



Introducción a la Arquitectura y Organización de Computadores

Alejandro Furfaro

22 de marzo de 2019

Agenda

1 Introducción

- Arquitectura y Organización
- Instruction Set Architecture (ISA)
- Organización y Hardware

2 Objetivos de estudiar organización y hardware

3 Orígenes

- En solo 80 años
- Pipeline

4 Nace un paradigma de Arquitectura: RISC

- Antecedentes

1 Introducción

- Arquitectura y Organización
- Instruction Set Architecture (ISA)
- Organización y Hardware

2 Objetivos de estudiar organización y hardware

3 Orígenes

4 Nace un paradigma de Arquitectura: RISC

Arquitectura vs Microarquitectura

Arquitectura

Es el conjunto de recursos accesibles para el programador, que por lo general se mantienen a lo largo de los diferentes modelos de procesadores de esa arquitectura (puede evolucionar pero la tendencia es mantener compatibilidad hacia los modelos anteriores).

- Registros
- Set de instrucciones
- Estructuras de memoria (descriptores de segmento y de página p. ej.)

Micro Arquitectura

Es la implementación en el silicio de la arquitectura. Lo que está detrás del set de registros y del modelo de programación. Puede ser muy simple o sumamente robusta y poderosa. Cambia de un modelo a otro dentro de una misma familia.

Definición de la Arquitectura de un Computador

- Es sin dudas la tarea mas ardua de un diseñador.
- Se trata de definir los aspectos mas relevantes en la arquitectura de un computador que maximicen su rendimiento, sin dejar de satisfacer otras limitaciones impuestas por los usuarios, como un costo económico que lo haga accesible, o un consumo de energía moderado.
- Comprende el diseño del set de instrucciones, el manejo de la memoria y sus modos de direccionamiento, los restantes bloques funcionales que componen el CPU, el diseño lógico, y el proceso de implementación
- La implementación comprende el diseño del circuito integrado, su encapsulado, montaje, alimentación y refrigeración.

Definición de la Arquitectura de un Computador

skills necesarios

No es posible realizar esta tarea con éxito sin tener manejar de manera solvente varias tecnologías diferentes:

- Diseño lógico.
- Tecnología de encapsulado
- Funcionamiento y diseño de compiladores y de Sistemas Operativos.

1 Introducción

- Arquitectura y Organización
- **Instruction Set Architecture (ISA)**
- Organización y Hardware

2 Objetivos de estudiar organización y hardware

3 Orígenes

4 Nace un paradigma de Arquitectura: RISC

Definiendo un ISA

Nos referimos a *Instruction Set Architecture*, como el set de instrucciones visibles por el programador. Es además el límite entre el software y el hardware.

Clases de ISA. ISAs con Registros de Propósito general vs. Registros dedicados. ISAs registro-memoria vs. ISAs Load Store.

Direccionamiento de Memoria. Alineación obligatoria de datos vs. administración de a bytes.

Modos de Direccionamiento. Como se especifican los operandos.

Tipos y tamaños de operandos. Enteros, Punto Flotante, Punto Fijo. Diferentes tamaños y precisiones.

Operaciones. Pocas Simples (RISC) o Muchas Complejas (CISC).

Instrucciones de control de flujo Saltos condicionales, calls.

Longitud del código Instrucciones de tamaño fijo vs. variable.

1 Introducción

- Arquitectura y Organización
- Instruction Set Architecture (ISA)
- Organización y Hardware

2 Objetivos de estudiar organización y hardware

3 Orígenes

4 Nace un paradigma de Arquitectura: RISC

Microarquitectura = Organización + Hardware

Organización

Se refiere a los detalles de implementación de la ISA. Es decir:

- Organización e interconexión de la memoria.
- Diseño de los diferentes bloques de la CPU y para implementar el set de instrucciones.
- Implementación de paralelismo a nivel de Instrucciones, y/o de datos.
- Podemos así encontrar procesadores que poseen la misma ISA pero una organización muy diferente.

Ejemplo:

Los procesadores AMD FX y los Intel Core i7, tienen la misma ISA, sin embargo organizan su caché y su motor de ejecución de maneras diferentes.

Microarquitectura = Organización + Hardware

Hardware

- Se refiere a los detalles de diseño lógico y tecnología de fabricación.
- Existirán por lo tanto, procesadores con la misma ISA y la misma organización, pero absolutamente distintos a nivel de hardware y diseño lógico detallado.

Ejemplo:

El Pentium 4, diseñado para desktop, y el Pentium 4 M para notebooks. Este último tiene una cantidad de lógica para control del consumo de energía, siendo su hardware muy diferente del del Pentium 4 desktop.

¿Porque necesitamos saber todo esto?

- Para comprender lo que ocurre debajo del software.
- Comprender como los diseños de hardware impactan en el software y en el programador
- Estar en condiciones de cruzar verticalmente las capas que separan el silicio de la aplicación.
- Comprender las nociones básicas que rigen el diseño de un sistema de cómputo moderno

