

Responde cada pregunta en una hoja distinta. Tiempo disponible: 3h

1. (2.5 puntos) Se desean mejorar las prestaciones de un dispositivo móvil que dispone de un procesador Qualcomm a 1.4GHz y una GPU Mali-G71 MP8, con un coste de producción de 300€. Para ello se desea evaluar desde el punto de vista coste/prestaciones diferentes alternativas.

Para evaluar las alternativas se empleará un software de test que da un tiempo de ejecución de 10.5 segundos en el dispositivo original. Se comprueba que dicho software emplea un 55 % del tiempo en operaciones con la CPU. El resto del tiempo realiza operaciones con la GPU y otros dispositivos.

En una primera prueba, cambiando la GPU por una Mali-G71 MP20, que ofrece una mejora del 150 % sobre la GPU original, se obtiene un tiempo de ejecución para el software de test de 8.9 segundos. El incremento de coste de esta nueva GPU es de 50€.

Responder a las siguientes cuestiones.

- ¿Cuál es la aceleración global obtenida al reemplazar la GPU?
- En el dispositivo original, ¿qué fracción de tiempo empleaba el software de test en operaciones con la GPU? ¿Y en operaciones con otros dispositivos?
- Si se sustituye el procesador del dispositivo original por uno con la misma arquitectura, pero a 2.0GHz y con un aumento del coste de 70€. ¿Qué aceleración global se obtendrá?
- ¿Qué aceleración global se obtendrá si aplicamos simultáneamente en el dispositivo original los cambios propuestos en los dos apartados anteriores? Considera que este cambio aumenta el coste en 100€.
- Finalmente, desde el punto de vista coste/prestaciones. ¿Qué alternativas son interesantes: cambiar la GPU, el procesador o ambos?

**Solución:**

- a) ¿Cuál es la aceleración global obtenida al reemplazar la GPU?

La aceleración global obtenida se calcula,  $S'_{gpu} = \frac{T_{eo}}{T_e} = \frac{10,5}{8,9} = 1,18$  por lo que se obtendrá una mejora de 1.18 ó del 18 %.

- b) En el dispositivo original, ¿qué fracción de tiempo empleaba el software de test en operaciones con la GPU? ¿Y en operaciones con otros dispositivos?

Aplicando la ley de Amdahl,

$$S'_{gpu} = \frac{1}{1 - F_{gpu} + \frac{F_{gpu}}{S_{gpu}}} \quad (1)$$

conocemos  $S'_{gpu}$  del apartado anterior, y  $S_{gpu} = 2,5$  ya que el enunciado indica que la nueva GPU ofrece una mejora del 150 %. Así podemos despejar la fracción de tiempo del componente mejorado  $F_{gpu}$ ,

$$F_{gpu} = \frac{S_{gpu}(S'_{gpu} - 1)}{S'_{gpu}(S_{gpu} - 1)} = \frac{2,5(1,18 - 1)}{1,18(2,5 - 1)} = 0,254 \quad (2)$$

Por lo tanto la fracción de tiempo empleada en la GPU originalmente era de 25,4 %.

El resto de componentes representan la fracción que queda tras quitar las fracciones del procesador y la GPU:  $F_{resto} = 100 - 55 - 25,4 = 19,6$  %.

- c) Si se sustituye el procesador actual por uno con la misma arquitectura, pero a 2.0Ghz y con un coste de 70€. ¿Qué mejora se obtendrá?

Aplicando la ley de Amdahl de nuevo,

$$S'_{cpu} = \frac{1}{1 - F_{cpu} + \frac{F_{cpu}}{S_{cpu}}} \quad (3)$$

tenemos  $F_{cpu} = 0,5$ , y podemos calcular  $S_{cpu} = \frac{2Ghz}{1,4Ghz} = 1,429$ . Así tenemos:

$$S'_{cpu} = \frac{1}{1 - 0,55 + \frac{0,55}{1,429}} = 1,198 \quad (4)$$

La aceleración global obtenida en este caso será de 1,198 ó del 19,8 %.

