¡Feliz navidad a todos!

Merry DCChristmas! :D

Ayudantía Examen (Parte 2)

IIC2343 – Arquitectura de Computadores

Sobre el curso

Algo que se debió decir en la clase 1, en parte son cosas que uno supone sabidas.

- NO ES FÁCIL.
- Es como matemáticas discretas, más que “que corra”, importan los fundamentos que hay detrás. Es del área de ciencias de la computación.
- Es cuesta arriba. La dificultad de la materia va en aumento.
- En el equipo docente creamos las preguntas para evaluar los distintos temas, la forma de demostrar que aprendieron es respondiendo correctamente.
- Que en un semestre anterior haya habido una pregunta similar con otro enfoque da lo mismo, puede que nos hayamos querido desmarcar de esa pregunta a propósito cambiando el foco.
- ¿Qué recomiendo para estudiar? Lo que yo hacía era revisar la materia y plantearme situaciones que me interesaran. A otros les basta con leerse los apuntes (aunque tienen errores). Una amiga me contó que estudiaba corrigiendo los apuntes.

Modalidad del examen

- 4 preguntas de dos letras independientes cada una:
  - Computador básico
  - Memoria caché
  - Pipelining
  - Representación de enteros e I/O
- Se revisarán las primeras 3 preguntas que respondan. Si quieren omitir alguna, déjenla en blanco.
- 9 de julio: tentativamente de 9:00 a 13:00 hrs.

Modalidad del examen

- 4 preguntas de dos letras independientes cada una:
  - Computador básico
  - Memoria caché
  - Pipelining
  - Representación de enteros e I/O
- Se revisarán las primeras 3 preguntas que respondan. Si quieren omitir alguna, déjenla en blanco.
- 9 de julio: tentativamente de 9:00 a 13:00 hrs.

En anteriores episodios de “Ayudantía para el examen”

- Vimos cambio de base y cálculo de complementos en distintas bases
- Vimos cómo son los flip flops D con flanco de subida en realidad, tanto los de las clases como los de los apuntes están “malos” (son otra cosa).
- Vimos cómo se sincroniza el computador completo gracias al clock.

Importante: todas las preguntas de estas ayudantías son preguntas descartadas del examen (salvo la de las décimas).

Preguntas de Caché

Identifíquese, caché (1)

- En este momento encarnas al oficial de policía computacional Rick Morty y debes realizar un control de identidad a una caché salvaje, pero esta no porta sus documentos. Determina su función de correspondencia y su política de reemplazo.

		Set 0				Set 1				
Acceso	**Binario**	**Bloque 0 (0b00)**		**Bloque 1 (0b01)**		**Bloque 2 (0b10)**		**Bloque 3 (0b11)**		
0	000000	**0**	**1**	-	-	-	-	-	-	Miss
3	000011	0	1	**2**	**3**	-	-	-	-	Miss
4	000100	0	1	2	3	**4**	**5**	-	-	Miss
2	000010	0	1	**2**	**3**	4	5	-	-	Hit
5	000101	0	1	2	3	**4**	**5**	-	-	Hit
8	001000	**8**	**9**	2	3	4	5	-	-	Miss
32	100000	**32**	**33**	2	3	4	5	-	-	Miss
29	011101	32	33	2	3	**28**	**29**	**-**	**-**	Miss
31	011111	32	33	2	3	28	29	**30**	**31**	Miss
5	000101	32	33	2	3	**4**	**5**	30	31	Miss

https://docs.google.com/spreadsheets/d/1MNAnkVZnplew6CHVLeKcV273aHO4JlK65zEZE1t66Yk/edit?usp=sharing

Identifíquese, caché (1)

S: directly mapped, no necesitamos política de reemplazo

- En este momento encarnas al oficial de policía computacional Rick Morty y debes realizar un control de identidad a una caché salvaje, pero esta no porta sus documentos. Determina su función de correspondencia y su política de reemplazo.

