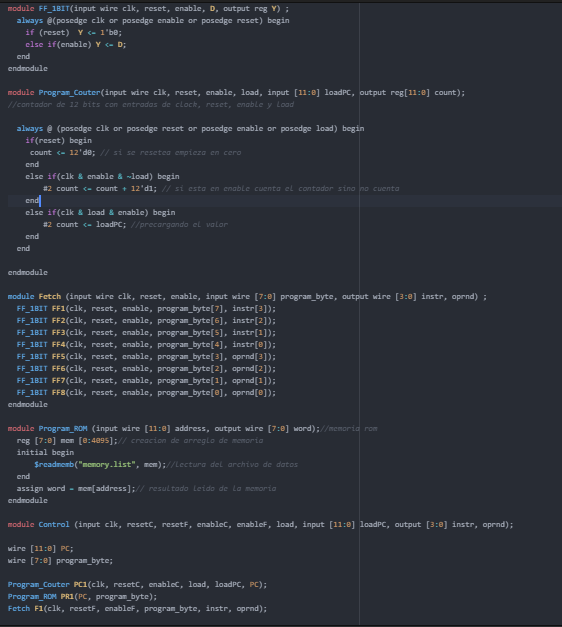
**Laboratorio #10**

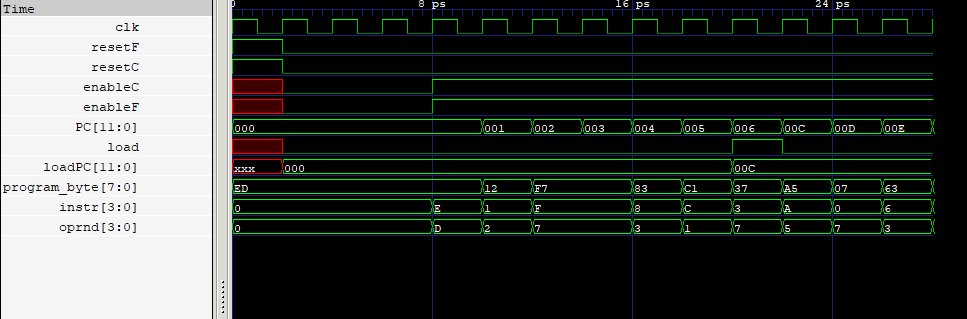
**Enlace al repositorio de GitHub:** <https://github.com/gar19421/Lab-10.git>

**Ejercicio 1:**

En este ejercicio se realizó un fragmento del microprocesador a trabajar en el laboratorio, en este ejercicio como se puede observar se implemento el program counter utilizando el contador codificado en laboratorios previos, el fetch el cual es un flip flop de 8bits que se implemento con 8 flipflops de 1bit codificados anteriormente. Por aparte tambien se implemento una program ROM junto con una lista de memoria de 32 direcciones, todo esto se implemento en un modulo control para manejar el fragmento del microprocesador.



Como se puede observar en el diagrama de timing, en cada flanco de reloj el contador aumenta y se obtiene el valor de memoria en dicha posición y cuando se activa el load en 1 se carga el valor de memoria en la posición precargada identificada como loadPC.

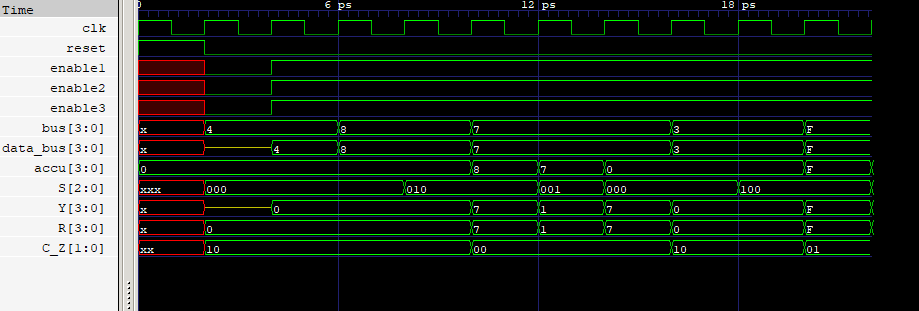


**Ejercicio 2:**

En este fragmento del microprocesador, como se puede observar, se implemento la ALU del microprocesador, de una ALU codificada en laboratorios previos, con la diferencia que en este se realizaban las 5 instrucciones propias del ALU del proyecto. Así mismo se le agregó un control de 2 bits que correspondía al bit de cero y carry on de las operaciones del ALU identificado como C\_Z. También se implemento el acumulador como un flip flop de 4 bits y un bus driver que se implemento 2 veces el cual consistía en un buffer triestado de 4 bits codificado en anteriores laboratorios. Todo esto unido en el modulo de control del fragmento del microprocesador.

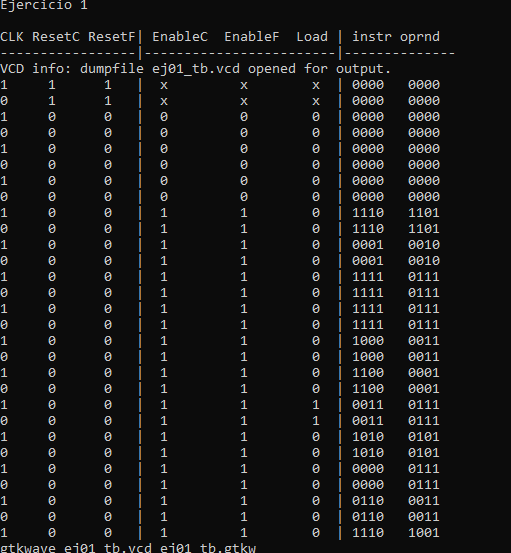
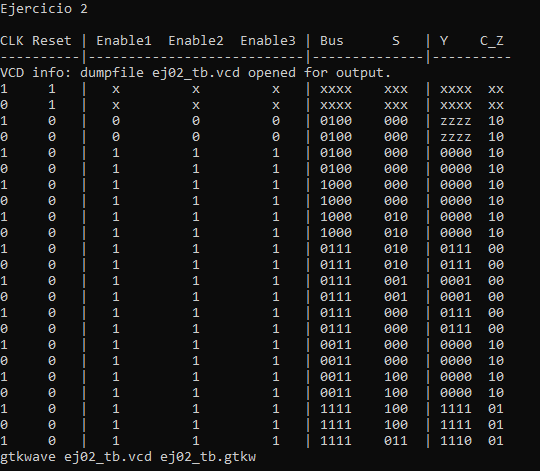


Como se puede observar en el diagrama de timing, se muestra que mientras los enables estén en cero la salida de la data\_bus se encuentra en alta impedancia al igual que Y. Por otro lado, se puede observar que respecto a la entrada del bus que representa la entrada del bus driver al ALU, dependiendo de la operación S que se realice la salida Y y R se actualiza. Así mismo la señal del C\_Z es 10 cuando la salida R, Y es cero y es 01, cuando existe un overflow en la salida.



**Otras evidencias (Screenshots):**

- Pruebas en consola:

- Testbech:

