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

IIC2343 - Arquitectura de Computadores (I/2020)

I_3

Arquitecturas y Pipeline.

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.

Instrucciones

Esta interrogación es de carácter **INDIVIDUAL**. Consta de 3 preguntas que forman parte de las 9 con las que se calcula el promedio de interrogaciones al final del semestre. Para entregar deberás enviar mediante un formulario de google:

- Un archivo en formato `pdf` por cada pregunta, con tus respuestas a la pregunta correspondiente.
- En el caso de la pregunta del código de honor, puede ser también en formato imagen.
- En el caso de la pregunta de Pipelining, puede ser también una hoja de cálculo (Excel). Si usas *google sheet* o alguna otra aplicación web, descárgalo y súbelo al *form*, no compartas el *link* (no se te considerará en caso de hacerlo).

Debes incluir todo tu procedimiento y/o desarrollo, así como todas tus fuentes en caso de que utilices material externo.[1]

Asegúrate de que tus respuestas no queden públicas en internet.

Si tienes una duda que puede dar pistas o la solución, mándalas a: `jmwielandt@uc.cl`

Importante a la hora de responder esta interrogación

Esta interrogación busca medir tu aprendizaje de los contenidos a través de distintas preguntas. No intentes basarte en pautas anteriores porque las preguntas son distintas y podrías equivocarte y no responder la pregunta correcta.

En cada pregunta hay un pequeño inciso que habla de a lo que apunta cada pregunta para que sepas mejor lo que se espera.

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.**

> Interrogación 3 - 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>

[1] Si tienes dudas con el formato de citación, usa APA.
[2] Si te preocupas mucho por el medio ambiente, puedes escribirlo a mano en digital.

6. Pregunta ISA

Los ayudantes del curso están creando una nueva ISA.[3] Pero no logran ponerse de acuerdo si construirla CISC o RISC, así que necesitan que les respondas algunas preguntas para ayudarles a tomar la decición.

En base a la materia de clases, responde:

1. ¿Podría la ISA RISC ser un subconjunto de las instrucciones de la ISA CISC? ¿Porqué?
   Lo esperado... Debes indicar "sí", "no" o "depende" y explicar los motivos, basándote en las características y propuestas de cada uno de los distintos tipos de ISA.

2. Si tuviéramos un programa escrito con la versión CISC de la ISA y quisiéramos convertir su código assembly a la ISA RISC. ¿Qué consideraciones deberíamos tener al hacer la conversión? Describe el proceso paso a paso.
   Lo esperado... Se espera que resaltes las dificultades a las que nos enfrentaríamos y cómo se resuelven, llegando a una serie de reglas con las que puedas crear tu algoritmo, basándote también en las características y propuestas de cada uno de los tipos de ISA.

7. Pregunta Microarquitecturas

Se te entregará una ISA, en base a ella implementa una micro-arquitectura **Von Neumann** de 8 *bits* que le de soporte. En tu implementación debes tener al menos:

- 4 registros (A, B, C y D)
  - Registros C y D para direccionamiento
- Una unidad aritmética
- Una unidad lógica
- Saltos y subrutinas
  - Se puede saltar a la posición que apunta D en memoria, también llamarla como subrutina.

Lo esperado... Se espera un diagrama claro, completo, legible y sin abstracciones. Utiliza nodos para las bifurcaciones o saltos para los cruces entre cables distintos. **No** uses componentes que no se han visto en clases sin explicar su funcionamiento ni detallar el circuito interno. No necesitas detallar el funcionamiento de la *control unit*. Lo más importante es que la arquitectura que diseñes sea funcional.

7.1. ISA

Condition Codes	Name	Description
o	Odd	Impar
e	Even	Par
c	Carry	Carry
v	Overflow	Overflow
d	Warning	Intentar dividir por 0

[3] Llamada "YadrISA".