					Set 0				Set 1				
Acceso	**Binario**	**Tag**	**Bloque**	**Posición**	**Bloque 0 (0b00)**		**Bloque 1 (0b01)**		**Bloque 2 (0b10)**		**Bloque 3 (0b11)**		
0	000000	000	00	0	**0**	**1**	-	-	-	-	-	-	Miss
3	000011	000	01	1	0	1	**2**	**3**	-	-	-	-	Miss
4	000100	000	10	0	0	1	2	3	**4**	**5**	-	-	Miss
2	000010	000	01	0	0	1	**2**	**3**	4	5	-	-	Hit
5	000101	000	10	1	0	1	2	3	**4**	**5**	-	-	Hit
8	001000	001	00	0	**8**	**9**	2	3	4	5	-	-	Miss
32	100000	100	00	0	**32**	**33**	2	3	4	5	-	-	Miss
29	011101	011	10	1	32	33	2	3	**28**	**29**	**-**	**-**	Miss
31	011111	011	11	1	32	33	2	3	28	29	**30**	**31**	Miss
5	000101	000	10	1	32	33	2	3	**4**	**5**	30	31	Miss

https://docs.google.com/spreadsheets/d/1MNAnkVZnplew6CHVLeKcV273aHO4JlK65zEZE1t66Yk/edit?usp=sharing

Identifíquese, caché (2)

- En este momento encarnas al oficial de policía computacional Rick Morty y debes realizar un control de identidad a una caché salvaje, pero esta no porta sus documentos. Determina su función de correspondencia y su política de reemplazo.

		Set 0				Set 1				
Acceso	Binario	Bloque 0 (0b00)		Bloque 1 (0b01)		Bloque 2 (0b10)		Bloque 3 (0b11)		
0	000000	**0**	**1**	-	-	-	-	-	-	Miss
4	000100	0	1	**4**	**5**	-	-	-	-	Miss
3	000011	0	1	4	5	**2**	**3**	-	-	Miss
5	000101	0	1	**4**	**5**	2	3	-	-	Hit
4	000100	0	1	**4**	**5**	2	3	-	-	Hit
8	001000	**8**	**9**	4	5	2	3	-	-	Miss
32	100000	**32**	**33**	4	5	2	3	-	-	Miss
27	011011	32	33	4	5	2	3	**26**	**27**	Miss
31	011111	32	33	4	5	**30**	**31**	26	27	Miss
3	000011	32	33	4	5	30	31	**2**	**3**	Miss

https://docs.google.com/spreadsheets/d/1MNAnkVZnplew6CHVLeKcV273aHO4JlK65zEZE1t66Yk/edit?usp=sharing

Identifíquese, caché (2)

S: 2-way associative, política es LFU desempatando con FIFO... o también puede ser random. Nunca se olviden de random.

- En este momento encarnas al oficial de policía computacional Rick Morty y debes realizar un control de identidad a una caché salvaje, pero esta no porta sus documentos. Determina su función de correspondencia y su política de reemplazo.

					Set 0				Set 1				
Acceso	Binario	Tag	Conjunto	Posición	Bloque 0 (0b00)		Bloque 1 (0b01)		Bloque 2 (0b10)		Bloque 3 (0b11)		
0	000000	0000	0	0	**0**	**1**	-	-	-	-	-	-	Miss
4	000100	0001	0	0	0	1	**4**	**5**	-	-	-	-	Miss
3	000011	0000	1	1	0	1	4	5	**2**	**3**	-	-	Miss
5	000101	0001	0	1	0	1	**4**	**5**	2	3	-	-	Hit
4	000100	0001	0	0	0	1	**4**	**5**	2	3	-	-	Hit
8	001000	0010	0	0	**8**	**9**	4	5	2	3	-	-	Miss
32	100000	1000	0	0	**32**	**33**	4	5	2	3	-	-	Miss
27	011011	0110	1	1	32	33	4	5	2	3	**26**	**27**	Miss
31	011111	0111	1	1	32	33	4	5	**30**	**31**	26	27	Miss
3	000011	0000	1	1	32	33	4	5	30	31	**2**	**3**	Miss

https://docs.google.com/spreadsheets/d/1MNAnkVZnplew6CHVLeKcV273aHO4JlK65zEZE1t66Yk/edit?usp=sharing

Preguntas de Pipelining

(ILP)

Frecuencia del clock

S: 6ns, es el tiempo que tarda la ETAPA más lenta del pipeline. No puede ser menor ya que, en ese caso, las etapas MEM no alcanzarían a completarse.

