2

La CPU

Objetivos

En los problemas de este capítulo se tratan aspectos claves relativos al tema de la CPU analizados en las clases expositivas: mejora del rendimiento, soporte a los sistemas operativos multitarea y soporte a la virtualización.

Problema 10.

Se pretende realizar desde cero una CPU que implemente la arquitectura AMD-64. Para ello, en primer lugar se ha diseñado una versión preliminar secuencial de la CPU, la cual es capaz de ejecutar 200 millones de instrucciones por segundo trabajando a la máxima frecuencia. La ejecución de cualquier instrucción se divide en diez etapas de igual duración que funcionan de forma secuencial. Cada una de estas etapas necesita un ciclo de reloj para completarse.

❑ **10.1** ¿Cuál es la frecuencia de reloj que debe emplearse con dicha CPU para alcanzar ese número de instrucciones por segundo?

2 GHz

❑ **10.2** ¿Cuál es tiempo mínimo necesario para ejecutar cualquier instrucción en dicha CPU? Contestar en nanosegundos.

5 ns

❑ **10.3** ¿Cuántas instrucciones puede ejecutar por cada ciclo de reloj (IPC)? Ejemplo: 1.2 instrucciones/ciclo.

0.1 instrucciones/ciclo

Una vez se ha realizado la versión inicial de la CPU, esta se ha modificado empleando la técnica de segmentación. Para ello, se modifica la unidad de control de tal forma que se permite que las diez etapas funcionen en paralelo. Debes despreciar el retardo que introducen los registros de la segmentación.

❑ **10.4** ¿Cuál es ahora el tiempo mínimo necesario para ejecutar cualquier instrucción, y cuántas instrucciones por segundo puede ejecutar, a lo sumo, la versión segmentada de la CPU?

5 ns
2000 MIPS

❑ **10.5** ¿Qué modificación en la organización interna de la CPU propondrías para conseguir que ejecutase 3000 MIPS?

Por ejepmlo, podría incrementarse el número de etapas de segmentación, incrementando así la frecuencia de reloj. También podría convertirse la CPU en superescalar.

Problema 11.

Dos compañías fabricantes de CPU se disputan el mercado. La primera de ellas dispone de la CPU A10, capaz de ejecutar 3000 MIPS trabajando a una frecuencia de 1.5 GHz.

La segunda compañía pretende competir con la primera poniendo a la venta la CPU modelo B20-PRO, la cual trabaja a una frecuencia de 2 GHz, e implementa el mismo juego de instrucciones que la CPU de la competencia. Para obtener este producto, la compañía parte de una CPU secuencial, modelo B20, que trabaja a 500 MHz y requiere una media de 4 ciclos de reloj para ejecutar cualquier instrucción.

❑ **11.1** ¿Cuál es el tiempo medio necesario para ejecutar una instrucción en la CPU B20?

4 ciclos $\times$ 2 ns/ciclo = 8 ns

❑ **11.2** La CPU B20-PRO se obtiene dividiendo la ejecución de las instrucciones en N etapas perfectamente balanceadas de duración 1 ciclo de reloj (de 2 GHz) que trabajan de forma concurrente. ¿Cuál debe ser el valor de N?

N = 8 ns / 0.5 ns = 16

❑ **11.3** ¿Qué CPU proporciona una mayor productividad, la A10 que trabaja a 1.5 GHz, o la B20-PRO que trabaja a 2 GHz? ¿Por qué?

La B20-PRO al estar segmentada es capaz de ejecutar una instrucción cada ciclo de reloj como máximo. Como su frecuencia de reloj es de 2 GHz, resulta que puede ejecutar 2000 MIPS, menor que los 3000 MIPS de la CPU A10.

Por lo tanto, la CPU A10 proporciona una mayor productividad aunque tenga una frecuencia de reloj menor.

Problema 12.

Se dispone de una CPU que es capaz de ejecutar 200 millones de instrucciones por segundo a la máxima frecuencia de funcionamiento. La ejecución de cualquier instrucción requiere completar secuencialmente 15 etapas, empleando en cada una de las etapas un ciclo de reloj. Dadas estas características de la CPU, contestar a las siguientes preguntas:

❑ **12.1** ¿Cuál es la máxima frecuencia a la que puede trabajar esta CPU en el modo no segmentado?

