Universidad Zaragoza
1542

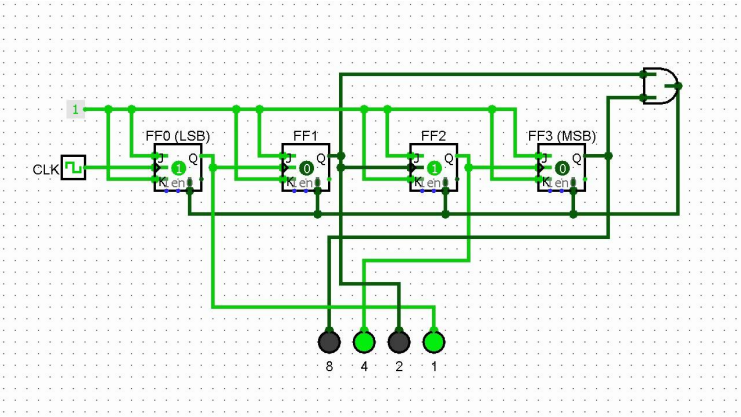
ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORES 2

2º DE GRADO EN INGENIERÍA INFORMÁTICA, CURSO 2020/2021

UNIVERSIDAD DE ZARAGOZA

PRÁCTICA 4

DISEÑO DE UN PROCESADOR SEGMENTADO EN 5 ETAPAS



Luis M. Ramos
José Luis Briz
Javier Resano
Francisco J. Martínez

Logisim
una herramienta de diseño y simulación de circuitos lógicos

Universidad Zaragoza
1542

CONTENIDO

1 INTRODUCCIÓN

En esta práctica vamos a continuar con el diseño de nuestro procesador multiciclo. Le añadiremos una memoria de datos, separada de la de instrucciones. Añadiremos también instrucciones de acceso a la memoria de datos (load y store) y una instrucción de salto condicional.

Una vez diseñado el procesador, lo programaremos y depuraremos hasta que funcione correctamente. Finalmente, haremos una versión segmentada y evaluaremos sus prestaciones.

Para poder realizar la práctica, debéis disponer de vuestro ***procesador multiciclo*** *diseñado en la segunda parte de la práctica anterior,* ***finalizado y funcionando correctamente****.* No obstante, tened en cuenta, que en función de cómo hubierais definido el formato de instrucción en la práctica anterior, puede ser necesario que os replanteéis tanto los formatos ya definidos, como el diseño de la Unidad de Control.

Duración: 3 sesiones de laboratorio (6 horas)

Universidad Zaragoza
1542

2 DISEÑO DE UN PROCESADOR CON MEMORIA DE DATOS

2.1 TRABAJO PREVIO: DISEÑO EN PAPEL

1. Partiendo del procesador multiciclo de la práctica anterior, modificad su diseño en papel teniendo en cuenta lo siguiente:

   - Añadir un registro de instrucción (RI) a la salida de la memoria de instrucciones.
   - Añadir una memoria de datos (RAM 64Kx32) y un registro en su salida (MEM).
   - El procesador deberá soportar las siguientes instrucciones:

```
mov rd, K       ; BR(rd) ← SignExt(K)    K constante de 16 bits
add rd, ra, rb  ; BR(rd) ← BR(ra) + BR(rb)
ld  rb, (ra)    ; BR(rb) ← MEM(BR(ra)15-0)
st  rb, (ra)    ; MEM(BR(ra)15-0) ← BR(rb)
beq ra, rb, K   ; si BR(ra) == R(rb) salta a @K
nop             ; no realiza ninguna acción (máquina segmentada)
```

2. Definid los formatos de instrucción y la codificación de las instrucciones.

3. Dibujad el autómata de la unidad de control usando Qfsm (NIA-multi2.fsm). Implementadla usando un registro (estado) y una única ROM (que incluya tanto la función de transición, como la función de salida).

4. Utilizando las instrucciones anteriores, escribid un programa en ensamblador (NIA-multi2.asm). Para ello, supondremos que tenemos los dígitos de nuestro NIA en las 6 palabras de la memoria de datos a partir de la dirección de memoria (NIA % 100), y que calcularemos la suma de los dígitos mediante un bucle, guardando el resultado en la siguiente palabra de la memoria. Asumid que los datos necesarios ya estarán cargados en la memoria RAM.