- Supón un pipeline con las siguientes etapas y sus duraciones:
  - IF: 3ns
  - ID: 1ns
  - MEM: 6ns
  - EX: 2ns
  - MEM: 6ns
  - WB: 2ns

¿Cuál es la máxima frecuencia/mínimo periodo que puede tomar el clock? Justifica.

ILP x ISA

- ¿Cuál es la relación entre tener un pipeline y el tipo de ISA que queramos usar?

Recordemos que CISC:

- Instrucciones que toman varios ciclos
- Instrucciones de memoria a memoria

Mientras que RISC:

- Instrucciones que en promedio duran un ciclo
- Instrucciones de registro a registro

S: ¡¡RISC es muchísimo más amigable con un pipeline!! Ya que si mi instrucción toma varios ciclos es muy complicado generar un pipeline que sirva. También, asumiendo que conseguí hacer un pipeline, si tengo muchas instrucciones de memoria a memoria voy a tener muchísimos hazards de datos y estructurales.

Preguntas de I/O

Viejito pascuero, para esta navidad te pido…

Indica qué dispositivo I/O le pedirías al viejito pascuero para cada una de las siguientes tareas y de qué forma te comunicarías con él (con el dispositivo, si por polling o por interrupciones).

- Esperar un intervalo de tiempo determinado (corto).
- Poner el timestamp en una nueva entrada de base de datos.
- Leer un archivo.

S1: el timer del sistema, su función es interrumpirme cuando le diga. Por interrupciones. Estar leyendo el RTC constantemente no es opción ya que leerlo toma tiempo y estaría desprovechando la CPU.
S2: el real time clock (RTC), es el reloj en tiempo real. Por polling ya que no necesito más que leer una vez el contenido y sería.
S3: el disco duro o SSD, por interrupciones ya que el disco es muy lento y por polling tendría detenida la CPU mucho rato.

Trivia: ¿quién quiere ser millonario?

- Es un sábado por la noche y estás navegando por sitios de dudosa reputación, cuando ¡anuncio salvaje aparece! (referencia). El anuncio te ofrece: “Haz click aquí para descargar más RAM”. ¿Le crees? ¿Por qué?

S: ¡NO! Nunca. La RAM es una pieza de hardware.
No tenemos la tecnología para digitalizar materia y transmitirla por internet.
Y aunque la tuviéramos, nuestro computador no es capaz de desdigitalizar materia y ponerla en la ranura de la RAM.

La canción del principio…

- Link: https://www.youtube.com/watch?v=3TV2JchT6ks
- Otra versión: https://www.youtube.com/watch?v=YR6YAgVIljg
- Documentación: https://www.youtube.com/watch?v=q72WoW4_VCo

Preguntas de Pipelining

(ILP)

La pregunta de las décimas

La pregunta de las décimas de regalo…

- En el anuncio:

“Nos dimos cuenta de que existe una ‘ligera’ discrepancia entre los apuntes y la idea de tener libertad frente a cómo manejar los saltos en esta arquitectura en particular y queremos generar un aprendizaje a partir de esta.”

