Pregunta 1. Disco duro

Considere los discos duros con las siguientes características.

Disco	Tiempo promedio de "*seek*"	RPM	Tasa de transferencia del disco	Tasa de transferencia del controlador
X	10 ms	7500	90 MB/s	100 MB/s
Y	7 ms	10000	40 MB/s	200 MB/s

A) Calcule el tiempo promedio para leer o escribir un sector de 1024 bytes para cada disco.

Tiempo = tiempo promedio seek + retraso rotacional + tiempo transferencia + overhead controlador

Retraso rotacional promedio = 0,5 rotación / RPM

Disco X: 14.02 ms
Disco Y: 10.03 ms

B) Calcule el tiempo promedio para leer o escribir un sector de 2048 bytes para cada disco.

Disco X: 14.04 ms
Disco Y: 10.06 ms

C) Para cada disco, determine el factor dominante de rendimiento. Específicamente, si pudiera introducir una mejora en algún aspecto del disco, ¿qué escogería? Si no hay factor dominante, explique por qué.

"*Seek*" time. Notar que tiempo acceso lectura cambia muy poco al doblar tamaño de sector.

Pregunta 2. Excepciones

Este ejercicio explora como el manejo de excepciones afecta el diseño de un *pipeline*. Considere las siguientes instrucciones:

Instrucción 1	Instrucción 2
BNE R1, R2, Label	LW R1, 0(R1)

A) ¿Qué excepciones puede gatillar cada una de estas instrucciones? Para cada excepción, identifique la etapa del *pipeline* en que es detectada

Instrucción 1: "*Invalid target address"* (EX). P. ej., "Label" fue mal escrita.
Instrucción 2: *"Invalid data address"* (MEM). P. ej: el contenido de R1 queda fuera de espacio de memoria

B) Si hay una dirección de manejo de excepciones distinta para cada excepción, muestre cómo debe ser cambiada la organización del *pipeline* para poder manejar esta excepción. Puede asumir que las direcciones de estos manejadores son conocidas cuando el procesador es diseñado.

El multiplexor que selecciona el siguiente PC debe tener entradas adicionales. Cada entrada es una dirección constante de un manejador de excepciones. Los detectores de excepciones deben ser agregados a las etapas apropiadas del *pipeline*, y la salida de estos detectores deben usarse para controlar el multiplexor pre-PC, y también para convertir en NOPs las instrucciones que estaban en el *pipeline* detrás de la instrucción que gatilló la excepción.

C) Si la segunda instrucción entra en etapa de "*fetch*" justo después de la primera instrucción, describa qué pasa en el *pipeline* cuando la primera instrucción causa la primera excepción indicada en A). Muestre el diagrama del *pipeline* desde el momento en que la primera instrucción entra en "*fetch*" hasta el momento en que la primera instrucción del manejador de excepciones es ejecutada.

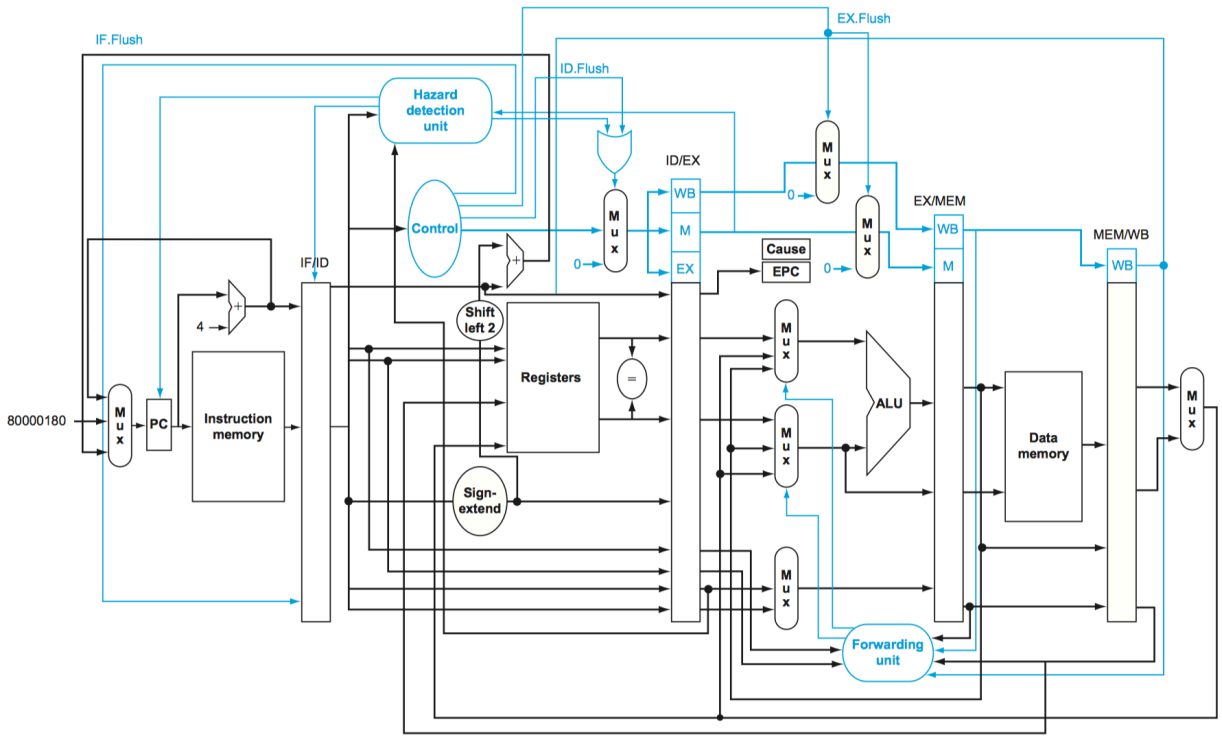
Las instrucciones entran en "*fetch"* de manera normal hasta que la excepción es detectada. Cuando la excepción es detectada, todas las instrucciones que están en el *pipeline* después de la primera instrucción son convertidas en NOP. Como resultado, la segunda instrucción nunca se complete y no afecta el estado del *pipeline*. En el ciclo que sigue inmediatamente al cicle en que la excepción es detectada, el procesador hará "*fetch*" de la primera instrucción del manejador de instrucciones.

