# Práctica AC

## Práctica 3

Evaluación del Rendimiento

Grado en ingeniería informática

Francisco Joaquín Murcia Gómez 48734281H

# Problema 1

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

1) **ld:** ld copia a un registro una palabra en una dirección de memoria, en este caso, copia en el registro "r2" una palabra de la posición de memoria "num+r0"
2) **dadd:** dadd suma con signo el registro "r8" con el registro "r9" y lo guarda en "r3"
3) **dsub:** dsub resta con signo "r6" con "r5" y lo guarda en "r10"
4) **dsll:** hace un desplazamiento a la izquierda del registro "r4" y lo guarda en "r1"
5) **sd:** guardar en "r4" a partir de la dirección de memoria dada por "num+r0"
6) **halt:** detiene el proceso

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

En la ventana data se muestran dos palabras de 64bits

## 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?**

  Se lee en la fase MEM en el ciclo 4.

- **¿En qué ciclo del total del programa se escribe el dato en la variable num2? ¿a qué fase corresponde de la instrucción store?**

  Se lee en la fase MEM en el ciclo 8

- **¿En qué ciclo del total del programa se escribe un dato en el registro r10? ¿a qué fase corresponde de la instrucción de resta?**

  Se escribe en la fase WB en el ciclo 7

- **Tras la ejecución, ¿se produce alguna detención en el cauce? Razona tu respuesta.**

  No se produce ninguna

- **¿Cuál es el promedio de ciclos por instrucción (CPI) en la ejecución de este programa?**

  10 ciclos / 6 instrucciones = 1,67 ciclos por instrucción

# Problema 2

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

1) **Ld r1,A(r0):** ld copia en el registro “r1” una palabra de la posición de memoria “A+r0”
2) **ld r2, B(r0):** ld copia en el registro “r2” una palabra de la posición de memoria “B+r0”
3) **ld r3, C(r0):** ld copia en el registro “r3” una palabra de la posición de memoria “C+r0”
4) **xor r5,r5,r5:** Realiza la operación de la puerta logica xor, en este caso, siempre aplica falso (0) al registro “r5”
5) **dadd r6,r2,r3:** dadd suma con signo el registro “r2” con el registro ”r3” y lo guarda en “r6”
6) **dadd r7,r6,r3:** dadd suma con signo el registro “r6” con el registro ”r3” y lo guarda en “r6”
7) **dadd r8,r7,r2:** dadd suma con signo el registro “r7” con el registro ”r2” y lo guarda en “r8”
8) **sd r6,D(r5):** guardar en “r6” a partir de la dirección de memoria dada por “D+r5”
9) **dadd r5,r5,r1:** dadd suma con signo el registro “r5” con el registro ”r1” y lo guarda en “r5”
10) **sd r7,D(r5):**guardar en “r6” a partir de la dirección de memoria dada por “D+r5”
11) **dadd r5,r5,r1:** dadd suma con signo el registro “r5” con el registro ”r1” y lo guarda en “r5”
12) **sd r8,D(r5):**guardar en “r8” a partir de la dirección de memoria dada por “D+r5”
13) **daddi r9,r5,8:** daddi suma con signo el registro “r5” con el numero 8 y lo guarda en “r9”
14) **ld r10,D(r5):**ld copia en el registro “r10” una palabra de la posición de memoria “D+r5”
15) **sd r10,D(r9):**guardar en “r10” a partir de la dirección de memoria dada por “D+r9”

## 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?**

  Hay 11

- **¿Qué instrucciones están generando las mismas (stalls) en el cauce?**

  Las instrucciones son las siguientes:

  dadd r6,r2,r3 ; dadd r7,r6,r3 ; dadd r8,r7,r2 ; sd r7,D(r5); sd r8,D(r5) ; sd r10,D(r9)

- **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?**

    Porque se queda a la espera que la instrucción ld r3, C(r0) escriba el valor en “r3”

  - **¿Es el mismo riesgo que se produce en la segunda detención? ¿Por qué tenemos 2 detenciones y sólo 1 en la primera detención?**

    No, la primera es porque aun no se a escrito el registro, sin embargo, la segunda esta esperando a que la anterior instrucción suma termine.

