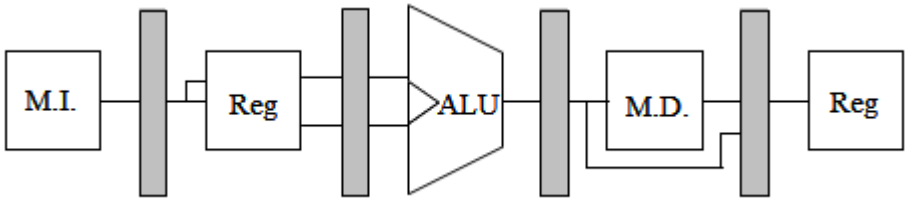
DISEÑO DE UN PROCESADOR SEGMENTADO EN 5 ETAPAS

Práctica 1 – sesiones 3 y 4



Arquitectura y Organización de Computadores 2
2º Grado Ingeniería Informática

Luis M. Ramos luisma@unizar.es
Alejandro Valero alvabre@unizar.es
José Luis Briz briz@unizar.es
Javier Resano jresano@unizar.es

Escuela de Ingeniería y Arquitectura
Universidad Zaragoza

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

1 RESUMEN

En las sesiones 3ª y 4ª 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.

***Al entrar al laboratorio** coloca el **trabajo previo del apartado 2.1** encima de la mesa, de forma visible. Antes de empezar a trabajar con Logisim conéctate a Moodle desde uno de los equipos del laboratorio y realiza el **control de asistencia a la sesión**.*

*La práctica finaliza cuando los procesadores funcionan correctamente y han sido entregados a través de **Moodle**. También hay que entregar los **resultados del apartado 3**.*

2 DISEÑO DE UN PROCESADOR CON MEMORIA DE DATOS

2.1 TRABAJO PREVIO: DISEÑO EN PAPEL

a) Partiendo del procesador multiciclo de la sesión anterior, modifica su diseño en papel teniendo en cuenta lo siguiente:
   - Añade un registro de instrucción (**IR**) a la salida de la memoria de instrucciones.
   - Añade una memoria de datos (RAM 64Kx32) y un registro en su salida (**MDR**).
   - El procesador debe soportar las siguientes instrucciones:

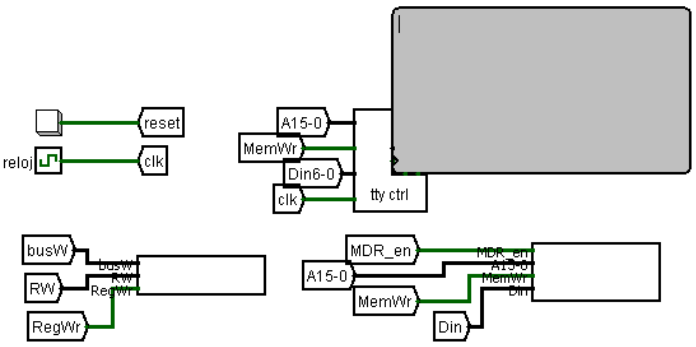
```
mov K, rb       ; SignExt(K) → BR(rb)          K cte de 16 bits
add ra,rb,rd    ; BR(ra) + BR(rb) → BR(rd)
ld  (ra), rb    ; MEM(BR(ra)15-0) → BR(rb)
st  rb, (ra)    ; BR(rb) → MEM(BR(ra)15-0)
beq ra, rb, K   ; si BR(ra)==BR(rb) salta a @ K
nop             ; no realiza ninguna acción
```

b) Define el nuevo formato de instrucción y la codificación de las instrucciones.
c) Dibuja el autómata de la unidad de control usando Qfsm (NIA-multi2.fsm). Al principio de la sesión se aconseja enseñar el autómata al profesor de prácticas. Implementa la UC usando un registro (estado) y dos ROM (función de transición y función de salida).
d) Utilizando las instrucciones anteriores escribe un programa ensamblador (NIA-multi2.asm) que calcule mediante un bucle la suma de los contenidos de las @ de memoria base+1, base+2, … base+6 (base = NIA % 100), guardando el resultado en la @0. Asume que los datos necesarios ya estarán cargados en la memoria RAM.