D) En el manejo de excepciones vectorizado, la tabla de direcciones de manejo de excepciones está en la memoria de datos en una dirección fija y conocida. Cambie el *pipeline* para implementar este mecanismo de manejo de excepciones.

Este enfoque requiere hacer "*fetch*" de la dirección del manejador desde memoria. Debemos agregar el código de la excepción a la dirección de la tabla vector de excepciones, leer la dirección del manejador desde memoria, y saltar a esa dirección. Una forma de hacer esto es tratarla como una instrucción especial que calcula la dirección en EX, carga la dirección en MEM y asigna el PC en WB.

E) Queremos emular el manejo de excepciones vectorizado en una máquina que solo tiene una dirección fija de manejador de excepciones. Escriba el código que debería estar en esa dirección. HINT: este código debería identificar la excepción, obtener la dirección apropiada de la tabla de excepciones vectorial y transferir la ejecución a ese manejador de excepciones.

Necesitamos una instrucción especial que nos permita mover un valor desde el registro de causa de la excepción ("*Cause register*") a un registro de propósito general. Primero debemos guardar el registro de propósito general (para restablecerlo mas tarde), cargar el registro de causa de la excepción en él, sumarle la dirección de la tabla vectorial, usar el resultado como una dirección para realizar un "*load*" que obtiene desde memoria la dirección del manejador de excepción correcto, y finalmente, saltar a ese manejador.



Pregunta 3. Multithreading

Considere las siguientes organizaciones de CPU:

- CPU SS: Un microprocesador de 2 *cores*, superescalar, que provee la capacidad de enviar instrucciones fuera de orden (*out-of-order*) en dos unidades funcionales. Un solo hilo (*thread*) corre en cada *core* al mismo tiempo.
- CPU MT: Procesador con *multithreading* de granularidad fina que permite que 2 instrucciones corran concurrentemente (hay dos unidades funcionales), pero solo instrucciones de un solo hilo pueden ser enviadas en un solo ciclo.
- CPU SMT: Un procesador SMT ("*Simultaneous Multi-Threading*") que permite que 2 hilos corran concurrentemente (hay dos unidades funcionales), e instrucciones de cualquiera de los hilos puede enviarse a ejecutar en cualquier ciclo.

Asuma que hay dos hilos, X e Y, para correr en estas CPU que incluyen las siguientes operaciones:

Hilo X	**Hilo Y**
A1: toma 3 ciclos en ejecutar	B1: toma dos ciclos en ejecutar
A2: sin dependencias	B2: conflicto por una unidad funcional con B1
A3: conflicto por una unidad funcional con A1	B3: depende del resultado de B2
A4: depende del resultado de A3	B4: sin dependencias y toma 2 ciclos en ejecutarse

Asuma que todas las instrucciones toman un ciclo de reloj en ejecutarse a menos que se diga explícitamente lo contrario o que se encuentren con un *hazard*.

A) Asuma que tiene 1 CPU SS. ¿Cuántos ciclos tomará la ejecución de estos dos hilos? ¿Cuántos espacios para enviar instrucciones son malgastados a causa de *hazards*?

Core 1	**Core 2**
A3	B1, B4
A1, A2	B1, B4
A1, A4	B2
A1	B3

B) Asuma que tiene 1 CPU MT. ¿Cuántos ciclos tomará la ejecución de estos dos hilos? ¿Cuántos espacios para enviar instrucciones son malgastados a causa de *hazards*?

FU1	**FU 2**
A1	A2
A1	
A1	
B1	
B1	
A3	
B2	
A4	
B3	B4
	B4

C) Asuma que tiene 1 CPU SMT. ¿Cuántos ciclos tomará la ejecución de estos dos hilos? ¿Cuántos espacios para enviar instrucciones son malgastados a causa de *hazards*?

FU1	FU 2
A1	B1
A1	B1
A1	B2
A2	B3
A3	B4
A4	B4