La pregunta de las décimas de regalo…

- En el anuncio:

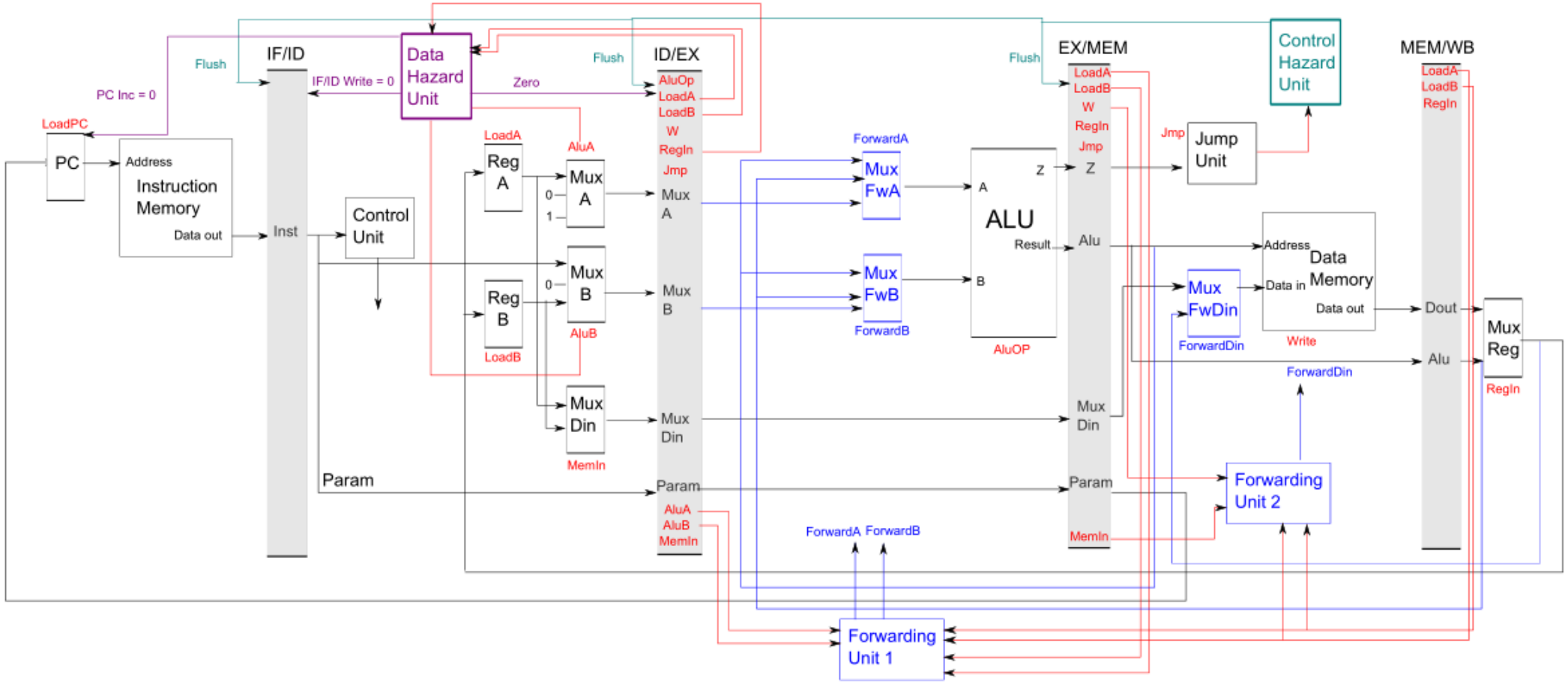
“Nos dimos cuenta de que existe una ‘ligera’ discrepancia entre los apuntes y la idea de tener libertad frente a cómo manejar los saltos en esta arquitectura en particular y queremos generar un aprendizaje a partir de esta.”

La pregunta de las décimas de regalo…

Un concepto importante a la hora de hablar del paralelismo a nivel de instrucción (ILP) por medio de un pipeline es la “ejecución especulativa” que consiste en intentar predecir si debo saltar o no ante una instrucción de salto y, en caso de equivocarme, debo restaurar el estado del computador por medio de hacer flush.