- d) ¿Qué aceleración global se obtendrá si sustituimos tanto el procesador como la GPU? Existe un único chip que contiene ambos elementos, así en este caso el coste sería de 100€.

De nuevo Amdahl, pero en este caso para varios componentes,

$$S' = \frac{1}{1 - F_{cpu} - F_{gpu} + \frac{F_{cpu}}{S_{cpu}} + \frac{F_{gpu}}{S_{gpu}}} \quad (5)$$

tenemos todos los valores que necesitamos,

$$S' = \frac{1}{1 - 0,55 - 0,254 + \frac{0,55}{1,429} + \frac{0,254}{2,5}} = 1,465 \quad (6)$$

En esta ocasión la mejora sería del 1,465 ó del 46,5 %.

- e) Finalmente, desde el punto de vista coste/prestaciones. ¿Qué alternativas son interesantes: cambiar la GPU, el procesador o ambas?

1) Nueva GPU: mejora del 18 %, incremento de coste es del  $50/300 = 16,67\%$ .

2) Nuevo procesador: mejora del 19,8 %, incremento de coste es del  $70/300 = 23,33\%$ .

3) Nueva CPU y procesador: mejora del 45,6 %, incremento de coste es del  $100/300 = 33,33\%$ .

Las alternativas 1) y 3) son interesantes desde el punto de vista de coste prestaciones, siendo la más interesante la 3).

□

2. (2.5 puntos) Sobre un microcontrolador Infineon ARM Cortex M4 con un reloj a 200 Mhz y con juego de instrucciones load/store se ejecutan varias tareas concurrentes. Dichas tareas comparten recursos y para sincronizar el acceso a los mismos se emplean operaciones *test-and-set* en las que se garantiza la atomicidad inhabilitando las interrupciones.

```

                                # test-and-set sobre la
                                # variable en memoria: lock
                                #
di                               # inhabilita interrupciones
addi r2,r0,#1                   # r2 <- 1
lb r1,lock(r0)                  # lee valor sobre r1
sb r2,lock(r0)                  # lock <- 1
ei                               # habilita interrupciones

```

Tras estudiar la carga que generan las tareas se obtiene la siguiente distribución de operaciones:

Operación	%	CPI
ALU	36	1
Load	24	2
Store	12	2
Salto	18	1.5
<i>di</i>	4	1
<i>ei</i>	6	1

Los ingenieros se están planteando introducir una modificación en la arquitectura para que la operación *test-and-set* se implemente en una sola instrucción. De este modo todo el código anterior se sustituiría por:

```
TaS r1, lock(r0) # cargar en r1 el valor de la posición look
                  # y luego asigna 1 a dicha posición
```

dicha instrucción tendría un CPI de 3. La frecuencia de reloj no cambia.

- Calcular el CPI del procesador original, así como el tiempo de ejecución en segundos de una tarea formada por  $n$  instrucciones.
- Sabiendo que en el procesador original el 50 % de las instrucciones *di* se emplean en operaciones *test-and-set*. Calcular la nueva distribución de instrucciones en el procesador modificado.
- Sobre el procesador modificado calcular el CPI, así como el número de instrucciones que tendría una tarea que en el procesador original tenía  $n$  instrucciones.
- Cuantifica la aceleración proporcionada por la nueva arquitectura.

#### Solución:

- Calcular el CPI del procesador original, así como el tiempo de ejecución en segundos de una tarea formada por  $n$  instrucciones.

$$CPI = 0,36 \times 1 + 0,24 \times 2 + 0,12 \times 2 + 0,18 \times 1,5 + 0,04 \times 1 + 0,06 \times 1 = 1,45 \quad (7)$$

$$T_e = I \times CPI \times T = n \times 1,45 \times 200^{-1} \times 10^{-6} = n \times 7,25 \times 10^{-9} \text{seg} \quad (8)$$

- Sabiendo que en el procesador original el 50 % de las instrucciones *di* se emplean en operaciones *test-and-set*. Calcular la nueva distribución de instrucciones en el procesador modificado.

