**Arquitectura de Computadores (AIC)**

Apuntes de Vicent Sornosa – 3G

**UT1 – Introducción a AIC**

1. Concepto de AIC
2. Análisis de prestaciones
3. Diseño del juego de instrucciones

**UT2 – Computadores segmentados**

1. Unidad de instrucción segmentada
2. Unidades multiciclo y gestión estática de instrucciones
3. Predicción dinámica de saltos
   1. **Concepto de AIC**

* Juego de instrucciones máquina con las que jugar.
* Recursos que se disponen en la CPU para guardar datos (registros) o para ejecutar instrucciones (operandos aritmético-lógicos)
* Conexiones y transistores…

La labor de un buen ingeniero de computadores es plantear el mejor diseño de un sistema partiendo de los recursos que dispone y de las limitaciones que tiene en cuanto a coste, energía, componentes hardware y características del sistema operativo.

Los transistores avanzan:

* El número de ellos crece exponencialmente debido a la reducción de sus tamaños.
* A screenshot of a computer

  Description automatically generated with medium confidenceLa potencia incrementa

Luego hay como 10 diapositivas donde nos cuentan *La maravillosa historia y evolución del rendimiento de los procesadores*, en resumen:

Los procesadores son como el vino, van a mejor con el paso de los años porque se han descubierto movidas rollo el **paralelismo** y eso redujo bastante el tiempo de computo.

Debido a estas grandiosas movidas, se han podido evolucionar cosas que ya nadie se acordaba como la Cache (se emocionaron y pusieron niveles y más niveles: L1, L2 y L3).

A *esa gente que entonces les gustaba AIC* decidieron pensar y pensar, y llegaron a la conclusión que un solo núcleo funcionando a full se frustra y empieza a causar problemas energéticos, por lo tanto, se preguntaron:

¿*Oye y si en vez de un núcleo grande tenemos varios pequeños dentro del procesador?*

Y así se dieron cuenta que todo iba más rápido con procesadores multinúcleo porque trabajando en equipo se consiguen más cosas.

Ah si, y también se les ocurrió segmentar procesadores y más tarde crear etapas únicamente para propagar señales… pero eso no llegó hasta el Pentium 4.

**1.2 Análisis de Prestaciones**

Estamos todos de acuerdo en que un usuario que utiliza un computador quiere tiempos de cómputo tan cortos como sea posible (no mola que se quede pensando ni que tarde una eternidad).

Para eso mismo, el admin del sistema quiere satisfacer al usuario y darle las mejores prestaciones (mayor productividad)

Cuando se están comparando dos computadores, se escoge al más lento como referencia **SIEMPRE** y cuando se está examinando un conjunto de computadores se determina un auxiliar como referencia **PARA TODOS**.

Esta relación básicamente nos dice que **X es S veces más rápido que Y**.

También se podría leer como que **X es n% más rápido que Y.**

Vale, sabemos la aceleración de un computador con respecto a otro, pero molaría saber el Tiempo de ejecución de un computador mientras hace una tarea, así ya sabemos más cositas de él. Vamos allá:

Donde:

* es el número de instrucciones que ejecuta (cuidado con los bucles)
* es el número de ciclos que ejecuta el procesador por instrucción
* Y finalmente, es cuanto tarda en ejecutar un ciclo.

La famosa Ley de Amdahl que cae en los parciales:

Cuenta la leyenda, que a Amdahl se le ocurrió segmentar los programas en fracciones.

¿Para qué? Buena pregunta, se supone que una/varias de esas fracciones se pueden paralelizar para ahorrarse un poquito de

A parte de pensar esta maravilla, Amdahl nos dejó una pequeña ecuación para calcular cual sería la aceleración de la actualización de un sistema con respecto al no modificado.

Graphical user interface, application

Description automatically generated

Y claro, a partir de esta genialidad podemos sacar la aceleración de un sistema frente a otro de la siguiente manera:

A white board with writing on it

Description automatically generated with medium confidence

Existe una cosa que se llama Principio de localidad de referencia que dice:

***‘El 90% del tiempo se ejecuta el 10% del código.’***

Por lo tanto, a la hora de optimizar/acelerar, es conveniente centrarse en ese 10% que se repite tanto.

Obviamente optimizar programas/fracciones de programas porque sí no iba a ser gratis y todos felices, todo tiene un coste.

Molaría saber si la optimización que se plantea llevar a cabo es rentable económicamente.

