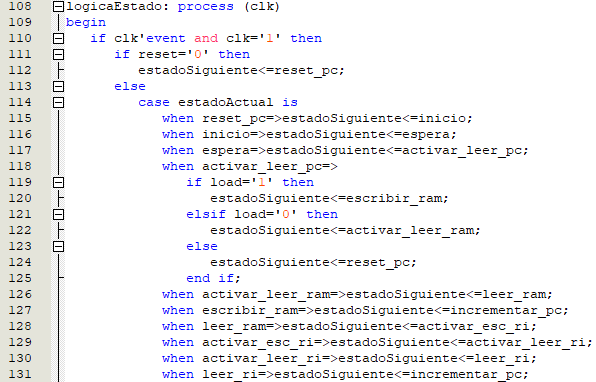
La misión del trabajo fue realizar, en descripción VHDL, un sistema microprocesado embebido de la siguiente manera:   
  
 En el cual se pedían los siguientes registros de 8 bits: Op1 – Op2 – Resultado – Status – Salida - PC (contador de programa) – RI (registro de instrucción) – Data (registro de datos) – Memoria RAM de 64 celdas y 8 bits por celda – ALU de 8 bits con 7 operaciones (suma, resta AND, OR, SHL, SHR, NOT), con salida de resultado y registro de status con banderas resultantes de la operación realizada (Cy, OV, Z).   
 Dichos elementos deberían ser implementados como componentes al sistema general.  
 El ciclo de instrucción era de la siguiente manera:  
 - Leer instrucción y cargarla en RI.  
 - Decodificar la instrucción presente en RI.  
 - Ejecutar instrucción.  
 - Incrementar PC + 1.  
  
 Se piden 15 instrucciones específicas que veremos punto por punto en la realización del proyecto.  
 La señal de control está compuesta por las señales de clock (1 bit), load (1 bit) y reset (1 bit).

Cuando reset está en cero el PC se coloca a cero.

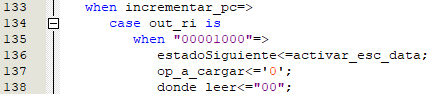
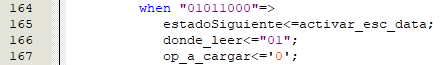
Cuando Load está en cero, se ejecuta el programa en forma normal.

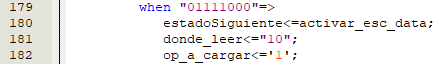
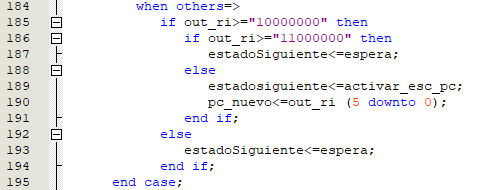
Cuando Load está en uno, por cada clock de reloj se carga un valor en la memoria por la entrada Data\_in.

Por último, para corroborar que todo funciona como corresponde, se pide realizar un testbench, por medio de un programa que utilice todas las instrucciones.  
  
  
 **REALIZACION DEL PROYECTO  
***\*Estas fueron las librerías que utilizamos.*  
  
*\*Entidad con sus respectivas entradas y salidas como fueron pedidas.*  
 A partir de ahora veremos la arquitectura del programa, cabe aclarar que dicho programa fue realizado con una máquina de estado de la forma que vimos en clases, con sus respectivas “*Lógica de Estado*” y “*Lógica de salida*”.  
  
  
*\*Estados utilizados.*  
  
  
*\*Señales utilizadas.   
  
\*Alias utilizados para la separación de la señal de control.  
  
  
\*Implementación de los programas (alu, registro8b y memoria8b) como componentes.***A continuación comienza el funcionamiento del programa en sí:**  
  
*\*Unión entre entradas y salidas de los componentes con las respectivas señales de nuestro programa general.*

  
\**Comienzo de la lógica de estado, con las instrucciones básicas para el funcionamiento general del programa, como la utilización del control (clk, load, reset) además de la utilización básica de los componentes “memoria8b” y “registro8b).*