Las 5 instrucciones originales que implementaban el *test-and-set* serán sustituidas en el nuevo procesador por una única instrucción. Como referencia sabemos que el 50 % de las instrucciones *di* del procesador original se usaban para este fin. Mirando la tabla vemos que las instrucciones *di* aparecen en un 4 % de las ocasiones, así que por cada 4 instrucciones *di* en el procesador original,  $4 \times 0,5 = 2$  son empleadas para implementar el *test-and-set*. Por ello, deberemos restar este número de instrucciones *di* a las originales, y en la misma proporción a las *ALU*, *LOAD*, *STORE* y *ei* que tampoco se emplearán (ver tabla adjunta). Por otra parte aparecerán en esta misma proporción instrucciones *TaS*.

Dado que la nueva distribución (ver columna #’ de la tabla) no suma 100, sino 92, habrá que normalizar por este valor para obtener los porcentajes de la nueva distribución (ver columna %’ de la tabla).

Operación	%	CPI	#’	%’	CPI’
ALU	36	1	36-2=34	37	1
Load	24	2	24-2=22	23.9	2
Store	12	2	12-2=10	10.9	2
Salto	18	1.5	18	19.6	1.5
<i>di</i>	4	1	4-2=2	2.2	1
<i>ei</i>	6	1	6-2=2	4.4	1
<i>TaS</i>	-	-	+2=2	2.2	3
Total	100	-	92	100	-

- c) Sobre el procesador modificado calcular el CPI, así como el número de instrucciones que tendría una tarea que en el procesador original tenía  $n$  instrucciones.

$$CPI' = 0,37 \times 1 + 0,239 \times 2 + 0,109 \times 2 + 0,196 \times 1,5 + 0,022 \times 1 + 0,044 \times 1 + 0,022 \times 3 = 1,49 \quad (9)$$

En el nuevo procesador, una tarea que originalmente tuviera  $n$  instrucciones tendría ahora  $0,92 \times n$  instrucciones.

- d) Cuantifica la mejora, si existe, de la nueva arquitectura.

Necesitamos conocer el nuevo tiempo de ejecución,

$$T'_e = I' \times CPI' \times T' = 0,92 \times n \times 1,49 \times 200^{-1} \times 10^{-6} = n \times 6,854 \times 10^{-9} \text{ seg} \quad (10)$$

por tanto la mejora se calcularía,

$$S = \frac{T_e}{T'_e} = \frac{n \times 7,25 \times 10^{-9}}{n \times 6,854 \times 10^{-9}} = 1,0578 \quad (11)$$

Por tanto la nueva arquitectura ofrece una mejora del 5.78 %.

□

3. (2.5 puntos) Se dispone de un procesador MIPS en el cual se ejecuta el siguiente bucle:

```
loop:  l.d f1, 0(r1)
      l.d f2, 0(r2)
      add.d f4, f1, f7
      add.d f5, f2, f8
      div.d f7, f6, f1
      div.d f8, f6, f2
      mul.d f5, f4, f5
      sub.d f5, f5, f10
      s.d f5, 0(r3)
      daddi r1, r1, 8
      daddi r2, r2, 8
      daddi r3, r3, 8
      bne r1, r5, loop
      trap 0
      <sgte1>
      <sgte2>
      <sgte3>
```

El procesador dispone de los siguientes operadores multiciclo de coma flotante:

- Sumador/Restador. Lat= 2, IR= 1, etapas A1, A2.
- Multiplicador. Lat= 3, IR= 1, etapas M1, M2, M3.
- Divisor. Lat= 5, IR= $\frac{1}{5}$ , etapas D1, D2, D3, D4, D5.