200 millones de instrucciones/s × 15 ciclos/instrucción = 3 GHz

❑ **12.2** ¿Cuánto tiempo, expresado en nanosegundos, requiere la ejecución de una instrucción?

1 s / 200 millones instrucciones = 5 ns/instrucción

❑ **12.3** ¿Cuántas instrucciones puede ejecutar esta CPU por ciclo de reloj?

1 instrucción / 15 ciclos = 0.0667 instrucciones/ciclo

Se supone ahora otra CPU con idéntica arquitectura del juego de instrucciones que la anterior, con una frecuencia de trabajo de 1.5 GHz, y en este caso, segmentada en 10 etapas que pueden trabajar en paralelo, requiriendo cada etapa un ciclo de reloj.

❑ **12.4** ¿Cuál es el número máximo de instrucciones por segundo que podrá ejecutar esta CPU en las condiciones descritas?

1500 MIPS

❑ **12.5** ¿Cuánto tiempo, expresado en nanosegundos, requiere cualquier instrucción para ejecutarse en esta nueva CPU?

10 ciclos $\times(1/(1.5 \times 10^9))$ ns/ciclo = 6.67 ns

❑ **12.6** ¿Cuántas instrucciones por ciclo de reloj puede ejecutar esta CPU?

1 instrucción por ciclo de reloj

Problema 13.

Se dispone de una CPU que implementa la arquitectura del juego de instrucciones de MIPS64. Se trata de una CPU no segmentada donde la ejecución de las instrucciones se divide en cinco pasos cuyas duraciones son 50 ns, 50 ns, 60 ns, 50 ns y 40 ns.

❑ **13.1** ¿Cuál es tiempo necesario para ejecutar una instrucción en esta CPU?

$50 + 50 + 60 + 50 + 40 = 250$ ns

❑ **13.2** ¿Cuál es el número de instrucciones por segundo que puede ejecutar la CPU?

4 MIPS

Esta CPU se modifica para convertirla en segmentada, de tal forma que los 5 pasos anteriores se convierten en las etapas de la segmentación.

❑ **13.3** ¿Cuál es ahora el tiempo necesario para ejecutar una instrucción en la versión segmentada de la CPU?

$60 \times 5 = 300\,\texttt{ns}$

❑ **13.4** ¿A qué frecuencia de reloj trabajaría la versión segmentada de la CPU?

$16.67\,\texttt{MHz}$

❑ **13.5** ¿Cuántas instrucciones por segundo sería capaz de ejecutar la CPU segmentada en situaciones ideales?

$16.67\,\texttt{MIPS}$

❑ **13.6** ¿Cuál es la aceleración de la productividad en la ejecución de instrucciones que se obtiene al segmentar la CPU suponiendo el caso ideal?

$250/60 = 4.17$

Problema 14.

Indica cuál o cuáles de las siguientes afirmaciones son CIERTAS. Contesta NINGUNA si crees que ninguna lo es.

A) La ejecución de un programa sobre una CPU segmentada provoca *riesgos de control* en la segmentación cuando existen instrucciones que provocan cambios en el flujo del programa: saltos, llamadas a funciones, etc.

B) En una CPU segmentada se produce un *riesgo de tipo estructural* cuando al ejecutar instrucciones varias etapas intentan acceder al mismo dato simultáneamente.

C) En una CPU segmentada se dice que existe un *riesgo de datos* cuando varias instrucciones intentan operar sobre datos, y tienen que esperar porque necesitan utilizar la misma unidad funcional que no está replicada.

D) En las condiciones más favorables posibles, una CPU segmentada puede alcanzar una productividad de una instrucción cada ciclo de reloj.

E) En una CPU segmentada que dispone de unidades de ejecución monociclo y multiciclo, el tiempo requerido para que se ejecute cualquier instrucción es siempre el mismo.

`A y D`

Problema 15.

Se dispone de una CPU MIPS64 que presenta las siguientes latencias:

- Etapa `IF`: 0.25 ns.
- Etapa `ID`: 0.12 ns.
- Etapa `EX` para instrucciones aritmético-lógicas: 0.15 ns.
- Etapa `MEM`: 0.25 ns.
- Etapa `WB`: 0.15 ns.
- Unidad de multiplicación de enteros: 0.7 ns.

