Informe de prácticas

CPU monociclo con Entrada / Salida

Carlos Gregorio Martín Pérez
Georg Sperl
Sara Báez
Leonor Priego García

Codificación elegida para cada una de las instrucciones.

Los cuatro tipos diferentes de instrucciones que usaremos son los siguientes:

- ALU

Registro destino (4b)	Registro OperandoA (4b)	Registro OperandoB (4b)	Opcode (4b)

Para las operaciones aritmético-lógicas usaremos un prefijo común → 0xxx. Para cada una de las operaciones específicas emplearemos los 3 bits menos significativos (los que están como 'x')

- CARGA

Registro (4b)	Valor constante (8b)	Opcode (4b)

La instrucción cargará en la dirección de *registro* introducida el valor constante. El opcode para la instrucción de carga → 1000

- SALTO ABSOLUTO

Dirección de salto (10b)	Opcode (6b)

La instrucción moverá el *program counter* (PC) a la *Dirección de salto.* El opcode que se ha elegido es → 001001

- SALTO CONDICIONAL ==

Dirección de salto (10b)	Opcode (6b)

La instrucción moverá el *program counter* (PC) a la *Dirección de salto* siempre y cuenado se cumpla una condición, es decir, que el *falg* de control *Z* esté activado(1). En otro caso, la ejecución del programa se desarrollara de forma secuencial, es decir, PC+1. El opcode que se ha elegido es → 001010

- SALTO CONDICIONAL !=

Dirección de salto (10b)	Opcode (6b)

La instrucción moverá el *program counter* (PC) a la *Dirección de salto* siempre y cuenado se cumpla una condición, es decir, que el *falg* de control *Z* esté desactivado(0). En otro caso, la ejecución del programa se desarrollara de forma secuencial, es decir, PC+1. El opcode que se ha elegido es → 001011

- SALIDA DE MEMORIA DE PROGRAMA

Inmediato (8b)	Puerto (2b)	Libre (2b)	Opcode (4b)

Consiste en enviar al *Puerto* (hay 4 puertos direccionables) un valor *inmediato* almacenado en memoria. Dicho valor se almacena en un *registro de salida*. El opcode que se ha elegido es → 1100

- SALIDA DE BANCO DE REGISTROS

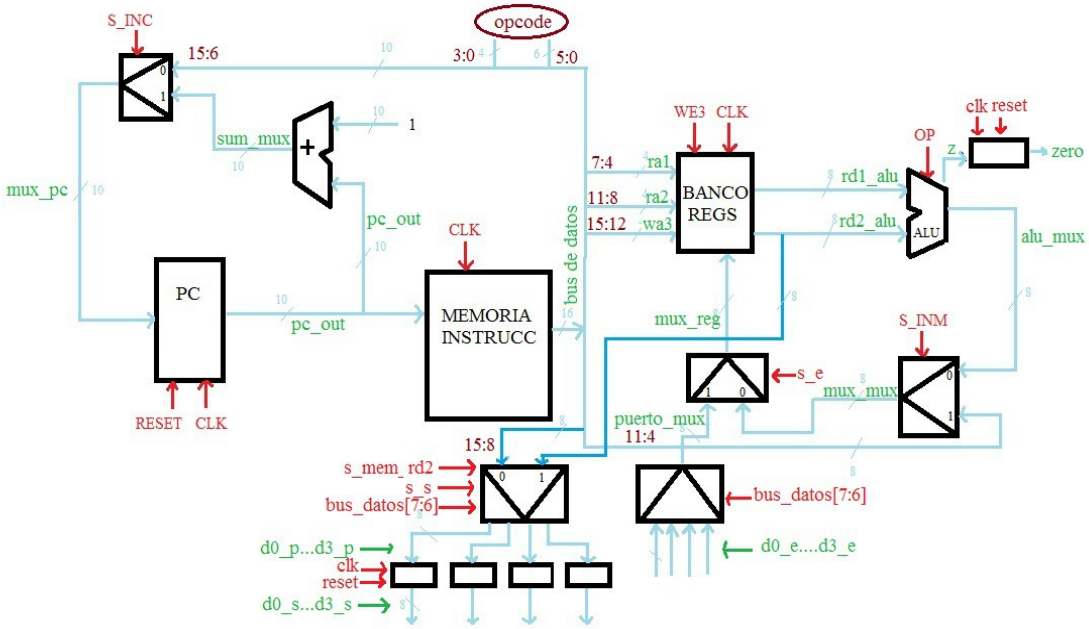
Libre (4b)	Registro Origen (4b)	Puerto (2b)	Opcode (6b)

Consiste en enviar al *Puerto* (hay 4 puertos direccionables) un valor almacenado en el *Banco de Registros* (*Registro Origen*). Dicho valor se almacena en un *registro de salida*. El opcode que se ha elegido es → 001110

- ENTRADA A REGISTRO

Libre (4b)	Registro Destino (4b)	Puerto (2b)	Opcode (6b)

La instrucción carga un valor de entrada que se lee del *Puerto* indicado y se almacena en el *Banco de Registros* (*Registro* Destino). El opcode que se ha elegido es → 011110

