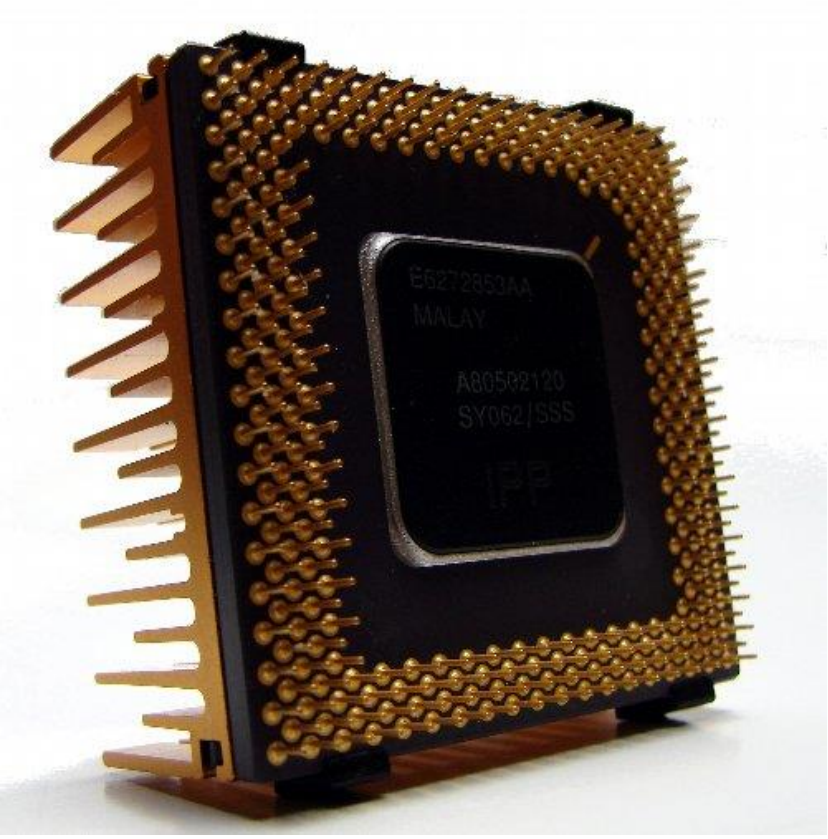
MATERIAL INDIVIDUAL F-III

PROYECTO



2020

Rendimiento en arquitecturas RISC

F-III. Estudio de los riesgos de la segmentación

Arquitectura de los Computadores
Grado en Ingeniería Informática
Dpto. Tecnología Informática y Computación
Universidad de Alicante

Material individual de la Fase III

RENDIMIENTO EN ARQUITECTURAS RISC

I. OBJETIVOS

La nueva FaseIII de la asignatura se centra en el estudio del rendimiento de arquitecturas RISC, en particular se centra en el análisis de los riesgos de la segmentación y el adelantamiento de operaciones.

El objetivo es estudiar conceptos relativos a la segmentación de una máquina. Concretamente analizaremos la arquitectura MIPS64. Para ello, se simulará la ejecución de programas ante diferentes estrategias de resolución de los riesgos de segmentación

Para la realización de esta práctica se utilizará la herramienta WinMIPS64. Con esta aplicación, además de escribir y ejecutar programas en ensamblador, podremos visualizar el progreso de ejecución de las instrucciones a través de las distintas etapas de la ruta de datos segmentada.

En el anexo 1 se proporciona un resumen del conjunto de instrucciones MIPS y en el anexo 2 se da una breve descripción sobre la utilización del simulador.

En el tema 4 de teoría se estudia una arquitectura segmentada para el procesador basada en el DLX. El DLX es una versión simplificada del MIPS que proporciona un modelo arquitectural sencillo que facilita la demostración de los principios de la segmentación y en ella los únicos riesgos por dependencia de datos que pueden ocurrir son los RAW (Read after write), es decir, riesgos que surgen porque una instrucción intenta leer un operando antes de que una instrucción haya escrito en él. **Es importante recalcar que en esta práctica no se estudiará el DLX y por tanto en esta nueva arquitectura sí pueden aparecer riesgos WAR (Write after Read) y WAW (Write after Write) aunque no los estudiaremos. Este simulador tampoco usa la técnica de implementación basada en partir el ciclo para las lecturas del banco de registros en la segunda mitad del ciclo y las escrituras en la primera mitad.**

II. DESARROLLO DE LA PRÁCTICA

La práctica consiste en un ejercicio guiado en el que se proponen 5 programas sencillos que deberéis introducir en el simulador WinMIPS64 para estudiar cómo se ejecutan las instrucciones en una máquina segmentada, comprobar lo que sucede cuando ocurre un riesgo de segmentación y analizar su posible solución.

