PROBLEMA 1

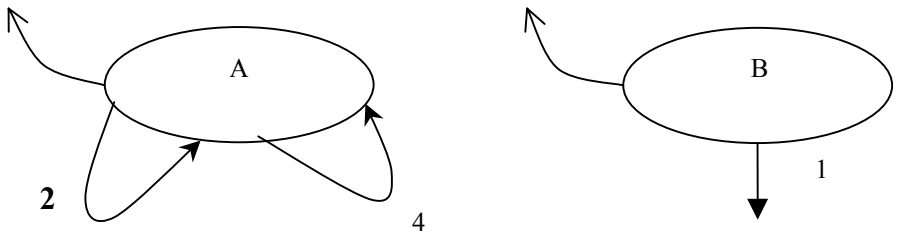
A	**X**								**X**
B		**X**	**X**					**X**	
C				**X**					
D				**X**	**X**				
E						**X**	**X**		
F	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**

La figura muestra una tabla de reserva de un procesador segmentado. Contestad razonadamente a las siguientes cuestiones:

a) Obtener la lista de latencias prohibidas.
b) Deducir el límite inferior del MAL (valor de la latencia media alcanzable).
c) Calcular el vector de colisiones inicial.
d) Obtener y representar el diagrama reducido de estados (grafo simplificado).
e) Listar todos los lazos ansiosos (ciclos "greedy").
f) Calcular el valor del MAL indicando para qué secuencia de iniciaciones se consigue.
g) Calcular la utilización de cada etapa para la secuencia óptima de indicaciones.
h) Diseñar un controlador de iniciaciones
   - Suponiendo datos siempre disponibles y
   - Suponiendo datos no siempre disponibles.
i) ¿Es posible mejorar la utilización de las etapas? ¿hasta qué límite? ¿por qué?
j) Proponer una sencilla inserción de retardos para que la nueva tabla de reserva tenga un MAL igual al valor reducido en el apartado (b).

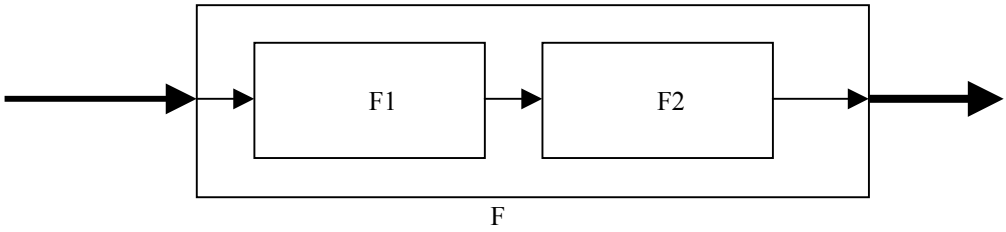
PROBLEMA 2

En la figura se muestra parte del grafo de estados de una unidad funcional segmentada. Los estados A y B representados en la figura no tienen más salidas que las dibujadas. Determinar el vector de colisión inicial indicando claramente los pasos seguidos para obtenerlo.



PROBLEMA 3

Se dispone de 2 unidades funcionales segmentadas F1 y F2 con vectores de colisión inicial VCI1 = 110011 y VCI2 = 1000011 respectivamente.
Se desea diseñar una unidad funcional segmentada F conectando directamente la salida de F1 con la entrada de F2.

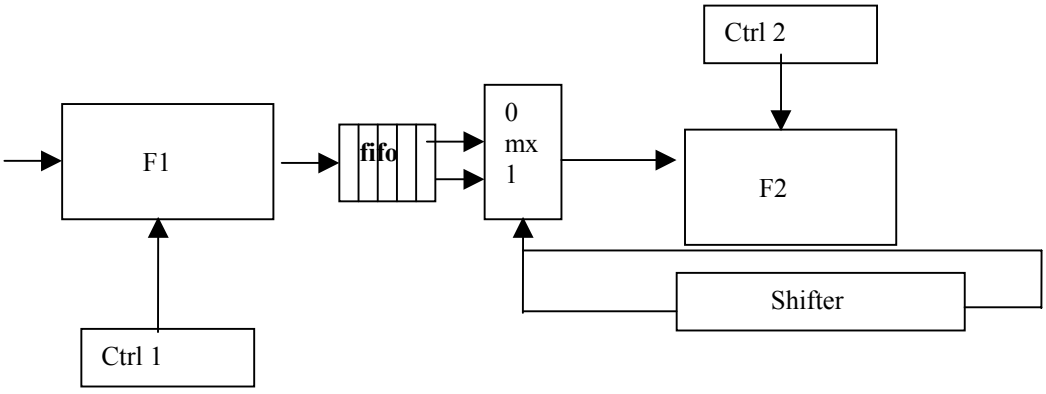


a. Se pide:

a) ¿Cuál es la secuencia de latencias de iniciación permitidas en la entrada de F para conseguir la máxima velocidad de iniciaciones media?
b) ¿Cuál es el MAL de F?
c) ¿Se podría reducir el MAL de F introduciendo una secuencia de registros entre la salida de F1 y la entrada de F2?. Justificar la respuesta.
(Nota: Una secuencia de registros es una secuencia lineal de etapas que no modifican los datos, solo los retardan).
d) Suponiendo que siempre hay datos disponibles a la entrada de F, diseñar una unidad de control de entradas para F que consiga máxima velocidad media de iniciaciones.

b. Suponer que se han diseñado los controladores de las entradas de F1 y F2 considerando que siempre hay datos disponibles en sus entradas. Estos controladores están diseñados de forma que obtienen la menor latencia media de entrada para cada una de la U.F. segmentadas.

Las unidades de F1 y F2 junto con sus controladores se encadenan mediante un sistema como el que se muestra en la figura para poder asi implementar la función F.



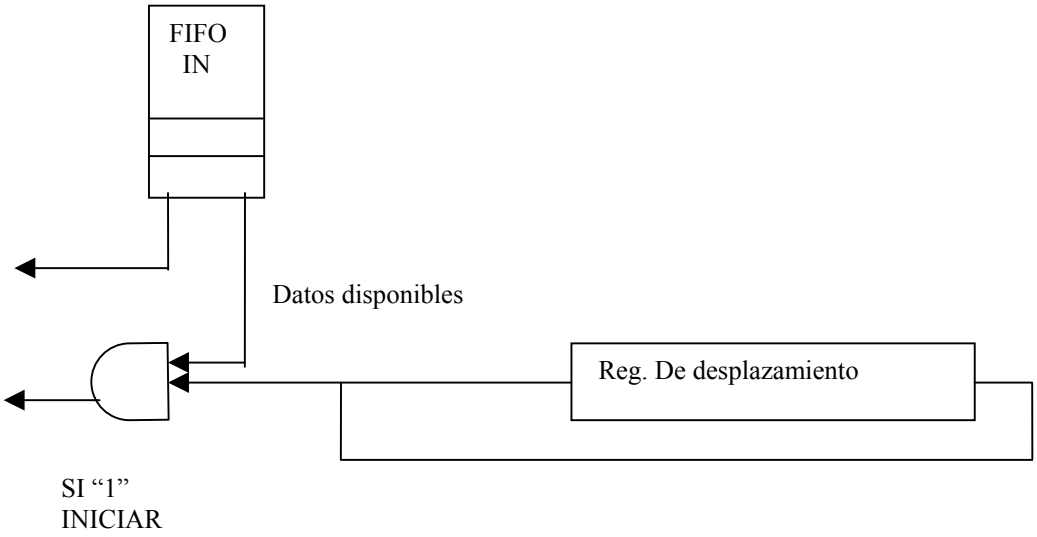
Se pide:
a) ¿Tamaño de la memoria FIFO?
b) ¿Cuál es el número de bits y contenido inicial del registro de desplazamiento?
c) Completar el diseño para que una señal indique en que ciclos los datos de salida de F2 son válidos.

PROBLEMA 4

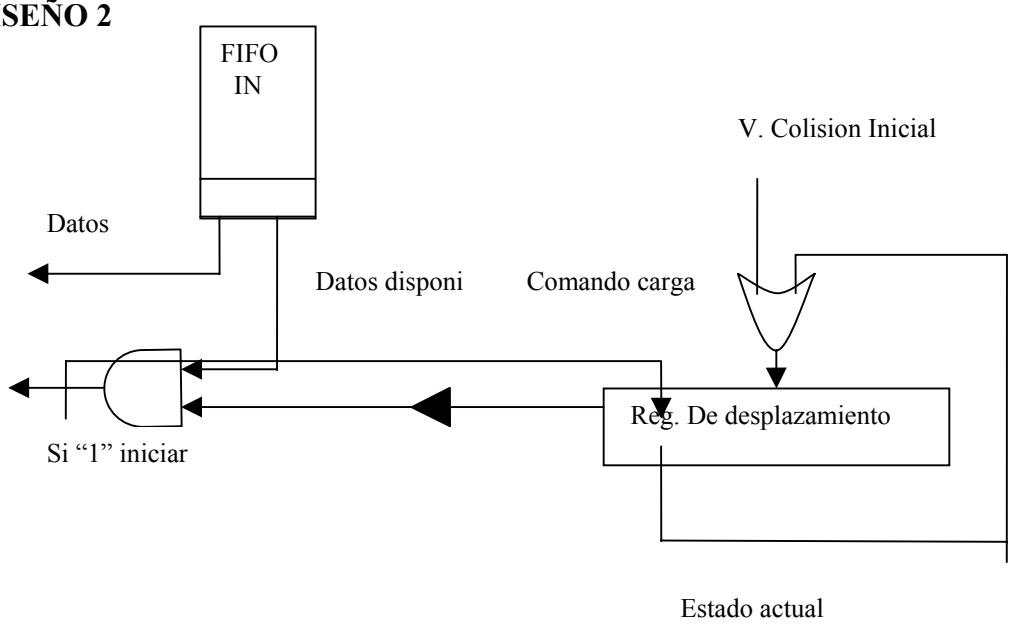
Para efectuar el control externo de entradas de datos a un pipeline, suponiendo que no siempre hay datos disponibles para iniciar una operación, se puede utilizar alguno de los tres diseños de la figura 1.