1 Introducción

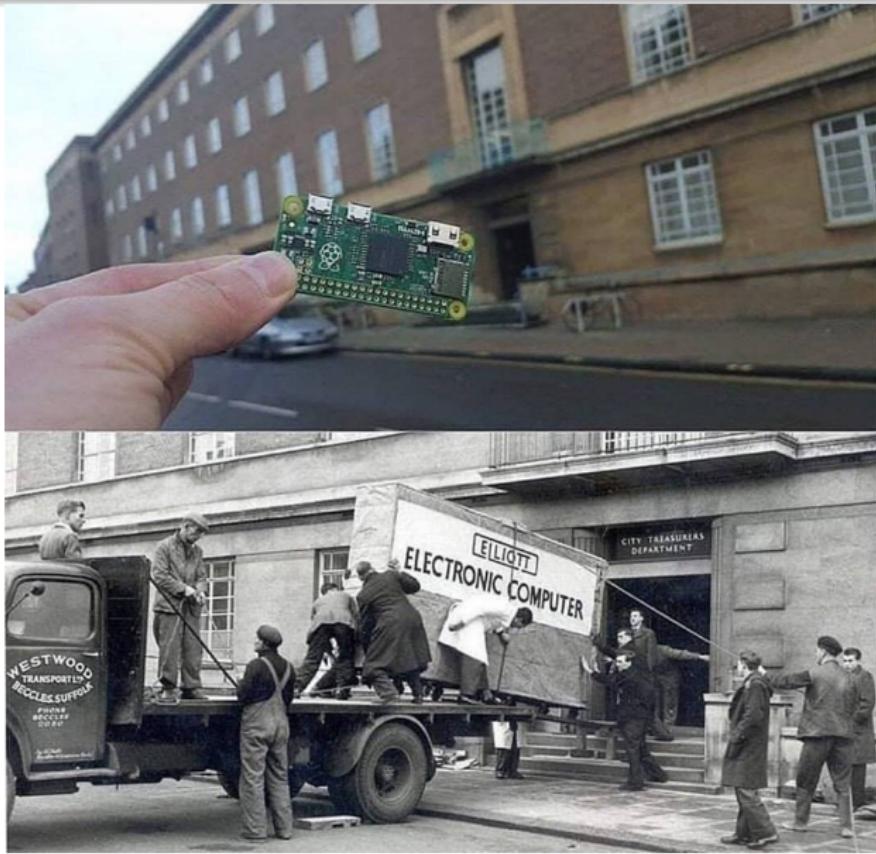
2 Objetivos de estudiar organización y hardware

3 Orígenes

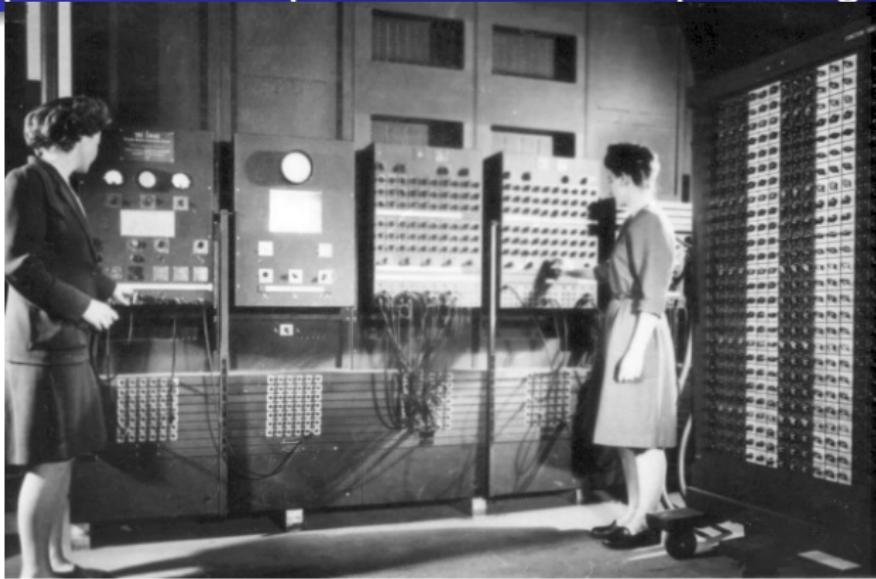
- En solo 80 años
- Pipeline

4 Nace un paradigma de Arquitectura: RISC

En solo algo menos de 80 años



ENIAC "primer" Computadora de Propósito general

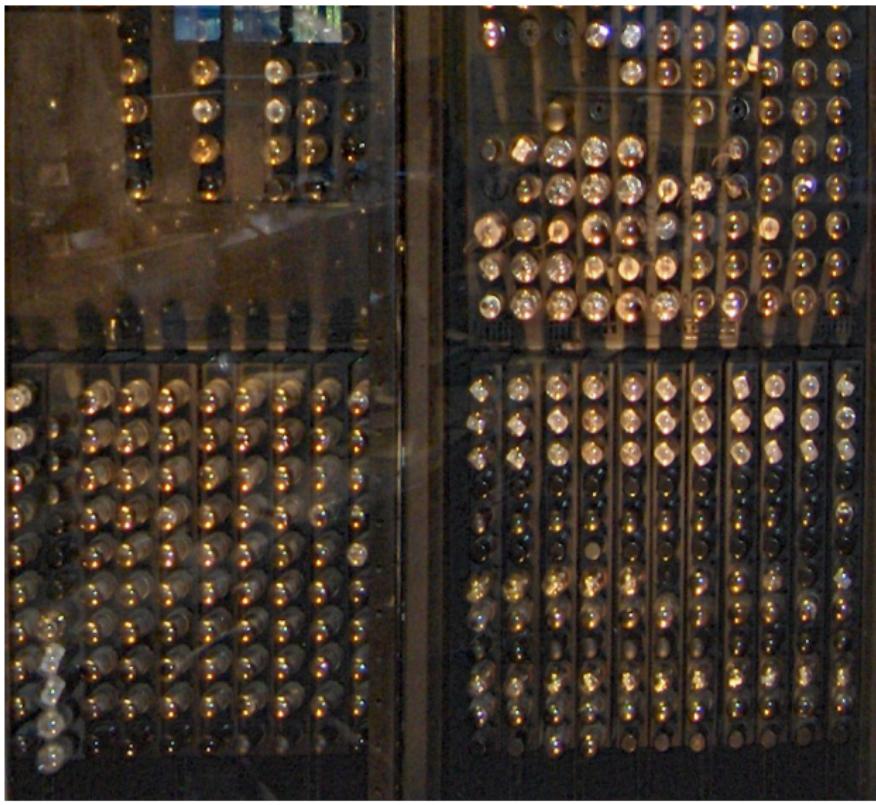


Tal parece que la Z1 en Alemania fue algo anterior. Pero esta está considerada la primera.