INTRODUCCIÓN A LA SEGMENTACIÓN

En esta parte de la práctica se pretende que el estudiante se familiarice con el entorno del simulador, ejecute un programa y reflexione sobre los beneficios que aporta la segmentación a las arquitecturas RISC.

Programa 1.

Introduce el siguiente código en el simulador. Para ello genera un archivo de texto con este código y grábalo con la extensión .s, luego introdúcelo con el simulador a través del comando "Load" del menú "File":

```
.data

num: .word 7
num2: .word 8

.code

       ld  r2, num(r0)
       dadd  r3, r8, r9
       dsub  r10, r5, r6
       dsll  r1, r4, 1
       sd r4, num2(r0)

halt
```

a) Indica qué función realiza cada una de las instrucciones del código

b) Indica qué variables de datos usa este programa y dónde se muestran en el simulador

c) Ejecutad el programa en el simulador con la opción ***Configure*/*Enable Forwarding*** deshabilitada. Analizar ciclo a ciclo su funcionamiento con la opción "Single cycle" del menú Execute o bien presionando F7 sucesivamente. Examinad las distintas ventanas que se muestran en el simulador y responder:

- ¿En qué ciclo del total del programa se lee el dato que hay en la variable num? ¿a qué fase corresponde de la instrucción load?
- ¿En qué ciclo del total del programa se escribe el dato en la variable num2? ¿a qué fase corresponde de la instrucción store?
- ¿En qué ciclo del total del programa se escribe un dato en el registro r10? ¿a qué fase corresponde de la instrucción de resta?
- Tras la ejecución, ¿se produce alguna detención en el cauce? Razona tu respuesta.
- ¿Cuál es el promedio de ciclos por instrucción (CPI) en la ejecución de este programa?

RIESGOS DE LA SEGMENTACIÓN POR DEPENDENCIA DE DATOS

En esta parte de la práctica se pretende que el estudiante estudie los riegos de la segmentación por dependencia de datos. Se pretende que analice la necesidad de utilizar hardware de adelantamiento para reducir el número de detenciones.

Programa 2.

Introduce el siguiente código en el simulador.

```
.data
   A: .word  8
   B: .word  6
   C: .word 3
   D: .word 0,0,0,0

.code
   Ld r1, A(r0)
   ld r2, B(r0)
   ld r3, C(r0)
   xor r5,r5,r5
   dadd r6,r2,r3
   dadd r7,r6,r3
   dadd r8,r7,r2
   sd r6,D(r5)
   dadd r5,r5,r1
   sd r7,D(r5)
   dadd r5,r5,r1
   sd r8,D(r5)
   daddi r9,r5,8
   ld r10,D(r5)
   sd r10,D(r9)

halt
```

a) Comenta cada una de las líneas del código y explica brevemente qué realiza el código

b) Ejecuta el programa en el simulador con la opción ***Configure*/*Enable Forwarding*** deshabilitada. Analizar paso a paso su funcionamiento, examinad las distintas ventanas que se muestran en el simulador y responde:

- ¿Cuántas detenciones RAW aparecen?
- ¿Qué instrucciones están generando las mismas (stalls) en el cauce?
- Explica por qué se producen las detecciones teniendo en cuenta las instrucciones y los registros implicados. Para ello, piensa en:
  - ¿Por qué se produce la primera detención en el ciclo 6?. ¿Con qué registro e instrucción?
  - ¿Es el mismo riesgo que se produce en la segunda detención?. ¿Por qué tenemos 2 detenciones y sólo 1 en la primera detención?

        - ¿Son todos los mismos tipos de riesgos? Compara el primer riesgo producido con la última instrucción que se detiene (sd)
    - ¿Cuál es el promedio de ciclos por instrucción (CPI) en la ejecución de este programa bajo esta configuración?

b) Una forma de solucionar las paradas por dependencia de datos es utilizar el adelantamiento de operandos o Forwarding. Ejecuta nuevamente el programa anterior con la opción ***Enable Forwarding*** habilitada y responde:

    - ¿Por qué no se presenta ninguna parada en este caso? Explicar la mejora teniendo en cuenta:
        - ¿Cómo se ha solucionado el primer riesgo? ¿Desde qué unidad funcional se ha adelantado el dato para resolver el riesgo que se producía en el ciclo 6?.
        - ¿Se resuelve de la misma forma la segunda detención anterior?. ¿Desde qué unidad se adelanta?
        - Compara la solución del primer riesgo producido con la del último (sd)
    - ¿Cuál es el promedio de ciclos por instrucción (CPI) en este caso? Comparar con el anterior.