a. Responder justificadamente a las siguientes preguntas relacionadas con la utilización de los diseños de la figura 1 para conseguir máxima eficiencia cuando el vector de colisión inicial vale VCI = 110011.
   a) ¿Cuál debe ser el tamaño y el contenido inicial del registro de desplazamiento que aparece en el diseño 1?
   b) ¿Cuántos estados distintos (valores del registro de desplazamiento) puede tener el pipeline si se controla su entrada con el diseño 2?
   c) ¿Cuántos estados distintos (valores del registro de desplazamiento) puede tener el pipeline si se controla su entrada con el diseño 3?
   d) ¿Para qué valores de entrada a la tabla del diseño 3 (estados del pipeline) la salida de la tabla vale 1?
b. La utilización eficiente de un diseño u otro se decidirá en función de varios parámetros, entre los cuales destacamos: el coste del hardware, el grafo de estados y la distribución de llegada de datos a la cola de entrada.
   a) ¿Bajo qué condiciones sería justificado elegir el diseño 1?
   b) ¿Bajo qué condiciones sería justificado elegir el diseño 3 frente al 2?

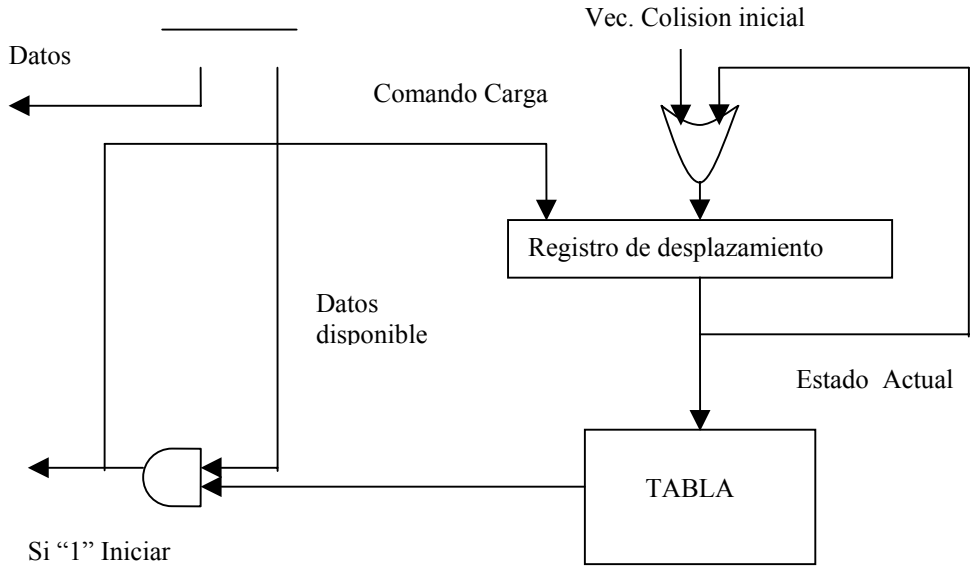
DISEÑO 1



DISEÑO 2

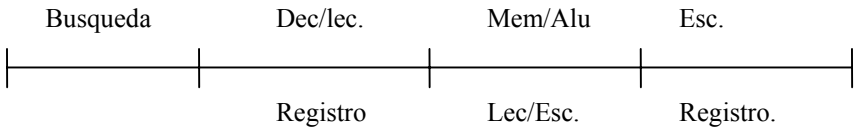


DISEÑO 3



PROBLEMA 5

Suponer un procesador segmentado con la siguiente estructura de segmentos:



Los tipos de instrucciones que se ejecutan en este procesador son los siguientes:

oper	Ri, Rk, Rj	Ri < -- Rk + Rj
load	Ri, Rk, Rj	Ri < -- Mem (Rk) Rk_{nuevo} < -- Rk_{viejo} + Rj
store	Ri, Rk, Rj	Mem(Ri) < -- Rj Ri_{nuevo} < -- Ri_{viejo} + Rk
br	R0, desp1, desp2, cond	salta a la instrucción cuya dirección esté en $R0_{viejo}$ si se cumple la f condición que ha establecido una instrucción anterior, sino sigue en secuencia. En cualquier caso: $R0_{nuevo}$ < -- PC + desp1 R1 < -- $R0_{nuevo}$ + desp2

Pregunta:

a) Indicar cuantos recursos son necesarios y en que etapas se utilizan para que en cualquier ciclo se pueda estar ejecutando cualquier combinación de instrucciones.
Los registros R0, R1 y PC no pertenecen al banco de registros generales.
Recursos a considerar: caminos de acceso a memoria
caminos de acceso al banco de registros (lectura y escritura)
caminos de acceso a los registros R0, R1, y PC (lectura y escritura)
ALU y sumadores
NOTA: un recurso se ocupa durante todo un ciclo.

PROBLEMA 6

El lenguaje máquina de una determinada arquitectura segmentada dispone de las siguientes instrucciones:

OP Rf1 Rf2 Rd (Op. arit/log con operandos en registros)

OP	Rf1	MEM	Rd	(Op. arit/log con un operando en memoria)
LOAD	Rd	MEM		
STORE	Rf	MEM		
JMP	offset			(salto incondicional)
JMPC	Rf1	Rf2	offset	(comparación y salto R-R)
JMPI	Rf1	MEM	offset	(comparación y salto R-M)

Rf1, Rf2 y Rd son registros, MEM se compone de un registro base

*falta parte de la fotocopia.

PROBLEMA 7

Queremos diseñar un procesador con el siguiente formato de instrucciones.

Código de Operación	R1	R2	R3

donde S3 es un registro o una constante.
Los tipos de instrucciones que soporta son los siguientes:
a) Aritméticas R1 = R2 op S3
b) Acceso a memoria (load, store)
- load: R1 = Mem (R2 + S3)
- store: Mem (R2 + S3) = R1; S3 es un constante
c) Saltos condicionales
- Si cond (R1 - R2) entonces PC = PC + S3; S3 es una constante

La arquitectura tiene los siguientes recursos:
a) Un conjunto de registros (tiempo de acceso 25 ns).
b) Una ALU (tiempo de cálculo 35 ns).
c) Para acceder a la memoria debe de enviarse en primer lugar la dirección (15 ns) y posteriormente se efectua el acceso (50 ns).
d) La búsqueda de instrucciones se efectua en un tiempo de 50 ns (se incluye el tiempo de enviar la dirección a memoria).
e) Existe un sumador que permite efectuar las operaciones asociadas con el PC (incremento y suma con S3) en un tiempo de 10 ns.
f) La evaluación de la condición se efectua en 5 ns.

Preguntas:
a. Proponer un diseño segmentado del procesador de forma que se ejecuten el mayor número de instrucciones de forma concurrente.
   a) Indicar las operaciones que deben de efectuarse en cada ciclo.
   b) ¿Cuál es el tiempo de ciclo?

(Nota: algún recurso puede utilizarse dos veces en un ciclo por la misma o distintas etapas si su tiempo de acceso o de cálculo lo permite)
b. ¿Cuál es el número de caminos necesarios para acceder a memoria sin que existan conflictos (riesgos estructural) entre dos instrucciones cualesquiera?
c. ¿Cuál es el menor número de buses necesarios para acceder al conjunto de registros?
d. ¿Cuántos cortocircuitos se han de construir en el camino de datos entre las unidades funcionales (ALU, memoria) para resolver las dependencias de datos (riesgos de datos)?
e. ¿Qué riesgos de datos no pueden resolverse mediante los cortocircuitos?
f. ¿Cuál es el número de ciclos (retardo) que tarda en ser efectivo un salto condicional?

PROBLEMA 8

En la interpretación de una instrucción se pueden distinguir las siguientes suboperaciones:
a) búsqueda de la instrucción (50 ns).
b) decodificación y lectura de operandos (50 ns).
c) escritura de registros (50 ns).
d) acceso a memoria (100 ns).
e) unidad aritmético lógica (50 ns).

En paréntesis se ha especificado el tiempo que tarda cada suboperación.
Partiendo de ellas se ha diseñado un procesador segmentado con cinco etapas
a) B: búsqueda de la instrucción (1 ciclo).
b) D/L: decodificación y lectura de operandos (1 ciclo).
c) E: escritura de registros (1 ciclo).
d) M: acceso a memoria (2 ciclos).
e) A: unidad aritmético lógica (1 ciclo).

En paréntesis se especifican los ciclos de cada etapa.
La funcionalidad de las etapas es la misma que la del procesador lineal explicado en clase.
En este procesador existen tres tipos diferentes de instrucciones en función de como se utilizan las etapas. En la figura se indican las tablas de reserva para los distintos tipos de instrucciones.