No utiliza programa almacenado. Utilizaba 6000 interruptores para programarse.

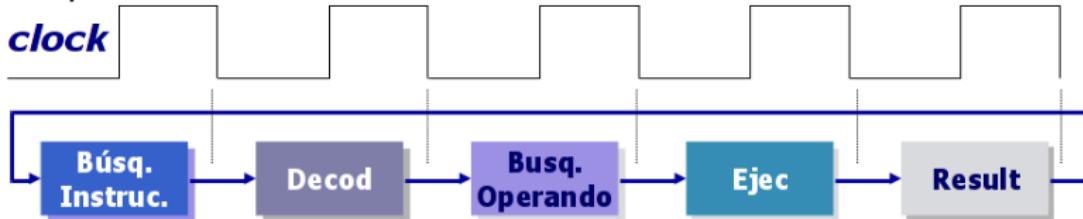
Consumo 160 KW, 27 Toneladas, 167 m²

ENIAC BackPlane. Observar las válvulas termoiónicas



Modelo de Von Neumann

- ① Modelo de programa almacenado.
- ② Datos y programa en el mismo (único) banco de memoria.
- ③ Máquina de estados



- ④ No está claro cual fue la primera implementación, pero Manchester Baby, IBM-SSEC, EDVAC, BINAC, entre otras fueron de las primeras (alrededor de 1948 todas ellas)

Y llegó UNIX en 1969



Esta máquina es a la que se portó la primer versión escrita en C.
16 Kbytes de memoria. 512 Kbytes de almacenamiento...

¿Porqué esta clase de historia?

- Actualmente la preocupación central de los fabricantes de microprocesadores es el consumo de energía eléctrica. todos investigan y trabajan diariamente para presentar chips que consuman el mínimo de energía para el mismo trabajo.
- Durante el corto lapso que lleva esta industria, la memoria fue durante muchas décadas el recurso mas escaso (y en rigor sigue vigente la máxima de Von Neumann "*la cantidad de memoria siempre será insuficiente*"). Pero como pudimos apreciar, en los albores esta escasez fue dramática.
- 16 Kbytes de RAM (8 para el sistema operativo y 8 para las aplicaciones) en 1969,... huelgan las palabras.

Uso mínimo de memoria: un “must”

Por lo tanto así como hoy se diseñan procesadores con foco puesto en el mínimo consumo de energía, hace 4 o 5 décadas, se diseñaban pensando en que los programas consuman la mínima cantidad de memoria posible.

Instrucciones complejas

- ① Los diseñadores de la época apuntaban a microprocesadores con instrucciones muy complejas
- ② En ese contexto se han diseñado Microprocesadores capaces de integrar código de alto nivel.
- ③ Ejemplos extremos: POLY (VAX computer)

Listado1

```
; To compute P(x) = C0 + C1*x + C2*x**2  
; where C0 = 1.0 , C1 = .5 , and C2 = .25
```

```
POLYF X,#2,PTABLE
```

```
.
```

```
.
```

```
.
```

```
PTABLE: .FLOAT 0.25 ; C2  
.FLOAT 0.5 ; C1  
.FLOAT 1.0 ; C0
```

Instrucciones complejas: Conclusión

Con los bloques de datos no hay mucho que hacer para ahorrar memoria. Pero en el caso del código, si podemos utilizar menos instrucciones para el mismo trabajo, se ahorra memoria. Este paradigma estuvo presente en todos los diseños durante décadas. Intel desarrolló su arquitectura actual con este paradigma. Y se comprometió a mantener **compatibilidad**.

