Practica 2: diseño de la unidad de control de un procesador

***Resumen**: En esta práctica vais a participar en el diseño de un procesador. Partiendo de un repertorio de instrucciones sencillo y de una ruta de datos multiciclo debéis diseñar la unidad de control de tal forma que cada instrucción tarde el mínimo número de ciclos posible. A continuación trabajaremos con una versión segmentada del mismo procesador y de nuevo debéis diseñar la unidad de control. Además hay que identificar los riesgos y eliminarlos, generando la menor penalización posible. Por último, realizaréis un análisis del rendimiento de ambos diseños.*

Duración: 2 sesiones de laboratorio (4 horas)

Objetivos:

- Entender cómo funciona un procesador multiciclo y un procesador segmentado.
- Ser capaces de comparar el rendimiento de distintas implementaciones de un procesador y comprender las técnicas que utilizamos para mejorarlo.

Especificación del procesador

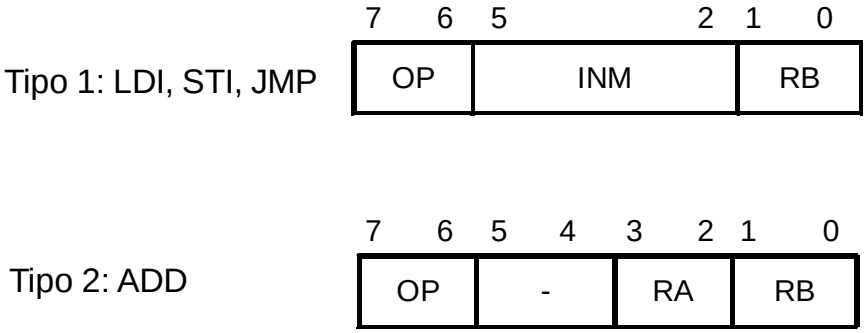
En esta práctica vais a formar parte del equipo que diseña el procesador gaz8. Las características principales de este procesador son:

- 4 instrucciones de 8 bits (con dos formatos)
- Memoria de instrucciones y datos separadas. Cada una de ellas tiene 16 palabras de 1 byte, numeradas de la 0 a la 15 (16 bytes en total).
- Banco de registros: 4 registros de 8 bits.
- Tipos de datos: se trabaja siempre con números enteros de 8 bits representados en Ca2.
- Arquitectura load/store
- Modos de direccionamiento: inmediato y registro
- Ejecución multiciclo y segmentada. Las ejecución de las instrucciones se divide en las mismas etapas que el procesador MIPS (IF; ID; EX; MEM;WB)

Repertorio de instrucciones

- LDI RB, [INM]: RB ← Mem(INM); PC ← PC+1; Código operación: 00
- STI RB, [INM]: Mem(INM) ← RB; PC ← PC+1; Código operación: 01
- ADD RB, RA: RB ← RA+RB; PC ← PC+1; Código operación: 11
- JMP INM: PC ← INM; Código operación: 10

La siguiente figura muestra el formato de estas instrucciones.



Paso 1: diseño de la unidad de control de un procesador multiciclo

En el archivo *multiciclo.cct* tenéis la ruta de datos multiciclo del procesador gaz8. En la página siguiente podéis ver un esquema de la ruta de datos y sus señales de control. En este procesador, la ejecución se ha dividido en cinco etapas similares a las estudiadas en clase con el procesador MIPS. Todas las instrucciones pasan por IF e ID, y a continuación c*ada instrucción pasa por las etapas que necesita.* En *multiciclo.cct* también está la estructura de la unidad de control (registro de estado + memorias ROM), pero las memorias ROM están vacías. En este paso debéis:

- Estudiar la ruta de datos
- Ver cuántos ciclos necesita cada instrucción
- Diseñar el diagrama de estados de la unidad de control
- Calcular el valor correcto de las señales de control en cada estado
- Trasladar estos valores a la ROM correspondiente

Los pasos anteriores deben hacerse en papel antes de comenzar a modificar el diseño de LogicWorks. Al acabar, debéis comprobar que el programa se ejecuta correctamente. Para ello en la memoria de instrucciones está cargado el siguiente programa:

0. LDI R0, [#0]
1. ADD R0, R0
2. ADD R0, R0
3. STI R0, [#3]
4. LDI R1, [#1]
5. LDI R2, [#2]
6. ADD R2, R1
7. STI R2, [#4]
8. JMP #0

En la memoria de datos hay un 03, un 02 y un 05 en las posiciones #0, #1 y #2. Por lo que al acabar la ejecución en la posición 3 debe haber un 0C y en la 4 un 07. Nota: como la memoria de datos permite escrituras es posible que borréis los datos iniciales sin daros cuenta. Si eso ocurre debéis editar la memoria y restaurarlos.

