[Resumen clase 6](https://drive.google.com/file/d/1nuoEQ3h-fhW4pXkPSopUChotmkUKKQjS/view?usp=sharing)

RISC - Computadoras de repertorio reducido de instrucciones

# Historia de evolución

* El concepto de familia:
  + Introducido por IBM en su System/360 en 1964
  + DEC PDP-8
  + Separa la arquitectura de la implementación
* Unidad de control microprogramada
  + Idea propuesta por Wilkes en 1951
  + Introducida por IBM en la línea S/360 en 1964
* Memoria cache
  + En 1968 en el IBM S/360 Modelo 85
* RAM de estado sólido
* Microprocesadores
  + Comienzo con Intel 4004 en 1971
  + Variantes:
    - Propósito general
    - Embebidos/empotrados
* Procesadores múltiples

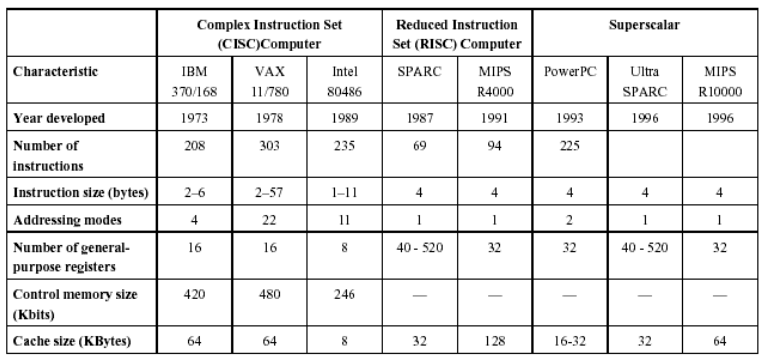
# RISC

Computadoras de repertorio reducido de instrucciones.

Características principales:

* Gran número de registros de uso general o mejor tecnología de compiladores para optimizar el uso de los registros
* Repertorio de instrucciones limitado y sencillo
* Énfasis en la optimización de la segmentación de instrucciones

Comparación de procesadores



# CISC

## Finalidad

* Facilitar el trabajo del escritor de compiladores
* Mejorar la eficiencia de la ejecución: secuencias complejas de operaciones en microcódigo
* Dar soporte a HLL más complejos

## Inconvenientes

* El software es mucho más caro que el hardware
* El nivel del lenguaje es cada vez más complicado
* Salto semántico: diferencias entre operaciones HLL y operaciones de la Arquitectura
* Todo esto conduce a:
  + Repertorios de instrucciones grandes
  + Más modos de direccionamiento
  + Varias sentencias de HLL implementadas en al hardware (Por ejemplo, el CASE del VAX)

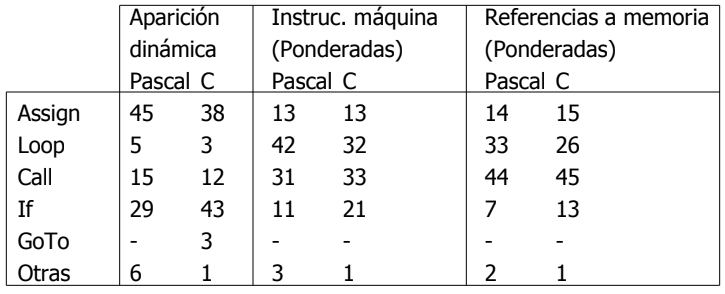
## Características de la ejecución

Estudios sobre programas escritos en HLL

* Operaciones realizadas: Funcionamiento del procesador e interacción con memoria
* Operandos usados: Tipos y frecuencia de uso (Organización de la memoria y modos de direccionamiento)
* Secuenciamiento de la ejecución: Organización del control y del cauce