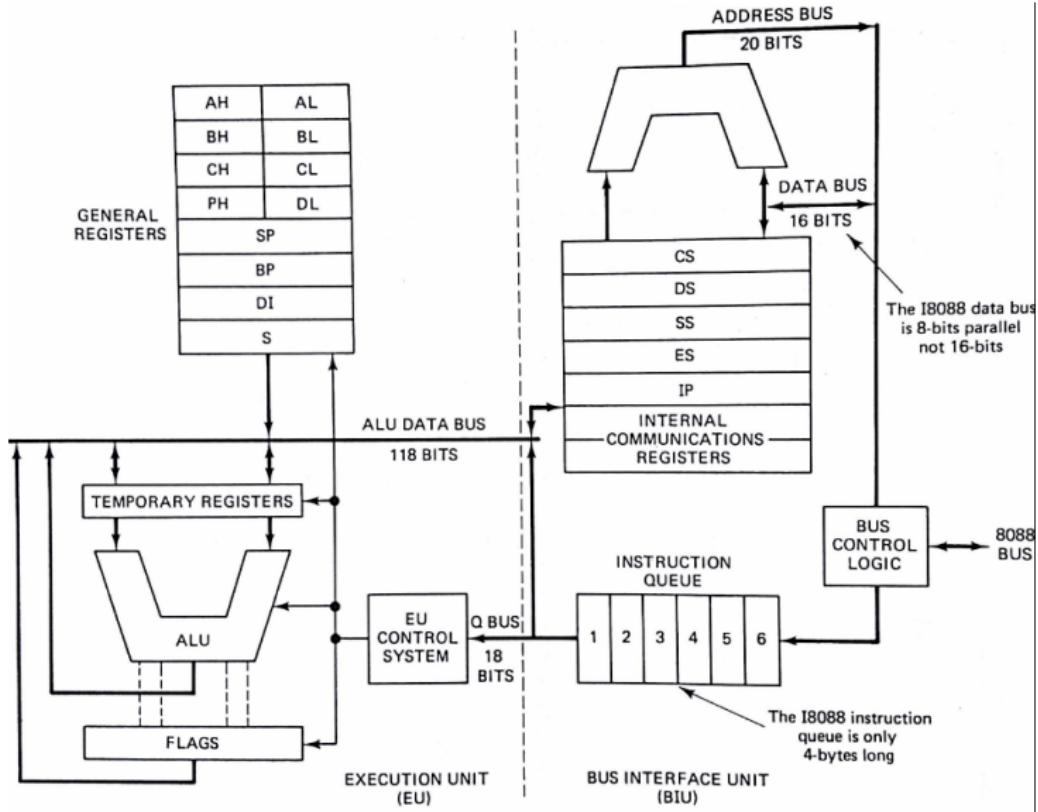
Otros problemas a resolver

- En los '70, los usuarios de computadores demandaban disminuir el costo de actualización de sus equipos
- Cada upgrade obligaba a reemplazar no solo el hardware sino además el sistema operativo y las aplicaciones.
- Esta dinámica dificultaba a los proveedores establecer un cash flow que permitiese incrementar inversiones en I+D.

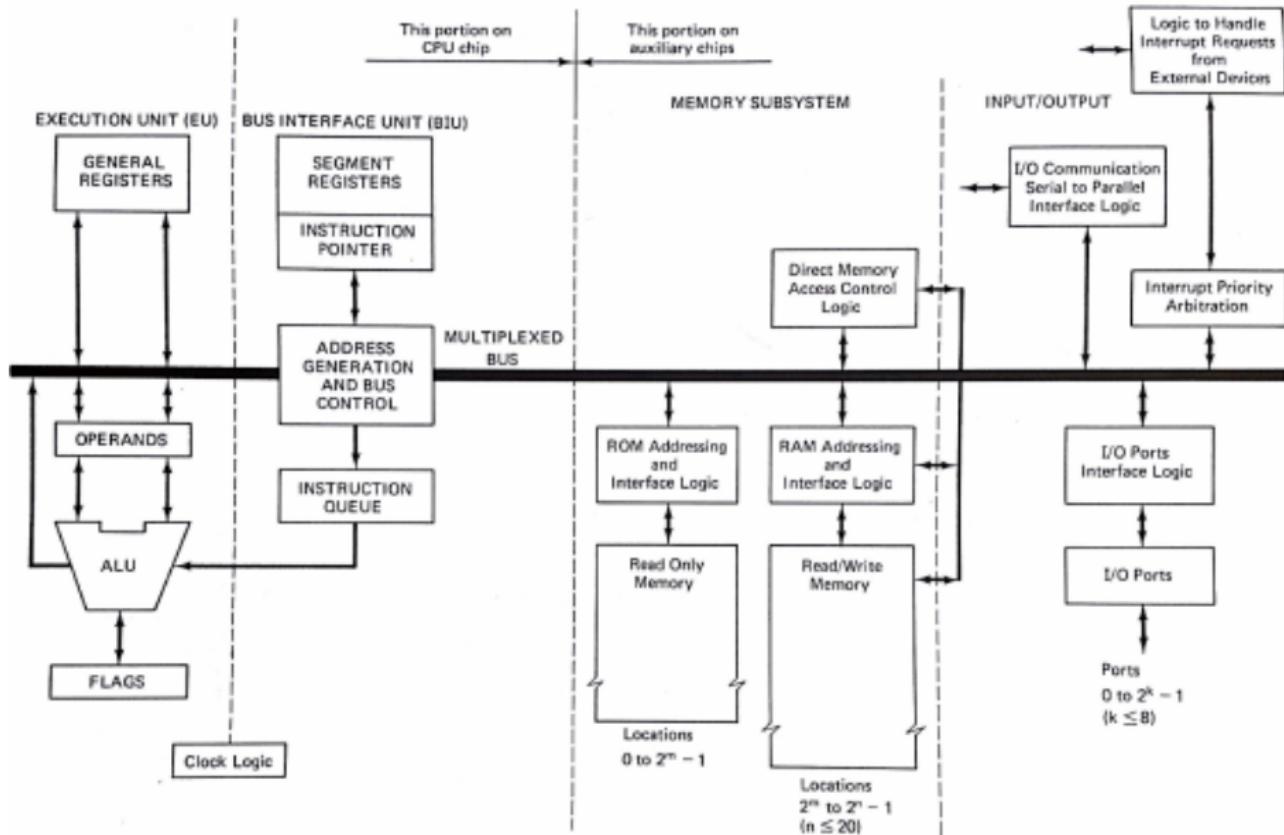
Intel dá el golpe

- En 1978 presenta la familia iAPx86 de la mano de su nave insignia procesador 8086,
- Se compromete a mantener compatibilidad ascendente en los modelos subsiguientes de procesadores para satisfacer esta demanda.
- Esto fue uno de los motivos de la decisión de IBM de adoptar esta familia de procesadores como base para su PC.

