DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
ESCUELA DE INGENIERÍA
PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

IIC2343 - Arquitectura de Computadores (I/2020)

Examen

Fecha: 9 de julio, 2020.

Instrucciones

I. Esta evaluación es estrictamente **INDIVIDUAL**.

II. Lee el enunciado completo de manera atenta y detallada.

III. Incluye tu procedimiento ordenado. Cualquier respuesta sin desarrollo o no explicada no recibirá puntaje.

IV. Referencia de manera clara y ordenada en APA[1] en caso de utilizar material que esté público en internet.

V. Pruebas de otros semestres no son una fuente válida que referenciar.

VI. Para preguntar dudas utiliza las issues del Syllabus en GitHub. Si darías la respuesta o pistas para ella, escríbenos a `iic2343.puc@gmail.com`. Las issues serán consideradas un canal de comunicación oficial y deberás tener en cuenta lo que allí se indique, tal como lo que se anota en la pizarra en una evaluación presencial.

Entrega

La entrega será mediante este formulario de google.

- Tienes hasta las 15:00 horas del día 9 de julio de 2020 para entregar. A partir de las 12:00 sólo se contestarán dudas administrativas.
- Entrega un documento PDF con tus respuestas **para cada pregunta** de manera independiente. En el caso de la pregunta del código de honor, puedes entregar también un archivo en formato de imagen.
- Es preferible que tus respuestas estén escritas en digital, para evitar que el ayudante no entienda tu letra. Queda bajo tú responsabilidad el decidir hacerlo a mano.

[1] O en algún otro formato estándar.

Política de Integridad Académica

Los alumnos de la Escuela de Ingeniería deben mantener un comportamiento acorde al Código de Honor de la Universidad:

"Como miembro de la comunidad de la Pontificia Universidad Católica de Chile me comprometo a respetar los principios y normativas que la rigen. Asimismo, prometo actuar con rectitud y honestidad en las relaciones con los demás integrantes de la comunidad y en la realización de todo trabajo, particularmente en aquellas actividades vinculadas a la docencia, el aprendizaje y la creación, difusión y transferencia del conocimiento. Además, velaré por la integridad de las personas y cuidaré los bienes de la Universidad."

En particular, se espera que mantengan altos estándares de honestidad académica. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un procedimiento sumario. Específicamente, para los cursos del Departamento de Ciencia de la Computación, rige obligatoriamente la siguiente política de integridad académica. Todo trabajo presentado por un alumno (grupo) para los efectos de la evaluación de un curso debe ser hecho individualmente por el alumno (grupo), sin apoyo en material de terceros. Por "trabajo" se entiende en general las interrogaciones escritas, las tareas de programación u otras, los trabajos de laboratorio, los proyectos, el examen, entre otros. Si un alumno (grupo) copia un trabajo, los antecedentes serán enviados a la Dirección de Docencia de la Escuela de Ingeniería para evaluar posteriores sanciones en conjunto con la Universidad, las que pueden incluir reprobación del curso y un procedimiento sumario. Por "copia" se entiende incluir en el trabajo presentado como propio partes hechas por otra persona. Está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, siempre y cuando se incluya la cita correspondiente.

Código de honor

Escribe **a mano en un papel**[2] el siguiente texto completo y fírmalo. A continuación tómale una foto o escanéalo y súbelo al formulario. **No hacerlo equivale a un 1 en todas las preguntas de la tarea.**

> Examen - IIC2343
>
> Yo, <tu nombre>, afirmo que respetaré el código de honor de la universidad y no cometeré ninguna falta a la ética ni a la política de integridad académica.
>
> <número de alumno>
> <firma>

[2]Si te preocupas mucho por el medio ambiente, puedes escribirlo a mano en digital.

1. Computador básico

Refiriéndonos únicamente al computador básico de las clases con soporte para subrutinas, sin pipeline ni I/O (el del capítulo 5.5 de los apuntes), que es de 8 bits, funciona con números enteros y tiene una instrucción para saltar si detecta *overflow*.

a) Dale al computador básico la capacidad para realizar la instrucción `MUL A,B`. No puedes utilizar la ALU ni modificarla. Utiliza registros auxiliares distintos de A y de B para hacer tus cálculos intermedios de ser necesario. Explica cómo funciona tu circuito.

