Universidad de Santiago de Chile
Facultad de Ingeniería
Departamento de Ingeniería Informática

Laboratorio 3

Organización de Computadores
Primer Semestre 2019

Profesor: Leo Medina
Ayudantes: Alexander Palma
Sebastián Pinto-Agüero
Ariel Undurraga

1. Objetivos

En la organización de computadores, el *pipeline* es una técnica que implementa una forma de paralelismo a nivel de instrucciones mediante la separación del *hardware* en varias etapas de procesamiento de las instrucciones.

Al incorporar múltiples etapas en un *pipeline* de un procesador, surgen potenciales problemas con las dependencias de los datos, el flujo de instrucciones, o con la estructura misma del procesador. Estos problemas son llamados *hazards* o riesgos de datos, de control y estructurales, respectivamente.

El objetivo de este laboratorio es construir un sencillo simulador de MIPS para detectar riesgos, simulando el *pipeline* en programas simples. El simulador debe leer un programa en MIPS (código fuente) e indicar en qué momentos se producen los riesgos de datos y de control a lo largo de su ejecución.

2. Procedimiento

Usted debe desarrollar un programa en el lenguaje de programación C, que contenga los siguientes puntos como funcionalidad:

- Debe simular un procesador MIPS con *pipeline* de 6 etapas, idéntico al procesador con *pipeline* de 5 etapas discutido en clases, pero en el que la etapa EX (ejecución) ha sido dividida en dos: una donde opera la ALU que recibe como operandos los datos desde el archivo de registros, y otra donde opera el sumador para obtener la dirección de destino de un operación de bifurcación.
- Debe leer un archivo conteniendo un programa MIPS a ser ejecutado en este procesador.
- Debe detectar riesgos de datos y de control, indicando la línea del programa y el ciclo de reloj en que ocurrieron.
- Debe tener la capacidad de solucionar los riesgos detectados en plena ejecución incorporando esperas o "burbujas" en el *pipeline*. Las burbujas debe implementarlas de la forma discutida en clases. La solución de cada uno de los riesgos debe ser registrada en un archivo de salida, y en este archivo se deben incluir los riesgos de control que no requirieron una acción correctiva.
- Debe mostrar el valor almacenado por todos los registros al final de la ejecución del programa (*$sp* lo puede considerar como un arreglo en su implementación).

Además, asuma lo siguiente:

- Esquema de predicción estático de las bifurcaciones: *branch not taken* o bifurcación no tomada.
- El procesador NO tiene adelantamiento y los riesgos se solucionan solamente mediante burbujas.

3. Consideraciones

Su simulador debe tener la capacidad de procesar las siguientes instrucciones:

- add
- sub
- addi
- subi
- beq
- bne
- j
- lw
- sw

Su programa debe entregar en la salida el valor final de los registros *$zero*, *$v[0-1]*, *$a[0-3]*, *$t[0-9]*, *$s[0-7]*, *$k[0-1]*, *$sp* y *$ra*, por lo cual dentro del programa ingresado como entrada puede encontrarse cualquiera de dichos registros. Para la inicialización del programa, debe asumir que todos los registros, a excepción de *$sp,* parten de 0 (almacenan un 0), y que el *stack* en memoria se encuentra vacío. Cualquier otra consideración se presentará en las respectivas clases de laboratorio y a través del Classroom del curso.

A continuación se indican los 3 archivos que deberá entregar su programa junto con ejemplos del contenido de estos archivos.

Archivo RIESGOS_DETECTADOS.txt:

1- Riesgo de datos línea de instrucción 4, CC 6.
2- Riesgo de control línea de instrucción 10, CC 20.

Archivo SOLUCION_RIESGOS.txt

1- Solucionable a través de 2 NOPs (burbujas).
2- Bifurcación no tomada, no fue necesario agregar burbujas.

Archivo REGISTROS.txt

#No se muestran todos los registros en este ejemplo
v0 = 0
v1 = 3
t0 = 1
..
t6 = 10
..
ra = 0x0035000

s0 = 5
s1 = 1
..
sp = 0x0010000 #Valor con el que consideraron el stack pointer
Contenido SP:
[1,6,2,9,3] #Recordar que stack pointer puede ser considerado como un arreglo

4. Exigencias

- Debe entregar un archivo Makefile junto a su código fuente para su compilación.
- El programa debe permitir al usuario ingresar el nombre del archivo de entrada (programa MIPS).
- El programa debe cumplir con un mínimo de calidad de *software*, incluyendo que las funciones estén separadas y los nombres tanto de funciones como de variables sean de fácil comprensión.
- El informe escrito debe ser entregado en formato PDF y no puede exceder 10 páginas de contenido, sin considerar portada ni índice. En caso contrario, por cada página extra se descontará 5 décimas a la nota final.
- Tanto el código fuente con su Makefile como el informe deben ser enviados por medio de la plataforma Google Classroom en un archivo comprimido, cuyo nombre debe incluir el RUT del alumno (ej.: lab3_12345678-9.zip).

5. Recomendaciones

- Puede usar estructuras de datos para almacenar la información de los registros y señales de control.
- Utilizar la plantilla de LaTeX disponible en el Classroom del curso.
- Consultar a los ayudantes en sus correspondientes clases de laboratorio.
- En caso de dudas o problemas en el desarrollo, asista a la sesión de laboratorio.

6. Descuentos

- Por cada exigencia no cumplida se descontarán dos décimas a la nota, a excepción las que ya mencionan su descuento.
- Por cada día de atraso se descontará un punto a la nota.
- Por cada tres faltas ortográficas o gramaticales en el informe, se descontará una décima a la nota.
- Por cada falta de formato en el informe se descontará una décima a la nota.

7. Evaluación

- La nota del laboratorio será el promedio aritmético del código fuente con el informe, y en caso de obtener una nota inferior a 4 en alguna de las dos calificaciones, se evaluará con la menor nota.
- En caso que no se entregue alguno de los dos, se evaluará con la nota mínima.
- Este laboratorio debe ser entregado el 7 de junio de 2019, hasta las 23:59 hrs. Los descuentos por atraso corren a contar de las 00:01 hrs del día 8 de junio de 2019.