8086 Organización



Sistema básico de 8086



1 Introducción

2 Objetivos de estudiar organización y hardware

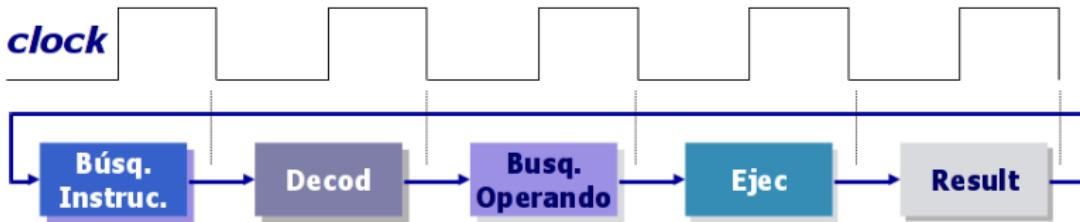
3 Orígenes

- En solo 80 años
- Pipeline

4 Nace un paradigma de Arquitectura: RISC

Máquina de estados elemental

- En la década del 40 Von Newman definió un modelo básico de CPU, que a esta altura está mas que superado.
- Sin embargo algunos conceptos de ese modelo viven en los mas modernos procesadores.
- Uno de ellos es la máquina de ejecución

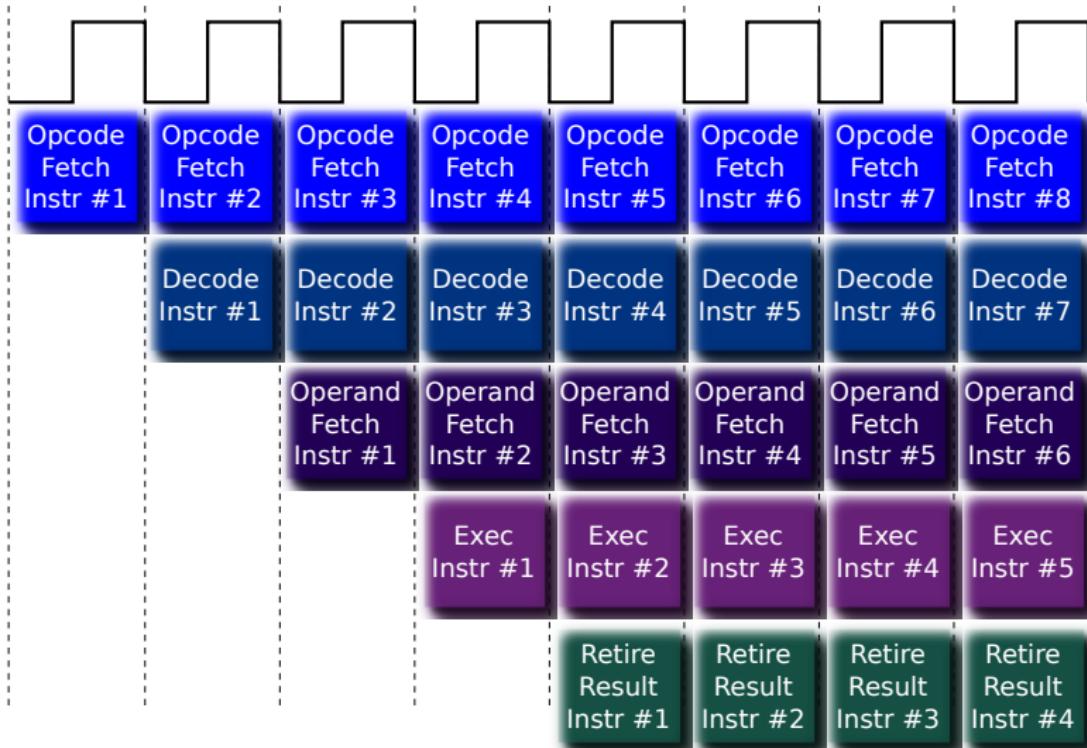


- En las primeras generaciones de microprocesadores cada etapa de este ciclo se ejecutaba en un ciclo de clock, y la CPU entera estaba dedicada a esa tarea.
- Ejecutar una instrucción insumía de varios ciclos de clock...

Pipeline

- Arquitectura que permite crear el efecto de superponer en el tiempo, la ejecución de varias instrucciones a la vez.
- Con él se formaliza el concepto de ***Instruction Level Parallelism (ILP)***.
- Requiere muy poco o ningún hardware adicional.
- Solo necesita que los bloques del procesador que resuelven la máquina de estados para la ejecución de una instrucción, operen en forma simultánea.
- Se logra si todos los bloques funcionales trabajan en paralelo pero cada uno en una instrucción diferente.
- Es algo parecido al concepto de una línea de montaje, en donde cada operación se descompone en partes, y se ejecutan en un mismo momento diferentes partes de diferentes operaciones.
- Cada parte se denomina etapa (stage)

