Organización y Arquitectura de Computadoras
2018-1
Práctica 10: Simulación de la ruta de datos de un procesador

Profesor: José Galaviz Casas
Ayudante: Luis Soto Martínez

1. Objetivos

Generales:

- El alumno conocerá el funcionamiento general de una unidad central de proceso.

Particulares:

En el desarrollo de la práctica, el alumno:

- Implementará un simulador de la ruta de datos de un procesador, reforzando los conocimientos teóricos del diseño de conjuntos de instrucciones y la unidad de control.
- Relacionará los conocimientos previos de la programación en lenguaje ensamblador con el lenguaje de alto nivel C.

2. Requisitos

- **Conocimientos previos:**
  - Unidad de control.
  - Diseño de conjuntos de instrucciones.
  - Modos de direccionamiento.
  - Programación en el lenguaje C:
    - Tipos de datos.
    - Estructuras de control.
    - Estructuras y uniones

- Arreglos y apuntadores.
- Entrada y salida.

- **Tiempo de realización sugerido:**
  10 horas.
- **Número de colaboradores:**
  Equipos de 2 integrantes.
- **Software a utilizar:**
  GCC.

3. Planteamiento

En esta práctica se desarrollará un simulador que ejecutará un programa siguiendo la ruta de datos de un procesador DLX.

4. Desarrollo

4.1. Ruta de datos

En la figura 1 se muestra la ruta de datos para la arquitectura DLX, la cual se especificó en la práctica anterior. En las siguientes secciones se da una breve explicación de sus componentes y su funcionamiento, si se requieren mayores especificaciones o información sobre el diseño, se puede consultar [**Patterson**].

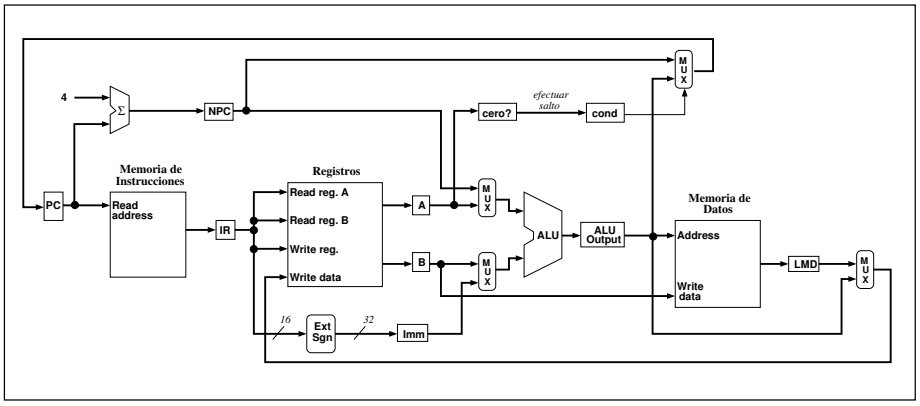

Figura 1: Ruta de datos para la arquitectura DLX.

4.2. Componentes

A continuación se descruben los registros y unidades funcionales de la ruta de datos de la figura 1:

- **PC** (*Program Counter*). Registro con la dirección de memoria de la instrucción a ejecutar.
- **NPC** (*Next Program Counter*). Registro con la dirección de memoria de la siguiente instrucción a ejecutar, es decir, `PC + 4`.
- **ExtSing**. Rellena los bits más significativos del inmediato recibido dependiendo de su signo.
- **Imm**. Registro con el operando inmediato a 32 bits.
- **A** y **B**. Registros temporales con los datos leidos de los registros especificados por la instrucción.
- **cero?**. Evalúa si el valor recibido es igual a cero. Se utiliza para instrucciones de salto.
- **cond**. Registro con el resultado de la evaluación de **cero?**.
- **IR** (*Instruction Register*). Decodifica la instrucción, entregando a las otras unidades funcionales y registros los datos necesarios para operar.
- **ALU** (*Arithmetic-Logic Unit*). Realiza las operaciones artiméticas y lógicas.
- **ALU Output**. Registro con el resultado de la operación realizada por la **ALU**.
- **LMD** (*Load Memory Data*). Registro con el dato leido por la memoria.
- **Banco de registros**. 32 registros de propósito general, el registro 0 siempre es cero, no puede ser modificado.
- **Memoria**. La memoria sigue el modelo de arquitectura Harvard: dividida en una sección para datos y otra para texto (código).

4.3. Unidad de control

Una vez que el `IR` ha decodificado la instrucción, la unidad de control determina las acciones que serán llevadas a cabo por las unidades funcionales, enviando las siguientes señales de control:

1. **RegWrite**. Recibida por el banco de datos.
   - 1 - El banco debe guardar el dato recibido en `Write Data` en el registro especificado en `Write Register`.
   - 0 - Ignora las entradas `Write Data` y `Write Register`.
2. **ALUSrc1**. Recibida por el multiplexor conectado al primer operando de la **ALU**. Determinana la salida del multiplexor.

- 1 - Registro **A**.
- 0 -**NPC**.

3. **ALUSrc2**. Recibida por el multiplexor conectado al segundo operando de la **ALU**. Determina la salida del multiplexor.
   - 1 - **Imm**.
   - 0 - Registro **B**.
4. **ALUCtrl** Recibida por la **ALU**. Determina la operación que será realizada. Los valores dependen de la codificación elegida en la práctica anterior.
5. **PCSrc**. Recibida por el multiplexor conectado a **PC**. Junto con **cond**, determina la salida del multiplexor.
   - 1 - **NPC** o **ALU**, dependiendo del valor de **cond**.
   - 0 - **NPC**.
6. **MemRead**. Recibido por la memoria de datos, determina si la memoria debe entregar el contenido de la dirección indicada en `Address`.
   - 1 - Entrega el dato.
   - 0 - Ignora la lectura.
7. **MemWrite**. Recibido por la memoria de datos, deterina si la memoria debe escribir el contenido de `Write Data` en la dirección `Address`.
   - 1 - Escribe el dato.
   - 0 - Ignora la escritura.
8. **RegSrc**. Recibido por el multiplexor conectado al banco de registros. Determina la salida del multiplexor.
   - 1 - **ALU Output**.
   - 0 - **LMD**.

5. Entrada

El simulador recibirá por medio de la línea de comandos el nombre de un archivo objeto generado por el ensamblador desarrollado en la práctica anterior.

6. Salida

El simulador entregará dos archivos de texto plano, el primero con el contenido de los registros y el segundo el de la memoria. Ambos archivos estarán codificados con caracteres '0' y '1', simulando una codificación binaria, cada línea será del tamaño de una palabra, es decir, de 32 carcateres.

7. Variables libres

El tamaño de la memoria fue definido por el alumno en la práctica anterior.

8. Procedimento

Se debe entregar un archivo de código fuente escrito en el lenguaje de programación C con la solución al ejercicio 3. Además del código fuente del ensamblador de la práctica anterior.

El programa debe estar completamente documentado y cumplir con las convenciones de código.

9. Ejercicios

1. De acuerdo a la codificación de las instrucciones que realizaste en la práctica anterior, define cómo se llevará a cabo la decodificación por parte del **IR**.

2. Analiza cómo deben activarse las señales de control para cada una de las instrucciones.

3. Desarrolla el simulador del procesador siguiendo el flujo de datos, debes simular el funcionamiento de cada una de las unidades funcionales.

10. Preguntas

1. ¿Cuál es la capacidad máxima de memoria que éste procesador puede direccionar bajo este diseño? Justifica tu respuesta.

11. Bibliografía