Memoria Técnica del Proyecto 1

Arquitectura y Organización de Computadores 2

2º curso, Grado de Ingeniería en Informática, 2018-2019

Patricia Briones Yus, 735576

Índice de contenidos

1. Diseño del MIPS

El procesador MIPS segmentado implementado, aparte de soportar las instrucciones básicas (NOP, aritméticas, LW, SW y BEQ), también permite la ejecución de dos instrucciones nuevas: LA y BNE. LA se encarga de calcular la dirección y de guardarla en un registro, mientras que BNE tiene la condición contraria a la instrucción BEQ. Salta si los registros no son iguales.

Como todo procesador MIPS, existe una unidad de control para regular las señales y datos de cada etapa. Las dependencias y riesgos se controlan mediante la unidad de anticipación y el apartado de riesgos, que hemos implementado, habiendo pensado en cada uno de los posibles casos de error (que más tarde se comentarán) y así evitar traspiés en los datos.

Por último, para resolver los riesgos de control se utiliza un predictor de saltos, que usa la información que se va generando durante la ejecución para predecir el salto, almacenando los datos para luego comprobar si ha acertado o no.

2. Modificación del hardware

2.1 Nuevas instrucciones

Para que el procesador MIPS dé soporte a dos nuevas instrucciones, primero hay que modificar la unidad de control. La instrucción LA supone añadir otra línea a los casos de "IR_op_code" con código "001000" , ALUSrc=1 y RegWrite=1.

En cambio BNE ("000101") requiere aparte de lo anterior, añadir una señal nueva para poder diferenciar cuándo estamos realizando un BNE o un BEQ. A esta señal se le llamará Zerosrc_A, la cual está inicializada a 1 y es de tipo "out STD_LOGIC". También hay que modificar la señal "Saltar" para que cuando se cumplan las condiciones del BEQ o del BNE se ponga a 1.

Las instrucciones de salto necesitan modificar el PCSrc para controlar el dato que se transmite al PC, pero también dependerán de los riesgos de control, por lo que se explicará más adelante en uno de los siguientes subapartados.

2.2 Riesgos de datos

Se ha modificado la unidad de anticipación para evitar las dependencias que pueda haber en las etapas MEM y WB. Para poder darse una dependencia, la señal RegWrite de cualquiera de las dos etapas tiene que estar activada. En caso de ser el registro destino en MEM igual al registro rs o rt en EX, se selecciona la entrada "01" (ALU_out_MEM) de los multiplexores Mux_A y Mux_B. Si el registro destino en WB es igual al registro rs o rt en EX, se selecciona la entrada "10" (busW) y en cualquier otro caso la entrada "00" (BusA_EX/ BusB_EX).

En el MIPS no hay una unidad de detección realmente implementada, sino que está formada por señales. Los riesgos de datos pueden encontrarse al ejecutar un LW o BEQ o BNE.

Un riesgo del LW es debido a que la siguiente instrucción que se va a ejecutar necesita el valor de rs del LW para uno de sus registros rs o rt. El riesgo solo existe cuando el LW está en la etapa EX, pero no antes ni después. La instrucción NOP puede crear una falsa dependencia al poder ser su registro rs o rt igual al registro destino del LW, lo cual se evita comprobando que la instrucción en ID sea distinta a "000000". Por último solo habría que verificar que el registro rs de LW sea igual al registro rs o rt de la etapa ID.

Los riesgos que puede haber con las instrucciones de salto, pueden ser debido a que antes de estas se haya ejecutado una instrucción aritmética o un LW, que causarían dos bloqueos. Las dos instrucciones pueden crear un riesgo en la etapa EX o MEM, entonces RegWrite_EX/MEM=1 y se compara el registro destino de la instrucción en EX/MEM con los registros rs o rt de la etapa ID.

En caso de haber un riesgo, ya sea por LW o por las instrucciones de salto, la señal avanzar_ID valdrá 0 y enviará una NOP a la etapa EX, si no valdrá 1.