B	X			
D/L		X		
A			X	
M				
E				X
	1	2	3	4

B	X					
D/L		X				
A			X			
M				X	X	
E						X
	1	2	3	4	5	6

B	X				
D/L		X			
A			X		
M				X	X
E					
	1	2	3	4	5

Nótese que las instrucciones tipo load y store utilizan durante dos ciclos consecutivos la etapa M. Esto es, la memoria tiene un tiempo de acceso de 2 ciclos y no está segmentada.
La unidad de control detecta riesgos estructurales y riesgos de dependencias de datos y tiene capacidad de bloquear el procesador hasta que los riesgos desaparecen. Además, puede indicarle a la etapa D/L que efectue la lectura de operandos tanto del banco de registros como de los cortocircuitos.
Hay dos buses para leer los registros y un bus para escribir los registros. La escritura se efectúa durante la primera mitad del ciclo y las lecturas se efectúan en la segunda mitad del ciclo.
Cualquier operación debe tener todos los datos que necesita al finalizar la etapa de D/L.
Se pregunta:

a) Dibujar un esquema simplificado (registros, ALU, M, no es necesaria la etapa de búsqueda) del camino de datos de este procesador identificando claramente las etapas.
b) Indicar en que situaciones se debe bloquear el procesador y durante cuantos ciclos para eliminar los riesgos estructurales.
c) Indicar en que situaciones, en las que aparecen riesgos debidos a dependencias de datos, se debe bloquear el procesador. Determinar el número de ciclos de bloqueo.
d) Indicar el número mínimo de cortocircuitos necesarios para hacer desaparecer los riesgos debidos a dependencias de datos cuando sea posible. Justificar la respuesta indicando que tipo de combinaciones de instrucciones los utilizan.
e) Dibujar otra vez el esquema de la pregunta a) e indicar de forma clara los cortocircuitos.
f) Modificar la ocupación de las etapas de los tres tipos de instrucciones para conseguir:
   - una latencia de iniciación de instrucciones mínima y constante.
   - que no se produzcan riesgos estructurales.

Indicar el valor de la latencia. Dibujar las tablas de reserva que son distintas al enunciado indicando que se hace en cada etapa e indicar que riesgos estructurales y riesgos debidos a dependencias de datos se han eliminado.

PROBLEMA 9

Sea un procesador segmentado lineal con las siguientes instrucciones:

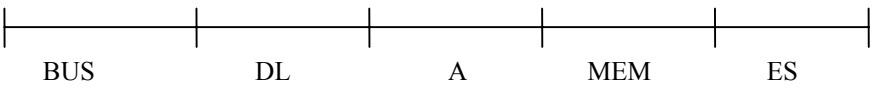
aritmético lógicas:	Ri = Rj op Rk
acceso a memoria	load Ri, M (X (Rj))
	store M (X (Rj)), Ri
saltos	if cond (Ri) then goto X + PC

Dos posibles diseños segmentados para este procesador (PA y PB) son los siguientes

PA

BUS: búsqueda de la instrucción e incremento del PC
EJ: decodificación delas instrucciones
lectura de registros
operación en UAL
cálculo de la dirección efectiva
resolución de los saltos
MEM: acceso a memoria
ES: escribir en registros

PB



Se ha separado EJ en dos etapas:

DL: decodificación y lectura de registros
A: el resto de EJ

Contestar para cada uno de los dos diseños PA y PB:

a) Suponer que no hay implementado ningún cortocircuito. Hay dependencias de datos que pueden causar pérdidas de ciclos por bloqueo del procesador. Mostrar, para cada una de esas dependencias, un ejemplo con dos instrucciones consecutivas e indicar el número de ciclos que dura el bloqueo en cada caso.
b) Contestar a las mismas cuestiones del apartado 1 suponiendo ahora que están implementados todos los cortocircuitos posibles.
c) Suponiendo que el procesador usa la técnica del salto retardado, justificar cuál es la latencia del salto.
d) Suponiendo, por el contrario, que el procesador usa mecanismos hardware de apuesta por salto tomado (efectivo) con lectura de dirección efectiva de salto en tabla BTB. Evaluar justificadamente el coste en número de ciclos perdidos en los 2 casos posibles (acierto o fallo en la apuesta).

PROBLEMA 10

Tenemos un procesador segmentado en el que instrucciones de salto son de tipo salto retardado (delayed branch) con latencia 2. Considerar el siguiente código máquina con 8 instrucciones, en el que todas las Si son sentencias de asignación:

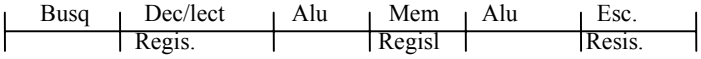
```
      S1
etil: S2
      S3
      S4
      IF condición THEN GO TO etil
      NOP
      S7
      S8
```

Se desea optimizar el tiempo de ejecución de este programa en este procesador, para lo cual se propone reestructurar el código.
Responder justificadamente a las siguientes preguntas:

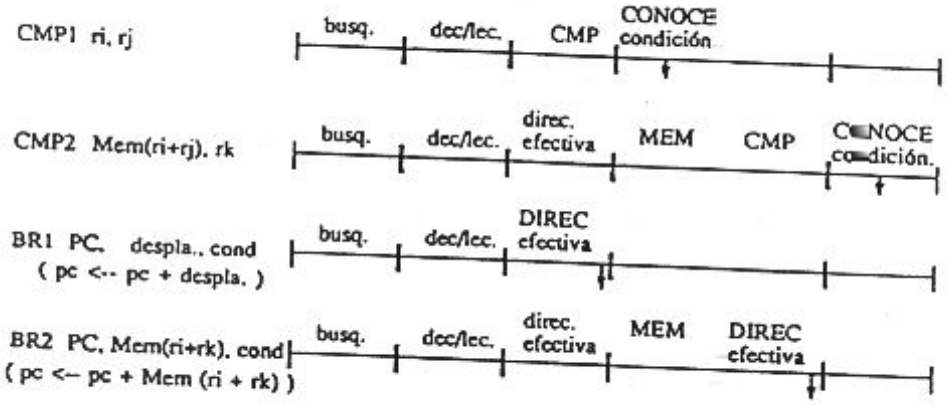
a) ¿En qué casos (condiciones en las dependencias entre instrucciones) es posible sustituir la sentencia NOP por la S3?
b) Escribir el código reestructurado resultante en cada uno de los casos anteriores.

PROBLEMA 11

Un procesador segmentado tiene la siguiente estructura de segmentos:



Se han implementado dos instrucciones de comparación y dos instrucciones de salto que funcionan de la siguiente forma:



Las instrucciones de salto originalmente implementan la técnica de salto retardado. Ahora bien, para aumentar el rendimiento se quiere comprobar si es posible implementar de forma sencilla la técnica de salto retardado con anulación. La anulación afecta a las instrucciones que se ejecutan entre la instrucción de salto y la instrucción a la que se bifurca. La anulación de una instrucción consiste en impedir que esa instrucción modifique el estado.
Pregunta:
Es posible implementar la técnica con el par de instrucciones?

a) CMP1 y BR1.
b) CMP2 y BR2.
Razonar las respuestas.

PROBLEMA 12

Tenemos un procesador segmentado, con un juego de instrucciones del cual solo nos interesan aqui las de cálculo, que son de la forma Rd <--- Rf1 op Rf2, donde op solo puede ser SUM, MPY o DIV.
Sabemos, de momento, que este procesador:
a) No puede hacer ni fetch ni decode de más de una instrucción por ciclo.
b) Tiene una unidad funcional para SUM, con tiempo de cálculo igual a 1 ciclo.
c) Tiene una unidad funcional para MPY, con tiempo de cálculo igual a 2 ciclos.
d) Tiene una unidad funcional para DIV, con tiempo de cálculo igual a 4 ciclos.
e) Ninguna de estas unidades funcionales está segmentada.
Queremos estudiar la ejecución en este procesador de un fragmento de código con la forma general de la figura 1 y con un diagrama ideal de ejecución como el de la figura 2 que complementaría los cálculos en 13 ciclos.
Hay 4 fases en cada instrucción:
F: Fetch (1 ciclo)
D/L: Decode/Lee registros (1 ciclo)
E: Execute (1, 2 o 4 ciclos)
W: Write resultado en registro (1 ciclo)
Cada fase ocupa totalmente el ciclo o ciclos.

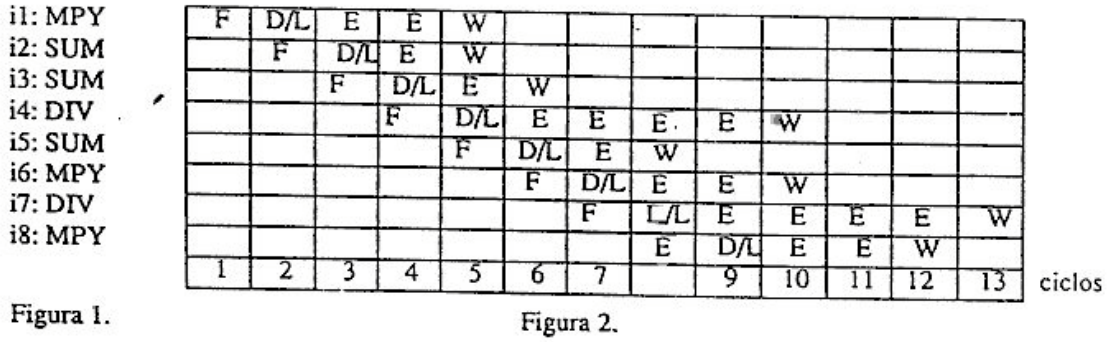


Figura 1.