❑ **15.1** ¿Qué productividad en MIPS máxima puede alcanzar una versión secuencial de la CPU? Redondea para no mostrar decimales.

`1087 MIPS`

❑ **15.2** ¿Cuál es la frecuencia de trabajo de la versión segmentada de la CPU?

`4 GHz`

❑ **15.3** Con estos parámetros temporales y en condiciones ideales, ¿cuál sería la aceleración que se obtiene en la versión segmentada respecto a la secuencial? Redondea a dos decimales.

`3.68`

Sobre la versión segmentada de la CPU se ejecuta el siguiente código. El *pipeline* de la CPU no incluye ninguna mejora, si bien se permite la terminación de las instrucciones fuera de orden.

```
1         daddi r1, r0, 200
2 loop:
3         ld    r2, 0(r2)
4         dmul  r2, r3, r1
5         daddi r1, r1, -1
6         dmul  r2, r3, r2
7         bnez  r1, loop
8         dadd  r9, r2, r3
```

❑ **15.4** Completa el cronograma de ejecución del código mostrado hasta el ciclo 15 teniendo en cuenta las características del *pipeline*.

Instrucción \ Ciclo	4	5	6	7	8	9	10	11	12	13	14	15
daddi r1, r0, 200	MEM	WB										
ld r2, 0(r2)	EX	MEM	WB									
dmul r2, r3, r1	ID	ID	EX	EX	EX	MEM	WB					
daddi r1, r1, -1	IF	IF	ID	EX	MEM	WB						
dmul r2, r3, r2			IF	ID	ID	ID	ID	EX	EX	EX	MEM	WB
bnez r1, loop				IF	IF	IF	IF	ID	EX	MEM	WB	
dadd r9, r2, r3								IF	IF	IF		
ld r2, 0(r2)											IF	ID
dmul r2, r3, r1												IF

Problema 16.

Sea una microarquitectura MIPS64 que, además de la unidad de ejecución de propósito general, dispone de una unidad de ejecución de instrucciones de multiplicación/división de enteros no segmentada, con una latencia de 4 ciclos de reloj. Además, se permite la ejecución y terminación de instrucciones fuera de orden. Para el programa siguiente:

```
1 dmul r2, r5, r6
2 xor  r1, r10, r3
3 dmul r4, r6, r8
4 ld   r3, 100(r2)
```

❑ **16.1** Indica las detenciones que se producen en el programa y la duración de las mismas.

```
Estructural de 2 ciclos de duración
Datos RAW de 1 ciclo de duración

                  1   2   3   4      5      6   7    8   9   10   11   12
 dmul r2, r5, r6  IF  ID  EX  EX     EX     EX  MEM  WB
 xor r1, r10, r3      IF  ID  EX     MEM    WB
 dmul r4, r6, r8          IF  IDstr  IDstr  ID  EX   EX  EX  EX   MEM  WB
 ld r3, 100(r2)               IF     IF     IF  ID   ID  EX  MEM  WB
```

❑ **16.2** ¿Cuántos ciclos de reloj requiere para su ejecución el programa anterior? ¿Cuál sería el CPI descontando el transitorio inicial de 4 ciclos?

12 ciclos; CPI $= (12-4)/4 = 2$

❑ **16.3** Si fuese necesario dar soporte a excepciones precisas, ¿cuál sería el tiempo de ejecución del programa?

Las instrucciones xor y ld no pueden acabar antes que las instrucciones dmul que las preceden. El tiempo de ejecución sería de 13 ciclos.

	1	2	3	4	5	6	7	8	9	10	11	12	13
dmul r2, r5, r6	IF	ID	EX	EX	EX	EX	MEM	WB					
xor r1, r10, r3		IF	ID	EX	EX	EX	EX	MEM	WB				
dmul r4, r6, r8			IF	IDstr	IDstr	ID	EX	EX	EX	EX	MEM	WB	
ld r3, 100(r2)				IF	IF	IF	ID	ID	EX	EX	EX	MEM	WB

❑ **16.4** Supóngase que la unidad de multiplicación/división está ahora segmentada y se soportan excepciones precisas, ¿qué ocurre con la detención estructural? ¿Cuánto tiempo tarda en ejecutarse el programa? ¿Cuál será la aceleración respecto al apartado anterior ignorando el transitorio inicial?