2.3 Riesgos de control

El predictor de saltos puede tener dos tipos de errores:

- Tomó la decisión contraria (Saltar/=prediction_ID)
- Decidió saltar pero se saltó a una dirección incorrecta (la predicción de saltar era correcta y address_predicted_ID/=DirSalto_ID).

En caso de haber un error de predicción por uno de estos dos errores, se activará la señal update_predictor para que se actualicen el valor del salto (prediction_in <= Saltar) y la dirección de salto correcta (branch_address_in <= DirSalto_ID) del predictor. Si hay error en uno de los dos casos, se sustituye la instrucción que se acaba de leer por una NOP para anular la antigua.

Por último, el multiplexor del PC tiene cuatro entradas:

- La entrada “01” (la dirección de salto que proporciona el predictor) se elegirá cuando no haya ningún error de predicción y se tenga que saltar
- La entrada “11” (la dirección de salto calculada en la etapa ID) se elegirá cuando el predictor sea erróneo y se haya calculado salta (Saltar=1)
- La entrada “10” (el PC+4 de la instrucción que está en ID) se elegirá cuando haya un error en la predicción y se haya calculado no saltar (Saltar=0).
- Si no se cumple ninguna de estas opciones, se escogerá la entrada “00” (PC actual + 4).

3. Impacto de la gestión de los riesgos

Para calcular el impacto en el rendimiento del procesador, se comparará el tiempo de ejecución de un programa sencillo finito en la versión antigua del procesador y en la actualizada.

El programa consta de varias dependencias de datos, de riesgos de LW y de riesgos de saltos.

Las instrucciones de la versión 1 se encuentran en el fichero “programa1_1” y las de la versión 2 en el fichero “programa1_2”.

(1) Versión con NOPS y salto retardado:

```
BUC   La r0,4(r0)
      NOP
      NOP
      Add r1,r0,r1
      NOP
      NOP
      Sw r0,0(r1)
      Bne r0,r1,FIN
      NOP
      Lw r1,0(r1)
      NOP
      NOP
      Beq r1,r1,BUC
      NOP
FIN   Sw r1,2(r1)
```

(2) Versión fully equipped:

```
BUC   La r0,4(r0)
      Add r1,r0,r1
      Sw r0,0(r1)
      Bne r0,r1,FIN
      Lw r1,0(r1)
      Beq r1,r1,BUC
FIN   Sw r1,2(r1)
```

	Versión 1	Versión 2
Tiempo de ejecución 1ª iteración	155	125
Tiempo de ejecución resto de iteraciones	100	60

$$T_{sup} = \frac{155}{125} = 1.24$$

4. Verificación del correcto funcionamiento

Para confirmar que el procesador MIPS funciona correctamente, se ha creado un pequeño programa en el que se pueden apreciar varios riesgos de datos, riesgos de control y posibles falsas dependencias con la instrucción NOP. La RAM de instrucciones se encuentra en el fichero "programa2". No ha sido necesario modificar el contenido de datos de la RAM.

```
BUC1    Lw r4,0(r4)
        Lw r0,12(r0)
        Add r1,r0,r4
BUC2    La r3,0(r0)
        Lw r2,4(r1)
        Sw r1,4(r2)
        La r0,0(r3)
        Beq r0,r1,BUC2
        Add r3,r2,r1
        Lw r0,8(r3)
        NOP
        Bne r0,r1,BUC1
```

Los valores iniciales de los datos de la RAM son:

0x00	00000001
0x04	00000002
0x08	00000004
0x0C	00000000

```
BUC1    Lw r4,0(r4)
        Lw r0,12(r0)
        Add r1,r0,r4
```