Explica clara y detalladamente por qué el computador del diagrama de computador básico del curso con pipeline que aparece en (…) es incapaz de saltar inmediatamente sino que debe esperar tres ciclos, incluso si se trata de un salto incondicional.

La pregunta de las décimas de regalo…



La pregunta de las décimas de regalo…

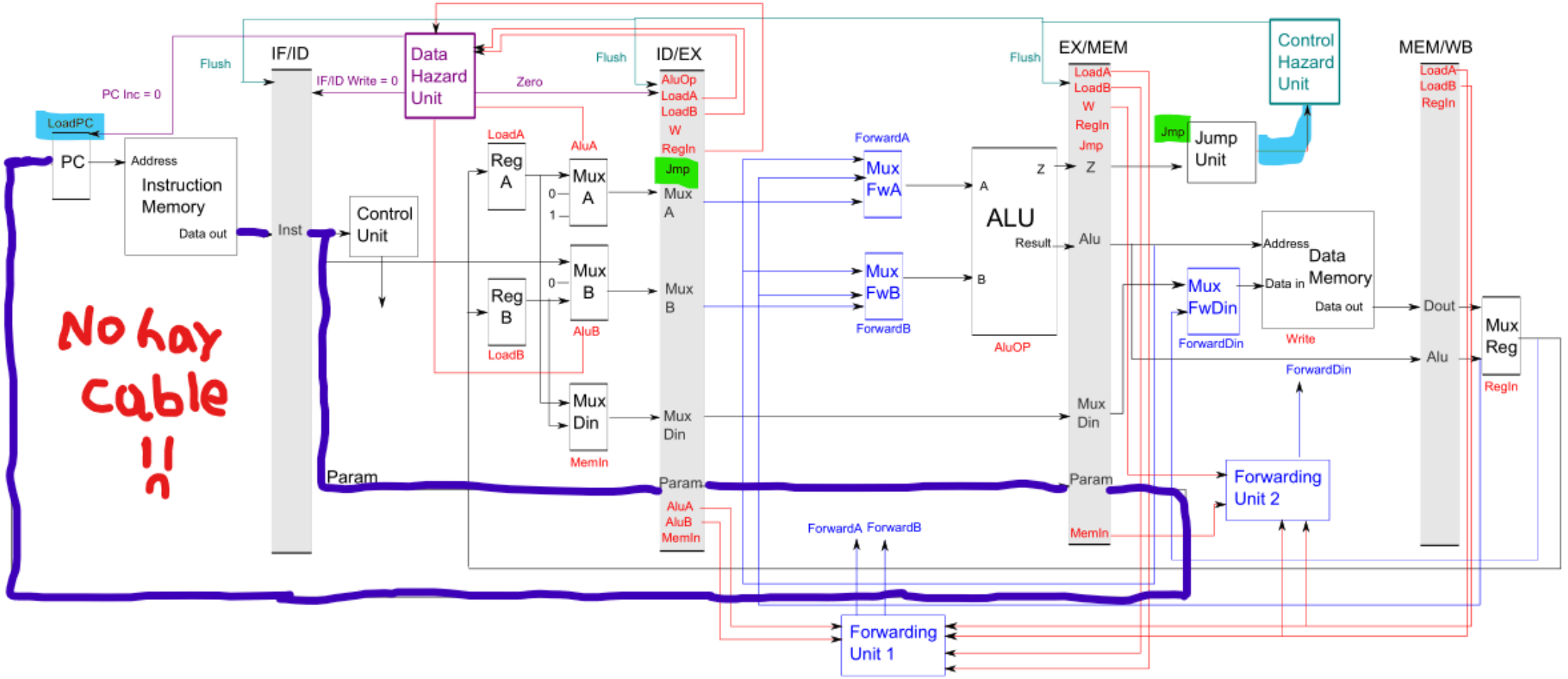
Un concepto importante a la hora de hablar del paralelismo a nivel de instrucción (ILP) por medio de un pipeline es la “ejecución especulativa” que consiste en intentar predecir si debo saltar o no ante una instrucción de salto y, en caso de equivocarme, debo restaurar el estado del computador por medio de hacer flush.