La detención estructural desaparece. El programa tarda ahora 11 ciclos de reloj. La aceleración será $(13-4)/(11-4) = 1.29$

	1	2	3	4	5	6	7	8	9	10	11
dmul r2, r5, r6	IF	ID	EX1	EX2	EX3	EX4	MEM	WB			
xor r1, r10, r3		IF	ID	EX	EX	EX	EX	MEM	WB		
dmul r4, r6, r8			IF	ID	EX1	EX2	EX3	EX4	MEM	WB	
ld r3, 100(r2)				IF	ID	ID	ID	ID	EX	MEM	WB

Problema 17.

El programa siguiente se ejecuta con detenciones por dependencia de datos.

```
1 dadd r9, r5, r7
2 xor  r4, r6, r7
3 ori  r9, r6, 1
4 ld   r3, 100(r4)
```

❑ **17.1** Indica la instrucción que sufre la detención, así como la duración y el tipo de esta.

ld r3, 100(r4). La detención dura 1 ciclo y es de tipo RAW sobre el registro r4

❑ **17.2** ¿Cuántos ciclos de reloj requiere para su ejecución el programa anterior?

9

❑ **17.3** El compilador podría generar código cambiando el orden de las instrucciones anteriores de tal forma que se eliminase la detención. Indica un nuevo orden de las instrucciones que elimine la detención y no cambie la semántica del programa.

```
xor r4, r6, r7
dadd r9, r5, r7
ori r9, r6, 1
ld r3, 100(r4)
```

❑ **17.4** ¿Cuál será el nuevo tiempo de ejecución del programa?

El nuevo tiempo de ejecución será de 8 ciclos (4 del transitorio inicial más 4 del número de instrucciones más 0 del número de detenciones).

❑ **17.5** Para conseguir el mismo efecto, ¿podría moverse la instrucción dadd justo después de la ori? ¿Por qué?

No puede moverse. Entre las instrucciones dadd y la ori hay una dependencia de datos WAW, por lo que ese movimiento cambiaría la semántica del programa.

Problema 18.

Se ha diseñado una microarquitectura MIPS64 que implementa las rutas de reenvío `Salida EX`→`Entrada EX` y `Salida MEM`→`Entrada EX`, y se ejecuta en dicha microarquitectura este programa:

```
1 dadd r9, r5, r7
2 xor  r2, r6, r7
3 ld   r4, 100(r9)
4 dsub r2, r4, r9
```

❑ **18.1** Indica la evolución del cauce durante la ejecución del programa.

```
                  1   2   3   4    5    6    7   8    9
dadd r9, r5, r7   IF  ID  EX  MEM  WB
xor r2, r6, r7        IF  ID  EX   MEM  WB
ld r4, 100(r9)            IF  ID   EX   MEM  WB
dsub r2, r4, r9               IF   ID   EX   EX  MEM  WB
```

❑ **18.2** ¿Cuántos ciclos de reloj se han ahorrado en la ejecución con las rutas de reenvío?

La ruta de renvío Salida MEM→Entrada EX ahorra 1 ciclo de reloj, pues adelanta 1 ciclo la disponiblidad del operando con respecto a disponer de él en la etapa WB. Como hay dos reenvíos de este tipo, se ahorran 2 ciclos de reloj.

Problema 19.

Una microarquitectura MIPS64 emplea un reloj de 2 GHz, implementa una evaluación agresiva de saltos en la etapa ID y todas las rutas de reenvío necesarias. Sobre esta microarquitectura se ejecuta el siguiente fragmento de código, almacenado en memoria a partir de la dirección 0000 0000 0000 A200h:

```
1         ori   r1, r0, 2  ; r1 = 2
2 startf:
3         beqz  r1, endf   ; jump to endf if r1 zero
4         daddi r1, r1, -1 ; r1 = r1-1
5         j startf ; jump to startf
6 endf:
7         ld    r5, 100(r3)
```

❑ **19.1** Indicar el tiempo necesario para ejecutar el fragmento de programa anterior.