2.2 TRABAJO EN EL LABORATORIO: DISEÑO CON LOGISIM

a) Abre en Logisim el circuito de la sesión anterior. Guárdalo en un fichero nuevo (NIA-multi2.circ).
b) Coloca los componentes del apartado 2.1 e interconéctalos según tu diseño. Se aconseja configurar la memoria de datos como *Separate load and store ports*.

c) Cierra Logisim, descarga el fichero aoc2.jar de la carpeta Logisim de Moodle y guárdalo en la misma carpeta que tu fichero .circ. Coloca en el extremo superior del circuito:
   - el botón de **reset**;
   - el componente *Reloj*;
   - el componente *RTL1 Viewer*;
   - un componente *RTL2 Viewer*; conéctale las señales **MDR_en** (permiso de carga de **MDR**), **A_{15-0}** (dirección de memoria), **MemWr** (permiso de escritura en memoria) y **Din** (dato a escribir en memoria);
   - Añade una pantalla teletipo (*Input-Output / TTY*). Descarga el fichero tty-ctrl.circ de la carpeta Logisim de Moodle y guárdalo en la misma carpeta que tu fichero .circ. Carga la librería con el controlador de la pantalla (*Proyecto / Cargar Librería / Librería Logisim / tty-ctrl.circ*). Coloca el componente *tty-ctrl* en la esquina inferior izquierda de la pantalla teletipo. Conéctale las señales **A_{15-0}**, **MemWr**, **Din_{6-0}** y **clk**.



d) Programa la memoria RAM con los valores que se calculan en webdiis.unizar.es/~luisma/aoc2/nia2.php a partir de tu NIA. Guarda el contenido de la RAM en un fichero (NIA-RAM.txt).
e) Añade instrucciones ensamblador en tu programa del apartado 2.1.d de forma que se compruebe si el resultado de la suma es el correcto y se imprima en pantalla "OK" o "NO OK". Para escribir un carácter hay que escribir su código ASCII en la dirección 0xffff de memoria RAM. Traduce tu programa a hexadecimal y programa la ROM.
f) Ejecuta ciclo a ciclo hasta que funcione correctamente.
g) Finalmente entrega tu circuito a través del recurso Moodle "Entrega NIA-multi2". Antes de entregar comprueba que has marcado **main** como circuito principal y que usas un único componente *Reloj*. Debes entregar también el autómata de la unidad de control NIA-multi2.fsm (dibujado con la herramienta Qfsm disponible en Moodle).

2.2.1 Diseño segmentado

Vamos a diseñar ahora una versión segmentada del procesador. El procesador segmentado debe soportar las mismas instrucciones que el multiciclo2, pero en este caso no detecta riesgos de datos ni de control, por lo que habrá que insertar instrucciones nop para evitarlos.

a) Haz una copia del fichero del procesador multiciclo2 y renómbrala NIA-seg.circ.
b) Modifica la ruta de datos para que el procesador ejecute de forma segmentada en 5 etapas (IF/ID/EX/MEM/WB). Tendrás que añadir los registros y multiplexores necesarios. Recuerda que todas las instrucciones deben pasar por todas las etapas, aunque no necesiten realizar ninguna acción en alguna de ellas.
c) Diseña la unidad de control para el procesador segmentado. Usa una ROM para obtener el vector de señales de control de la instrucción y varios registros por los que se vaya propagando etapa a etapa.
d) Detecta las dependencias que existen en tu programa e inserta instrucciones nop para evitar los riesgos de datos y control (NIA-seg.asm). Guarda el contenido de la memoria ROM de instrucciones en un fichero (NIA-seg.txt).
e) Ejecuta ciclo a ciclo hasta que el procesador segmentado funcione correctamente.

3 EVALUACIÓN DE PRESTACIONES

a) ¿Cuántos ciclos tarda el procesador multiciclo2 en ejecutar tu programa ($nciclos_{multi2}$)?
b) ¿Cuántos ciclos tarda el procesador segmentado ($nciclos_{seg}$) en ejecutar el programa con nops?
c) Aplica técnicas de planificación estática de código (reordenando instrucciones) a tu programa para minimizar el número de nops a insertar (NIA-segR.asm). Programa la ROM de instrucciones y guarda su contenido en un fichero (NIA-segR.txt). Guarda el circuito con las instrucciones reordenadas en un fichero nuevo (NIA-segR.circ). ¿Cuánto tarda el procesador segmentado ($nciclos_{segR}$) en ejecutar el programa reordenado?
d) Cuando tengas todos los cálculos realizados introdúcelos a través del recurso Moodle "Evaluación de prestaciones multiciclo2 vs segmentado".
e) Finalmente entrega tus circuitos a través del recurso Moodle "Entrega NIA-seg y NIA-segR". Antes de entregar comprueba que has marcado **main** como circuito principal y que usas un único componente *Reloj*.