En el ADD existe una dependencia productor-consumidor con r4 y r0 de los LW. Cuando el ADD esté en la etapa EX, el segundo LW está en MEM, por lo que existe un riesgo del load (se detiene un ciclo). Para obtener los valores correctos de r4 y r0 se hace uso de la UA. **Figura 1.**
R4=00000001 R0=00000000 R1=00000001

```
BUC2    La r3,0(r0)
        Lw r2,4(r1)
        Sw r1,4(r2)
```

Como en el caso anterior, para obtener el valor de r3 se hará uso de la UA. Existe un riesgo LW en el r2, por lo que se detiene un ciclo para que la UA pueda más tarde proporcionar el valor correcto.
R3=00000000 R2=00000002 0x06 es 0x04 -> @0x04=00000001

La r0,0(r3)

Beq r0,r1,BUC2

En este caso hay un riesgo del BEQ, ya que hay que esperar a que LA llegue a WB y BEQ espere en ID dos ciclos. Al ser r0/=r1 no se toma el salto y por tanto la predicción ha sido correcta. **Figura 2.**
R0=00000000

Add r3,r2,r1

Lw r0,8(r3)

NOP

Bne r0,r1,BUC1

Existe una dependencia productor-consumidor con r3, por lo que podría haber una falsa dependencia entre el LW y el NOP, ya que LW escribe en r0, pero no debe suceder y por tanto no se debe detener. Existe un riesgo del beq con el load en r0, pero al haber una nop de por medio, solo se detendrá un ciclo de reloj. Al ser r0/=r1 se tomará el salto, el predictor falla y por tanto se actualizan sus valores. **Figura 3.**
R3=00000003 @0xB->@0x8 R0=00000004