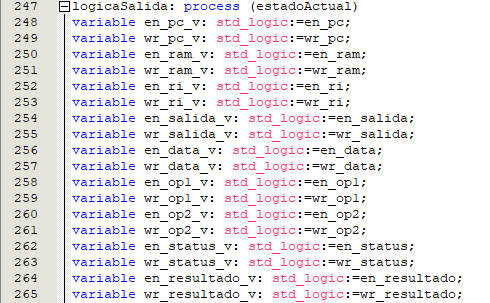
**A continuación, veremos las instrucciones pedidas, luego de eso, como fueron implementadas en la lógica de estado para su utilización.****

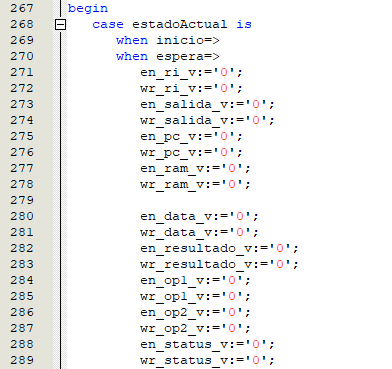
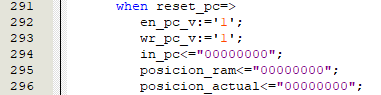
  
*\* Instrucción 1.* *\* Instrucción 2.* *\* Instrucción 3. (podemos observar la utilización del componente “alu”).*  
 *\* Instrucción 4.* *\* Instrucción 5.* *\*Instrucción 6.* *\* Instrucción 7.* *\*Instrucción 8.* *\*Instrucción 9.* *\*Instrucción 10.* *\*Instrucción 11.* *\*Instrucción 12.*

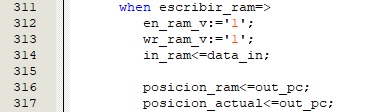
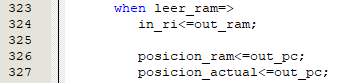
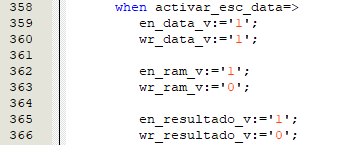
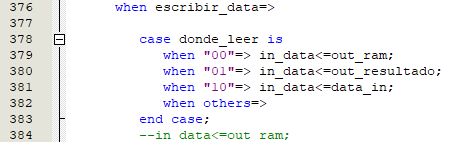
  
\**Instrucción 13.  
  
\*Instrucción 14.  
  
\*Instrucción 15.*

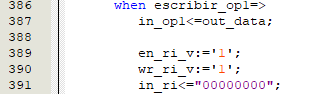
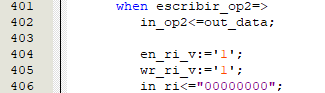
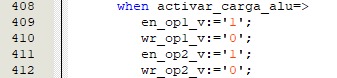
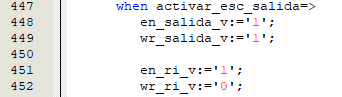
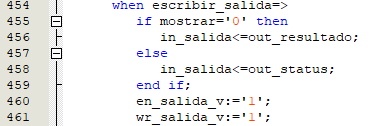
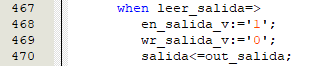
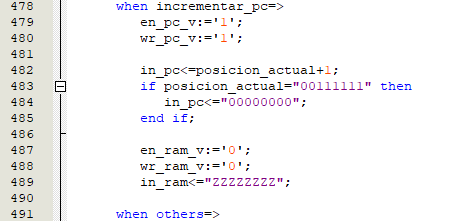
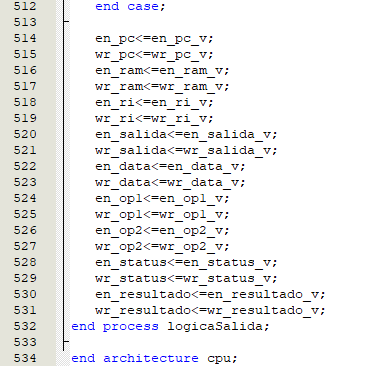
*  
\*Aquí observamos los estados restantes de la lógica de estados, donde utilizamos las señales para escribir o leer los operadores, enviamos los resultados a la salida, etc.*