19 ciclos => 9.5 ns.
El programa anterior ejecuta 9 instrucciones: ori, beqz, daddi, j, beqz, daddi, j, beqz y ld. Por lo tanto, el número de ciclos coincide con el transitorio inicial más el número de instrucciones más el número de ciclos de detención. Solo hay detención por riesgos de datos: la primera vez que se ejecuta beqz necesita en la etapa ID el valor que la instrucción anterior (ori) está computando al mismo tiempo en su etapa EX. Las detenciones estructurales no son posibles, pues no hay instrucciones multiciclo. Cada instrucción de salto condicional o incondicional introduce 1 ciclo de detención, por lo que en total resultan (4+9+1+5) = 19 ciclos.

❑ **19.2** ¿Qué rutas de reenvío se activan y qué valores se transmiten a través de las mismas?

Se activa la ruta de reenvío Salida EX→Entrada ID y se transmite el valor 2.
Se activa durante la primera ejecución de la instrucción beqz por la dependencia que tiene en r1 con la instrucción ori.

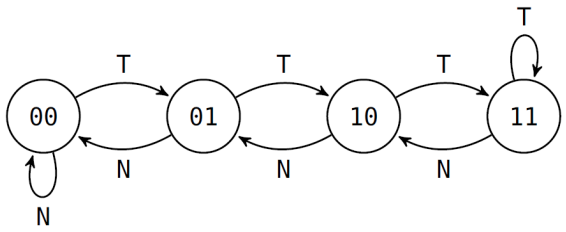

Figura 2.1: Máquina de estados del predictor de dos bits

❑ **19.3** Suponiendo que se realiza una predicción de saltos "siempre no tomado", ¿cuál sería el nuevo tiempo de ejecución?

```
17 ciclos => 8.5 ns.
Cuando se acierta en la predicción en un salto condicional no
se incurre en detención. Cuando se falla se incurre en una de-
tención de 1 ciclo. Los saltos incondicionales siempre producen
una detención de 1 ciclo. Las dos primeras ejecuciones de beqz
tienen acierto y la última fallo. Por lo tanto el número de
ciclos es (4+9+1+3) = 17 ciclos.
```

❑ **19.4** Suponiendo ahora que se realiza una predicción de saltos con historial de 2 bits, ¿cuál sería el nuevo tiempo de ejecución? ¿Cuál sería el estado final de la tabla de saltos (BTB)? Debe suponerse la máquina de estados de la figura 2.1 y que la predicción de las instrucciones de salto condicional arranca en el estado 01 (*weak not taken*).

```
16 ciclos => 8 ns.
A204h A210h 01 (instrucción beqz)
A20Ch A204h 11 (instrucción j)


La primera vez que se ejecuta la instrucción de salto incon-
dicional se produce una detención de 1 ciclo en la etapa IF,
pues hay que esperar a que se calcule la dirección destino del
salto en la etapa ID y se almacene en el BTB. En las ejecucio-
nes restantes ya no se produce detención en la etapa IF, pues
la dirección destino del salto ya está disponible en la etapa
IF al leerse del BTB. En cuanto a la instrucción de salto con-
dicional, la primera vez se acierta pues el estado de partida
es 01 (weak not taken) en la etapa IF. No obstante, se produce
detención de datos sobre r1 como se vio anteriormente. Se cal-
cula la dirección destino del salto en la etapa ID y se guarda
en el BTB. La segunda vez que se ejecuta el salto condicional
también se acierta, pues el estado es 00 en IF y no hay deten-
ción. La tercera vez se falla pues el salto se toma, por lo que
en la etapa ID el estado pasa a tomar el valor 01. En total hay
1 detención del salto incondicional, 1 por riesgo de datos y 1
del salto condicional. Resulta un número de ciclos (4+9+3) = 16
ciclos.
```

Problema 20.

Se dispone de una CPU MIPS64 con estas características: sin rutas de reenvío, predicción de saltos siempre no tomado, evaluación agresiva de saltos (en la etapa ID), unidad de multiplicación segmentada de 3 ciclos, ejecución de instrucciones fuera de orden (cuando se emplean unidades de ejecución diferentes) y terminación de instrucciones fuera de orden. Sobre esta CPU se ejecuta el siguiente fragmento de programa:

```
1         ori    r1, r0, 8 ; r1 = 8
2         daddi  r2, r0, 2 ; r2 = 2
3  loop:
4         beqz   r1, dest
5         dmul   r3, r2, r7
6         xor    r6, r4, r5
7         dadd   r6, r7, r6
8         dsub   r1, r1, r2
9         j      loop
10 dest:
11        dmul   r5, r4, r2
```

❑ **20.1** Indica el tiempo de ejecución en ciclos de este código en una versión no segmentada de la CPU descrita.