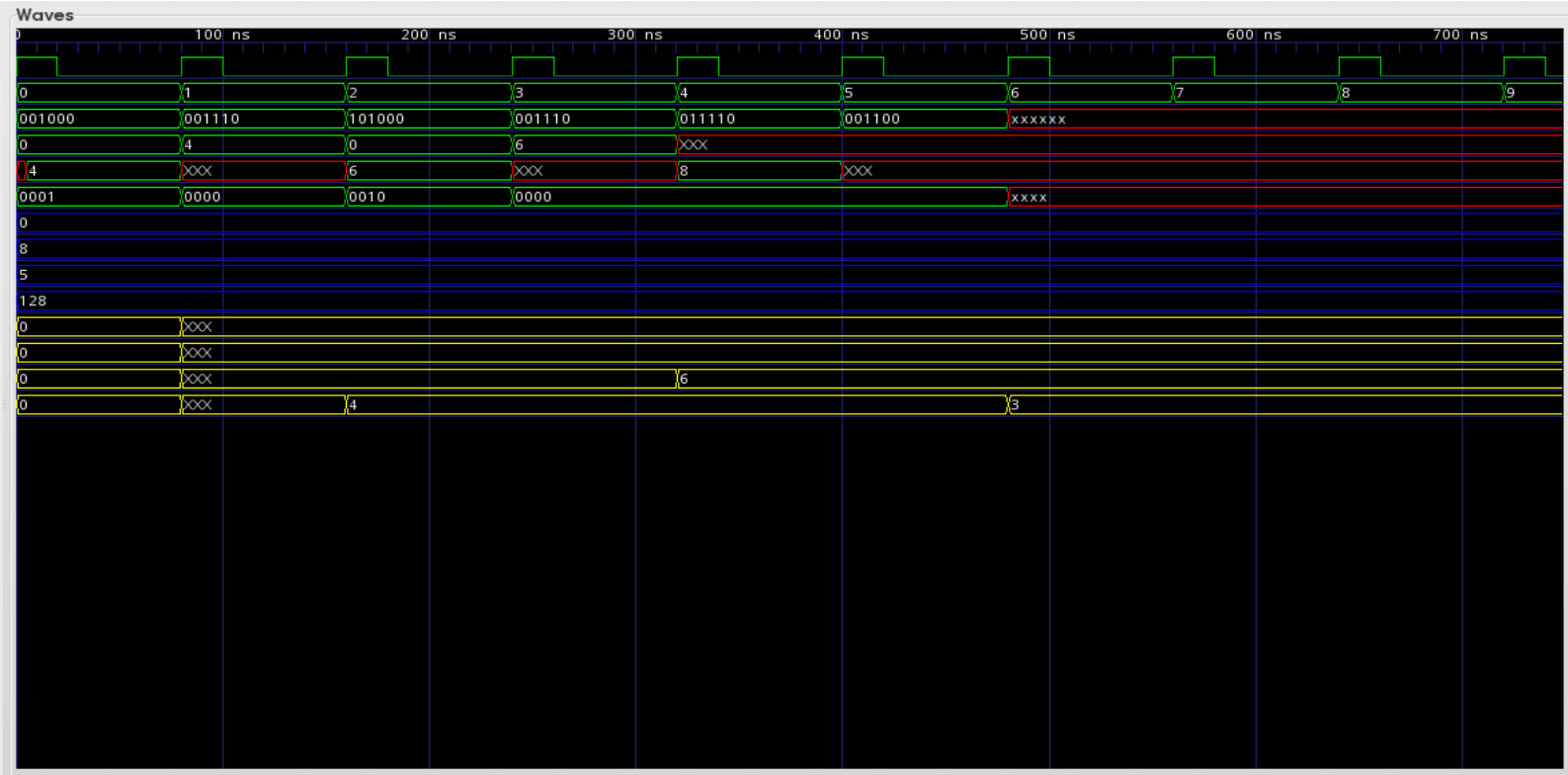


Se ha modificado el camino de datos para poder añadir los puertos de entrada y salida al procesador. Como se observa en el diagrama superior, para la entrada de daots hemos insertado dos multiplexores, uno que habilita la entrada desde la ALU o de un inmediato almacenado en memoria, que funciona con la señal *s_e* que se activa (1) si se va a usar como entrada un puerto (*d0_e* … *d3_e*), que es seleccionado mediante otro mux. Con la una señal proveniente del *bus de datos [7:6]* que corresponde a *Puerto* en la codificación de la instrucción.

Para la salida de datos, hemos creado otro multiplexor, con varias señales de control: *s_mem_rd2* que selecciona lectura de datos entre un inmediato de la memoria (*bus de datos [15:8]*) o desde el banco de registro (*rd1_alu*), otra señal de control *s_s* que está activa (1) cuando es una instrucción de salida y una última para seleccionar el puerto (*bus_datos[7:6]*). La salida de datos queda almacenada en un registro, para que no haga falta sincronización entre el dispositivo y la CPU.

Programa probado

```
0001_0000_0100_1000 #Carga 4 en Reg.1
0000_0001_0100_1110 #Salida de registro Reg.1 a Puerto 1
0010_0000_0110_1000 #Carga 6 en Reg.2
0000_0010_1000_1110 #Salida de registro Reg.2 a Puerto 2
0000_0011_0101_1110 #Entrada desde Puerto 1 hacia Reg.3
0000_0011_1100_1100 #Salida de mem. prog. Salida de un 3 por el puerto 3
```



Las señales son: *clk, PC, opcode, rd2_alu, wd3 (mux_reg), wa3*. Las señales azules corresponden a los dispositivos de entrada (inicializados en el *test_bench*) las amarillas a los puertos de salida.

Problemas y dificultades

En esta segunda entrega de nuestro Procesador hemos tenido que diseñar nosotros mismos los dispositivos y métodos de conexión a los puertos de entrada y salida (con los multiplexores) y eso es lo que nos ha parecido más complicado. Hemos tenido, también, que ajustar los *opcodes* de las instrucciones de la entrega anterior para tener varias más libres sin conflicto para implementar estas tres nuevas de entrada y salida