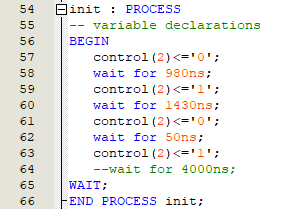
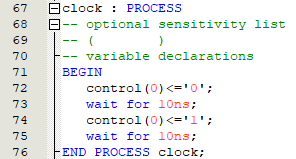
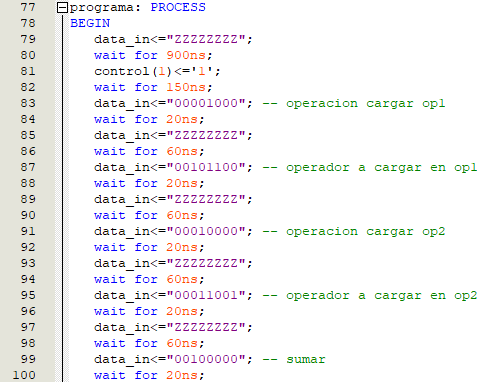
A partir de este punto observaremos todo el funcionamiento de la lógica de salida.

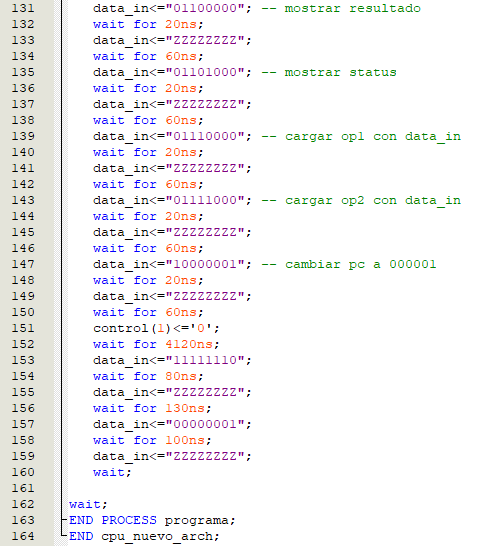
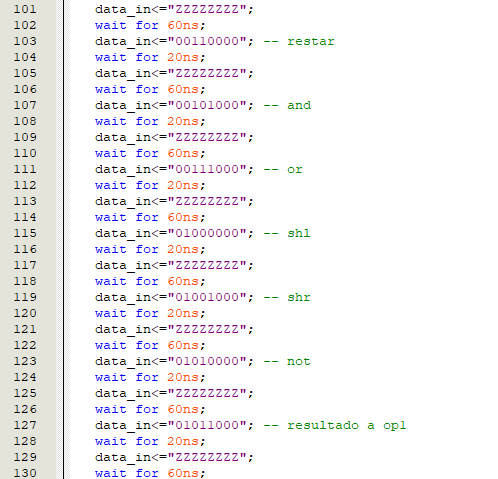
*  
\*Variables que vamos a utilizar.*

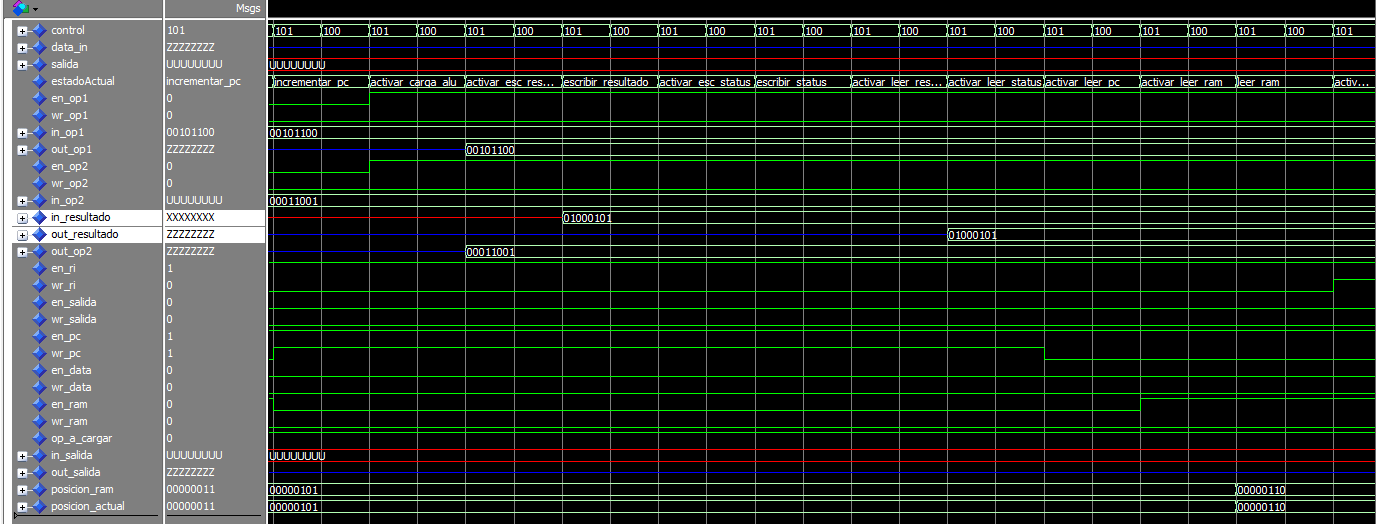
*  
\*Configuración de salidas en el estado “estadoActual”.  
  