$10 + 4 \times (5 \times 6 + 2) + 5 + 7 = 150$ `ciclos`

❑ **20.2** Indica la primera vez que aparece cada una de las dependencias de datos RAW, WAW y WAR identificando las instrucciones involucradas y el registro que crea la dependencia. Ejemplo de respuesta, `RAW: dsub y dadd, r4`.

```
WAW: xor y dadd, r6
RAW: ori y beqz, r1
WAR: beqz y dsub, r1
```

❑ **20.3** Identifica las tres primeras detenciones que se producen en la ejecución de este código. Identifica el tipo de detención (**(D)**atos, **(E)**structural o **(C)**ontrol), la instrucción que la sufre y el número de ciclos que dura. Ejemplo de respuesta: `(D) - dsub - 3 ciclos`.

```
(D) - beqz - 1 ciclo
(D) - dadd - 2 ciclos
(C) - j / dmul - 1 ciclo
```

Se mejora la microarquitectura de la CPU anterior implementando todas las rutas de reenvío posibles.

❑ **20.4** ¿Qué rutas de reenvío se activarán durante la ejecución del código anterior? Ejemplo de respuesta: `Salida MEM dmul` $\rightarrow$ `Entrada EX dsub`.

```
Salida EX ori → Entrada ID beqz
Salida MEM daddi → Entrada EX dmul
Salida EX xor → Entrada EX dadd
```

❑ **20.5** Rellena una tabla con la evolución del pipeline desde el ciclo 1 al ciclo 13 con las rutas de reenvío activadas.

Instr. \ Etapa	1	2	3	4	5	6	7	8	9	10	11	12	13
ori r1, r0, 8	IF	ID	EX	MEM	WB								
daddi r2, r0, 2		IF	ID	EX	MEM	WB							
beqz r1, dest			IF	ID	EX	MEM	WB						
dmul r3, r2, r7				IF	ID	EX	EX	EX	MEM	WB			
xor r6, r4, r5					IF	ID	EX	MEM	WB				
dadd r6, r7, r6						IF	ID	EXstr	EX	MEM	WB		
dsub r1, r1, r2							IF	ID	ID	EX	MEM	WB	
j loop								IF	IF	ID	EX	MEM	WB
dmul r5, r4, r2										IF			
beqz r1, dest											IF	ID	EX

Problema 21.

Se ejecuta el siguiente fragmento de código en una CPU MIPS64 que dispone de una unidad de multiplicación y división de enteros que requiere 4 ciclos de reloj para completarse. La microarquitectura segmentada incluye todas las rutas de reenvío necesarias y no implementa excepciones precisas.

```
1 daddi r3, r0, 5
2 ddiv  r2, r4, r5
3 xor   r3, r6, r8
4 dsub  r2, r9, r10
```

❑ **21.1** Indicar el cronograma de ejecución.

```
                  1   2   3   4    5   6    7   8    9    10
daddi r3, r0, 5   IF  ID  EX  MEM  WB
ddiv r2, r4, r5       IF  ID  EX   EX  EX   EX  MEM  WB
xor r3, r6, r8            IF  ID   EX  MEM  WB
dsub r2, r9, r10              IF   ID  EX   EX  EX   MEM  WB
```

❑ **21.2** Supóngase que se emplea renombrado de registros, para lo cual se dispone de 64 registros físicos de enteros denominados rr0 a rr63. Para el renombrado se dispone de una cola FIFO que originalmente contiene los registros rr32 a rr63 y a la que se van añadiendo los registros disponibles. ¿Cuál será la aceleración respecto al caso anterior ignorando el transitorio inicial? ¿Qué registros arquitectónicos cambian de registro físico al final de la ejecución del programa anterior y qué registros físicos tienen asignados?

```
                  1   2   3   4    5   6    7    8    9
daddi r3, r0, 5   IF  ID  EX  MEM  WB
ddiv r2, r4, r5       IF  ID  EX   EX  EX   EX   MEM  WB
xor r3, r6, r8            IF  ID   EX  MEM  WB
dsub r2, r9, r10              IF   ID  EX   MEM  WB
A = (10-4)/(9-4)= 1.2
Cambian los registros r3 y r2 que pasan a estar asignados a los
registros físicos rr34 y rr35, respectivamente.
```

Problema 22.