	Op1	Op2	Descripción
MOV	R1	R2	Coloca del valor de R2 en R1
SWP	R1	R2	Intercambia los valores de R1 y R2
ADD	R1	R2	Suma R1 con R2
NEG	R1	R2	Multiplica R2 por -1 (inverso aditivo) y lo guarda en R1
MUL	R1	R2	Toma la mitad menos significativa de los bits de cada registro y los multiplica
DIV	R1	R2	Guarda el cociente en R1 y el módulo en R2, si R2 es 0, retorna R1 y R2 normalmente y pone en 1 el condition code d.
NAND	R1	R2	R1 = NOT (R1 AND R2)
NOR	R1	R2	R1 = NOT (R1 OR R2)
SHL	R1	R2	R1 << R2
SHR	R1	R2	R1 >> R2
JMP	LIT		Salto incondicional
JCR	LIT		Salto si hay carry
JOV	LIT		Salto si hay overflow
JOD	LIT		Salto si es impar (odd)
JEV	LIT		Salto si es par (even)
MOV	R1	(C)	
MOV	R1	(D)	
MOV	(C)	R1	
MOV	(D)	R1	
MOV	R1	(LIT)	
MOV	(LIT)	R1	
JMP	D		Usa el valor guardado en el registro D como dirección de salto
JCR	D		Usa el valor guardado en el registro D como dirección de salto
JOV	D		Usa el valor guardado en el registro D como dirección de salto
JOD	D		Usa el valor guardado en el registro D como dirección de salto
JEV	D		Usa el valor guardado en el registro D como dirección de salto
CALL	LIT		Usa el valor LIT como dirección de la subrutina
CALL	D		Usa el valor guardado en el registro d como dirección de la subrutina
RET			
PUSH	A		Guarda el valor del registro A en la posición del stack apuntada por SP
PUSH	B		Guarda el valor del registro B en la posición del stack apuntada por SP
POP	A		Lee el valor del stack en la posición apuntada por SP y la guarda en el registro A
POP	B		Lee el valor del stack en la posición apuntada por SP y la guarda en el registro B
INC	SP		Incrementa el stack pointer
DEC	SP		Decrementa el stack pointer

8. Pregunta Pipelining

8.1. Parte A

Simula la ejecución del siguiente código assembly (a mano) e indica, por cada instrucción, cuando ocurren hazards; incluye la dependencia, la resolución y la respectiva forwaring unit asociada, stalling y flushing. Ante saltos **condicionales**, asume que siempre saltaremos. **Esto es sólo para la sección CODE, no incluyas la sección DATA.**

```
DATA:
    dividendo   0
    divisor     0
    cociente    0
    resto       0

CODE:
JMP main
div:                                                    // división sin signo
    MOV (dividendo),A
    MOV (divisor),B
    while_dividendo_mayor_o_igual_que_divisor:
        MOV A,(dividendo)
        MOV B,(divisor)
        CMP A,B
        JLT end_while
        SUB A,B
        MOV (dividendo),A
        MOV A,(cociente)
        ADD A,1
        MOV (cociente),A
        JMP while_dividendo_mayor_o_igual_que_divisor
    end_while:
        MOV (resto),A
        MOV A,(cociente)
        MOV B,(resto)
    JMP end
main:
    MOV A,10
    MOV B,6
    JMP div    // calcula A / B
end:
```

Importante: en lugar de representar las dependencias y su resolución con flechas (como se hace en las slides), puedes indicarlas de manera clara y ordenada **con texto** a un costado por cada instrucción.

Lo esperado... Esta pregunta apunta a que, siendo capaces de entender el assembly del computador básico, puedan detectar los hazards y resolverlos correctamente.

8.2. Parte B

Tenemos un computador Von Neumann con pipeline que tiene las siguientes etapas: IF, ID, MEM, EX y WB, en ese orden. Recuerda que en un computador Von Neumann tenemos una única memoria, que puede entregar un dato o (*exclusive or*) una instrucción en un mismo acceso a ella.

¿Qué tipos de hazards pueden ocurrir? Muéstralo con el diagrama de etapas del pipeline, indicando las dependencias y cómo se resuelven.

Lo esperado... En lugar de mostrar todas las posibles combinaciones y cómo resolverlas, esperamos que muestres los casos de manera aislada y que sean representativos (tal y como se hace en las clases).