Figura 1

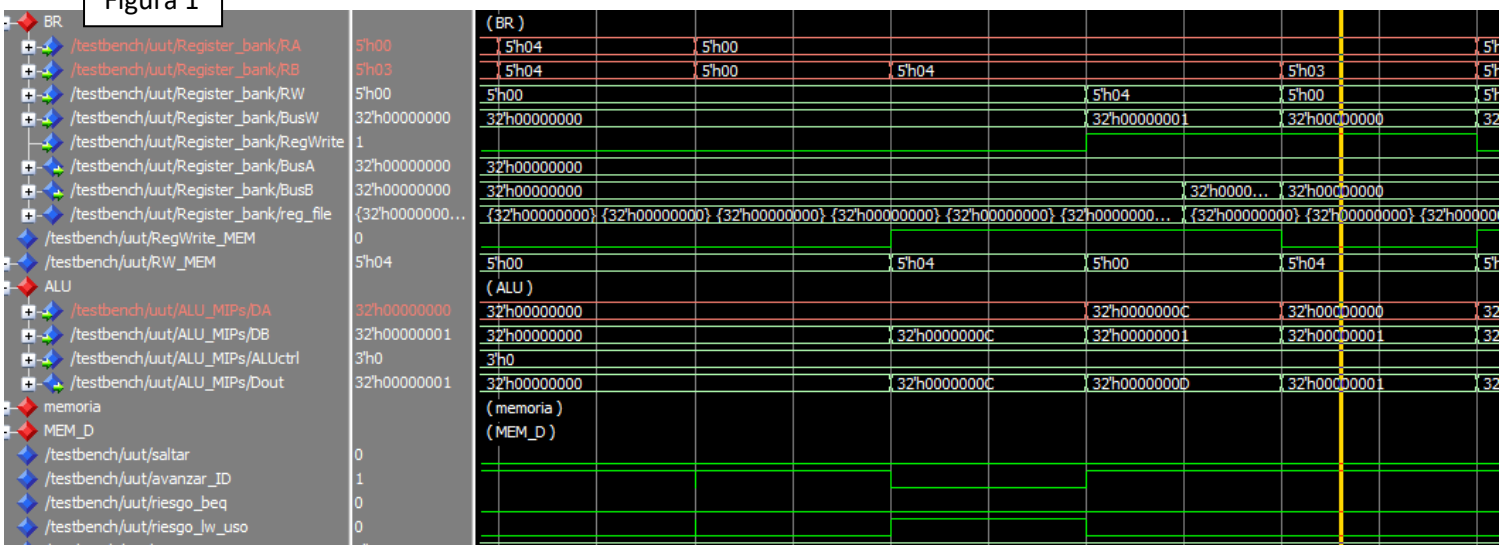


Figura 2

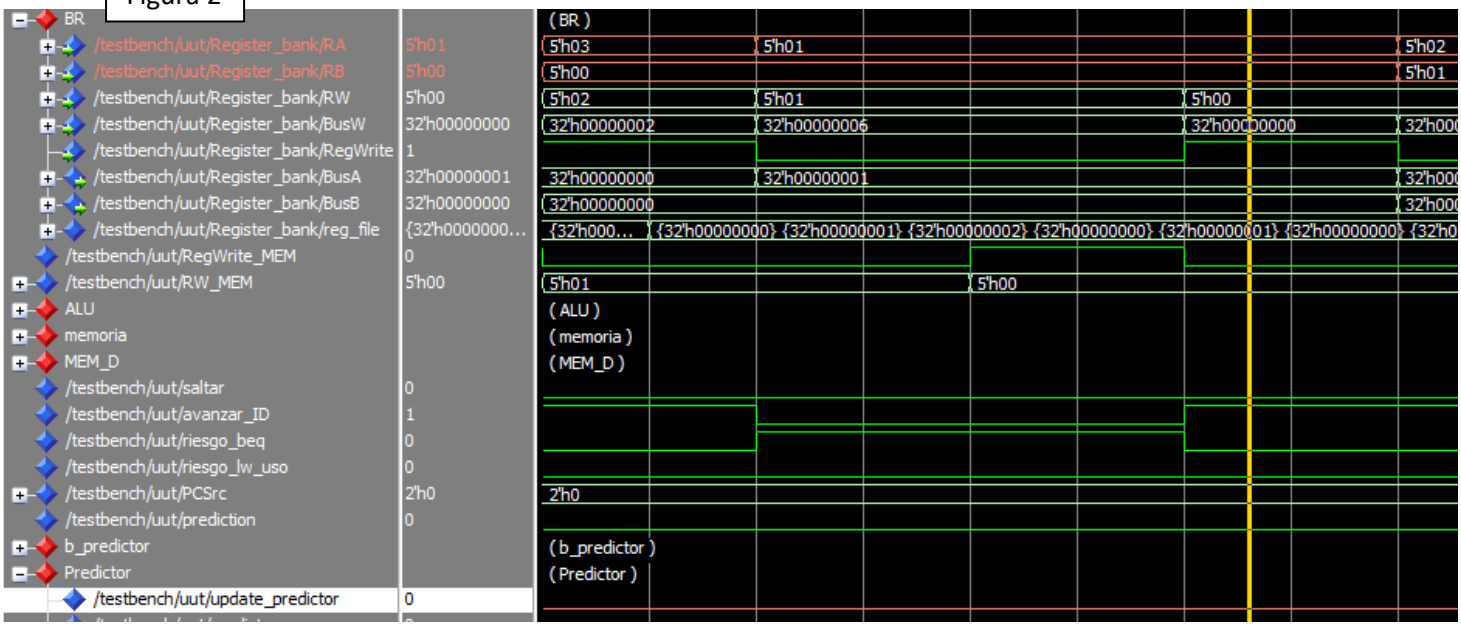
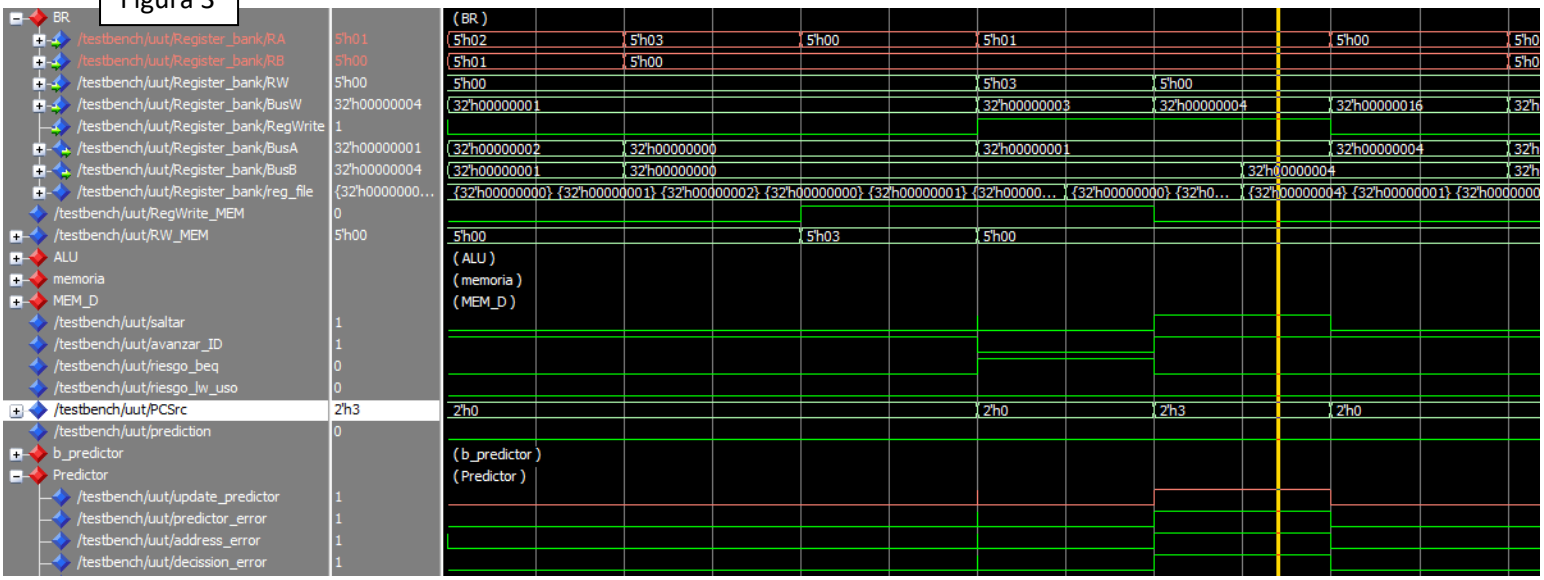