Figura 2.

Este diagrama ideal supone que no hay nunca ninguna causa de bloqueo del procesador. Parece lógico pensar que, en general, se tardarán más de 13 ciclos debido a distintas dificultades prácticas:

a) Dependencias entre instrucciones por operandos.
b) Conflictos por una misma unidad funcional.
c) Varios accesos simultáneos al banco de registros.
d) Imposibilidad de iniciar y/o acabar la ejecución de varias instrucciones por ciclo.
Para analizar la eficiencia de varias alternativas arquitéctonicas en este procesador, compararemos la ejecución de un código concreto con la forma general del de la figura 1 que es el siguiente:
i1: R1 <--- R3 * R4
i2: R2 <--- R4 + R2
i3: R3 <--- R1 + R5
i4: R2 <--- R4 / R6
i5: R4 <--- R5 + R3
i6: R1 <--- R1 * R1
i7: R4 <--- R4 / R3
i8: R2 <--- R4 * R1
Se pide hacer esta comparación a base de representar cada vez el correspondiente diagrama real de ejecución, indicando cada vez cuales son los ciclos de bloqueo, cual es la causa, y cuantos ciclos totales resultan. Identificar cada causa de bloqueo situando dentro de la casilla correspondiente del diagrama, alguna(s) de las siguientes cifras identificadoras:
(1) = Falta bus de acceso a registros y no puede leer o escribir en el banco.
(2) = Falta(n) operando(s) por leer.
(3) = No puede adelantarse en escritura.
(4) = Unidad funcional ocupada y no se lanza a ejecutar.
(5) = Instrucción decodificada pero encolada esperando operando(s) de unidad(es) funcional(es).
Las alternativas a estudiar son las 7 siguientes:
a) Sus características son:

- el procesador solo puede obtener los operandos leyendo del banco de registros.
- hay dos buses de acceso al banco de registros, uno para lectura o escritura y otro para lectura. La escritura es prioritaria frente a la lectura.
- No se puede iniciar (lanzar) la ejecución de más de una instrucción por ciclo y se debe seguir el orden secuencial del programa.
- La escritura en registros de la instrucción j no puede hacerse antes que la de la instrucción i si se hizo primero el fetch de la i. No obstante, la U.F. queda libre y el resultado queda encolado.
- Cualquier riesgo o conflicto bloquea el Fetch y Decode de instrucciones.
- Además, las características expuestas al principio (Sabemos, de momento...)

b) Sus características son:
- Las mismas que a) excepto que ponemos cualquier cantidad de unidades funcionales de cálculo.

c) Sus características son:
- Las mismas que a) excepto que las tres unidades funcionales son segmentadas, con MAL = 1.

d) Sus características son:
- Las mismas que a) excepto que ponemos 3 buses de acceso al banco de registros, 2 para lectura o escritura y 1 para escritura. La escritura es prioritaria frente a la lectura.

e) Sus características son:
- Las mismas que d) excepto que ponemos todos los cortocircuitos que hagan falta (Forwarding de datos).

f) Sus características son:
- Las mismas que e) excepto que ahora puede hacerse lectura parcial de operandos en la fase D/L y acabar las lecturas en ciclo(s) posterior(es). Tanto el lanzamiento a ejecución como la escritura puede hacerse en cualquier orden que respete al algoritmo.

g) Reorganización de código.
- Analizar ahora las posibles mejoras de eficiencia para este código ejecutado en el procesador f) si el compilador es capaz de reordenarlo. Para ello, hay que estudiar el grafo de dependencias.

PROBLEMA 13

Supongamos un procesador segmentado con las siguientes características:

a) Existen dos sumadores (uno para números codificados en coma flotante y otro para números enteros) y un multiplicador para números codificados en coma flotante. Estas unidades aritméticas están segmentadas con un MAL = 1, y un número ilimitado de unidades funcionales virtuales en cada una de ellas.
b) Los tiempos de ejecución de cada una de las instrucciones son los siguientes: Load 11 ciclos, Jump, Mul y Add 5 ciclos y Store, Addi 1 ciclo. No se han contabilizado, porque no afectan al problema, los ciclos necesarios para hacer la búsqueda y la decodificación. Las direcciones de memoria se calculan en un sumador aparte que tarda un ciclo y forman parte de la ejecución de las instrucciones correspondientes. Los registros ai contienen números enteros y los ri números reales.
c) Cuando exista una dependencia que no se puede resolver se bloqueará la unidad que lanza a ejecutar las instrucciones. Es posible que en un mismo ciclo se lancen a ejecutar la instrucción que estaba inhibiendo la decodificación de instrucciones y alguna encolada en las unidades funcionales virtuales.

Suponiendo que las instrucciones del siguiente programa están en un buffer interno del procesador se trata de evaluar el número de ciclos que tarda el procesador en ejecutar dos iteraciones del programa. Hacer un diagrama de tiempos en la tabla adjunta indicando:

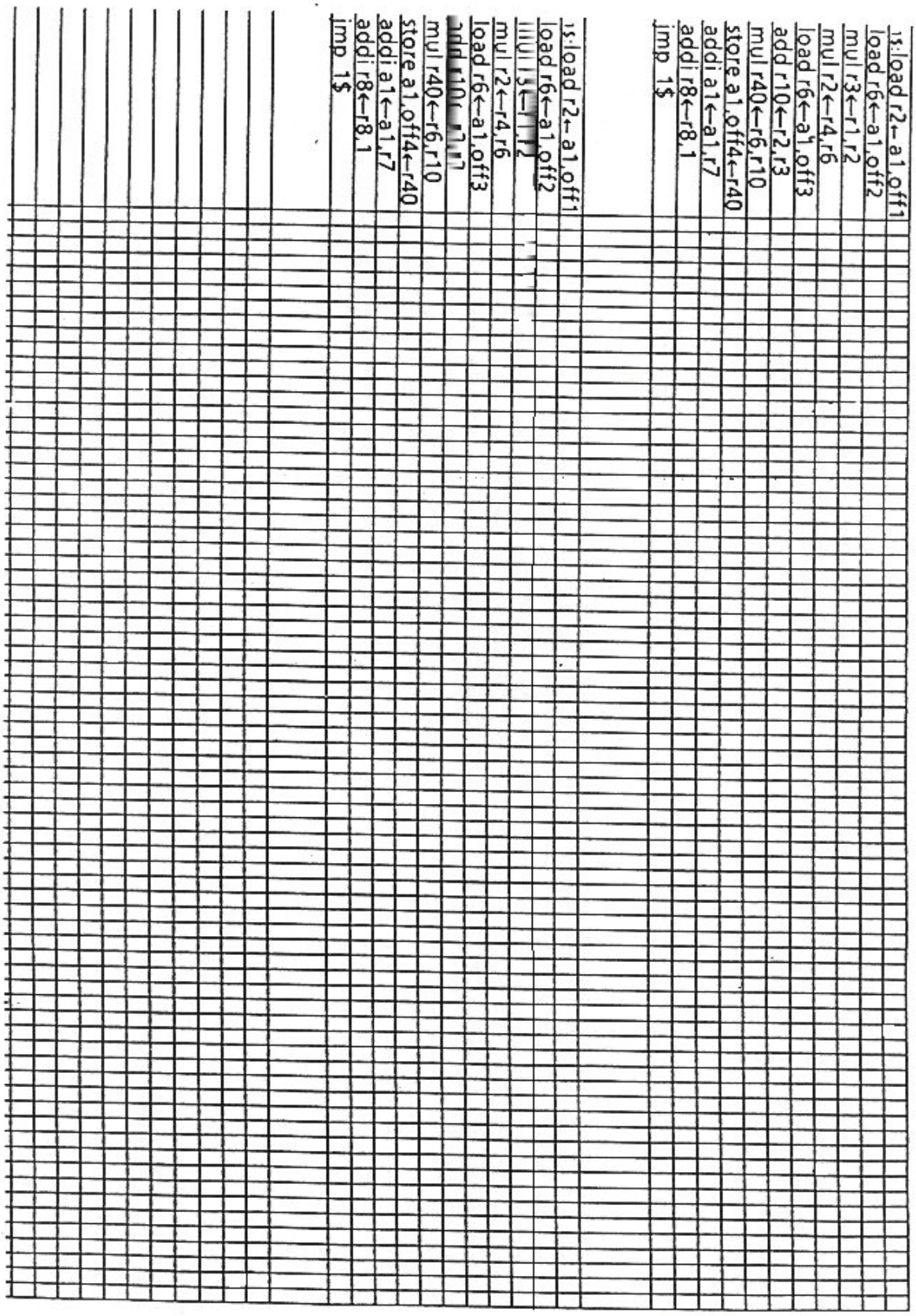
■ instrucción en ejecución

☒ instrucción inhibiendo la decodificación de instrucciones. Está bloqueada.

▪ Instrucción encolada en la unidad virtual.

Realizaremos la evaluación para los siguientes casos:

a) El procesador posee mecanismos hardware para resolver todas las dependencias de datos.
b) El procesador no puede resolver las dependencias de tipo $R(i) \cap R(j) \neq 0$.
c) Indicar si una reescritura del programa permite mejorar el rendimiento obteniendo en alguno de los dos casos anteriores explicando el porqué. Dibujar un diagrama de tiempos del caso en el que se obtenga un incremento más significativo del rendimiento.