Ten cuidado con el manejo de los ciclos, en caso de que tu solución requiera varios para funcionar. Tienes la Control Unit estándar del computador básico, si quieres más funcionalidades debes implementarlas tú de forma separada. **Puedes describir las modificaciones con palabras en lugar de hacer el diagrama, pero debes ser especifico en tus descripciones.**

b) Diseña el circuito que detecta el *overflow* en la ALU y tiene como salida el valor de la señal V. Utiliza únicamente compuertas lógicas. Puedes tomar como entrada el valor del resultado de la operación que se realice.

Lo esperado... **UN** circuito correctamente dibujado, de forma clara y ordenada. Usa nodos y/o saltos.

2. Caché

a) ¿Cuál es la ventaja de *N-Way associative* frente a *directly mapped* que aumenta el Hit Rate?

b) Dada una memoria caché *fully associative* de 4 bloques y 2 palabras cada uno. Nuestra memoria principal es de 32 bytes. Determina qué política de reemplazo se está utilizando (tus opciones son: FIFO, LFU, LRU y random). Considera los siguientes accesos a memoria:

0, 1, 2, 3, 4, 14, 15, 28, 29, 4, 15, 5, 4, 3, 2, 1.

Y el estado de la caché tras cada acceso:

N° Fila	Dirección	Binario	Bloque 0		Bloque 1		Bloque 2		Bloque 3		
1	0	00000	0	1	-	-	-	-	-	-	Miss
2	1	00001	0	1	-	-	-	-	-	-	Hit
3	2	00010	0	1	2	3	-	-	-	-	Miss
4	3	00011	0	1	2	3	-	-	-	-	Hit
5	4	00100	0	1	2	3	4	5	-	-	Miss
6	14	01110	0	1	2	3	4	5	14	15	Miss
7	15	01111	0	1	2	3	4	5	14	15	Hit
8	28	11100	0	1	2	3	28	29	14	15	Miss
9	29	11101	0	1	2	3	28	29	14	15	Hit
10	4	00100	0	1	2	3	28	29	4	5	Miss
11	15	01111	0	1	2	3	14	15	4	5	Miss
12	5	00101	0	1	2	3	14	15	4	5	Hit
13	4	00100	0	1	2	3	14	15	4	5	Hit
14	3	00011	0	1	2	3	14	15	4	5	Hit
15	2	00010	0	1	2	3	14	15	4	5	Hit
16	1	00001	0	1	2	3	14	15	4	5	Hit

Puedes consultar la tabla en este link.

3. Pipelining

a) ¿Por qué no tenemos *hazards* estructurales en el computador básico del curso? ¿Y por qué tenemos que esperar a la etapa MEM para resolver los hazards de control?

b) ¿Cómo el pipelining nos ayuda a aumentar la *performance*?

4. Pregunta extra

Si no quieres responder alguna (sólo una) de las 3 preguntas de arriba, entonces puedes dejarla en blanco y responder esta en su lugar. Si contestas las 3 de arriba, esta no se revisará (independientemente del resultado de las de arriba).

a) **[Números]** Transforma el número 10645_7 a base 14. Incluye tu procedimiento.

b) **[I/O]** Un amigo te dice que está haciendo un juego en el que cambia la luz según la hora del día (DNS, que significa *day-night system*), haciendo que se vea más oscuro durante la noche y más claro durante la mañana. Hasta ahora había tenido un rendimiento impecable, siendo muy eficiente con el uso de la memoria, usando los mejores algoritmos y optimizando sus procedimientos para que la memoria caché lo ayude lo más posible, sin embargo luego de implementar el DNS el rendimiento del juego cayó enormemente. Sabe que el problema no está ni tiene que ver con la tarjeta gráfica. Da una posible explicación para que el rendimiento haya caído tanto e indica una posible solución para el juego de tu amigo.

Código del bucle del juego (es un ejemplo ilustrativo):

```
1  int main() {
2      GameState game_state = new_game();              // el estado del juego, contiene todos los objetos del juego
3      int close_game = 0;                             // una variable de control
4      while (!close_game) {                           // el "game loop"
5          read_inputs(&game_state);                   // leo los dispositivos I/O
6          close_game = update_state(&game_state);     // actualizo el estado de mi juego
7          write_outputs(&game_state);                 // escribo en los dispositivos I/O
8      }
9      return 0;
10 }
```