Los riesgos estructurales y de datos se detectan en la fase ID, insertando tantos ciclos de parada como sean necesarios y, en el caso de los riesgos de datos, utilizando cortocircuitos siempre que sea posible.

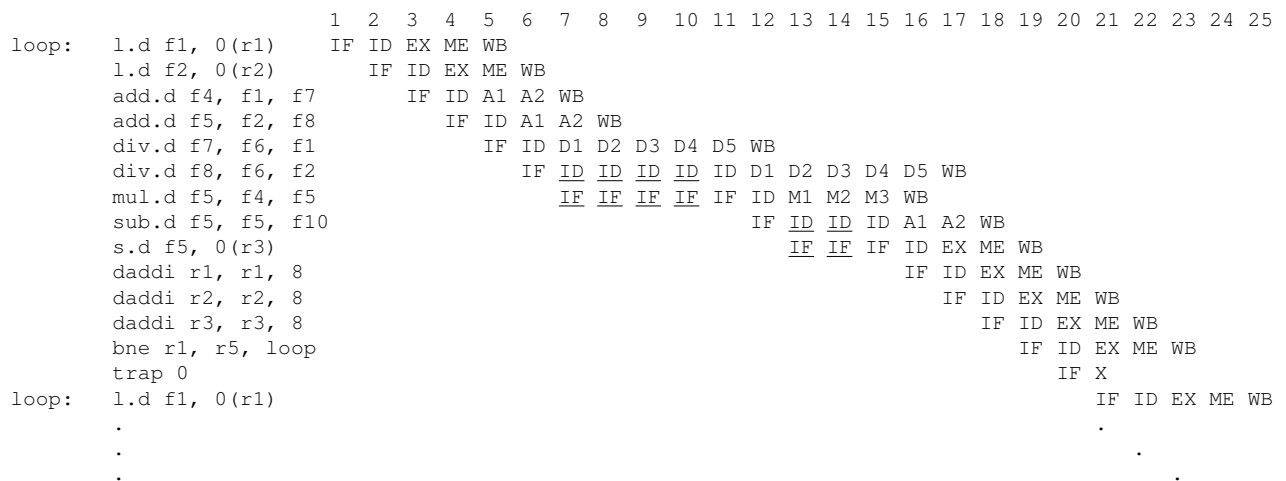
Los riesgos de control se resuelven mediante la técnica predict-not-taken. El cálculo de la condición de salto, el de la dirección destino, y la escritura del PC, se realizan en la etapa ID.

Teniendo en cuenta que las operaciones de enteros y de memoria utilizan las 5 etapas clásicas: IF, ID, EX, ME y WB, mientras que las instrucciones multiciclo las etapas IF, ID, <ejecución en el operador multiciclo correspondiente> y WB, y que se dispone de dos bancos de registros (1 flotante y 1 entero) con 2 puertos de lectura y 1 puerto de escritura cada banco, se pide:

- El diagrama instrucciones-tiempo de la primera iteración, incluyendo la instrucción que se ejecuta después de la bne.
- Si en el apartado 3a ha sido necesario introducir ciclos de parada debido a riesgos, identifique para cada caso el tipo de riesgo y las instrucciones involucradas explicando el motivo de la detención.
- A partir del diagrama anterior, indique el tiempo de ejecución de una iteración (en ciclos) cuando el predictor acierta y cuando el predictor falla.
- ¿Que ocurriría si este mismo código (**sin ninguna modificación**) se ejecutara en un procesador que resolviera los riesgos de control mediante la técnica del salto retardado? Razone la respuesta.

### Solución:

- El diagrama instrucciones-tiempo de la primera iteración, incluyendo la instrucción que se ejecuta después de la beqz.



- Si en el apartado 3a ha sido necesario introducir ciclos de parada debido a riesgos, identifique para cada caso el tipo de riesgo y las instrucciones involucradas explicando el motivo de la detención.