Y ahí es cuando entra la relación prestaciones-coste:

Diagram

Description automatically generated

**Otras medidas de prestaciones:**

1. La suma aritmética / Tiempo total de ejecución:
2. La media aritmética:
   * Siendo n el número de sistemas a evaluar
3. La media geométrica:

Diagram

Description automatically generated

1. MIPS (**M**illones de **I**nstrucciones **P**or **S**egundo)
2. MFLOPS (**M**illones de **F**loating **O**perations **P**er **S**econd)

**1.3 Diseño de los juegos de instrucción**

Primero antes de tocar nada molaría saber que es un juego de instrucciones…

* Básicamente, es la **interfaz** que conecta un **programa con la ruta de datos.**
* Para que una **instrucción** del propio juego de instrucciones sea útil **se debe de compilar** por un compilador, **si no, es totalmente inútil**, es como si fuese una línea que no hace nada.

**Cada juego de instrucciones tiene sus propiedades** que son totalmente distintas de otros juegos de instrucciones como por ejemplo cuánto ocupa cada registro (en bits), cómo se handlean las operaciones, cómo se controla el flujo de datos…

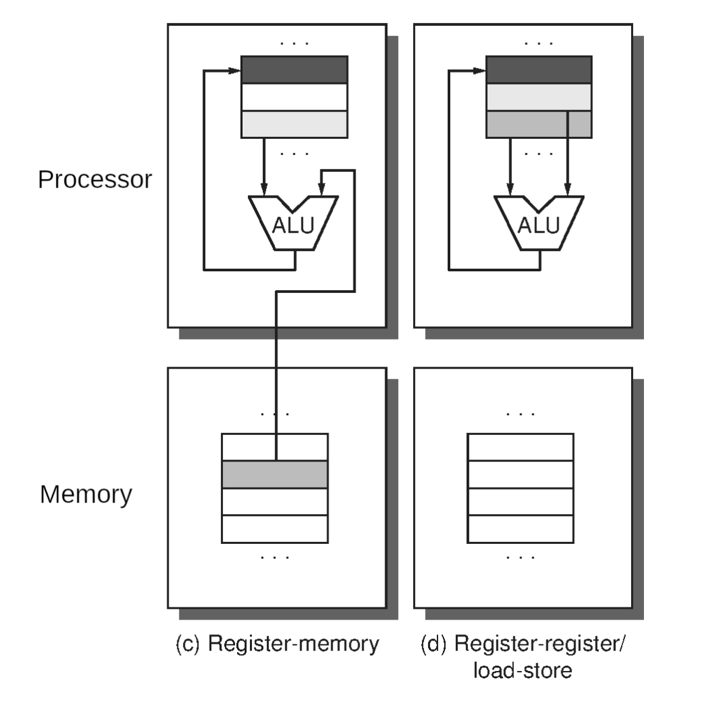
En el caso de AIC, se va a utilizar como juego de instrucciones el MIPS64 porque tiene una serie de utilidades y aplicaciones bastante curiosas.

* Las instrucciones son de 32 bits
* MIPS64 contiene 64 registros (de 64 bits):
  + - Contiene 32 registros de propósito general.
    - Y también contiene 32 registros destinados a operaciones en coma flotante.

Cuando queremos hacer operaciones tenemos que saber que los datos se obtienen de los registros en el banco de registros de la CPU o bien, de memoria directamente.

Cuando se obtienen esos datos, la ALU (Unidad Aritmético-Lógica) los pilla y los procesa, y finalmente cuando obtiene un resultado, envía ese resultado al banco de registros.

Esto proporciona una compilación eficiente hasta que dentro de unos años llegará algún genio con alguna movida nueva y lo sustituirá, pero hasta entonces esto tira bien.



**Existen 2 principales juegos de instrucciones:**

1. **Juego de instrucciones RISC:**

Propio de MIPS, ARM (**A**dvanced **R**ISC **M**achines) …

Instrucciones de longitud fija (32/64 bits).

Modelo Load/Store.

1. **Juego de instrucciones Intel Architecture (IA):**  
    Es compatible con AMD (**A**dvanced **M**icro **D**evices).

Instrucciones de longitud variable.

Modelo Registro-Memoria. (*nInstr* incrementa por la instrucción extra de acceso a memoria)

Graphical user interface, text, application

Description automatically generated

lw r1, 8(r2)

Diagram, schematic

Description automatically generated

Text

Description automatically generated