...VIENE DEL PROBLEMA 13

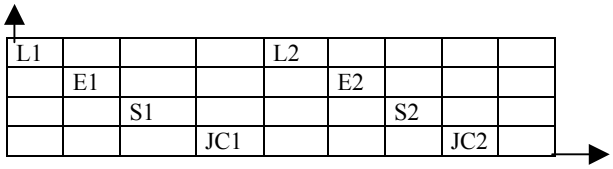
PROBLEMA 14

En un computador segmentado sin mecanismos de resolución de riesgos se bloquea la búsqueda de instrucciones cuando se detecta una dependencia en ejecución. Por ello, interesa reordenar las instrucciones de forma que se detecten el menor número posible de dependencias en ejecución. En concreto, interesa optimizar al máximo la ejecución de secuencias de código que se ejecutan de forma repetitiva (bucles).

Supongamos el siguiente bucle:

```
loop:  Li
       Ei
       Si
       JC loop
```

donde i representa la iteración i, Li representa una instrucción que trae operandos de memoria, Ei representa una instrucción de cálculo cuyos operandos han sido suministrados por Li, Si representa una instrucción que almacena en memoria el resultado que se acaba de calcular y JCi es la instrucción de control de secuenciamento.
Un posible diagrama de tiempos (ideal) de la ejecución del bucle anterior es el siguiente (se ha supuesto que todas las instrucciones tardan un ciclo):



En la figigura no se especifica ni el tiempo de búsqueda de la instrucción ni el de decodificación de las instrucciones, solo se especifica el tiempo de ejecución.
Reescribir el bucle de forma que se optimice el tiempo de ejecución del bucle suponiendo que el número de ciclos de ejecución de las instrucciones es el siguiente: $T_L = 4$, $T_E = 2$, $T_S = 1$ y $T_{JC} = 1$.
NOTA: Reescribir el bucle de forma que en una iteración del nuevo bucle se ejecuten instrucciones correspondientes a iteraciones distintas del bucle original. La reescritura del bucle tiene el siguiente aspecto:

preámbulo
NUEVO BUCLE
epílogo

Es decir hay que ejecutar unas cuantas instrucciones antes de empezar a ejecutar el NUEVO BUCLE y posteriormente es necesario ejecutar instrucciones que completen las iteraciones inacabadas del bucle original.

PROBLEMA 15

Un procesador segmentado tiene tres etapas (I, M y A). Existen, únicamente, dos tipos de instrucciones X eY y su ejecución utiliza las etapas en los ciclos especificados a continuación:

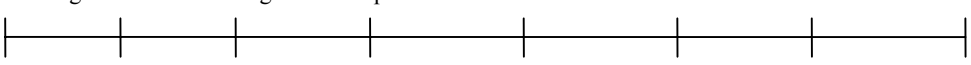
CICLOS	1	2	3
X	1	A	
Y	1	M	A

Dos instrucciones colisionan si durante su ejecución intentan utilizar la misma etapa en el mismo instante de tiempo.
Contestar a las siguientes preguntas:

a) Comprobar que es posible que exista colisión al ejecutar un programa si la latencia de inicialización de instrucciones es de un ciclo.
b) Cuál es la latencia mínima constante (MAL) que garantiza que no existe colisión en la ejecución de un programa.
c) Rediseñar los dos tipos de instrucciones para que la latencia mínima constante sea la menor posible. (NOTA: analizar el número de ciclos y utilización de las etapas).
d) Razonar si el nuevo diseño aumenta o disminuye el tiempo de ejecución de los programas al compararlo coin el caso a).

PROBLEMA 16

Sea un procesador segmentado con las siguientes etapas:



BU DL A1 A2 M A2 ES

Todas las instrucciones pasan por todas las etapas (un ciclo en cada etapa), aunque algunas instrucciones en algunas etapas no hagan nada. Todas las etapas se han implementado con hardware separado.
A efectos del problema nos interesan los siguientes tipos de instrucciones:

OPR: Rd <--- Rf1 op Rf2

OPM: Rd <--- M [Rf1 + X] op Rf2

LOAD: Rd <--- M [Rf1 + X]

En las etapas se efectúan las siguientes operaciones:

BU: búsqueda de la instrucción e incremento del PC.

DL: decodificación y lectura de operandos.
En caso de riesgo por dependencias, que no pueda resolverse con cortocircuitos, se bloquea la instrucción en esta etapa, hasta que se resuelve el riesgo. Ninguna otra instrucción posterior pasará a ejecutarse hasta que la instrucción bloqueada salga de la etapa DL (o sea, las instrucciones se ejecutan en orden).

A1: Se efectúa la operación en las instrucciones OPR y se calcula la dirección de memoria {Rf1 + X] en las instrucciones OPM y LOAD (X es un campo que está en la instrucción).

M: Se accede a memoria en las instrucciones OPM y LOAD.

A2: Se calcula la operación en las instrucciones OPM.

ES: Se escribe el resultado en el registro destino (al final del ciclo).

RESPONDER A LAS SIGUIENTES PREGUNTAS:

a) Identificar los 23 elementos [banco de registros (BR), unidades aritméticas (ALU's), memorias (MEM's), multiplexores (MPX's), PC, RI y registros entre etapas (REG)] de la figura que representa el camino de datos para poder ejecutar estos 3 tipos de instrucciones. Observar que no hay ningún cortocircuito (bypass). Usar la hoja de respuestas rellenando la tabla inferior.

En las preguntas siguientes se pide que indiquéis la ejecución de un código mediante diagramas de instrucciones-tiempo (como los hechos en clase). En estos diagramas se dedica una fila para cada instrucción y una columna para cada ciclo, indicándose en que etapa está cada instrucción en cada ciclo. Se deben indicar también (si es el caso) los ciclos de bloqueo y los cortocircuitos entre las etapas que se usan en cada ciclo (con flechas etiquetadas). Cada etiqueta (a, b, c, ...) se corresponde con un cortocircuito hardware, que deberéis dibujar y etiquetar sobre el camino de datos de la pregunta anterior.

b) Monitorizar en un diagrama instrucciones-tiempo la ejecución del siguiente fragmento de programa suponiendo que no existen cortocircuitos. Indicar el número total de ciclos.

CODIGO 1:

LOAD: R1 <--- M [R2 + X]

OPR: R2 <--- R1 op R2

LOAD: R3 <--- M[R2 + X]

OPM: R1 <--- M[R3 + X] op R2

OPM: R1 <--- M[R2 + X] op R1

c) Monitorizar en un diagrama instrucciones-tiempo la ejecución del código 1, usando todos los cortocircuitos que sean necesarios para minimizar el tiempo de ejecución. Usar la hoja de respuestas para marcar los cortocircuitos en la figura del camino de datos. Indicar el número total de ciclos.

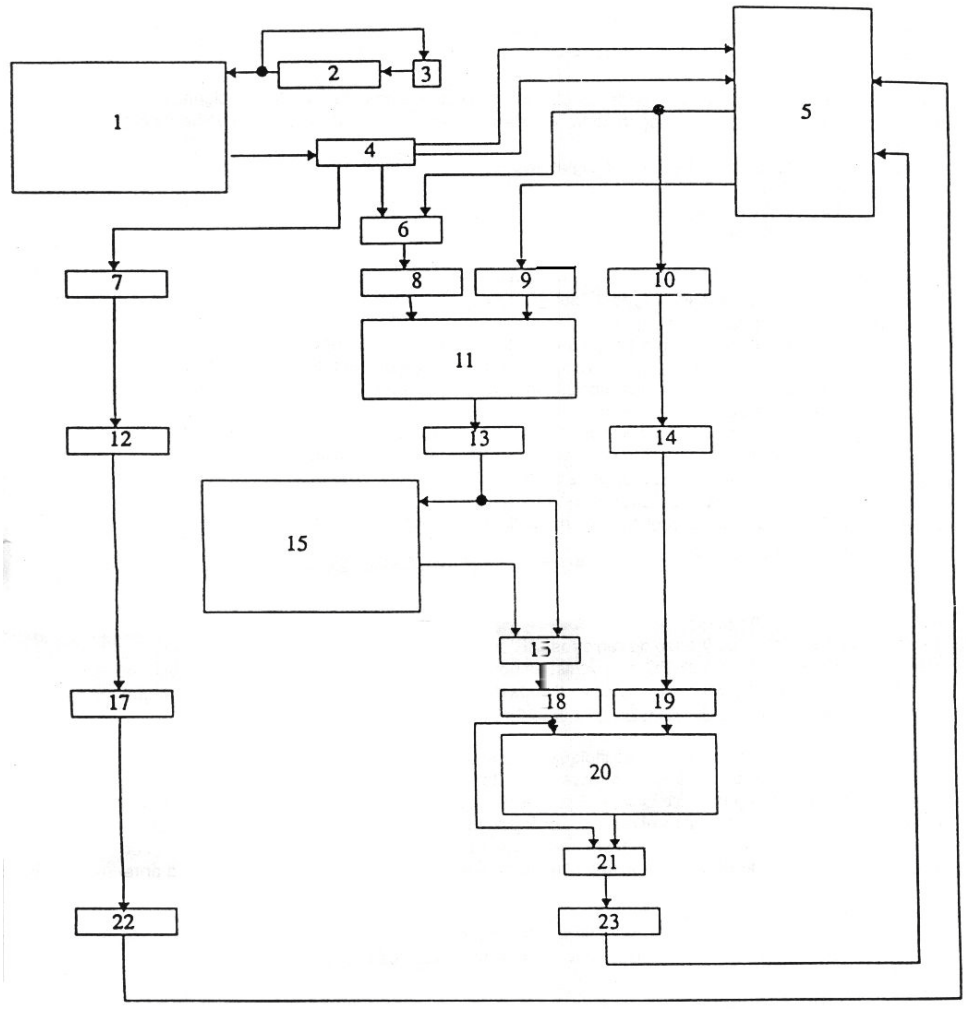
d) ¿Se puede reducir el tiempo de ejecución del código 1 (respecto al de la pregunta c) con alguna técnica software? ¿Con qué cortocircuitos, en caso de poderse reducir el tiempo?. Razonar la respuesta.

e) Repetir el apartado c) si la última instrucción del código 1 se cambia por:

OPM : R1 <--- M[R1 + X] op R2

Señalar qué cortocircuitos sirven. Indicar el número total de ciclos.