Se dispone de una CPU MIPS64 superescalar con ancho de emisión 3 y renombrado de registros, la cual dispone de las siguientes unidades de ejecución:

- 1 Unidad de carga/almacenamiento (2 ciclos).
- 3 Unidades de enteros de propósito general (1 ciclo).
- 1 Unidad de multiplicación y división de enteros (4 ciclos).
- 1 Unidad de saltos (1 ciclo).
- 1 Unidad de coma flotante (6 ciclos).

❑ **22.1** ¿Cuál es el mínimo CPI que puede obtenerse con esta CPU?

```
1/3 = 0.33
```

La CPU anterior ejecuta el siguiente fragmento de programa:

```
1 ld   r2, 120(r1)
2 dadd r2, r2, r5
3 beqz r2, etiqueta
```

Se supondrá que las tres instrucciones se distribuyen en el mismo ciclo de reloj.

❑ **22.2** ¿Cuál es el tiempo mínimo que debe esperar la instrucción dadd antes de ser emitida a su unidad de ejecución? ¿Dónde se almacena mientras espera?

```
Debe esperar a que la instrucción ld salga de su unidad de eje-
cución y reenvíe el valor de r2 => 2 ciclos de reloj. Mientras
tanto, se almacena en la estación de reserva de enteros de pro-
pósito general.
```

Para evitar la detención anterior se modifica el programa, sustituyendo la instrucción `dadd r2, r2, r5`, por la instrucción `dadd r2, r1, r5`.

❑ **22.3** ¿Cuánto tiempo debería esperar la instrucción dadd antes de ser emitida a su unidad de ejecución?

```
0 ciclos, pues desaparece la dependencia RAW respecto a ld. La
dependencia WAW con el registro destino r2 desaparece al ser
renombrado.
```

❑ **22.4** ¿Qué instrucción publicará antes en el bus de reenvío el valor del registro destino calculado, la instrucción ld, o la instrucción dadd? ¿Por qué?

```
La instrucción dadd, pues es emitida a su unidad de ejecución
a la vez que la instrucción ld, pero su unidad de ejecución
requiere un ciclo en lugar de los dos que requiere ld.
```

❑ **22.5** La instrucción beqz espera por el valor r2 en la estación de reserva de saltos. ¿Qué instrucción le proporcionará el valor de dicho registro arquitectónico a través del bus de reenvío? ¿Por qué?

La última instrucción que modifica r2, es decir, dadd r2, r1, r5. Al pasar la instrucción dadd por la etapa ID se produce el renombrado del registro r2 al registro rr33, por lo que el registro arquitectónico r2 pasa entonces a estar asociado a rr33. La instrucción siguiente beqz r2, etiqueta es equivalente entonces a beqz rr33, etiqueta, por lo que la instrucción daddd es la que proporciona el valor del registro arquitectónico r2. El valor que publica la instrucción ld r2, 120(r1) es rr32 que ya no está asociado al registro arquitectónico r2.

❑ **22.6** En cualquier caso, ¿qué instrucción es retirada antes, la instrucción ld o la instrucción dadd? ¿Por qué?

La instrucción ld, pues la retirada (o terminación) de las instrucciones se realiza en orden.

Problema 23.

Las CPU modernas son muchas de ellas superescalares, por lo que en condiciones ideales ejecutan tantas intrucciones por ciclo de reloj como su ancho de emisión. A medida que se aumenta el ancho de emisión aumenta rápidamente el número de transistores requerido para la implementación de la CPU. Actualmente las CPU superescalares tienen un ancho de emisión igual o inferior a 4, aunque la tecnología actual permitiría construir CPU superescalares con anchos de emisión mayores.

❑ **23.1** ¿Cuál crees que es la razón de no construir CPU superescalares con un ancho de emisión mayor?

Porque los programas tienen en la práctica un ILP limitado y el hecho de implementar microarquitecturas superescalares de mayor ancho de emisión no supodría una mejora de rendimiento.

❑ **23.2** ¿Por qué razón la mayor parte de las CPU actuales tienen frecuencias de reloj inferiores a los 4 GHz?