**Riesgo 1** estructural. El operador de división es no segmentado. Las instrucciones involucradas son `div.d f7, f6, f1` y `div.d f8, f6, f2`. La segunda división tiene que esperar a que termine la primera.

**Riesgo 2** de datos. La instrucción `sub.d f5, f5, f10` no puede proceder hasta que el valor del registro f5 no esté disponible, con lo que se insertan dos ciclos de parada y luego se procede en el ciclo 16 a activar el cortocircuito de WB a A1.

- A partir del diagrama anterior, indique el tiempo de ejecución de una iteración (en ciclos) cuando el predictor acierta y cuando el predictor falla.

La penalización por una mala predicción es de tan solo 1 ciclo de reloj, ya que la escritura del PC se realiza en la etapa ID

Acierto = 19 ciclos.

Fallo = 20 ciclos.

- d) ¿Que ocurriría si este mismo código (**sin ninguna modificación**) se ejecutara en un procesador que resolviera los riesgos de control mediante la técnica del salto retardado? Razone la respuesta.

Ocurriría que no iteraría ya que la instrucción `trap` formaría parte del *Delay-slot* y al ejecutarse terminaría el programa. La solución es poner una instrucción válida en el *Delay-slot* si se encuentra y si no poner una NOP, y la `trap` dejarla fuera del *Delay-slot*

□

#### 4. (2.5 puntos)

El siguiente programa copia las componentes distintas de cero del vector fuente al vector destino.

```
i = 0;
j = 0;
n = 10;
do
{
    if (fuente[i] != 0) /* Salto b1 */
    {
        destino[j] = fuente[i];
        j++;
    }
    i++;
    n--;
}
while (n != 0) /* Salto b2 */
```

Este programa se traduce por un compilador en el siguiente código ensamblador MIPS64.

```
        daddi r1,r0,0    # r1 = Dirección del vector fuente
        daddi r2,r0,80   # r2 = Dirección del vector destino
        daddi r3,r0,10   # r3 = Guarda del bucle (n)
loop:   ld r10,0(r1)
        daddi r1,r1,8
        beqz r10,fi      # Salto b1 (salta si fuente[i] == 0)
        sd r10,0(r2)
        daddi r2,r2, 8
fi:     daddi r3,r3,-1
        bnez r3,loop     # Salto b2 (salta si n != 0)
        trap 0
        nop
        nop
        nop
```

Este código se ejecuta en un procesador MIPS64 segmentado en las 5 etapas habituales, resultando en el siguiente diagrama de instrucciones-tiempo para la primera iteración del bucle.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
loop: ld r10,0(r1)	IF	ID	EX	ME	WB									
daddi r1,r1,#8		IF	ID	EX	ME	WB								
beqz r10,fi			IF	ID	EX	ME	WB							
sd r10,0(r2)				IF	ID	X								
daddi r2,r2,#8					IF	X								
fi:     daddi r3,r3,-1						IF	ID	EX	ME	WB				
bnez r3,loop							IF	ID	EX	ME	WB			
trap #0								IF	ID	X				
nop									IF	X				
loop: ld r10,0(r1)										IF	ID	EX	ME	WB

Teniendo en cuenta que el procesador cuenta con un predictor dinámico BTB con 1 bit para la condición de salto, y asumiendo que la BTB se encuentra vacía al principio de la ejecución del programa, se pide, **razonando las respuestas**:

- a) ¿En qué etapa se actualiza el PC en caso de salto efectivo? ¿En qué etapa se calcula la condición de salto?