Explica clara y detalladamente por qué el computador del diagrama de computador básico del curso con pipeline que aparece en (…) es incapaz de saltar inmediatamente sino que debe esperar tres ciclos, incluso si se trata de un salto incondicional.

¿Por qué no puedo saltar antes de llegar a MEM?

¿Cuál es la razón más fundamental por la que **no puedo decir** *“esto es un salto incondicional, no tengo por qué esperar más, mejor salto ahora”* antes de llegar a MEM?

La pregunta de las décimas de regalo…



Y la explicación esperada…

- Dado que la única forma de pasar al PC la dirección a la que hay que saltar es desde la etapa MEM y lo mismo ocurre con la señal que le da la orden de cargar el valor recibido, no importa lo que hagamos: siempre vamos a perder 3 ciclos con ese diagrama.

Se dan las décimas a los primeros 10 que hayan mencionado alguna de esas dos condiciones (señal || literal).

Algunas respuestas de ustedes a la pregunta de las décimas

- **¡OJO!** Recordemos que nada de lo que diga es personal ni busca hacer sentir mal a nadie. Los comentarios de feedback buscan transmitir lo que un ayudante piensa al enfrentarse a distintas respuestas y dar consejos de cómo mejorarlas.

Algunas respuestas de ustedes…

- *Debido al pipline de la figura señalada, el salto ocurre en la etapa MEM que ocurre tres instrucciones después de que se llega a la instrucción, es decir no sabe todavía con seguridad si saltar o no, por lo que podemos intentar predecir pero siempre sabremos con seguridad si salta o no luego de tres instrucciones (ciclos) desde que se llega al salto.*

Feedback: no me hablas de los saltos incondicionales. ¿Qué ocurre exactamente en MEM referente a los saltos? ¿Cómo se ejecuta el salto? ¿Por qué no se puede hacer antes del tercer ciclo?

Aquí no se puede dar puntaje porque la respuesta está muy incompleta.

Algunas respuestas de ustedes…

- *La arquitectura propuesta en el último diagrama del capítulo 11 – ILP de los apuntes oficiales no es capaz de hacer saltos condicionales ni incondicionales en menos de 3 ciclos de pipeline. En primer lugar, no se podrían hacer en el ciclo inmediatamente siguiente a una instrucción JMP, puesto que en el primer instante esta entrará en la etapa IF, y recién en el siguiente ciclo se sabrá en ID que corresponde a una instrucción de salto, por lo que de ninguna manera se podría haber cargado el PC con la dirección correcta para el salto incondicional en el ciclo siguiente a su paso por IF. Como segundo argumento, la única forma que tiene la arquitectura del diagrama para modificar arbitrariamente el valor del PC (en lugar del incremento por defecto) es usando la señal LoadPC, controlada por la Jump Unit, que se conecta inmediatamente después del registro intermedio EX/MEM. El salto condicional o incondicional toma la dirección establecida en el registro Param, que es cargado luego en el PC cuando la señal LoadPC así lo indica. Este último valor se decodifica y propaga desde ID y es almacenado en ID/EX y EX/MEM. De esta forma, tanto para saltos incondicionales como condicionales con speculative execution, se debe esperar a completar la etapa EX, tras la cual la Jump Unit puede leer los códigos de status Z y poner un uno en la señal LoadPC. En otras palabras, una instrucción JMP | JEQ | JNE addr pasa en el primer ciclo por IF, en el segundo por ID, el tercero por EX y recién en este instante recibe la Jump Unit la información necesaria para realizar el salto, demorando siempre 3 ciclos adicionales, tras la lectura en IF, para modificar el valor del PC. Una manera de agilizar el proceso es poner una unidad de salto incondicional en la misma etapa ID, que corresponde al primer instante en que se conoce si la instrucción corresponde o no a un salto, con lo que se podría reducir la espera a 2 ciclos. Estos siempre serán saltos tomados correctamente, mientras que un approach similar se podría tomar para la ejecución especulativa en jumps condicionales, en que se puede ubicar también la unidad de salto en ID y una unidad de revisión, corrección y flush (como la actual) luego de conocer el resultado de la operación tras pasar por EX.*