2.2 TRABAJO EN EL LABORATORIO: DISEÑO CON LOGISIM

2.2.1 DISEÑO MULTICICLO

a) Abrir en Logisim el circuito de la práctica anterior. Guardadlo en un fichero nuevo (NIA-multi2.circ).

b) Colocad los nuevos componentes comentados en el apartado 2.1 e interconectadlos según vuestro diseño en papel. Se aconseja configurar la memoria de datos como *Separate load and store ports*.

c) Programad la memoria ROM de instrucciones para que incluya el programa que habéis implementado en el apartado 2.1.4. Guardad el contenido de la ROM en un fichero (NIA-programa.txt).

d) Programad las @ de la RAM con las cifras de uno de vuestros NIA. Guardad el contenido de la RAM en un fichero (NIA-datos.txt).

e) Programad la memoria ROM dedicada al control, para que incluya tanto las señales de control, como las transiciones entre estados. Guardad el contenido de la ROM en un fichero (multi2-control.txt).

f) **Ejecutad ciclo a ciclo hasta que el programa funcione correctamente**.

g) Antes de entregar, comprobad que habéis marcado el procesador multiciclo (PROCESADOR o como lo hayáis llamado) como circuito principal y que usáis un único componente Reloj. Debéis entregar también el diseño en papel y el autómata de la unidad de control NIA-multi2.fsm (dibujado con la herramienta Qfsm disponible en moodle).

2.2.2 DISEÑO SEGMENTADO

a) Vamos a diseñar ahora una versión segmentada del procesador. El procesador segmentado deberá soportar las mismas instrucciones que el procesador multiciclo2, pero en este caso no será capaz de detectar los riesgos de datos ni de control, por lo que tendréis que modificar ligeramente el programa que habéis hecho en el apartado 2.1.4, insertando algunas instrucciones *nop* para evitarlos. Para ello, tendréis que detectar e indicar las dependencias que existen en vuestro programa e insertar instrucciones *nop* para evitar los riesgos de datos y control (NIA-seg.asm).

b) Haced una copia del procesador multiciclo2 y renombradla como NIA-seg.circ.

c) Programad la memoria ROM de instrucciones para que incluya el nuevo programa que habéis implementado en el apartado a). Guardad el contenido de la ROM en un fichero (NIA-programa-seg.txt).

d) Modificad la ruta de datos para que el procesador ejecute de forma segmentada en 5 etapas (IF/ID/EX/MEM/WB). Tendréis que añadir los registros intermedios y multiplexores necesarios. Recordad que todas las instrucciones deben pasar por todas las etapas, aunque no necesiten realizar ninguna acción en alguna de ellas.

e) Diseñad la Unidad de Control para el procesador segmentado. Usad una ROM para obtener el vector de señales de control de la instrucción, y varios registros por los que se vayan propagando etapa a etapa.

f) **Ejecutad ciclo a ciclo hasta que el procesador segmentado funcione correctamente**.

g) Antes de entregar, comprobad que habéis marcado el procesador segmentado (PROCESADOR o como lo hayáis llamado) como circuito principal y que usáis un único componente Reloj. Debéis entregar también un diagrama (similar a los que vemos durante el Tema 5) en el que se indique, para cada instrucción y ciclo de reloj, qué etapa de cada instrucción se está completando. Considerad sólo la primera iteración del bucle.

2.3 ANÁLISIS TEMPORAL: RESULTADOS

1. ¿Cuántos ciclos tarda el procesador multiciclo2 en ejecutar vuestro programa ($nciclos_{multi2}$)?

2. ¿Cuántos ciclos tarda el procesador segmentado ($nciclos_{seg}$) en ejecutar el programa usando nops?

3. Aplicad técnicas de planificación estática de código a vuestro programa (es decir, reordenando instrucciones), para minimizar el número de nops a insertar (guardadlo en NIA-segReordenado.asm). Programad la ROM de instrucciones de acuerdo a ese nuevo código reordenado y guardad su contenido en un fichero (NIA-segReordenado.txt).

   ¿Cuánto tardará el procesador segmentado ($nciclos_{segReordenado}$) en ejecutar el programa reordenado? ¿Cuál será la mejora obtenida comparada con la ejecución del programa sin ordenar?

4. Utilizando los datos de los retardos que hemos usado en las anteriores prácticas, ¿cuál será el tiempo de ciclo del procesador multiciclo2? ¿y el del procesador segmentado? Presentad en detalle los cálculos necesarios y justificad vuestras respuestas.

5. Teniendo en cuenta lo anterior. ¿Cuál será la mejora obtenida por la máquina segmentada ejecutando el código planificado comparada con la ejecución del programa en la máquina multiciclo2?

2.4 MATERIAL A ENTREGAR

Deberéis entregar todo lo realizado (diseño en papel, diseño en logisim, ficheros .circ y .txt, autómata .fsm y la propia memoria de la práctica en la que expliquéis el análisis temporal), dentro del plazo establecido, que se comunicará en moodle.

La forma de entrega será la misma que en las prácticas anteriores.