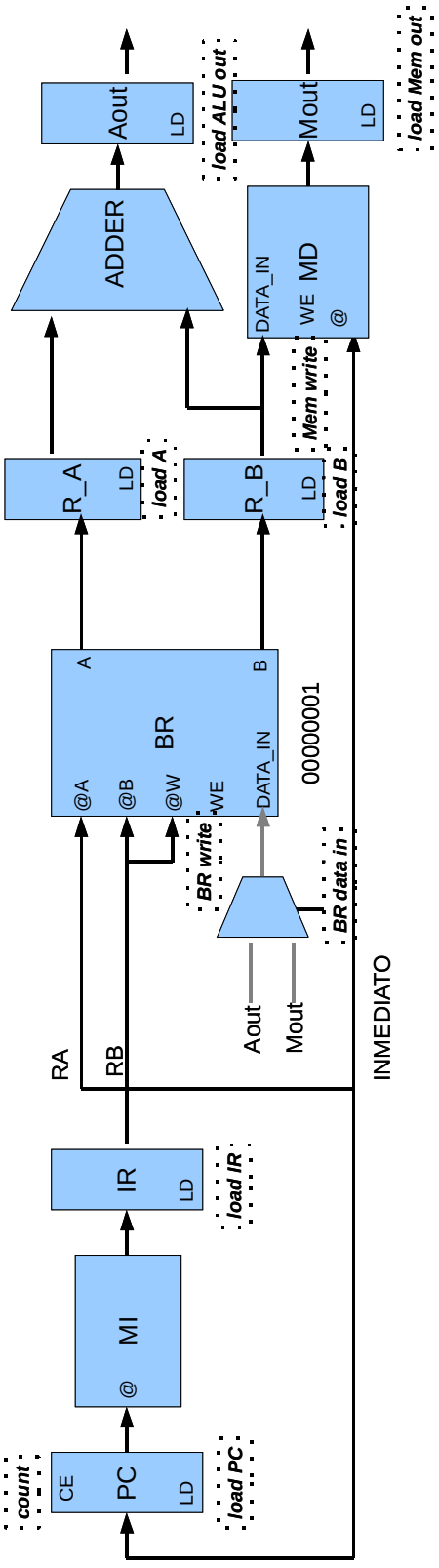


Ilustración 1: Diagrama del procesador multiciclo

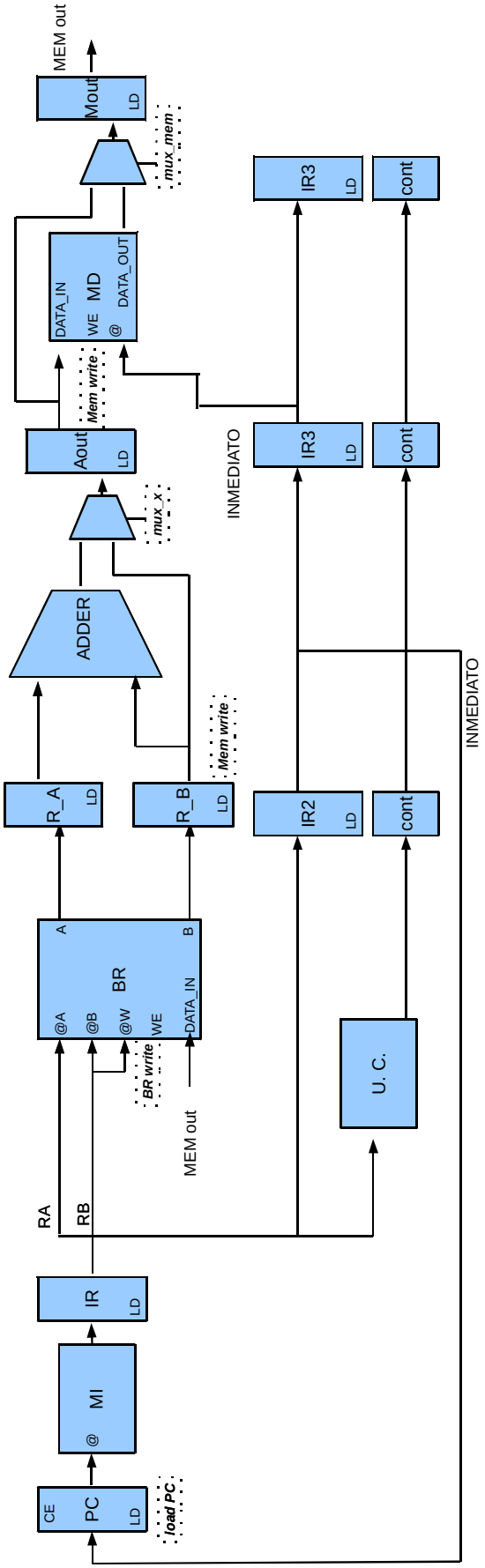


Ilustración 2: Diagrama del procesador segmentado

Paso 2: diseño de la unidad de control de un procesador segmentado

En el archivo *segmentado.cct* tenéis la ruta de datos segmentada del procesador gaz8. En la figura 2 podéis ver un esquema simplificado. De nuevo, las etapas son similares a las estudiadas en clase con el procesador MIPS. En este caso *todas las instrucciones pasan por las cinco etapas*. Las señales de control se generan en la etapa ID y viajan con la instrucción. Como en el caso anterior, junto a la ruta de datos está el esqueleto de la unidad de control, con una memoria ROM que debéis rellenar con los valores adecuados para que todas las instrucciones se ejecuten correctamente.

La ejecución segmentada puede generar riesgos, tanto de datos como de control. Este procesador no los detecta, y es responsabilidad del compilador/programador eliminarlos. Para ello utilizaréis las siguientes técnicas:

- Instrucciones NOP: en el procesador segmentado se define una nueva instrucción codificada en binario de la siguiente forma: **111**00000. A todos los efectos, esta instrucción actúa como una suma, pero cuando va a escribir el resultado en el banco de registros la unidad de control inhabilita la escritura, por lo que al final la instrucción no hace nada. Esto ya está implementado en el circuito que os damos, así que no debéis preocuparos de las señales de control de esta instrucción. La instrucción NOP nos puede resultar muy útil para separar a los productores/consumidores que generen riesgos de datos, y para evitar que se ejecuten instrucciones no deseadas debido a los riegos de control. **Introduciendo instrucciones NOP, podemos eliminar todos los riesgos del código, pero el objetivo es introducir el menor número de NOPs posible para no degradar el rendimiento del procesador.**
- Reordenación de código: Podéis cambiar las instrucciones para separar aquéllas que generan riesgos de datos y así reducir el número de NOPs necesario. Eso sí, sólo se pueden realizar aquellos cambios que no afecten de ninguna manera al resultado final del código.