  
\*Configuración de salidas en el estado “reset\_pc”.*

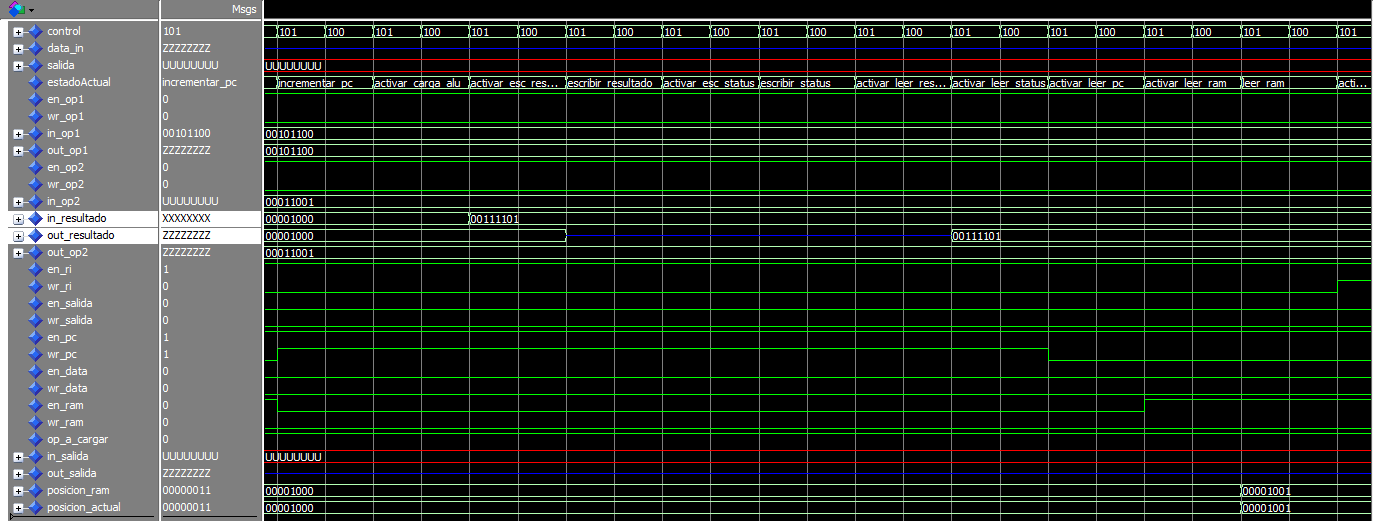
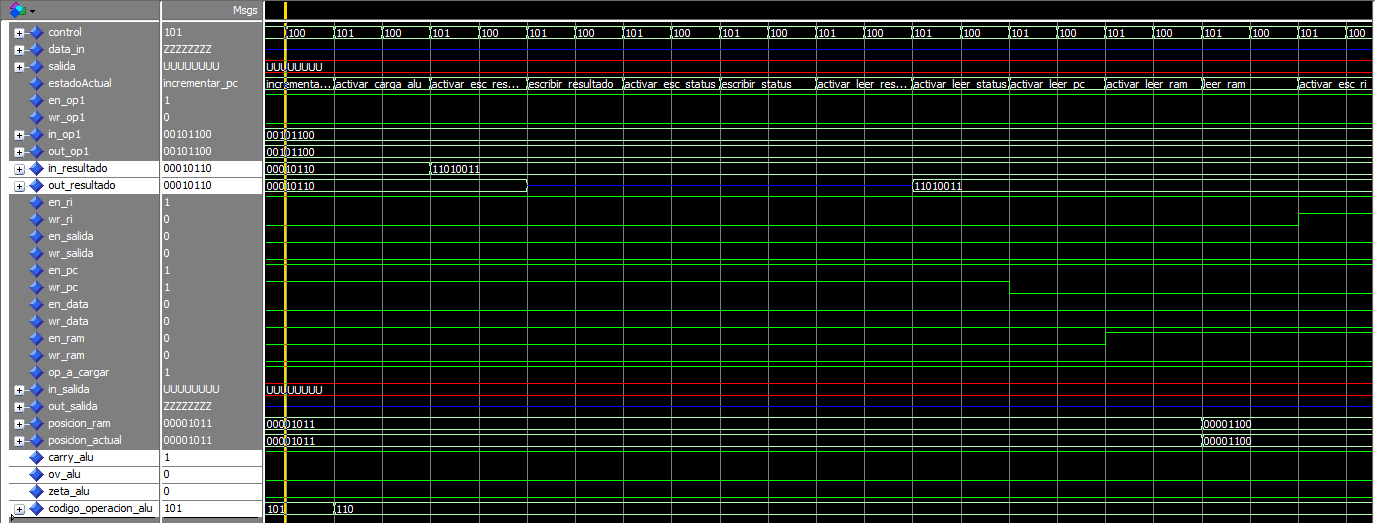
*  
\*Configuración de salidas en el estado “activar\_leer\_pc”.  
  