Figura 3


BR
/testbench/uut/Register_bank/RA
/testbench/uut/Register_bank/RB
/testbench/uut/Register_bank/RW
/testbench/uut/Register_bank/BusW
/testbench/uut/Register_bank/RegWrite
/testbench/uut/Register_bank/BusA
/testbench/uut/Register_bank/BusB
/testbench/uut/Register_bank/reg_file
/testbench/uut/RegWrite_MEM
/testbench/uut/RW_MEM
ALU
memoria
MEM_D
/testbench/uut/saltar
/testbench/uut/avanzar_ID
/testbench/uut/riesgo_beq
/testbench/uut/riesgo_lw_uso
/testbench/uut/PCSrc
/testbench/uut/prediction
b_predictor
Predictor
/testbench/uut/update_predictor
/testbench/uut/predictor_error
/testbench/uut/address_error
/testbench/uut/decission_error
5'h01
5'h00
5'h00
32'h00000004
1
32'h00000001
32'h00000001
{32'h0000000...
0
5'h00
2'h3
0
1
1
0
0
(BR)
(ALU)
(memoria)
(MEM_D)
(b_predictor)
(Predictor)

5. Gestión del proyecto

Paso 1 y toma de contacto con el simulador	5h
Diseño del paso 2	1h
Diseño del paso 3	1h
Diseño del paso 4	1h
Depuración y ajustes	14.30h
Memoria	4h
TOTAL	**= 26.30h**