Con la tecnología disponible para la fabricación de circuitos integrados, pequeños aumentos de frecuencia producen grandes aumentos de energía consumida. Frecuencias mayores producen tanto calor que su extracción no es viable económicamente.

❑ **23.3** ¿Qué ocurre con el coste y la disipación de energía cuando se implementa la misma microarquitectura sobre una tecnología de fabricación más avanzada?

```
Se reduce la energía disipada y el coste. La energía disipada
aumenta con la superficie y la frecuencia. Al usar una tecno-
logía más moderna, la superficie ocupada es más pequeña, lo que
reduce el coste (caben más chips en la oblea). Como la frecuen-
cia es la misma resulta que la potencia disipada es menor.
```

❑ **23.4** ¿A qué dedican los procesadores actuales la gran cantidad de transistores extra disponibles en comparación a los primeros procesadores?

```
Implementan microarquitecturas superescalares con muchas uni-
dades replicadas, extensiones de la arquitectura en forma de
conjuntos de instrucciones avanzadas SIMD, múltiples núcleos
dentro del chip del procesador, memorias cachés de varios nive-
les, así como otras unidades funcionales del computador.
```

Problema 24.

Indica cuál o cuáles de las siguientes afirmaciones son CIERTAS. Contesta NINGUNA si crees que ninguna lo es.

A) La programación multihilo tiene poca utilidad en sistemas con un único núcleo sin capacidades de multihilo simultáneo.

B) La tecnología de multihilo simultáneo es equivalente a disponer de varios núcleos.

C) Las extensiones SIMD permiten mejorar el rendimiento de programas multihilo, pero no de programas con un solo hilo de ejecución.

D) El disponer de varios núcleos en un procesador mejora la productividad, pero no el tiempo de respuesta de los programas que se ejecutan cuando estos son monohilo.

E) El muro de memoria impide que el número de núcleos de un procesador crezca por encima de un cierto límite.

```
D y E
```

Problema 25.

Indica cuál o cuáles de las siguientes afirmaciones son CIERTAS. Contesta NINGUNA si crees que ninguna lo es.

A) Según la clasificación establecida por la taxonomía de *Flynn*, un monoprocesador ejecutando un S. O. monotarea se corresponde con una arquitectura de tipo SIMD.

B) La GPU de los ordenadores personales modernos son un ejemplo de arquitectura SIMD, según la clasificación establecida por la taxonomía de *Flynn*.

C) Según la taxonomía de *Flynn*, la arquitectura MIMD es la generalización de la arquitectura *von Neumann*, representada en los equipos actuales con varios núcleos o varios procesadores.

D) Una aplicación con varios hilos tendrá un mayor rendimiento si sus hilos se pueden ejecutar en paralelo.

E) Para un programa monohilo es más interesante disponer de un *pipeline* (o paralelismo a nivel de instrucción) de gran rendimiento antes que disponer de paralelismo a nivel de hilo.

F) La tecnología *Simultaneous Multithreading* consiste en que sin hacer ningún tipo de replicación un núcleo pueda trabajar sobre dos hilos simultáneamente.

```
B, C, D y E
```

Problema 26.

Indica cuál o cuáles de las siguientes afirmaciones son CIERTAS. Contesta NINGUNA si crees que ninguna lo es.

A) La técnica de segmentación consiste en dividir la ejecución de instrucciones en etapas y estas etapas trabajan secuencialmente para obtener un incremento de rendimiento.

B) Cuando se realiza una llamada al sistema operativo, una interrupción o una excepción, se ejecuta un manejador ubicado en el espacio de direcciones de la tarea.

C) Las interrupciones no enmascarables se atienden siempre.

D) Las CPU que gracias a la replicación de componentes son capaces de trabajar en cada etapa de segmentación sobre varias instrucciones simultáneamente reciben el nombre de *superescalares*.

E) La CPU con nivel de privilegio de usuario puede acceder a las direcciones de memoria asociadas al sistema operativo.

```
C y D
```

Problema 27.

El empleo de VirtualBox como hipervisor permite la ejecución de una máquina virtual Linux en los equipos de prácticas. ¿Qué tipo de hipervisor es VirtualBox, tipo 1 o tipo 2? ¿Por qué?

```
Tipo 2, pues se ejecuta sobre un sistema operativo anfitrión
(Windows) y no directamente sobre el hardware.
```