Pipeline

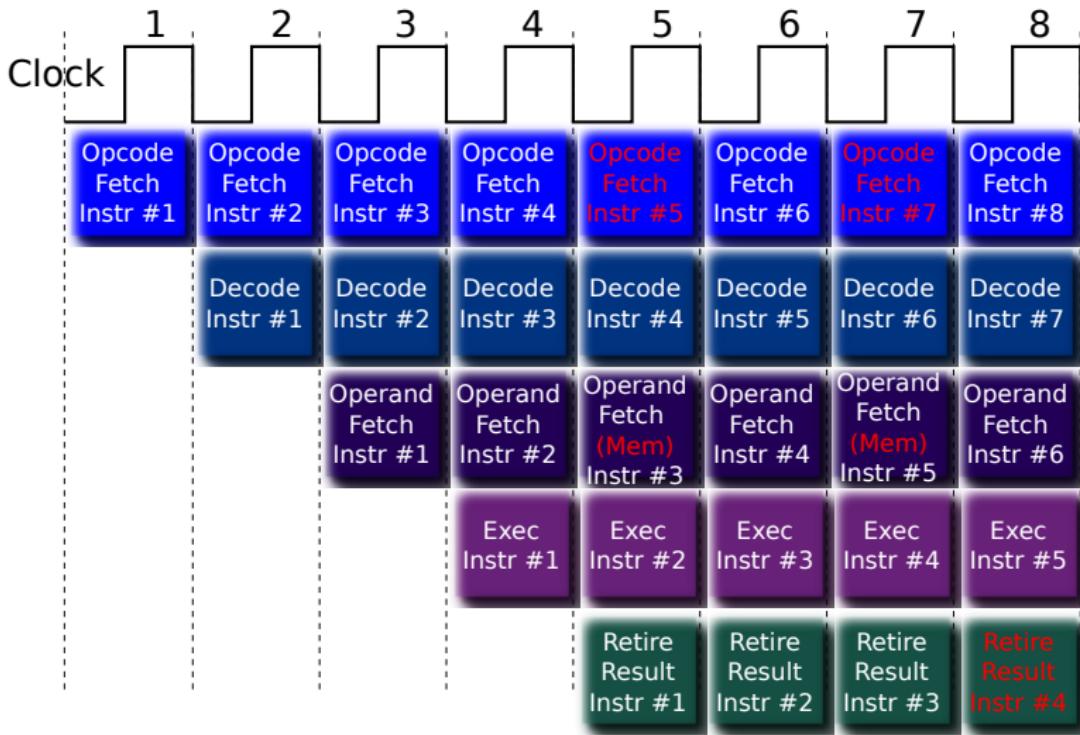


Este modelo es teórico y representa la situación ideal.

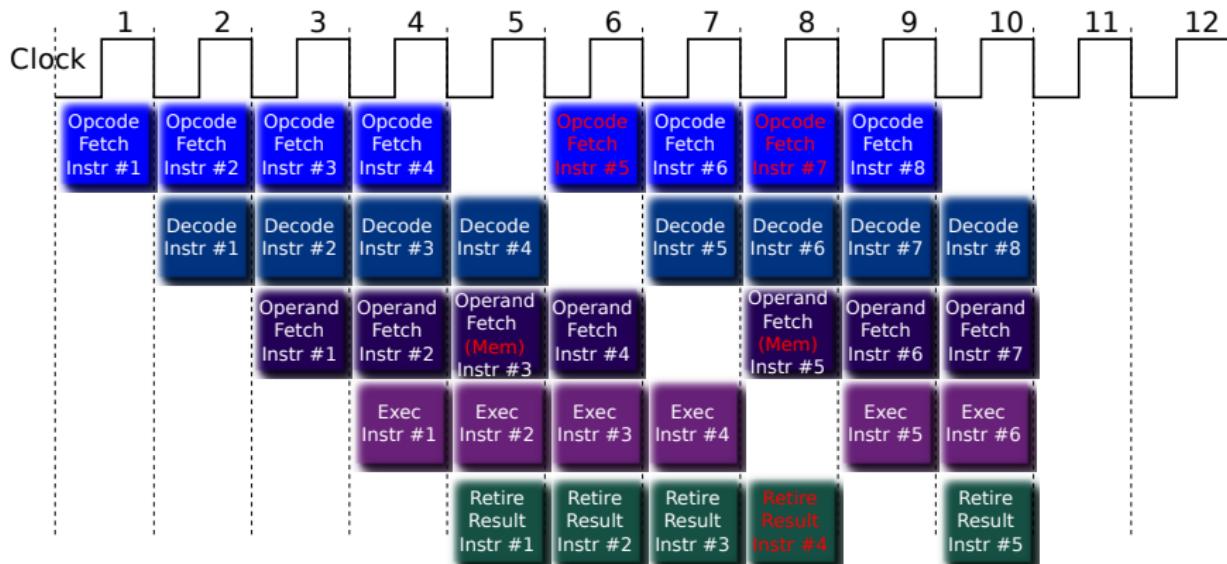
Obstáculos Estructurales: Ejemplo

- Tenemos un procesador que solo tiene una etapa para acceder a memoria y la comparte para acceso a datos e instrucciones.
- En el caso de que se necesite un operando de memoria, el acceso para traer este operando interferirá con la búsqueda del operando de una instrucción mas adelante del programa.
- También interferirá con el Fetch de la siguiente instrucción