....viene del problema 16

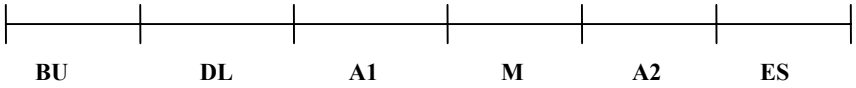


Rellenar números de 1 a 23

BR	ALUs	MEMs	MPXs	PC	RI	REGs

PROBLEMA 17

Sea un procesador segmentado con las siguientes etapas:



Todas las instrucciones pasan por todas las etapas (un ciclo en cada etapa), aunque algunas instrucciones en algunas etapas no hagan nada. Todas las etapas se han implementado con hardware separado.
A efectos del problema nos interesan los siguientes tipos de instrucciones:

OPR: Rd <--- Rf1 op Rf2

OPM: Rd <--- M[Rf1 + X] op Rf2

LOAD: Rd <--- M[Rf1 + X]

En las etapas se efectuan las siguientes operaciones:

BU: Búsqueda de la instrucción e incremento del PC.

DL: Decodificación y lectura de operandos.
En caso de riesgo por dependencias, que no pueda resolverse con cortocircuitos, se bloquea la instrucción en esta etapa, hasta que se resuelve el riesgo. Ninguna otra instrucción posterior pasará a ejecutarse hasta que la instrucción bloqueada salga de la etapa DL (o sea, las instrucciones se ejecutan en orden).

A1: Se efectua la operación en las instrucciones OPR y se calcula la dirección de memoria [Rf1 + X] en las instrucciones OPM y LOAD (X es un campo que está en la instrucción).

M: Se accede a memoria en las instrucciones OPM y LOAD.

A2: Se efectua la operación en las instrucciones OPM.

ES: Se escribe el resultado en el registro destino (al final del ciclo).

RESPONDER A LAS SIGUIENTES PREGUNTAS

a) Dibujar el camino de datos (banco de registros, buses, ALUs, memorias, multiplexores, PCs, Irs, registros separadores de etapas, y todas sus interconexiones) para ejecutar los tres tipos de instrucciones. Suponer en este apartado, que no existen cortocircuitos.

En las preguntas siguientes se pide que indiqueis la ejecución de fragmentos de programas mediante diagramas de instrucciones-tiempo (como los hechos en clase). En estos diagramas se dedica una fila para cada instrucción y una columna para cada ciclo, indicándose en qué etapa está cada instrucción en cada ciclo. Se deben indicar también (si es el caso) los ciclos de bloqueo y los cortocircuitos entre las etapas que se usan en cada ciclo (con flechas etiquetadas). Cada etiqueta (a,b,c,...) se corresponde con un cortocircuito hardware, que debemos dibujar y etiquetar sobre el camino de datos de la pregunta anterior.

b) Monitorizar en un diagrama instrucciones-tiempo la ejecución del siguiente fragmento de programa suponiendo que no existen cortocircuitos.

CODIGO 1:

OPR: R2 <--- R1 op R2

LOAD: R3 <--- M[R1 + X]

OPM: R1 <--- M[R3 + X] op R2

c) Monitorizar con un diagrama instrucciones-tiempo la ejecución del código 1 si se dispone de los cortocircuitos que sean necesarios para reducir el tiempo de ejecución.

d) ¿Se puede reducir el tiempo de ejecución del código 1 (respecto al de la pregunta 3) con alguna técnica software? ¿Son necesarios cortocircuitos?. Monitorizar los resultados con un diagrama instrucciones-tiempo.

PROBLEMA 18

Considerar un procesador segmentado con varias unidades funcionales de diferente tiempo de ejecución y control dinámico de riesgos. Considerar solo instrucciones registro-registro.

a) Indique DOS razones por las cuales es ventajoso escribir en los registros destino en orden.

b) Para controlar la escritura en orden se dispone de un buffer B de N posiciones y de un vector de bits V de N bits. Los resultados de las operaciones junto con el número del registro destino se almacenan en el buffer y durante el ciclo de escritura se obtienen del buffer. Describir las acciones que deben efectuarse durante las etapas de envio (lanzamiento) de instrucciones, de ejecución y de escritura para controlar la escritura en orden.

c) Suponiendo que los riesgos RAW se solucionan como en el método de Tomasulo, es decir los resultados de las unidades funcionales se envian a las estaciones de reserva que los requieren, explicar las modificaciones necesarias para incluir la escritura en orden (controlada como en 2).

d) ¿Qué efecto tiene la escritura en orden en el número de instrucciones ejecutadas por ciclo? Indicar como aumentan los riesgos estructurales y de datos.

e) Co el método de control de riesgos RAW indicado en 3), explicar como se solucionan los riesgos WAW y WAR para los dos casos siguientes:

- las instrucciones escriben en los registros destino en orden.
- las instrucciones pueden escribir en los registros destino fuera de orden.

PROBLEMA 19

Dibujar el grafo de dependencias (en el espacio de sentencias únicamente) indicando la distancia y tipo (verdadera, antidependencia y de salida) de las dependencias para el siguiente bucle:

```
DO I= 1 , N
    S1:  A(I - 1) = C(I) + 2
    S2:  C(I) = A(I + 2)
    S3:  D(I + 1) = A(I - 2)
    S4:  C(I + 2) = D(I - 2)
ENDDO
```

PROBLEMA 20

Un procesador tiene dos tipos de instrucciones, X e Y. Las de tipo X tardan un ciclo en ejecutarse y las de tipo Y tardan dos ciclos.
Se rediseña el procesador y se comprueba que ahora todas sus instrucciones pueden ejecutarse en un ciclo. Ahora bien, el nuevo tiempo de ciclo es 1.2 veces el del original.
¿Cuál es la mínima proporción de instrucciones del tipo Y que debe contener un programa para que en el nuevo diseño el tiempo de ejecución del programa sea inferior?

PROBLEMA 21

Suponer que tenemos las siguientes unidades funcionales de coma flotante:
GRUPO 1: 2 sumadores con latencia de 2 ciclos.
GRUPO 2: 2 multiplicadores con latencia de 5 ciclos.
GRUPO 3: 2 divisores con latencia de 19 ciclos.
Se han diseñado dos procesadores segmentados, donde además de la ALU para aritmética entera se han utilizado, en cada uno de ellos, todas las unidades funcionales de coma flotante. En cada procesador existe un conjunto de registros para enteros y otro para números en coma flotante. Las unidades funcionales de coma flotante únicamente operan con los registros de coma flotante.

PROCESADOR A: Se utiliza la técnica del scoreboard para tratar los riesgos en ejecución (recordar que los operandos se leen todos en el mismo ciclo). Cada grupo de unidades funcionales tiene dos caminos de lectura y uno de escritura al conjunto de registros.

PROCESADOR B: Se utiliza la técnica de Tomasulo para tratar los riesgos en ejecución (recordar que existe un único bus al que se conecta la salida de todas las unidades funcionales y del que pueden leer las unidades funcionales, que los operandos pueden leerse en distintos ciclos y que existen dos caminos de lectura y uno de escritura al conjunto de registros).

Se quiere estudiar el retardo en el inicio de ejecución (acabar de leer o leer los datos y empezar a ejecutar) de una secuencia de instrucciones, debido al número limitado de recursos disponibles para que las unidades funcionales se alimenten de datos y escriban en el conjunto de registros.
PREGUNTAS: (Justificar las respuestas)

a) Dar una secuencia de instrucciones inferior a 6 que muestre la técnica del scoreboard retarda el inicio de una instrucción y que la técnica de Tomasulo no la retarda.
b) Dar una secuencia de instrucciones inferior a 6 que muestre que la técnica de Tomasulo retarda el inicio de una instrucción y que la técnica de scoreboard no la retarda.

PROBLEMA 22

Una unidad funcional segmentada tiene una tabla de reserva que permite el ciclo o lazo simple (2,3,7). Contestar a las siguientes preguntas:

a) Obtener un vector de colisión inicial que permita ese ciclo simple. Uno de los estados del ciclo se corresponde con el vector de colisión inicial. Justificar de forma detallada la respuesta.
b) Deteminar la latencia media del ciclo.
c) Suponiendo que siempre existen datos disponibles a la entrada de la unidad funcional diseñar el controlador para cumplir el ciclo del enunciado.