RIESGOS DE LA SEGMENTACIÓN QUE REQUIEREN DETENCIONES

En esta parte de la práctica se pretende que el estudiante estudie los riegos de la segmentación por dependencia de datos que requieren detenciones. Se analizará la necesidad de utilizar el hardware de interbloqueo en determinadas situaciones para detener el cauce y la posibilidad de utilizar el adelantamiento para reducir el número de detenciones. También se analizará la posibilidad de reorganizar el código por parte del compilador para eliminar los riesgos que requieren detenciones.

Programa 3.

Introduce el siguiente código en el simulador. Para ello genera un archivo de texto con este código y grábalo con la extensión .s, luego introdúcelo con el simulador a través del comando "Load" del menú "File":

```
.data
A: .word32  2
B: .word32  3
C: .word32  0
D: .word32  4
E: .word32  5
F: .word32  0
.code
     lw r1, A(r0)
     lw r2, B(r0)
     dadd r3,r1,r2
     sw r3, C(r0)
     lw r4, D(r0)
     lw r5, E(r0)
     dadd r6,r4,r5
     sw r6, F(r0)
halt
```

c) Indica qué función realiza cada una de las instrucciones del código. ¿Podrías expresar el código mediante dos instrucciones de un lenguaje de alto nivel?

d) Indica qué tipos de datos se están utilizando y relaciónalo con las instrucciones que se utilizan.

e) Ejecutad el programa en el simulador con la opción ***Configure*/*Enable Forwarding*** deshabilitada. Analizar ciclo a ciclo su funcionamiento con la opción "Single cycle" del menú Execute o bien presionando F7 sucesivamente. Examinad las distintas ventanas que se muestran en el simulador y responder:

- ¿Qué ocurre en los ciclos 5, 6, 13 y 14 con las instrucciones dadd?
- ¿Qué ocurre en los ciclos 8, 9, 16, 17 con las instrucciones sw?
- ¿Cuántos ciclos se consumen en total y cuantos de ellos son detenciones?
- ¿Calcula el CPI?

f) Ejecutad el programa en el simulador con la opción ***Configure*/*Enable Forwarding*** habilitada. Analizar ciclo a ciclo su funcionamiento con la opción "Single cycle" del menú Execute o bien presionando F7 sucesivamente. Examinad las distintas ventanas que se muestran en el simulador y responder:

- ¿Qué ocurre en los ciclos 6 y 11 con las instrucciones dadd? ¿Se producen adelantamientos? En su caso indica cuales.
- ¿Se producen adelantamientos en los ciclos 8 y 13 vinculados a las instrucciones sw? En su caso indica cuales.
- ¿Cuántos ciclos se consumen en total y cuantos de ellos son detenciones?
- ¿Calcula el CPI?

c) Propón una reorganización del código para reducir el número de detenciones manteniendo el resultado final del programa sobre los registros y memoria.

- Escribe el código reorganizado.
- Con el adelantamiento activado:
  - ¿Cuántos ciclos se consumen en total y cuantos de ellos son detenciones?
  - ¿Calcula el CPI?

RIESGOS DE CONTROL

En esta parte de la práctica se pretende estudiar los riesgos de control. En MIPS64 los saltos condicionales (beq, beqz y bnea, etc) requieren comprobar si dos registros son iguales o si un registro es igual a cero. En MIPS64 es posible completar esta decisión al final del ciclo ID moviendo los test de igualdad o desigualdad a este ciclo. Además, para aprovechar este adelantamiento en la decisión de salto se calculan previamente los dos valores del PC, el efectivo y el no efectivo. Para ello se añade un sumador adicional en la etapa ID que calcula la decisión de salto ya que de otro modo no estaría disponible hasta la etapa EXE. De esta manera sólo será necesario un ciclo de detención en los saltos.

Además, el MIPS64 predice el salto como no efectivo. En este esquema se permite al hardware que continúe como si el salto no se ejecutase, es decir, se realiza la búsqueda de instrucciones como si no ocurriera nada. Si el salto es no efectivo (se determina en ID), simplemente se continúa con la instrucción que se acaba de leer en la etapa IF. Si resulta que el salto es efectivo se detiene la segmentación, se reanuda la búsqueda de instrucciones en la dirección de salto y se deshacen los cambios de estado limpiando la segmentación.

En el ejemplo que sigue se verá cómo predice el MIPS los saltos como no efectivos observando un ciclo de detención cuando éstos sean efectivos. Además se estudiará cómo reducir las penalizaciones de los saltos en la segmentación mediante los saltos retardados. Con esta técnica se ejecutan instrucciones independientes del salto en los ciclos de retardo (delay slot). Estas instrucciones se van a ejecutar siempre, independientemente de que el salto sea efectivo o no, por tanto se deberá incluir una instrucción a continuación del salto que se tenga seguridad que se puede ejecutar independiente del resultado de la condición de salto.

Programa 4.