Obstáculos Estructurales: Accesos concurrentes a memoria

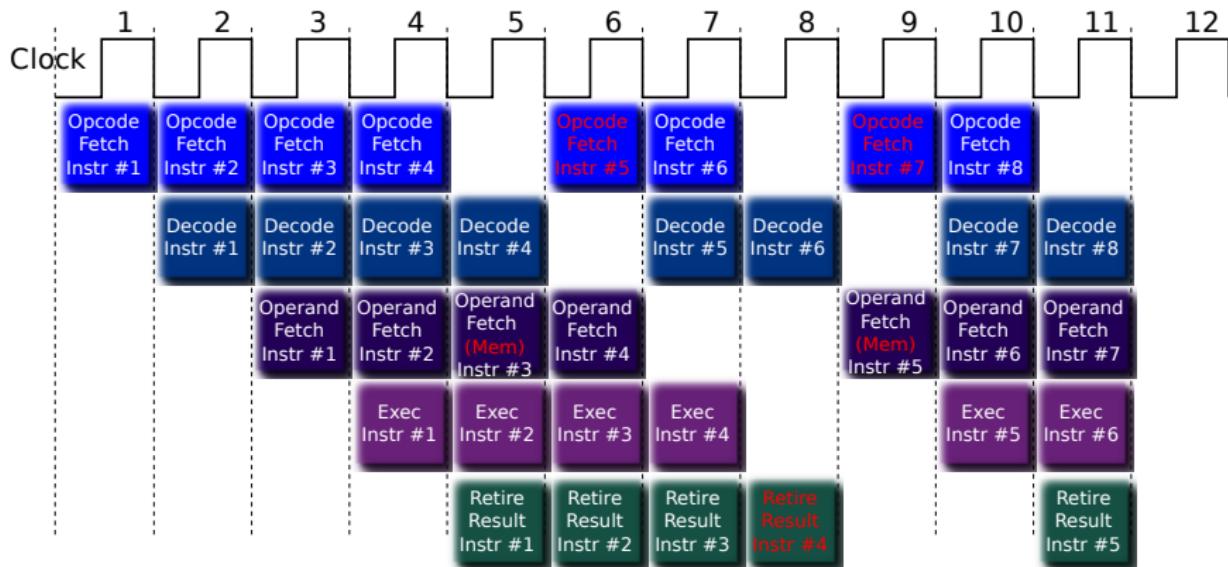


Obstáculos Estructurales - Efecto en CPI



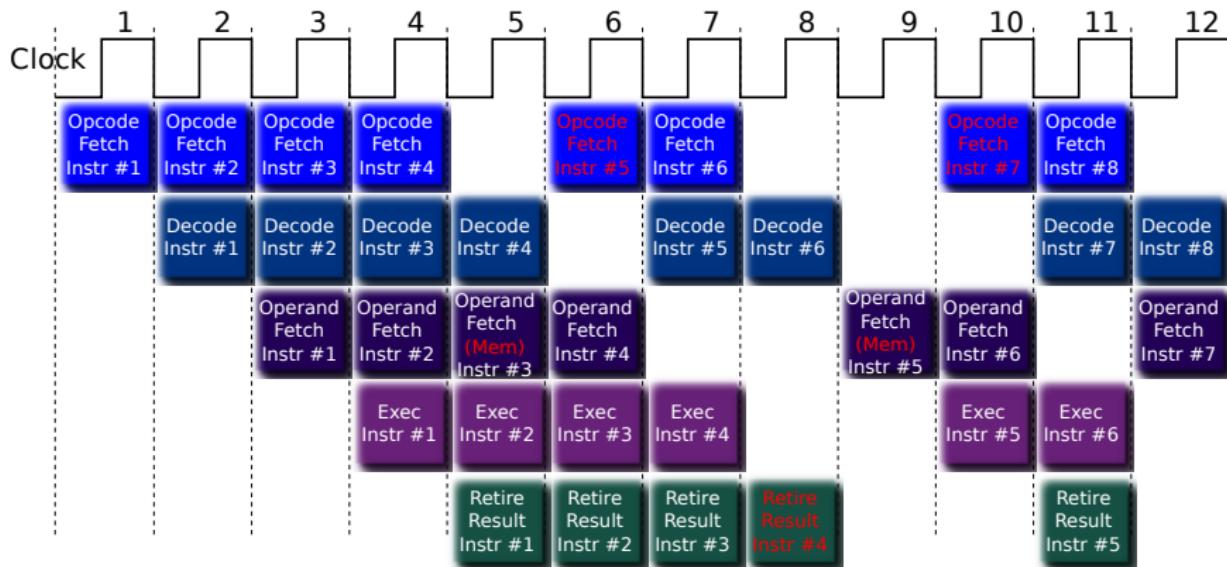
- Por cada Obstáculo, pospone una operación. $CPI = CPI + 1$
- En general la cantidad de CPI que se incrementan es igual a la cantidad de concurrencias menos 1, en el lapso considerado

Obstáculos Estructurales - Efecto en CPI



- Por cada Obstáculo, pospone una operación. $CPI = CPI + 1$
- En general la cantidad de CPI que se incrementan es igual a la cantidad de concurrencias menos 1, en el lapso considerado