PROBLEMA 23

Considerar la siguiente tabla de reserva para una Unidad Funcional Segmentada (UFS) de 4 etapas con un tiempo de ciclo de 20 ns.

S1	X					X
S2		X				

S3			X			
S4				X	X	
	1	2	3	4	5	6

RESPONDER JUSTIFICADAMENTE A LAS SIGUIENTES A LAS SIGUIENTES PREGUNTAS

a) ¿Cuáles son las latencias prohibidas y cuál es el vector de colisión inicial?
b) Dibujar el diagrama de estados reducido indicando las latencias entre estados.
c) Determinar que lazo voraz (greedy, avaricioso) tiene la menor latencia media.
d) Calcular la utilización de cada etapa si se sigue el lazo de la pregunta 3.
e) Determinar el límite inferior y superior de la latencia media mínima alcanzable.
f) Suponiendo que existen siempre datos disponibles en la entrada, indicar el vector de bits que controla las iniciaciones (entrada de datos en la UFS).

Para mejorar alguna de las características de la UFS se insertan retardos en la tabla de reserva. Por ejemplo, si se inserta un retardo de un ciclo en el ciclo 2 la nueva tabla es

S1	X						X
S2			X		X		
S3				X			
S4					X	X	
	1	2	3	4	5	6	7

Observar que se han retrasado 1 ciclo todas las marcas a partir del ciclo 2.

g) ¿Dónde se puede insertar un retardo de 1 ciclo en la tabla de reserva original para que en el grafo de estados existan lazos que incluyan la latencia 1 (ej.: (I1, I2, 1, I4, I5)).?
h) ¿Cuál es el lazo con menor latencia media para esta nueva UFS?

PROBLEMA 24

Tenemos un procesador segmentado lineal del tipo DLX (explicado en clase), con 5 etapas (Búsqueda, Decodificación y Lectura de Operandos, ALU, Memoria y Escritura). El data path tiene todos los cortocircuitos necesarios para reducir el efecto de los riesgos de datos.

Nuestro procesador NO dispone de instrucciones de salto retardado. Cuando se ejecuta una instrucción de salto condicional, el procesador se bloquea hasta que se tiene el resultado del salto. La evaluación de la condición y cálculo de la dirección efectiva se hace en la etapa de ALU, y al final de esta etapa ya se puede cargar el PC con el resultado del salto.

Se quiere ejecutar el siguiente trozo de programa en FORTRAN:

```
N = 0
DO I = 0,99
    N = N + A(I)
ENDDO
```

El código generado por el compilador es:

```
        MOV   R0,  #0          ; R0 es N
        MOV   R1,  #0          ; R1 es I
LOOP:   LOAD  R2, A(R1)        ; R2 = A(I)
        INC   R1               ; I = I + 1
        ADD   R0, R0, R2       ; N = N + A(I)
        CMP   R1,  #100
        BLESS LOOP
```

PREGUNTAS:

a) ¿Cuántos ciclos tarda en ejecutarse este bucle?
b) Se quiere ejecutar el mismo bucle en otro procesador segmentado lineal con las mismas características que el anterior pero con la diferencia de que SI que tiene instrucciones de salto retardado. Modificar el código en ensamblador para que se pueda ejecutar en este procesador. Se pide que el código optimice el tiempo de ejecución del bucle.
c) ¿Cuántos ciclos tarda en ejecutarse el nuevo código?

PROBLEMA 25

Tenemos un procesador segmentado del tipo DLX, lineal y con 5 etapas.

En cada ciclo comienza a ejecutar una nueva instrucción (puede ser una NOP) y nunca puede bloquear ni anular ninguna instrucción.

La escritura en el banco de registros ocupa todo el quinto ciclo.

Tenemos que ejecutar un programa que intercambie el contenido de dos posiciones de memoria.

PREGUNTAS

a) Escribir una secuencia óptima de instrucciones máquina para este programa, en cada uno de los 3 casos siguientes:

- el procesador no dispone de ningún "bypass" (cortocircuito) y todos los datos han de ser leidos en el segundo ciclo.
- el procesador dispone de "bypass" (cortocircuito) hacia la entrada de la ALU.
- el procesador dispone de "bypass" (cortocircuito) hacia la entrada de cualquier etapa que convenga.

b) Si sabemos que la frecuencia de reloj para este procesador es de 20 Mhz cuando no tiene "bypass", pero es de 16 Mhz cuando tiene "bypass", calcular los tiempos de ejecución del programa en cada uno de los 3 casos anteriores.

NOTA: Justificar todas las respuestas brevemente. En los apartados 1.2) y 1.3) indicar qué cortocircuitos se usan ???????.

PROBLEMA 26

Sea un procesador segmentado de 5 etapas, como el DLX.

La escritura en el banco de registros ocupa todo el último ciclo. Las instrucciones de salto (no son delayed branch) causan el bloqueo del procesador durante al menos 2 ciclos (tienen latencia 3 y en la tercera etapa comparan y cargan el PC).

Vamos a considerar los dos fragmentos de código siguientes:

	CODIGO 1		CODIGO 2
acumula:	R1 <--- Mem [R20 + K1]	copia:	R1 <--- Mem [R20 + K1]
	R5 <--- R1 + R5		Mem [R20 + K2] <--- R1
	R20 <--- R20 - K3		R20 <--- R20 - K3
	saltar a acumula si R20 > 0		saltar a copia si R20 > 0
	Mem [R20 + K1] <--- R5		

PARA CADA UNO DE LOS DOS CODIGOS SE PIDE:

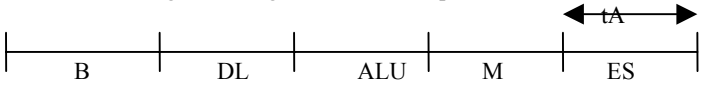
Calcular el número de ciclos que dura su ejecución, justificado con el dibujo de su cronograma, siendo N el número de veces que se ejecuta cada bucle, y para cada uno de los cinco casos siguientes:

a) El procesador no tiene ningún cortocircuito (bypass).
b) El procesador tiene bypass MEM / MEM .
c) El procesador tiene bypass MEM / MEM y bypass ALU / ALU.
d) El procesador no tiene ningún bypass. Reordenar las instrucciones de cada uno de los dos códigos para minimizar, si es posible, el tiempo de ejecución.
e) El procesador no tiene ningún bypass. Reescribir, aumentando el número de instrucciones, cada uno de los dos códigos a ejecutar. Pueden replicarse instrucciones. Se desea optimizar el tiempo de ejecución, hasta donde sea posible. Suponer que no hay limitación en el número de registros del procesador.

Suponer que el número N es múltiplo de 4.

PROBLEMA 27

En un procesador segmentado se distinguen las siguientes cinco etapas.



donde

DL: decodificación y lectura de registros
cálculo de la dirección efectiva y evaluación de la condición en instrucciones que modifican el secuenciamiento.
Otras etapas: acciones iguales a las explicadas en clase.

Una propuesta que permite reducir el tiempo de ciclo del procesador ($t_S < t_A$) es modificar la segmentación de la siguiente forma, usando siete etapas:

donde
B1, B2: búsqueda de la instrucción (llega al procesador al final de B2, 2 etapas)
DL: decodificación y lectura de registros.
ALU: operaciones aritmético-lógicas.
cálculo de la dirección efectiva para acceder a memoria
cálculo de la dirección efectiva y evaluación de la condición en instrucciones que modifican el secuenciamiento.
M1, M2: acceso a memoria para obtener el dato (al final de M2)
ES: escritura en registros.
En los dos procesadores se busca en memoria una instrucción por ciclo, se utiliza la técnica de salto retardado y se supone que existen todos los cortocircuitos posibles.
El compilador siempre separa lo máximo posible las instrucciones con dependencias de forma que se reduzcan los riesgos. Para el caso de salto retardado no efectua cálculo útil, tampoco lo efectuan las instrucciones posteriores a la k que pertenecen a las posiciones de retardo.
Para comparar el rendimiento de los dos procesadores se dispone de la siguiente información estadística sobre los programas:

a) El 50% de las instrucciones son operaciones aritméticas, el 20% son instrucciones de load y el 30% son instrucciones de salto.
b) Sea la instrucción i un load

- en el 20% de los casos la instrucción i +1 depende de la instrucción i.
- en el 75% de los casos la primera instrucción que depende de la instrucción i es la instrucción i + 2.
- en el 4% de los casos la primera instrucción que depende de la instrucción i es la instrucción i + 3.
- en el 1% de los casos la primera instrucción que depende de la instrucción i es la instrucción i + 4 o posteriores.

c) Sea la instrucción i un salto retardado, la utilidad para el algoritmo de las instrucciones ubicadas en las posiciones (slots) de retardo es la siguiente:

- en el 10% de los casos la instrucción i + 1 no es útil.
- en el 80% de los casos la primera instrucción no útil es la instrucción i + 2.
- en el 9% de los casos la primera instrucción no útil es la instrucción i + 3.
- en el 1% de los casos la primera instrucción no útil es la instrucción i + 4.

PREGUNTA:
Determinar la relación entre tiempos de ciclo de los dos procesadores para que un programa tarde en promedio el mismo tiempo en ejecutarse en uno y otro procesador.
JUSTIFICAR DE FORMA RAZONADA LA RESPUESTA.

