Examen Virtual de Arquitectura Del Computador Modalidad Manuscrito

Por favor firmar y aclarar todas las hojas y Se les recuerda a los estudiantes que, según la resolución RD-2020-197-E-UNC-DEC#FAMAF, en el examen en la modalidad manuscrito, el/la estudiante, deberá firmar todas las hojas de su examen antes de digitalizarlo y enviarlo para su corrección. Al final del mismo deberá introducir la leyenda "Por la presente declaro que la resolución de este examen es obra de mi exclusiva autoría y respetando las pautas y criterios fijados en los enunciados. Asimismo declaro conocer el régimen de infracción de los estudiantes cuyo texto ordenado se encuentra en el apéndice de la Res. Rec. 1554/2018", con una foto de su Documento Nacional de Identidad, ocultando su número de trámite, en carácter de Declaración Jurada.

Enviar por email apenas finalizado el examen a pablo.ferreyra@unc.edu.ar

Ejercicio 1: (Memoria Cache) (2 puntos) (Por favor justificar cada paso o figura mediante texto explicativo. Se evaluará fuertemente la claridad de la explicación)

La memoria principal de un sistema computacional está organizada en 64 bloques, con un tamaño de 8 palabras por bloque. La CACHE posee 8 líneas. En los apartados a) al c) se especifica el criterio de correspondencia para cada caso. Suponga que cada palabra del procesador es directamente direccionable en memoria principal.

a) Dibuje la organización de la CACHE de correspondencia DIRECTA, indicando todas sus dimensiones. Además indicar cuántos bits se utilizan en la identificación de cada campo (etiqueta, línea y palabra) en una dirección de memoria principal.
b) Repita el punto anterior para correspondencia FULL-ASOCIATIVA
c) Repita el punto a) para una CACHE de correspondencia ASOCIATIVA POR CONJUNTOS de 4 vías. Es posible?

Ejercicio 2 (Multiple Issue) (2 puntos): (Por favor justificar cada paso o figura mediante texto explicativo. Se evaluará fuertemente la claridad de la explicación)

Un procesador 2-issue de arquitectura LEGv8 posee las siguientes propiedades:

1. En cada *issue packet* una instrucción debe ser una operación de acceso a memoria y la otra de tipo aritmética/lógica o un salto.
2. El procesador tiene todos los caminos de forwarding posibles entre las etapas (incluyendo caminos a la etapa ID para la resolución de saltos).
3. El procesador predice los saltos perfectamente.
4. El compilador asume toda la responsabilidad de eliminar los hazard, organizar el código e insertar instrucciones "nop" para que el código se ejecute sin necesidad de generación de stalls.

Para el siguiente fragmento de código LEGv8:

```
        1>    ADD x3, xzr, xzr
loop:   2>    LDUR x0, [x2, #8]
        3>    ADD x9, x0, x5
        4>    ORR x10, x1, x9
        5>    AND x11, x9, x0
        6>    SUB x12, x0, x11
        7>    STUR x9, [x3, #24]
        8>    STUR x10, [x3, #16]
        9>    STUR x11, [x3, #8]
        10>    STUR x12, [x3, #0]
        11>    ADDI x3, x3, #32
        12>    ADDI x2, x2, x3
        13>    ADDI x13, xzr, #1
        14>    SUBI x14, x13, #8
```

```
15>    CBZ x14, loop
...
```

a. Analice en el código las dependencias de datos y determine cuales generan data hazards. En caso de detectar un hazard indique el operando en conflicto y los números de las instrucciones involucradas. Suponga que los saltos están perfectamente predichos, de modo que los *control hazard* son manejados por hardware.

b. Dibuje un diagrama de pipeline que muestre cómo se ejecuta el código LEGv8 dado en el procesador de 2-issue (sólo hasta completar una iteración del bucle). Sin modificar el orden de ejecución, organice el código para evitar la mayor cantidad posible de *stalls*. Deje indicados los caminos de *forwarding* utilizados.

