Universidad de Costa Rica
Escuela de Ciencias de la Computación e Informática

CI-0120 Arquitectura de computadores I-2021

Segundo examen parcial

Prof. Francisco Arroyo

Viernes 23 de Julio del 2021

Observaciones generales

i. La solución debe ser **presentada** como máximo el día **Viernes 2021/Jul/30** a las 11 p.m.
ii. La solución del examen puede ser presentada por grupos de dos personas o de manera individual
iii. Las soluciones deben seguir las buenas prácticas estudiadas
iv. Escriba sus suposiciones
v. Debe entregar en un archivo compactado todos los elementos de su solución, puede entregar un archivo "logisim" distinto para cada una de las respuestas. Cada respuesta debe tener todos los elementos para realizar las pruebas solicitadas. Favor colocar el número de carnet en los nombres de los archivos que van a entregar (carnet1-carnet2.zip o carnet1-carnet2-CPU.circ, etc.)
vi. Debe entregar su solución en "*Mediación Virtual"* y guardar una versión en su repositorio de control de versiones

Preguntas

A) (40 pts.) Completar su CPU desarrollada durante el curso con una **unidad de "forwarding"**.

- Tomando como guía el material revisado en clase, agregue a su CPU los componentes necesarios para controlar los "**data hazards**" por medio de la técnica de 'adelantamiento' de datos (casos 1a, 1b, 2a y 2b) o burbujas en los casos que sea necesario

- (5 pts.) Determine las entradas y las salidas necesarias para completar su tarea (**diseño**, en un documento aparte). Cada entrada o salida debe tener un pin y un tunel en esta sección, tal y como lo hemos presentado en varios circuitos durante las clases. Además haga un listado con todos los componentes que considere necesario agregar a su CPU. Explique cómo deben correrse las pruebas

- (20 pts.) Construya un circuito independiente que represente su "Forwarding Unit", agregue una sección para enseñar las entradas y salidas de este componente. Debe modificar su CPU para que contenga todos los elementos necesarios para resolver las pruebas indicadas. Los registros deben tener los valores pre-cargados y las instrucciones y los datos en las memorias respectivas. Los registros 1, 2, 3, 4, 5, 6 contiene un diez en decimal, puede agregarlos de manera manual en su "Register File". En la memoria de datos en la posición 32 tiene que colocar un **23**

- (15 pts.) Su solución tiene que ser capaz de ejecutar las siguientes instrucciones MIPS sin producir datos erróneos, los resultados correctos son mostrados en los comentarios

  - (5 pts.) Primera prueba ("dependencias de datos" sin insertar burbujas), debe entregar un archivo "logisim" aparte con estas instrucciones cargadas en su memoria y las modificaciones indicadas, tanto en registros como en memoria de datos
    - `add $2, $1, $1    // r2 = 20`
    - `add $3, $5, $2    // r3 = 30`
    - `add $4, $2, $6    // r4 = 30`
    - `add $5, $4, $4    // r5 = 60`
    - `sw  $5, 100($4)   // Guarda 60 en la posición 130 de la memoria`
    - `Ensambladas: 00211020, 00a21820, 00462020, 00842020, ac850080`

- (7.5 pts.) Segunda prueba (“dependencias de datos” con burbujas), debe entregar un archivo “logisim” aparte con estas instrucciones cargadas en su memoria y y las modificaciones indicadas, tanto en registros como en memoria de datos
  - `lw $2, 22($1)     // se carga de la posición de memoria 22 + 10 (tiene un 23)`
  - `add $4, $2, $3    // r4 = 23 + 10`
  - `add $5, $2, $3    // r5 = 23 + 10`
  - `add $6, $4, $2    // r6 = 33 + 23`
  - `slt $1, $4, $2    // r4 = 33, r2 = 23, r1 = 0`
- `Ensambladas: deben ensamblar estas instrucciones (2.5 pts.)`

- En el circuito entregado debe estar **completo**, todos los elementos necesarios para correr las pruebas deben estar contenidos en la solución entregada, las instrucciones indicadas deben ejecutarse correctamente, los valores de los registros tiene que ser correctos luego de la ejecución

B) (40 pts.) Construir en “logisim” una memoria para **datos** que permita la lectura y escritura de bytes, medias palabras y palabras en su CPU.