Introduce el siguiente código en el simulador:

```
.data
  cant:  .word 8
  datos: .word 1, 2, 3, 4, 5, 6, 7, 8
  res:   .word 0
.code
       dadd  r1, r0, r0
       ld  r2, cant(r0)
       loop:  ld  r3, datos(r1)
              daddi r2, r2, -1
              dsll  r3, r3, 1
              sd  r3, res(r1)
              daddi r1, r1, 8
              bnez  r2, loop
       halt
```

a) Indica cuál es el objetivo del código y cuál será el resultado.

b) Identifica los riesgos por dependencia de datos que pueden aparecer.

c) Ejecuta el programa en el simulador con la opción **Configure**/**Enable Forwarding** deshabilitada. Analizar ciclo a ciclo su funcionamiento con la opción “Single cycle” del menú Execute o bien

presionando F7 sucesivamente. Examina las distintas ventanas que se muestran en el simulador y responde a las siguientes cuestiones:

- ¿En qué ciclo ocurre la primera parada por dependencia de datos? ¿En qué instrucción?
- ¿En qué ciclo se producen dos detenciones en la misma instrucción? ¿A qué se debe?
- Observa que ocurre con la instrucción de salto. ¿En qué ciclo ocurre la primera parada por salto efectivo?
- Ejecuta el programa completo y observa que la última vez que se ejecuta el salto no hay parada debido a que el salto ha sido no efectivo.
- ¿Cuántos ciclos tarda el programa en ejecutarse? ¿Cuántos de ellos son detenciones? ¿Cuál es el CPI?

d) Ejecuta el programa en el simulador con la opción **Configure/Enable Forwarding** habilitada. Analizar ciclo a ciclo su funcionamiento con la opción "Single cycle" del menú Execute o bien presionando F7 sucesivamente. Examina las distintas ventanas que se muestran en el simulador y responder:

- ¿Cuántas dependencias de datos se han logrado solucionar?
- ¿Cuántas paradas por riesgos de control se han producido?
- ¿Cuántos ciclos de reloj tarda el programa en ejecutarse? ¿Cuál es el CPI?

e) Ejecuta el programa en el simulador con la opción **Configure/Enable Delay Slot** habilitada. Analizar ciclo a ciclo su funcionamiento con la opción "Single cycle" del menú Execute o bien presionando F7 sucesivamente. Examinad las distintas ventanas que se muestran en el simulador y responde a las siguientes cuestiones:

- ¿Cuántos ciclos se han consumido? ¿Por qué ha seguido ejecutándose la instrucción `halt` a partir del ciclo 9?
- Modifica el programa cambiando la instrucción `daddi r1, r1, 8` de lugar y colocándola después del salto (en el delay slot). Ejecuta de nuevo el programa y observa qué ocurre. ¿Es correcto el resultado?
- ¿Cuántas paradas por riesgo de control se han eliminado? ¿Se ha incrementado el número de instrucciones a ejecutar? ¿Cuántos ciclos de reloj se han consumido?
- ¿Cuál es el CPI?
- ¿Qué otra instrucción se hubiera podido colocar en el delay slot? ¿Habría sido necesario realizar algún otro cambio en el código?

f) El código del programa 5 realiza la misma función que el programa 4 que acabas de estudiar:

Programa 5.

Introduce el siguiente código en el simulador:

```
.data
  cant: .word 8
```

```
  datos: .word 1, 2, 3, 4, 5, 6, 7, 8
  res: .word 0
.code
  ld  r2, cant(r0)
  dadd  r1, r0, r0
  loop: beqz r2, fin
        ld  r3, datos(r1)
        daddi r2, r2, -1
        dsll  r3, r3, 1
         sd  r3, res(r1)
        daddi r1, r1, 8
  j loop
  fin: halt
```

- ¿Qué diferencias observas?
- Ejecuta el programa en el simulador con la opción **Configure/Enable Delay Slot** deshabilitada. ¿En qué instrucciones y en qué casos se produce detención por riesgo de control?
- Compara las estadísticas que se obtienen en el número de ciclos y CPI con las obtenidas en el programa 4 con las opción **Configure/Enable Forwarding** habilitada y deshabilitada.
- Coloca una instrucción válida en el delay slot y ejecuta con la opción **Configure/Enable Delay Slot** habilitada. ¿Cuántos ciclos se consumen? ¿Cuál es el CPI?.
- ¿Qué conclusiones puedes extraer al comparar el número de ciclos y CPI de los dos programas estudiados?

III. EVALUACIÓN

A causa de la modificación en el contenido de la Fase III de las prácticas de la asignatura, la evaluación de la parte práctica se establecerá de acuerdo con el siguiente criterio:

NP = 0,1x Fase I + 0,3 x NotaGrupalFaseII + 0,2 x NotaIndividualFaseII + 0,4 x NotaFaseIIIModificada