Organización del Computador II

Gianfranco Zamboni

14 de febrero de 2019

Índice

1. Introducción

La **arquitectura de computadoras** es la ciencia y arte de diseñar, seleccionar e interconectar hardware y diseñar las interfaces hardware/software para crear un sistema computacional que poosea los requerimientos funcionales, de perfomance, consumo (de energía) y de costo (económico) adecuados para realizar determinadas tareas.

Estas tareas son problemas que pasaron por varias transformaciones (desde su descripción en un lenguaje natural hasta convertirse en un programa) y deben ser ejecutadas por una computadora. La tarea del arquitecto consiste en diseñar el **Instruction Set Architecture** (ISA), un conjunto de instrucciones que usarán los programas compilados para decir al micropocesador que hacer. El ISA es implementado por un conjunto de estructuras de hardware conocidas como la **microarquitectura** del procesador.

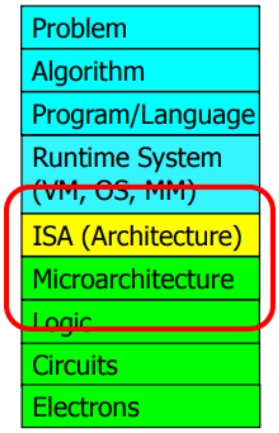

Figura 1: figure
La computadora definida en niveles de abstracción

El ISA y la microarquitectura sientan las bases para el diseño del procesador y, como se dijo anteriormente, un buen diseño debe tener en cuenta los objetivos de sus usuarios y conseguir el balance adecuado de los factores mencionados para llevarlos a cabo de la manera más optima posible. Habrá casos en los que daremos prioridad a un subconjunto de ellos en detrimento de otros (por ejemplo, podriamos elegir mejorar perfomance y aumentar el costo, o quitar performance para mejorar el consumo energético).

1.1. Componentes del ISA

El ISA es la especificación completa de la interfaz entre los programas y el hardware que debe llevar a cabo las operaciones. El ISA especifíca:

1.1.1. Regitros

- Los **registros**, celdas de memoria dentro dentro del cpu que son usados para almacenar temporalmente lo valores necesarios para ejecutar una instrucción. Estos registros son visibles al programa y se clasifican según su uso: Acumuladores, De dirección ó De Proposito General.
- Los mecanismos usados por la computadora para saber donde están almacenados los datos(**Espacio de memoria**),
- La cantidad de bloques univocamente distinguibles en memoria y el tamaño de cada uno de estos bloques (**Direccionamiento**)
- Un conjunto de instrucciones que pueden ser llevadas a cabo por la computadora. Cada instrucción está compuesta por su **opcode** (que se espera que la computadora haga) y sus **operandos** (a que datos debe hacerlo). En una ISA, podremos encontrar tres tipos de instrucciones:

  - De **Operacion**: Procesan datos
  - De **trasnporte de datos**: Transportan información entre la memoria, los registros y los dispositivos de entrada salida.
  - De **control**: Modifican la secuencia de instrucciones a ser ejecutada, es decir permiten ejecutar instrucciones que no están almacenadas en el proximo bloque de memoria.

  Depéndiendo que valores puedan modificar las instrucciones de operación, podremos clasificar las arquitecturas en **Arquitecturas Load/Store** (solo pueden operar en registros) o **Arquitecturas memory/memory** (se pueden modificar los valores directamente en memoria)

- los **Tipos de datos**, es decir que representación deben tener ciertos valores para que puedan ser interpretados por la microarquitectura
- Las formas en las que un operando puede ser accedido (**modos de direccionamiento**). Pueden ser:
  - **Inmediato:** El operando está incluido en la instrucción.
  - **Directo o absoluto:** El operando es la dirección de memoria donde se encuentra el valor a ser utilizado.
  - **Indirecto:** El operando es una dirección de memoria, donde se encuentra la dirección de memoria en la que está almacenado el valor deseado.
  - **De desplazamiento:** La instrucción toma como operandos una dirección de memoria que se toma como **base** y un **offset**, que es un número que indica cuanto hay que desplazar la base para encontrar el valor deseado, es decir $dir = base + offset$
  - **Indexado:** Lo mismo que el anterior, pero con el *offset* guardado en un registro.
  - **De Memoria Indirecta:** El operando es un registro en el que se encuentra guardada la dirección de memoria indirecta.
- Como comunicarse con los dispositivos de entrada/salida (**I/O Interface**), puede ser por medio de instrucciones especiales o mapeos de ciertas regiones memoria para esos dispositivos.
- Quien puede y quien no puede ejecutar certas instrucciones (**Modos de privilegios**)
- Qué debe suceder si una instrucción falla o cuando un dispositivo necesita usar el microprocesador (**Manejo de excepciones e interrupciones**)
- Si soporta o no el uso de **memoria virtual**, es decir, si cada programa tiene la ilusión de estar un espacio de memoria secuencial cuando en realidad el sistema operativo realiza el manejo de la memoria principal

1.1.2. Arquitectura de Von Neumann

Como se vió en Organización del computador I, las mayoría de las ISA usadas actualmente usan el modelo de Von Neumann. Este un ciclo de cinco etapas:

1. **Fecth:** Se utiliza un **program counter** que indica donde está almacenada la proxima instrucción a ser ejecutada.