  
\*Configuración de salidas en el estado “activar\_escribir\_pc”.  
  
  
\*Configuración de salidas en el estado “cambiar\_pc”.  
  
  
\*Configuración de salidas en el estado “escribir\_ram”.  
  
  
\*Configuración de salidas en el estado “activar\_leer\_ram”.  
  
  
\*Configuración de salidas en el estado “leer\_ram”.  
  
  
\*Configuración de salidas en el estado “activar\_leer\_ri”.  
  
  
\*Configuración de salidas en el estado “activar\_esc\_ri”.  
  
  
\*Configuración de salidas en el estado “leer\_ri”.  
  
  
\*Configuración de salidas en el estado “activar\_esc\_op1”.  
  
  
\*Configuración de salidas en el estado “activar\_leer\_op1”.  
  
  
\*Configuración de salidas en el estado “activar\_esc\_op2”.  
  
  
\*Configuración de salidas en el estado “activar\_leer\_op2”.  
  
  
\*Configuración de salidas en el estado “activar\_esc\_data”.  
  
  
\*Configuración de salidas en el estado “activar\_leer\_data”.  
  
  
\*Configuración de salidas en el estado “leer\_data”.  
  
  
\*Configuración de salidas en el estado “escribir\_data”.*

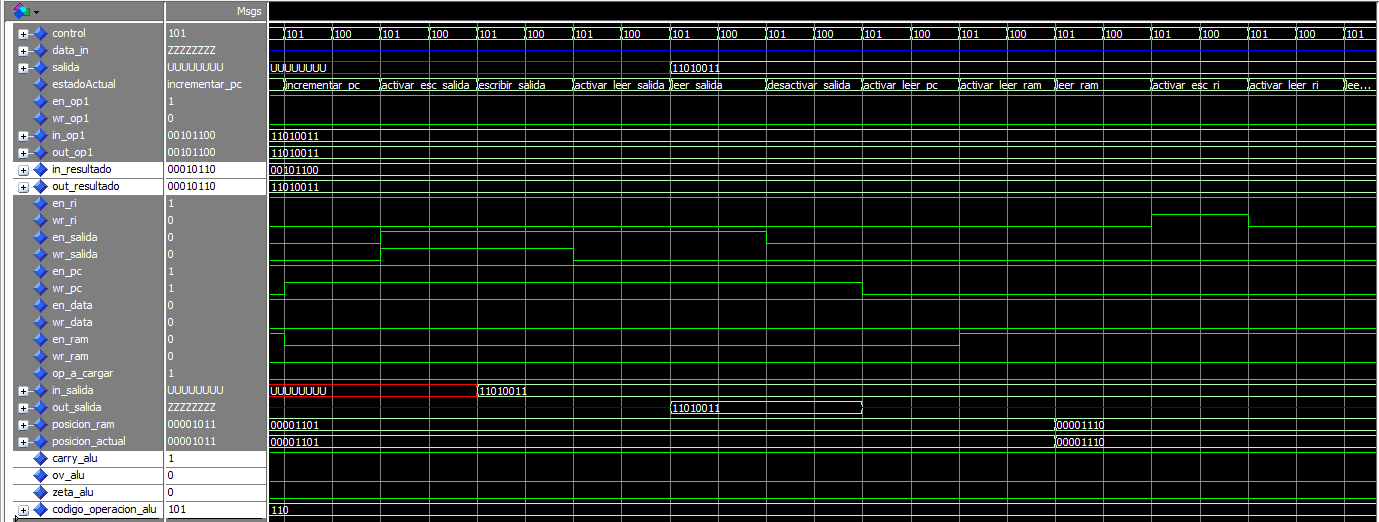
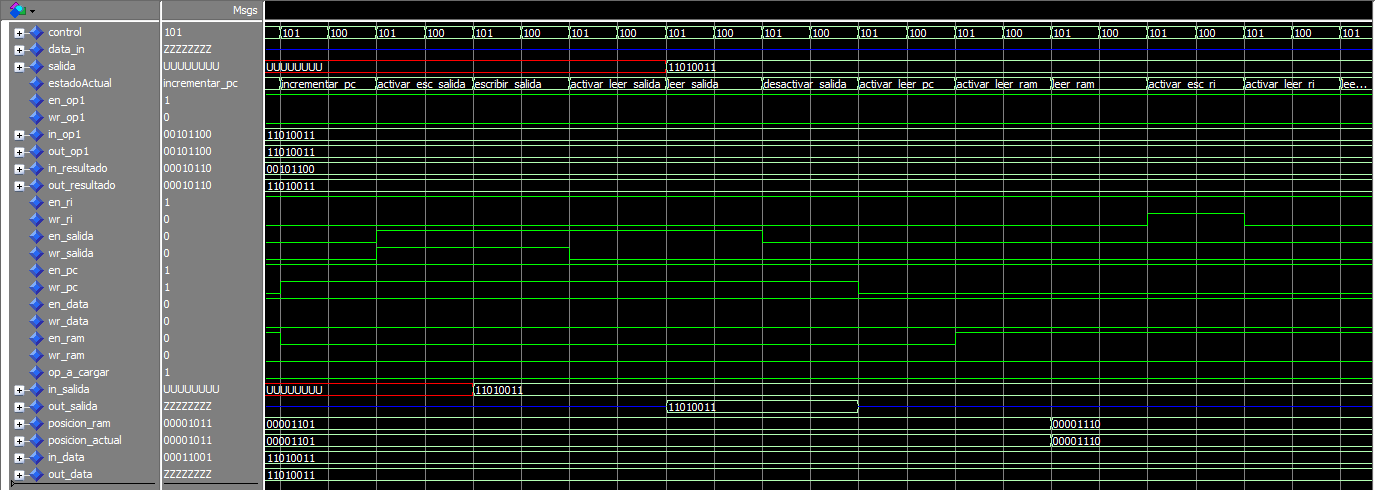
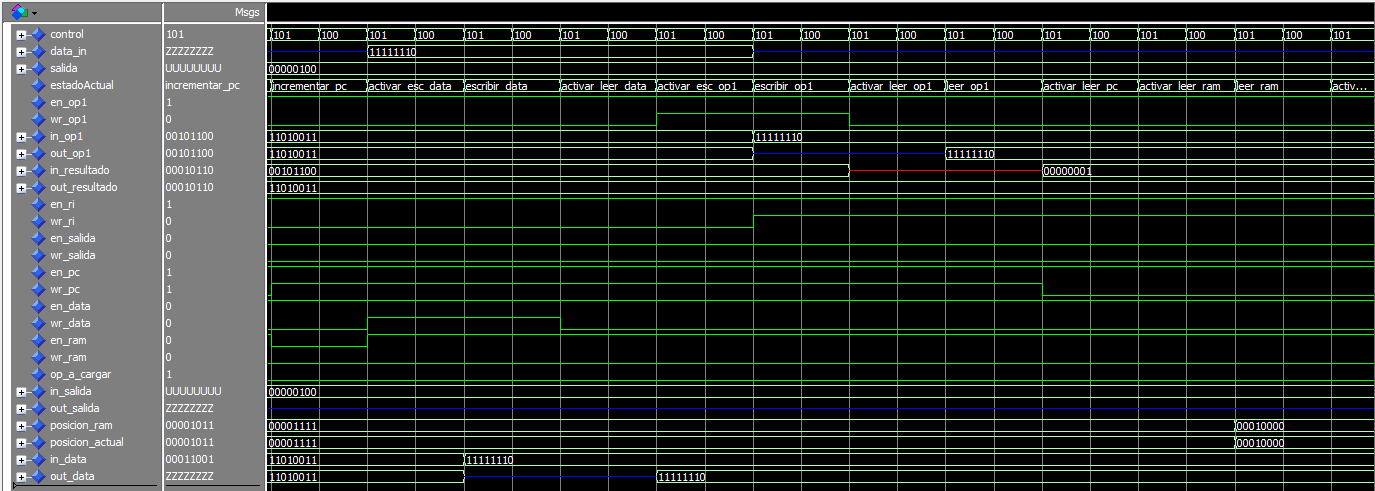
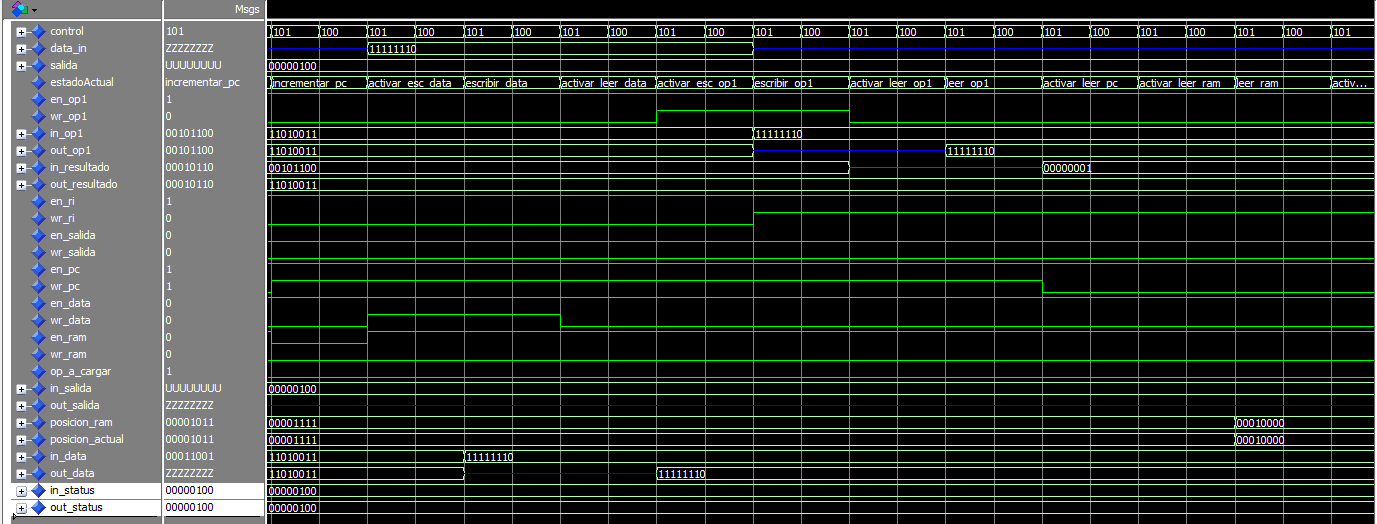
*  
\*Configuración de salidas en el estado “escribir\_op1”.  
  