- b) Completa en la hoja adjunta la traza de la ejecución del salto b2. Con respecto a este salto, ¿cuántas veces acertará el predictor?
- c) Asumiendo que el vector fuente contiene las componentes 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, completa en la hoja adjunta la traza de la ejecución del salto b1. Con respecto a este salto, ¿cuántas veces acertará el predictor?
- d) Se ha decidido sustituir el predictor de un bit por otro de dos bits con histéresis y estados *Strongly Not Taken* (SNT), *Weakly Not Taken* (WNT), *Weakly Taken* (WT), y *Strongly Taken* (ST). Asumiendo que antes de ejecutar el programa la entrada en la BTB del salto b1 se encuentra en el estado SNT y que el vector fuente contiene las mismas componentes que el apartado anterior, ¿cuántas veces acertará el predictor con respecto a salto b1?

### Solución:

- a) ¿En qué etapa se actualiza el PC en caso de fallo en la predicción? ¿En qué etapa se calcula la condición de salto?

Si se observa el siguiente detalle del diagrama instrucciones-tiempo,

		6	7	8	9	10	11	12	13	14
fi:	daddi r3,r3,-1	IF	ID	<b>EX</b>	ME	WB				
	bnez r3,loop		IF	ID	<b>EX</b>	ME	WB			
	trap #0			IF	ID	X				
	nop				IF	X				
loop:	ld r10,0(r1)				<b>IF</b>	ID	EX	ME	WB	

se comprueba que la instrucción correcta `ld r10,0(r1)` se busca justo después de la etapa EX del salto `bnez r3,loop` (ciclo 9), lo cual significa que el PC se actualiza en esa etapa.

Por otro lado, el contenido del registro r3, el cual se usa para calcular la condición de salto, se produce en la etapa EX de la instrucción `daddi r3,r3,-1` (ciclo 8), lo que significa que la condición de salto también debe calcularse también en la etapa EX del salto (ciclo 9).

- b) Completa en la hoja adjunta la traza de la ejecución del salto b2. Con respecto a este salto, ¿cuántas veces acertará el predictor?

Traza del funcionamiento del predictor:

Iteración:		1	2	3	4	5	6	7	8	9	10
Sí salta/No salta:	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	No
Predicción:	No	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí
Acierto/Fallo:	<b>F</b>	A	A	A	A	A	A	A	A	A	<b>F</b>

Se producen 8 aciertos en total.

- c) Asumiendo que el vector fuente contiene las componentes 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, completa en la hoja adjunta la traza de la ejecución del salto b1. Con respecto a este salto, ¿cuántas veces acertará el predictor?

Traza del funcionamiento del predictor:

Iteración:		1	2	3	4	5	6	7	8	9	10
fuelle[i]:		0	1	0	1	0	1	0	1	0	1
Sí salta/No salta:	Sí	No	Sí	No	Sí	No	Sí	No	Sí	No	No
Predicción:	No	Sí	No	Sí	No	Sí	No	Sí	No	Sí	Sí
Acierto/Fallo:		F	F	F	F	F	F	F	F	F	F

Se producen 0 aciertos en total.

- d) Se ha decidido sustituir el predictor de un bit por otro de dos bits con histéresis y estados *Strongly Not Taken* (SNT), *Weakly Not Taken* (WNT), *Weakly Taken* (WT), y *Strongly Taken* (ST). Asumiendo que antes de ejecutar el programa la entrada en la BTB del salto b1 se encuentra en el estado SNT y que el vector fuente contiene las mismas componentes que el apartado anterior, completa en la hoja adjunta la traza de la ejecución del salto b1. Con respecto a este salto, ¿cuántas veces acertará el predictor?

Traza del funcionamiento del predictor:

Iteración:	1	2	3	4	5	6	7	8	9	10
fuelle[i]:	0	1	0	1	0	1	0	1	0	1
Sí salta/No salta:	Sí	No	Sí	No	Sí	No	Sí	No	Sí	No
SNT/WNT/WT/ST:	SNT	WNT	SNT	WNT	SNT	WNT	SNT	WNT	SNT	WNT
Predicción:	No	No	No	No	No	No	No	No	No	No
Acierto/Fallo:	F	A	F	A	F	A	F	A	F	A

Se producen 5 aciertos en total.