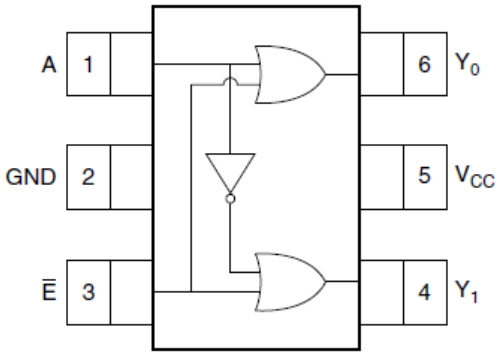
c. Suponga que el compilador es capaz de determinar que la cantidad de iteraciones del bucle se da en múltiplos de 2. Utilice la técnica estática "*loop unrolling*" para re-ordenar la ejecución del código. Está permitido utilizar otros registros si se considera necesario.

d. Determine mejora en el tiempo de ejecución del código resultante del punto (c) respecto al del punto (b) considerando la ejecución completa del fragmento de código dado.

Ejercicio 3 (System Verilog) (2 puntos): (Por favor justificar cada paso o figura mediante texto explicativo. Se evaluará fuertemente la claridad de la explicación)

1. Todos los programas en System Verilog que pueden ser simulados, también puede ser sintetizados? Si no, explique por qué.

2. La figura siguiente contiene la asignación de pines y el diagrama lógico del Fairchild NC7SP19 1-of-2 Decoder/Demultiplexer IC.



(a) Escriba la declaración de un módulo llamado nc7sp19. Etiquete las entradas y salidas con los mismos nombres que el diagrama. Ignore los número de pin y los pines de alimentación.

(b) Escriba el archivo completo del diseño para la entidad nc7sp19 que incluya una descripción estructural del mas alto nivel (top-level). Use un componente OR denominada or_2 y un componente not_1 para el inversor. Nombrar los componentes en el diagrama de U1 a U3.

Ejercicio 4 (Tomasulo) (4 puntos): (Por favor justificar cada paso o figura mediante texto explicativo. Se evaluará fuertemente la claridad de la explicación)

Considerando un microprocesador out-of-order execution implementado mediante el algoritmo de Tomasulo, muestre el contenido de las tablas de *Status,* las *Reservation stations* y el flujo de ejecución para la siguiente secuencia de código justo después del 4to ciclo de clk.

Instruction	Instruction status		
	Issue	Execute	Write result
1> addi X0, X0, #100	☐	☐	☐
2> ldur D0, [X1, #0]	☐	☐	☐
3> ldur D1, [X1, #0]	☐	☐	☐
4> fadd D1, D1, D0	☐	☐	☐
5> faddi D0, D0, #23	☐	☐	☐
6> fmuld D4, D2, D3	☐	☐	☐
7> lsr X2, X0, #4	☐	☐	☐
8> div X4, X4, X5	☐	☐	☐
9> fdiv D2, D3, D4	☐	☐	☐
10> ldur D4, [X1, #32]	☐	☐	☐
11> fadd D0, D1, D2	☐	☐	☐
12> add X0, X4, X2	☐	☐	☐

Harware
Issue = 4 instrucciones
Load = 4 RS / 1 clock cycle
Store = 4 RS / 1 clock cycle
Suma entera = 1 FU - 2 RS / 1 clock cycles
Multiplicación entera = 1 FU - 2 RS / 1 clock cycles
Suma flotante = 1 FU - 2 RS / 1 clock cycles
Multiplicación flotante = 1 FU - 2 RS / 2 clock cycles

Clk	Stage				
	IF	ID	IS	Ex	WB
0					
1					
2					
3					
4					

Name	Reservation stations						
	Busy	Op (Inst)	Vj	Vk	Qj	Qk	A
Load 1	☐						
Load 2	☐						
Load 3	☐						
Load 4	☐						
Store 1	☐						
Store 2	☐						
Store 3	☐						
Store 4	☐						
Add int 1	☐						
Add int 2	☐						
Mult int 1	☐						
Mult int 2	☐						
Add FP 1	☐						
Add FP 2	☐						
Mult FP 1	☐						
Mult FP 2	☐						

Register Status									
	D0	D1	D2	D3	D4	D5	D6	D7	D8
Qi									
	X0	X1	X2	X3	X4	X5	X6	X7	X8
Qi									