Feedback: es innecesariamente largo ☹, se ve como una respuesta poco decidida en la que simplemente pones mucha información, pero entre toda esa información está la respuesta. Intenta ser más preciso y ordenado, que se note un hilo conductor. Cuesta mucho corregir las respuestas de este estilo ya que hay que ponerse a buscar si respondes o no.

Algunas respuestas de ustedes…

- Tiene que ver principalmente con la arquitectura necesaria para poder ejecutar saltos condicionales con paralelismo a nivel de instrucción.
  Como el computador mostrado en esta sección de los apuntes ejecuta las comparaciones necesarias para un salto condicional en la misma instrucción en la que se pide el salto, la Jump Unit no recibe la señal de confirmar un salto condicional y la información para aprobarlo o no hasta 3 ciclos después pues esta no exista antes. Por lo que si se ejecuta un salto incondicional el computador no está construido para procesarlo inmediatamente. Si no que la señal necesaria para que se active la Jump Unit tiene que pasar por 3 ciclos antes de llegar a este.
  Si ignoramos restricciones de la arquitectura más técnica nos podemos encontrar con problemas en el orden en el que ocurren las cosas, es decir que si el salto incondicional se ejecuta en ID, lo más rápido si mantenemos que la Control Unit maneja señales, entonces si 3 ciclos atrás se empezó un salto condicional que si se aprueba y el ciclo anterior se lee de Memoria una instrucción de salto incondicional, el computador va a recibir dos señales y direcciones distintas de salto al mismo tiempo. que no está preparado para manejar a menos que agreguemos una suerte de Mux que priorice las señales de salto de más adelante. Pero esto agregaría tiempo a la ejecución de todas las señales en esta primera etapa y por tanto el ciclo tendría que durar más.
- En cambio si se leen una instrucción de salto condicional, que va a ser exitosa, seguida de una incondicional esto puede ser manejado por el computador como existe actualmente.
- Pero si ocurre que se ejecuta un salto incondicional en ID sin otra interrupción, inevitablemente esto significa que se leyó la instrucción que seguía a el salto en IF, por lo que habría que agregar otra Control Hazard Unit que haga flush en IF/ID en este caso.
- Con todos estos cambios a la arquitectura y al funcionamiento del computador este ya no es el mismo computador original. Por lo que me parece razonable decir que el computador no es capaz de decidir saltar inmediatamente.

Feedback: tu respuesta es muy imprecisa. Estás dejando demasiadas cosas que se pueden interpretar de distintas formas por lo que en una prueba no te podría poner puntaje al no saber si lo que yo entiendo es lo que querías decir o no. Tienes cosas que aparentan ser errores conceptuales graves, estos no se dejan pasar. Te terminas desviando completamente de lo que se pregunta. Es "¿por qué no puedo?", no "¿por qué no podría?".

Algunas respuestas de ustedes...

- No puede realizar los saltos inmediatamente ya que la Jump Unit se encuentra después de la ALU en el sector de memoria, es decir tres ciclos después de salida de la Instruction Memory. Lo anterior hace que aunque se trate de adivinar el resultado el cambio de la dirección del PC solo ocurriría cuando ya se tiene este. Para resolver este problema se podría "adelantar" la unidad de salto al nivel Instruction Decode, dejando en Memory una unidad que compruebe que se haya hecho bien el salto y arreglarlo en caso de error.

Feedback: ¿Por qué? Te falta especificidad respecto a la jump unit y por qué nos compete. La Data Hazard Unit sólo se encarga de un tipo de Hazard de datos. El nombre no siempre es tan fiel representante de nada.