  - **¿Son todos los mismos tipos de riesgos? Compara el primer riesgo producido con la última instrucción que se detiene (sd)**

    No, las instrucciones de suma esperan a que la instrucción anterior termine de escribir

- **¿Cuál es el promedio de ciclos por instrucción (CPI) en la ejecución de este programa bajo esta configuración?**

  ciclos / instrucciones =CPI, el CPI por tanto es: 1.9357

### c) 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?**

    Al ser ld una operación de carga, cuando pasa la fase "MEM" ya esta disponible el resultado, por lo tanto, dadd puede coger el dato y calcularlo en "EX"
  - **¿Se resuelve de la misma forma la segunda detención anterior? ¿Desde qué unidad se adelanta?**

    No, en este caso se resuelve debido a que dadd es una operación aritmética, es decir, que se resuelve en la ALU, por lo tanto, cuando termina "EX" el registro "r7" ya esta calculado y puesto que no hay que guardar en memoria la siguiente instrucción puede usar "r7"
  - **Compara la solución del primer riesgo producido con la del último (sd)**

    Si, pasa lo mismo, en "MEM" ld suelta el registro y sd ya puede usarlo.
- **¿Cuál es el promedio de ciclos por instrucción (CPI) en este caso? Comparar con el anterior.**

  ciclos / instrucciones =CPI, el CPI por tanto es: 1.25, con adelantamiento podemos ver que este este procedimiento acelera la ejecución notablemente.

# Problema 3

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

1. **lw r1, A(r0):** ld copia en el registro "r1" una palabra de la posición de memoria "A+r0"
2. **lw r2, B(r0):** ld copia en el registro "r2" una palabra de la posición de memoria "B+r0"
3. **dadd r3, r1,r2:** sumamos los registros equivalentes a las palabras de las posiciones de memoria A y B y lo guardamos en el registro "r3"
4. **sw r3, C(r0):** cargamos el resultado de A+B en la posición de memoria C
5. **lw r4, D(r0):** ld copia en el registro "r4" una palabra de la posición de memoria "D+r0"
6. **lw r5, E(r0):** ld copia en el registro "r5" una palabra de la posición de memoria "E+r0"
7. **dadd r6, r4,r5:** sumamos los registros equivalentes a las palabras de las posiciones de memoria D y E y lo guardamos en el registro "r6"
8. **sw r6, F(r0):** cargamos el resultado de D+E en la posición de memoria F

En un lenguaje de alto nivel seria de esta manera:

```
variable A=2;
variable B=3;
variable C=0;
variable D=4;
variable E=5;
variable F=0;

C=A+B;
F=D+E;
```

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

Datos de tipo entero. dadd acepta tanto positivos como negativos.

### 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:

- **¿Qué ocurre en los ciclos 5, 6, 13 y 14 con las instrucciones dadd?**

  En el ciclo 5 y 6 dadd se tiene que esperar a que lw desbloque el registro "r2" debido a que esta leyendo antes que lw escriba el registro, por lo que se produce una detención RAW, cuando termina "WB" dadd puede continuar porque el dato está escrito en "r2"; en los ciclos 13 y 14 ocurre lo mismo que en los ciclos 5 y 6 pero el conflicto esta vez es con el registro "r5".

- **¿Qué ocurre en los ciclos 8, 9, 16, 17 con las instrucciones sw?**

  En el ciclo 8 y 9 le ocurre lo mismo que en el 5 y 6 de dadd, dadd está escribiendo el registro "r3" y hasta que no completa la fase de escritura (WB) dicho registro no estará disponible; con 16 y 17 pasaría igual, dadd hasta que no termine la fase de escritura no desbloquea el registro "r6".

- **¿Cuántos ciclos se consumen en total y cuantos de ellos son detenciones?**

  21 ciclos de los cuales 8 son detenciones de tipo RAW.

- **Calcula el CPI**

  ciclos / instrucciones =CPI=2.65.

### d) 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.**

  En el ciclo 6 y 11 dadd tiene que esperar a ejecutar debido a que cuando se necesita el dato lw (que está en fase MEM) necesita el dato, en el ciclo siguiente se aplica el adelantamiento y como ya esta cargado el dato lw puede escribir y dadd puede ejecutar y continuar su ciclo.

- **¿Se producen adelantamientos en los ciclos 8 y 13 vinculados a las instrucciones sw? En su caso indica cuales**

  Si, se produce adelantamiento porque en las instrucciones aritméticas de antes (dadd al finalizar el ciclo EX ya se ha calculado el dato, por lo tanto, sw ya lo puede usar.

- **¿Cuántos ciclos se consumen en total y cuantos de ellos son detenciones?**

  15 ciclos de los cuales hay 2 detenciones.

- **Calcula el CPI**

  ciclos / instrucciones =CPI=1.667.