  
\*Configuración de salidas en el estado “leer\_op1”.  
  
  
\*Configuración de salidas en el estado “leer\_op2”.  
  
  
\*Configuración de salidas en el estado “escribir\_op2”.  
  
  
\*Configuración de salidas en el estado “activar\_carga\_alu”.  
  
  
\*Configuración de salidas en el estado “activar\_esc\_resultado”.  
  
  
\*Configuración de salidas en el estado “activar\_esc\_status”.  
  
  
\*Configuración de salidas en el estado “activar\_leer\_resultado”.  
  
  
\*Configuración de salidas en el estado “activar\_leer\_status”.  
  
  
\*Configuración de salidas en el estado “escribir\_resultado”.  
  
  
\*Configuración de salidas en el estado “escribir\_status”.  
  
  
\*Configuración de salidas en el estado “leer\_resultado”.  
  
  
\*Configuración de salidas en el estado “leer\_status”.  
  
  
\*Configuración de salidas en el estado “activar\_esc\_salida”.  
  
  
\*Configuración de salidas en el estado “escribir\_salida”.  
  
  
\*Configuración de salidas en el estado “activar\_leer\_salida”.  
  
  
\*Configuración de salidas en el estado “leer\_salida”.  
  
  
  
  
  
\*Configuración de salidas en el estado “desactivar\_salida”.  
  
  
\*Configuración de salidas en el estado “incrementar\_pc”.  
  
  
\*En esta parte indicamos que variable corresponde a determinada señal, luego de eso terminamos con la lógica de salida. Aquí termina toda la configuración del programa.*

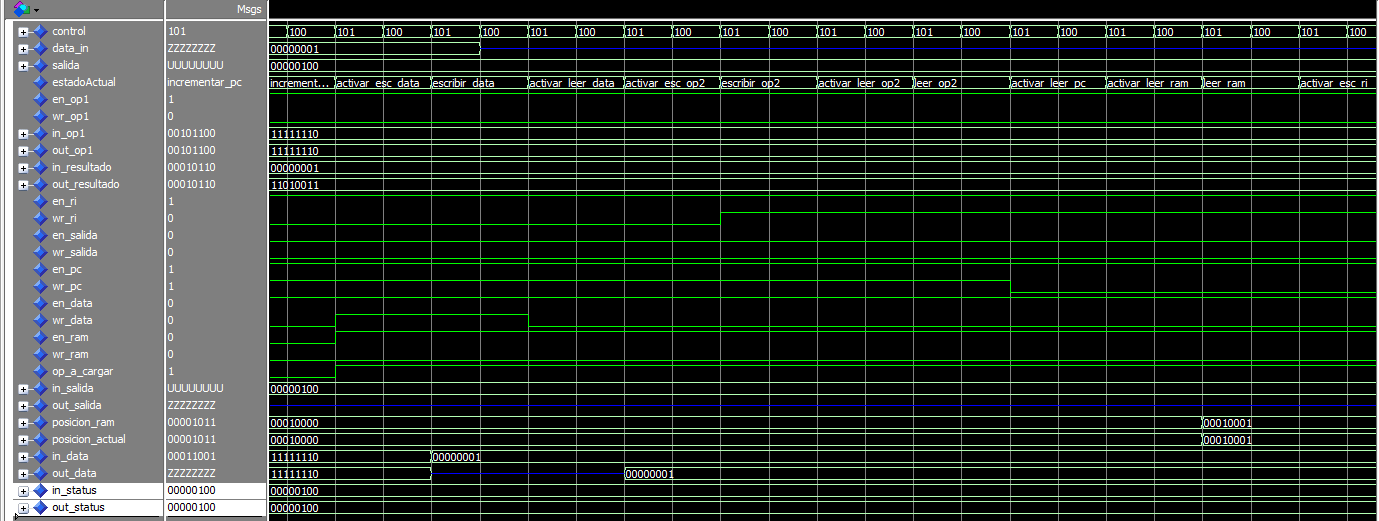
Configuración de la simulación TestBench:  
**  
*\*Proceso “init”, donde controlamos el reset.*  
  
  
*\*Proceso “clock” donde vamos activando el clk.*  
  
**Las siguientes imágenes corresponden al proceso del programa en general, donde le** **pedimos simular todas las instrucciones que fueron pedidas.**



**Resultados de la simulación:  
 *****\*Instrucción 1.  
  
\*Instrucción 2.  
  
\*Instrucción 3.  
  
Instrucción 4.*

*  
\*Instrucción 5.  
  
\*Instrucción 6.  
  
\*Instrucción 7.  
  
\*Instrucción 8.*

*  
\*Instrucción 9.  
  
\*Instrucción 10.  
  
\*Instrucción 11.  
  
\*Instrucción 12.*

*  
\*Instrucción 13.  
  
\*Instrucción 14.  
  
\*Instrucción 15.*