Obstáculos Estructurales - Efecto en CPI

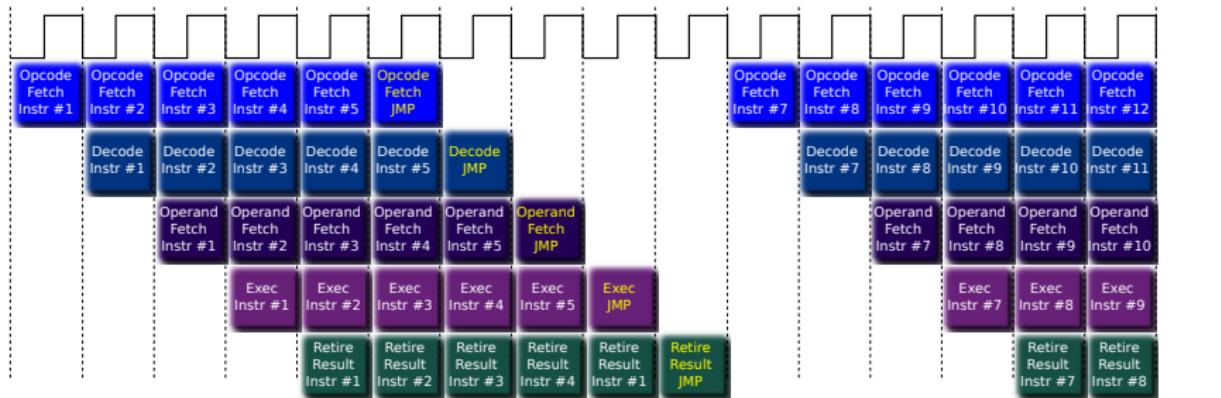


- Por cada Obstáculo, pospone una operación. $CPI = CPI + 1$
- En general la cantidad de CPI que se incrementan es igual a la cantidad de concurrencias menos 1, en el lapso considerado

Obstáculos de Control

- Un branch es la peor situación en pérdida de performance.
- Un branch es una discontinuidad en el flujo de ejecución.
- El pipeline busca instrucciones en secuencia.
- El branch hace que todo lo que estaba pre procesado deba descartarse. Y el pipeline se vacía debiendo transcurrir $n - 1$ ciclos de clock hasta el próximo resultado. Siendo n la cantidad de etapas del pipeline. Esto se conoce como **branch penalty**.

"La conspiración de los branches"



- 1 Introducción
- 2 Objetivos de estudiar organización y hardware
- 3 Orígenes
- 4 Nace un paradigma de Arquitectura: RISC
 - Antecedentes

Orígenes de RISC

- Antes de las Personal Computers, el mercado era dominado por los denominados mainframes.
- Los players eran IBM, Digital Electronic Corporation (DEC), Amdahl, entre otros. Sus computadores hoy son piezas de museo.
- Sobre los años 80 aparecen los primeros microprocesadores de 16 bits con ciertas capacidades para hacer frente a una nueva generación de equipos de cómputo:
 - Intel evoluciona el 8086/88 al 80286, y ensaya el iAPx432 citado como micromainframe processor, especialmente diseñado para ejecutar lenguajes de alto nivel como Ada.
 - Motorola evoluciona el 68000 (citado como m68k)
- Vimos que tienen todos un punto en común: Set de instrucciones de complejidad creciente

Historia de RISC

Líneas de Investigación

Durante los años '60 y '70, IBM, Control Data Corporation (CDC), y Data General (DG), incursionaron líneas de investigación tendientes a implementar procesadores con pocas instrucciones simples.

En los '80 David Patterson¹ ² (Universidad de Berkeley), y John Hennessy³ (Universidad de Stanford) publicaron los primeros trabajos con resultados concretos, que presentaban una arquitectura contrapuesta con la de los Computadores que en ese momento dominaban la industria. La denominaron **RISC** (**R**educed **I**nstruction **S**et **C**omputer). Y esto motivó que a los procesadores diseñados hasta entonces se los etiquetar como **CISC** (Por **C**omplex **I**nstruction **S**et **C**omputer)

¹ David A. Patterson, Carlo H. Sequin. RISC I: A Reduced Instruction Set VLSI Computer

² David A. Patterson, David R. Ditzel. The case for the reduced instruction set computer

³ Hennessy, J.L., Jouppi, N., Baskett, F., Gill, J. MIPS: A VLSI Processor Architecture. In Proceedings CMU Conference on VLSI Systems and Computations. Computer Science Press, October 1981

Relación con el pipeline

- Se analizaron los obstáculos en un pipeline.
- En particular los obstáculos de datos impactan cuando involucran accesos a memoria para buscar operandos.
- Claramente si se evita que los operandos se accedan en memoria, en lugar de cargarlos previamente en un registro tal vez el pipeline pueda mejorarse

Hennesy & Patterson



Los Mandamientos RISC

- ① Se dispone de un juego de registros numeroso, todos de propósito general.
- ② Las instrucciones se ejecutan en un solo ciclo de clock.
- ③ Las instrucciones derivan en códigos de operación de igual formato y tamaño.
- ④ Las instrucciones deben ser sencillas de decodificar. Los números de registros se deben tener la misma ubicación en los códigos de la instrucción y deben requerir la misma cantidad de bits para su decodificación.
- ⑤ No se utiliza micro código para decodificar instrucciones (no hay instrucciones complejas, como DIV o MUL).
- ⑥ Los datos en memoria se acceden mediante instrucciones simples de transferencia: LOAD y STORE.

Corolario

- 1 La validación de una máquina RISC es mucho más simple de realizar, y también es menor la probabilidad de liberar al mercado un procesador con defectos de lógica. (Recordemos el bug de división del Pentium@90Mhz)
- 2 Es esperable que el tamaño de los programas sea mayor, debido a que lo que los procesadores CISC resuelven con microcódigo, los RISC lo resuelven en el software. Una multiplicación se resuelve siempre por acumulación de sumas.

$$A * B = \sum_{i=1}^A B$$

- En microcódigo se implementa la instrucción MUL.
- Por software se implementa con una subrutina.

```

sub R2,R2,1
sumar: add R1,R1,R1
       sub R2,R2,1
       blez R2,sumar
    
```

Algunas comparaciones de código

Se requieren dos cosas:

- ① Ingresar a Compiler Explorer
- ② Para sacar conclusiones útiles, pensar y aplicar lo que vimos hasta aquí.