## e) 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.**

  lw r1, A(r0)
  lw r2, B(r0)
  lw r4, D(r0)
  lw r5, E(r0)
  dadd r3,r1,r2
  sw r3, C(r0)
  dadd r6,r4,r5
  sw r6, F(r0)

- **Con el adelantamiento activado:**

  - **¿Cuántos ciclos se consumen en total y cuantos de ellos son detenciones?**

    13 iclos y 0 detenciones

  - **Calcula el CPI:**

    ciclos / instrucciones =CPI=1.444.

# Problema 4

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

El objetivo es almacenar en la dirección de memoria "res" el resultado de la multiplicación de cada dato del vector "datos" por 2, porque desplazar un bit equivale a multiplicar por 2.

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

2, la instrucción ld r2, cant(r0) con daddi r2, r2, -1 con el registro r2 y dsll r3, r3, 1 con sd r3, res(r1) con el registro r3.

## 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 ocurre la primera parada por dependencia de datos? ¿En qué instrucción?**

  daddi r2, r2, -1 en el ciclo 5

- **¿En qué ciclo se producen dos detenciones en la misma instrucción? ¿A qué se debe?**

  En el ciclo 7 y 8 en la instrucción sd r3, res(r1)

- **Observa que ocurre con la instrucción de salto. ¿En qué ciclo ocurre la primera parada por salto efectivo?**

  En el ciclo 10 en la instrucción halt

- **¿Cuántos ciclos tarda el programa en ejecutarse? ¿Cuántos de ellos son detenciones? ¿Cuál es el CPI?**

  87 ciclo de los cuales hay 25 detenciones.

  Su CPI seria ciclos / instrucciones =CPI=1.706.

### 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?**

  Todas

- **¿Cuántas paradas por riesgos de control se han producido?**

  7, las instrucciones halt.

- **¿Cuántos ciclos de reloj tarda el programa en ejecutarse? ¿Cuál es el CPI?**

  62 cliclos y su CPI seria ciclos / instrucciones =CPI=1.216.

### 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?**

  13 cliclos, porque hemos eliminado los riesgos de control ya que ha tomado halt como delay slot

- **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?**

```
0000  0000000000000008  cant: .word 8
0008  0000000000000001  datos: .word 1, 2, 3, 4, 5, 6, 7, 8
0010  0000000000000002
0018  0000000000000003
0020  0000000000000004
0028  0000000000000005
0030  0000000000000006
0038  0000000000000007
0040  0000000000000008
0048  0000000000000002  res: .word 0
0050  0000000000000004
0058  0000000000000006
0060  0000000000000008
0068  000000000000000a
0070  000000000000000c
0078  000000000000000e
```

Este seria el resultado, como absorbamos es el mismo

- **¿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?**

  Si, sin la modificación nos daba 7 riesgos y ahora ninguno, el numero de instrucciones se ha mantenido, sin embargo, los ciclos se han reducido de 62 a 51 ciclos.

- **¿Cuál es el CPI?**

  ciclos / instrucciones =CPI=1.078.

- **¿Qué otra instrucción se hubiera podido colocar en el delay slot? ¿Habría sido necesario realizar algún otro cambio en el código?**

  La instrucción daddi r2, r2, -1, sin embargo, habría que inicializar cant a 7.

# Problema 5

- **¿Qué diferencias observas?**

  La diferencia es que en el 4 hacia un bucle tipo do while y aqui un tipo while

- **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?**

  Ocurre en las instrucciones halt porque el procesador intenta ejecutar la instrucción de después del salto, pero como ha saltado aborta.

- **Compara las estadísticas que se obtienen en el número de ciclos y CPI con las obtenidas en el programa 4 con la opción Configure/Enable Forwarding habilitada y deshabilitada.**

| | programa 4 | programa 5 |
|---|---|---|
| **Cliclos sin Forwarding** | 87 | 98 |
| **Cliclos con Forwarding** | 62 | 74 |
| **CPI sin Forwarding** | 1.706 | 1.633 |
| **CPI con Forwarding** | 1.216 | 1.233 |

  En sin adelantamiento tenemos una ganancia de 1.04 del programa 5 respecto al 4 y con adelantamiento una del 0.98, es decir en este caso el programa 4 es mejor que el 5

- **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?**

  Si colocamos daddi r1, r1, 8 como delay slot obtenemos 66 ciclos y un CPI de 1.086

- **¿Qué conclusiones puedes extraer al comparar el número de ciclos y CPI de los dos programas estudiados?**

  La configuración del programa 5 es más útil si no se puede usar ni adelantamiento ni delay slot, sin embargo el 4 premia con estas configuraciones por algunas decimas (si miramos su CPI); en el caso de con delay slot activado la diferencia es casi irrelevante, pero existente.