PROBLEMA 28

Sean las siguientes secuencias de código.

A)

```
        load R6 <---  #100
1S:     Fload FR1 <--- XA(R4)
        Fload FR2 <--- XB(R4)
        Fadd FR3 <--- FR1, FR2
        Fstore Xc(R4), FR3
```

B)

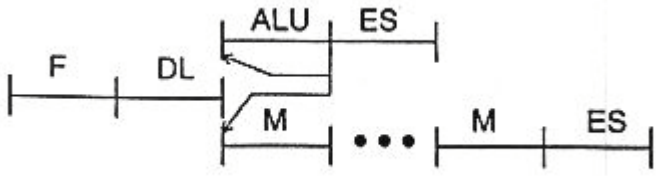
```
        load R6 <---  #100
        Fload FR1 <--- XB(R2)
1S      Fload FR2 <--- XA(R2)
        Fadd FR1 <--- FR1, FR2
        Fmul FR1 <--- FR1, FR7
```

```
Iadd R4 <--- R4, 1          Iadd R2 <--- R2, 1
Isub R6 <--- R6, 1          Fstore X_B(R2), FR1
Jnz 1$                      Isub R6 <--- R6, 1
                            Jnz 1$
```

donde Fx e Ix indican instrucciones de coma flotante y de aritmética entera respectivamente.
Suponer un procesador segmentado con el mecanismo de Tomasulo, que nunca bloquea la búsqueda de instrucciones porque suponemos ilimitado el número de unidades funcionales virtuales (estaciones de reserva). Además, el procesador dispone de un mecanismo para encolar un ilimitado número de instrucciones store, de forma que se puedan ejecutar en paralelo con otras instrucciones.
Tiene una UF de suma en coma flotante y una UF de multiplicación en coma flotante. Ambas están no segmentados y sus latencias son 12 y 15 ciclos respectivamente. La aritmética entera se realiza en otra UF con latencia 1. La latencia de la memoria (cálculo de la dirección efectiva y acceso a memoria) se supone que es un ciclo.
PREGUNTA:
Indicar para cada secuencia de código si LA SEGMENTACIÓN DE LAS U.F.s mejoraria el rendimiento (reducción del tiempo de ejecución).
JUSTIFICAR DE FORMA RAZONADA LA RESPUESTA
NOTA: Para contestar a la pregunta no es necesario efectuar una ejecución ciclo a ciclo. Es suficiente con observar el código.

PROBLEMA 29

Un procesador segmentat té les següents etapes



La memòria de dades es pot considerar com una unitat funcional no segmentada amb temps d'operació variable. En concret, el temps d'operació pot ser 1 o 3 cicles.
El processador disposa de 2 curtcircuits: ALU / ALU i ALU / M. L'escriptura en el banc de registres es fa en la primera mitad d'un cicle, mentre que la lectura es fa en la segona mitad.
El processador es bloqueja davant un risc estructural i de dependències de dades. El bloqueig inhibeix la búsqueda i descodificació de noves instruccions mentre duri el conflicte.
L'escriptura de resultats en el banc de registres no té perque seguir l'ordre del programa. Hi han dos camins d'escriptura al banc de registres (des de l'etapa ALU i des de l'etapa M).
Les instruccions de salt són Delayed Branch amb latència 2 cicles.

Volem avaluar el rendiment del processador amb diferents mecanismes. Per això considerarem el següent programa de prova

```
                                  R2 <--- N
                                  R0 <--- A
                                  R1 <--- A + 4
                                  R3 <--- B
for i=0 to N-1            bucle   R4 <--- Mem [R0]    | 1 |
a (i+1) := a (i) + a (i+1)        R5 <--- Mem [R1]    | 3 |
b (i) := a (i+1)                  R2 <--- R2 -1
k := k + 2                        R0 <--- R0 + 4
endfor                            R1 <--- R1 + 4
                                  R8 <--- R8 + 2
                                  R4 <--- R4 + R5
                                  Mem [R0] <--- R4    | 3 |
                                  Mem [R3] <--- R4    | 3 |
                                  if R2 ≠ 0 then goto bucle
                                  R3 <--- R3 + 4
```

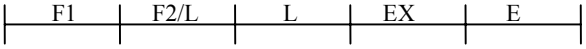
on el valor que figura en el requadre al costat de les instruccions de Load/Store indica el temps de resposta de la memòria.
Es demana:
a) El temps d'execució del bucle suposant que aquest es repeteix N vegades. Justifiqueu la resposta mitjançant un cronograma amb l'execució de les dues primeres iteracions.

Suposem que afegim al processador una cua FIFO entre les unitats de descodificació i de memòria. Aquesta FIFO té capacitat per emmagatzemar 3 instruccions Load/Store pendents. En cas de que la memòria no estigui desponible quan es descodifica una instrucció Load/Store, s'encua l'operació.

b) Indiqueu els camps de les entrades de la FIFO.
c) Dibuixeu un cronograma de les dues primeres iteracions del bucle. Quin és ara el temps total d'execució del bucle?
d) Suposem que substituim la cua FIFO per dues: una cua pels Loads i una altra pels Stores. Proposeu un algoritme de gestió de les cues que possibiliti una millora del rendiment del processador. L'única restricció que s'imposa és que les escriptures (Stores) a memòria es facin en l'ordre del programa. Justifica la resposta mitjançant un fragment de codi (amb només instruccions Load/Store) i el cronograma d'execució associat.

PROBLEMA 30

Tenim un processador segmentat multicicle amb 5 etapes:



F1: inicia fetch i actualitza comptador de programa.
F2/D: acaba fetch i descodifica.
L: verifica riscs i llegeix (en la 2ª meitat del cicle) operants del banc de registres.
EX: execució (durant els cicles que duri cada operació).
E: escriptura en el banc de registres (en la 1ª mitat del cicle).

Coneixem aquestes característiques del processador:

- pot descodificar una instrucció per cicle, si no està bloquejat.
- quan detecta risc, per conflicte en unitat funcional (UF) o per dependència de dades, el control bloqueja el processador durant els cicles que dura el risc.
- no hi ha cap camí de curtcircuit (bypass).
- hi han tres camins amb el banc de registres, dos de lectura i un d'escriptura.
- l'escriptura de resultats es fa sempre en ordre, i es manté ocupada la UF fins el cicle anterior al d'escriptura en el banc de registres.
- té dues unitats funcionals: UF1 i UF2.
  La UF1 està segmentada en 4 etapes; la utilització de les etapes es descriu mitjançant la següent taula de reserva,

X				
	X		X	
		X		X
			X	

el control.lador d'iniciacions de UF1 sempre segueix la seqüència òptima,
la UF2 no està segmentada i la seva operació consumeix tres cicles

Per estimar les prestacions d'aquest processador s'utilitza sempre el codi de prova següent:

1: F3 <--- F1 op1 F2
2: F5 <--- F4 op1 F1
3: F6 <--- F2 op2 F4
4: F3 <--- F2 op1 F7
5: F1 <--- F8 op1 F9
6: F5 <--- F6 op2 F4
7: F2 <--- F2 op2 F6

on op1 és l'operació que realitza la UF1 i op2 la que realitza la UF2.

Es demana respondre justificadament:

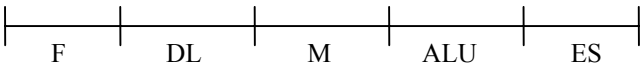
a) Calcular el número de cicles que triga en executar-se el codi de prova, representant amb claredat el cronograma d'execució cicle a cicle. (Marcar amb un creu cada cicle de bloqueig i amb línia discontínua cada retard d'escriptura).
b) Repetir l'apartat a) suposant que existeix qualsevol bypass i que s'utilitzen només si es redueixen els cicles que triga en executar-se el codi de prova.
c) Repetir l'apartat a) suposant que no existeix cap bypass, però hi ha un buffer de resultats que permet a les UF quedar lliures en acabar l'operació i escriure en aquest buffer en qualsevol ordre. L'escriptura en el banc de registres es manté ordenada.

PROBLEMA 31

Es vol dissenyar un processador segmentat per executar, entre d'altres, les següents instruccions:

càlcul	Rk <--- Ri op Rj
Load	Rk <--- Mem [Ri]; Ri <--- Ri + Rj ; sempre Ri ≠ Rk
Store	Mem [Ri] <--- Rk; Ri <--- Ri + Rj

El processador està segmentat en 5 etapes:



on

F:	fetch de l'instrucció
DL:	descodificació, comprovació de riscs, lectura operants en registres.
M:	accés a memòria
ALU:	operacions aritmètiques i lògiques
ES:	escriptura resultats a registres (al final del cicle).

Davant la presència d'un risc el processador bloqueja la búsqueda d'instruccions fins que desapareix el risc.

Es demana:

a) Disseny del Data-Path per poder executar les 3 classes d'instrucció (no cal especificar l'etapa de Fetch). El Data-Path haurà de disposar dels recursos suficients per evitar riscs estructurals. Per aquest apartat i el següent, el Data-Path no inclourà cap bypass.
b) Indicar els casos en que el processador es bloquejarà degut a riscs per dependències entre dades. Il.lustrar amb un exemple de codi cada situació junt amb un cronograma d'execució.
c) Per a cadascún dels casos de l'apartat anterior, especificar el bypass que resolgui el risc de dependències de dades, indicant per a cada bypass les etapes font i destí i la reducció de cicles de bloqueig que s'obté.