- Por limitaciones de "logisim" la dirección de 32 bits debe ser convertida a 12 bits (10 bits para dirección y 2 para offset)
- Debe soportar las instrucciones: LB, LH, LW, SB, SH, SW (LW y SW ya deben estar completadas)
- Deben utilizar cuatro unidades de memoria (RAM) de 8 bits para los datos y 10 bits para las direcciones
- Tanto la entrada como salida de datos de la memoria deben ser de 32 bits
- Si la lectura es de 8 o 16 bits, serán los menos significativos de esta salida
  - Las escrituras deben alinearse en la memoria de acuerdo con su dirección
  - Los datos de 32 bits solo pueden escribirse en direcciones que sean divisibles por 4 (0, 4, 8, etc.)
  - Los datos de 16 bits (half word) solo pueden escribirse en direcciones que sean divisibles por 2 (0, 2, 4, 6, etc.)
  - Los datos de 8 bits (bytes) pueden escribirse en cualquier dirección de memoria
- Debe tener dos entradas que indiquen si estamos haciendo una lectura o una escritura (*MemWrite* y *MemRead*)
- Pueden agregar **una** entrada (de control) para establecer el tipo de acceso que se desea realizar
- (5 pts.) Determine las entradas y las salidas necesarias para completar su tarea (**diseño**, en un documento aparte). Cada entrada o salida debe tener un pin y un tunel en esta sección, tal y como lo hemos presentado en varios circuitos durante las clases. Además haga un listado con todos los componentes que considere necesario agregar
- (20 pts.) Dentro de su CPU construya un circuito “Data Memory” independiente para resolver este problema de manera **correcta**, con las condiciones expuestas en este enunciado. Agregue una sección para diferenciar las entradas y salidas, como lo hemos expuestos durante las clases
- (10 pts.) Debe modificar su CPU para que pueda ejecutar la instrucciones indicadas y comprobar que su memoria funciona correctamente
- (5 pts.) Debe crear una prueba con las instrucciones indicadas, ensamblarla, colocarla en la memoria de instrucciones y ejecutarla. Utilice valores significativos (distintos de cero) que demuestren la correctitud de su solución

C) (20 pts.) Escoger **una** de las dos siguientes preguntas:

1. Completar su CPU desarrollada durante el curso con una **unidad de "hazard detection"**
   - Tomando como guía el material revisado en clase agregue a su CPU la funcionalidad para controlar los "control hazards"
   - Debe realizar las modificaciones a su CPU para lograr rebajar la pérdida de instrucciones de tres a una instrucción en el caso de que el "branch" deba ser tomado
   - Su solución debe tener todos los elementos necesarios para controlar la CPU e insertar las burbujas o "stall" en el ciclo de ejecución cuando es necesario
   - (5 pts.) Debe agregar un ejemplo para demostrar que su modificación funciona correctamente, puede utilizar como ejemplo las instrucciones propuestas en las filminas (#49)

2. Implantar en en su CPU el uso de los registros especiales **LO** y **HI**
   - Su nueva versión debe ser capaz de ejecutar las siguientes instrucciones que utilizan los registros LO y HI: MFHI, MFLO, MTHI, MTLO, MULT, DIV
   - Estos registros deben formar parte de su "Register File"
   - Debe realizar los cambios necesarios a su unidad de control para generar las señales a su CPU
   - En su "Register File" puede agregar una entrada (de control) para indicar el uso de estos registros
   - (5 pts.) Debe agregar un ejemplo con las instrucciones solicitadas y ensambladas para demostrar que su modificación funciona correctamente