Hardware vs Software

Arquitectura de software o Instruction Set Architecture (ISA): Contiene todos los aspectos de diseño visibles para un desarrollador de software. Sería la interfaz que conecta el software con el hardware.

Arquitectura de Hardware o Microarquitectura: Se refiere a una implementación de la ISA. Cantidad de núcleos, frecuencia, instrucciones.

Esta separación entre arquitectura y microarquitectura da garantías de compatibilidad, es decir que software viejo puede correrse en hardware nuevo, también, permite actualizar el hardware sin afectar el software.

La Arquitectura cuenta con 4 partes:

**Set de instrucciones:** Conjunto de instrucciones disponibles en el procesador y las reglas para utilizarlas. Este define la cantidad de instrucciones, el tipo de estas (RISC / CISC), formatos y el ancho del path (tamaño en bits de los registros, memoria direccionable).

Clasificación de las instrucciones: Aritméticas, lógicas, relacionales, control, punto flotante, transferencia de datos, desplazamientos, manipulación de bits, sistema.

**Organización de registros:** Cantidad, tamaño y reglas para su uso.

**Organización de la memoria y direccionamiento.**

**Modos de operación:** Modos de operación del procesador (modo user y modo system).

Clasificación de las Arquitecturas:

Por tipo de instrucciones: **CISC (Complex Instruction Set Computers):** Instrucciones complejas, de largo variable, modos de direccionamiento variable, tiende a ser lento. **RISC (Reduced Instruction Set Computers):** Instrucciones simples, de largo fijo, modos de direccionamiento simple, tiende a ser veloz, ocupando más memoria.

Por tipo de memorias: **von Neumann:** Memoria de datos y programa esta unificada, requiere solo de una memoria, permite modificar el código, tiene un uso más general. **Harvard:** Memoria de datos y programa están separadas, requiere de memorias separadas físicamente, tiene mayores velocidades.

Microprocesador vs Microcontrolador:

Microprocesador: Tiene un propósito más general, rendimiento por velocidad, paralelismo, no incluye periféricos, y no ve a la potencia como un problema, no le importa ser eficiente en este sentido.

Microcontrolador: Tiene propósitos más específicos, rendimiento más balanceado, cuenta con gran cantidad de periféricos, y es eficiente en términos de potencia.

Formas del código fuente:

Código de maquina: Son los programas que el procesador ejecuta, a nivel de bytes (binario).

Código Assembly: Es una representación en formato texto del código de máquina.

Sobre la CPU:

**PC (Program Counter):** Guarda la dirección de la próxima instrucción.

**Condition Codes:** Información sobre la última operación aritmética, se usan para saltos condicionales.

**Memoria:** Es un arreglo direccionable, guarda datos, mantiene el stack.

**Registro:** Almacena dato que el programa está usando.

Condition flags: Teniendo la operación t = a + b

**CF Carry Flag** (para unsigned), si el carry o borrow llega a ser 1 (unsigned overflow)

**ZF Zero Flag**, t == 0

**SF Sign Flag** (para signed), t < 0 (signed)

**OF Overflow Flag** (para signed), si hay overflow en complemento a dos (signed)

Soporte para procedimientos:

Transferencia de control, llamadas y retornos.

Pasaje de datos, a través de registros o el stack.

Manejo de la memoria

Todos los mecanismos se implementan a través de instrucciones

El stack en los procedimientos: Un programa maneja el almacenamiento requerido por los procedimientos usando el stack (como llamadas y retornos), el stack y los registros del programa guardan información requerida para control, y data. El stack crece para las direcciones más bajas, la memoria se ve como un arreglo de bytes. El stack se divide en frames, el %rsp es el stack pointer, y él %rbp es el frame pointer, los frames contienen información de retorno, almacenamiento local.

Calling conventions:

Función caller = invocante

Función callee = invocada

El contenido de los registros puede ser sobre escrito por la función callee.

**Convención caller saved:** La función caller guarda los registros en el stack antes de la invocación. Los registros son %rax, %rdi, %rsi, %rdx, %rcx, %r8, %r9, %r10, %r11.

**Convención callee saved:** La función callee guarda los registros en el stack antes de modificarlos, y reestablece los valores de los registros modificados antes de retornar. Los registros son %rbx, %r12, %r13, %r14, %r15, %rbp, %rsp.

Acceso a elementos de un arreglo

Siendo A[i][j] siendo un tipo de dato T, de tamaño K bytes, y siendo C la cantidad de columnas, seria **A + i \* (C \* K) + j \* K = A + (i \* C + j) \* K**

Representación de estructuras:

Una estructura se representa como un bloque de memoria, lo suficientemente grande como para contener todos los campos. Los campos se ordenan de acuerdo a la declaración, el compilador determina el tamaño total y los offsets de los campos (en lenguaje de maquina no se tiene conocimiento de las estructuras en el código fuente).

Alineamiento:

Si el dato requiere B bytes, la dirección donde comienza debe ser múltiplo de B, la motivación de esto es que la memoria se accede de a pedazos. El padding es el relleno que agrega el compilador para que quede alineado, hay padding interno y externo, el interno es cuando esta entre dos tipos de datos, y el externo es cuando esta al final.