2. **Decode:** Se decodifica la instrucción fetcheada y se consiguen los operandos (literales y registros) correspondientes.

3. **Execute:** En esta etapa se busca en memoria los datos requeridos (si es necesario) y se procesa los datos acorde a la instrucciónу se almacenan sus resultados en el lugar correspondiente.

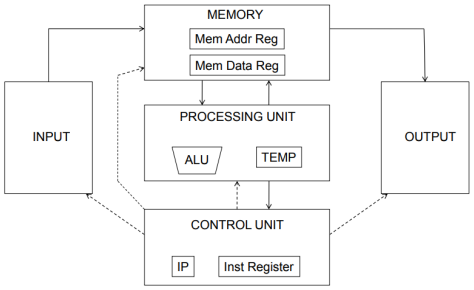

Figura 2: Arquitectura de Von Newmman

Cada instrucción, es extraida de la memoria usando la dirección indicada por el **Instruction Pointer**. La unidad de control se encarga de indicar a la memoria si son necesarios otros valores para poder llevar a cabo la ejecución de la instrucción y luego pasa todo los datos a la unidad de procesamiento.

Sin embargo, la implementación de las ISA (microarquitectura) fue cambiando drásticamente para mejorar el rendimiento de este ciclo.

2. Microarquitectura

La micoarquitectura es la implementación a nivel hardware de la ISA, es decir, es un conjunto de componentes electrónicos organizados de cierta manera para que respeten esas especificaciones. En la sección 1.1.2, vimos como el usuario ve el ciclo de instrucciones. Desde el punto de vista de la implementación (hardware), el ciclo es realizado por unidades de procesamiento que operan sobre los datos de acuerdo a ciertas señales.

Cada instrucción es una señal que usa el procesador de instrucciones para decidir que conjunto de componentes electronicos deben ser activados para poder llevar a cabo la operación deseada. Especificamente, las instrucciones indican:

- **Datapath:** Que elementos deben manejar y transformar los datos (unidades de procesamiento, de almacenamiento y estructuras de hardware que permiten el flujo de datos)
- **Control Logic:** Que elementos de hardware determinan las señales de control que indican al datapath lo que debe hacer con los datos.

En este nivel, ocurren muchas *cosas* que no son visibles para el programador y están destinadas a optimizar el tiempo de ejecución de las instruccions. Estas *cosas* pueden ser algoritmos implementado a nivel electrónico o agregado de componentes que permiten acelerar alguna etapa del ciclo. Algunas de las características encontradas en las microarquitecturas actuales son:

- Pipelining

- Ejecución de múltiple instrucciones simultáneamente.
- Ejecución fuera de orden
- y Cachés de dato e instrucciones separadas, entre otros.

2.1. Medidas de Performance

La escala y complejidad de los sistemas de software modernos, junto con las técnicas usadas por los diseñadores de hardware para mejorar el rendimiento de los dispositivos, ha logrado hacer que el rendimiento pueda depender de varios factores.

2.2. Response time:

A veces, mediremos el rendimiento de una computadora en base a su tiempo de respuesta (**response time** o **execution time**) - el tiempo entre que pasa entre que una tarea empieza y termina -. Este tiempo se mide en segundos por programa y mide el tiempo total que toma completar una tarea, incluyendo accesos a memoria, operaciones del sistema, etc.

En la práctica, la computadora es compartida por varios programas y los microprocesadores deben ejecutar varias tareas simultáneamente. En estos casos, deberemos tener en cuenta que la tarea que la tarea que se está evaluando, no está siendo ejecutada todo el tiempo por lo qué habrá que distinguir el tiempo que el procesador pasa ejecutando la tarea **CPU Time** del tiempo en el que está procesando otros programas.

Entonces, el tiempo de ejecución se define de la siguiente manera:

$$Response\ Time = \frac{Clock\ Cycles\ spent\ on\ task}{Clock\ Rate}$$

Donde *Clock Cycle* son la cantidad de ciclos de reloj (ticks) que se utilizaron en la tarea y *Clock Rate* es el tiempo que dura cada tick del reloj.

Referencias

[1] *18-447 Introduction to Computer Architecture - Spring 2015 Course of Carnegie Mellon University*, 2015.

[2] Yale N. Patt. Requirements , bottlenecks , and good fortune : Agents for microprocessor evolution. 2001.

[3] Sanjay Patel Yale Patt. *Introduction to Computing Systems: From bits and gates to C and beyond.* McGraw-Hill, 2nd international edition, 2005.