- Saltos retardados: Este procesador no para después de una instrucción JMP, a pesar de que no se puede cargar la instrucción destino del salto hasta pasados N ciclos (debéis calcular vosotros el valor de N). Una solución sencilla, pero no muy eficiente, es colocar N instrucciones NOPs después de cada JMP. De esta forma se garantiza que no se ejecutan instrucciones que no tocan, pero se desperdician N ciclos. La técnica de saltos retardados, consiste en colocar en las N posiciones que hay después del salto instrucciones válidas que antes estaban antes del salto. De nuevo hay que garantizar que los cambios en el código no afecten de ninguna manera al resultado final.

Estudio previo:

- Estudia el circuito que os damos y calcula los valores correctos que debe generar la unidad de control.
- Identifica qué instrucciones producen datos y cuáles los consumen. Teniendo en cuenta que las escrituras en el banco de registros se realizan en la etapa WB, y que no se puede leer un dato hasta al ciclo siguiente de escribirlo, identifica qué combinación de instrucciones puede generar riesgos de datos y durante cuántos ciclos.
- Cuantifica el efecto de los riesgos de control.
- Identifica los riesgos en el código original y aplica las técnicas anteriores para eliminarlos. Actualiza la memoria de instrucciones con el nuevo código.

Estudio final: Una vez realizado el diseño y modificado el código original, comprobad de nuevo que se ejecuta correctamente. Por último, comparar el rendimiento de ambos procesadores. Para ello responded a las siguientes preguntas:

1. ¿Cuántos ciclos tarda cada implementación para ejecutar el código que os damos?
2. Si suponemos que los dos procesadores trabajan con el mismo reloj y teniendo en cuenta las instrucciones NOPs introducidas, ¿cuál sería el speedup del procesador segmentado con respecto al multiciclo? ¿y el aumento de velocidad en MIPS? (en el cálculo de los MIPS las instrucciones nop cuentan como una instrucción más).
3. Repite el apartado anterior haciendo una estimación de la frecuencia de reloj del procesador segmentado. Para ello, asume que la etapa EX es la que marca el camino crítico del sistema. Compara ambas etapas e identifica los retardos en cada caso (los retardos del sumador, registros y muxes son los mismos que en la práctica anterior).
4. Si introducimos en el procesador segmentado la posibilidad de realizar cortocircuitos para leer sus operandos antes de que se escriban en el banco de registros, ¿cuántas instrucciones NOP podéis eliminar?

Anexo I: Pasos para modificar el contenido de una PROM/RAM:

1. abrir el circuito y seleccionar el dispositivo que queremos editar

3. abrir el menú de simulación

4. seleccionar el PROM/RAM/PLA wizard

5. seleccionar "edit selected device" (ya está seleccionado por defecto) y dar a siguiente

6. meter el contenido manualmente, o desde un archivo