Estudios dinámicos: medir durante la ejecución

## Frecuencia dinámica relativa

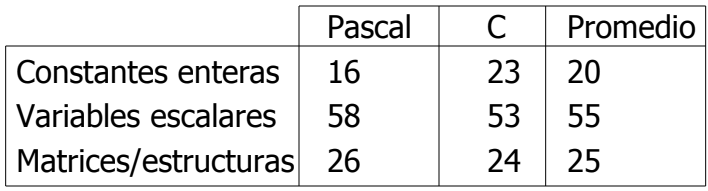


## Operaciones

* Asignaciones: movimiento de datos
* Estamentos condicionales (if, loop): control secuencial
* El procedimiento llamada/retorno consume mucho tiempo
* Algunas instrucciones HLL conducen a muchas operaciones de código máquina

## Operandos

* Principalmente variables escalares locales
* La optimización debe concentrarse en el acceso a las variables locales



## Llamadas a procedimientos

* Se consume mucho tiempo
* Depende del número de parámetros tratados
* Depende del nivel de anidamiento
  + La mayoría de los programas no tienen una larga secuencia de llamadas seguida por la correspondiente secuencia de retornos
  + La mayoría de las variables son locales
  + Las referencias a operandos están muy localizadas

**Ventana de profundidad**

## Consecuencias

Se puede ofrecer mejor soporte para los HLL optimizando las prestaciones de las características más usadas y que más tiempo consumen.

* Usar un gran número de registros: optimizar las referencias a operandos
* Prestar cuidadosa atención al diseño de los cauces de instrucciones: predicción de bifurcaciones, etc
* Es recomendable un repertorio con instrucciones simples (reducido)

## Amplio banco de registros

* Aproximación por software:
  + El compilador es necesario para asignar registros
  + Asignación de registros a las variables que se usen más en un período de tiempo dado
  + Requiere el uso de sofisticados algoritmos de análisis de programas
* Aproximación por hardware:
  + Utilización de más registros
  + De esta manera, más variables pueden mantenerse en registros durante períodos de tiempo más largos

## Registros para variables locales

* Muchos registros → reducir el acceso a memoria
* Por estudios anteriores → almacenar las variables escalares locales en registros
* Problema: cada llamada de procedimiento/función cambia la “localidad”:
  + Los parámetros deben ser pasados
  + Los resultados tienen que ser devueltos
  + Las variables de los programas de llamada tienen que ser restauradas

## Ventanas de registro

Por estudios realizados y con los registros:

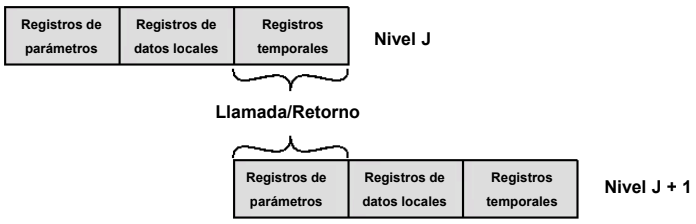
* Se requieren pocos parámetros y variables locales en cada llamada
* Hay limitación en la “profundidad” de llamadas”
* Utilización de múltiples conjuntos pequeños de registros para cada llamada distinta
* La llamada cambia el conjunto de registros a usar
* Los retornos vuelven a cambiar al anterior conjunto de registros utilizado

Tres áreas dentro de un conjunto de registros:

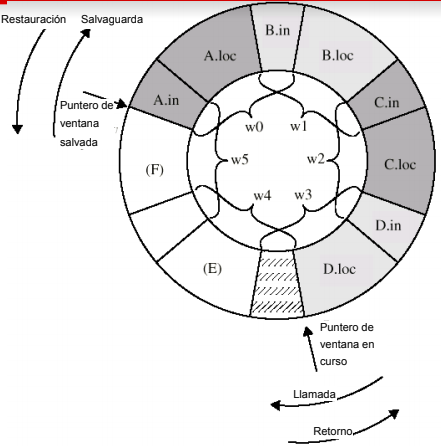
* Registros de parámetros
* Registros de datos locales
* Registros temporales

Los registros temporales de un conjunto se solapan con los registros de parámetros del nivel más bajo adyacente. Esto posibilita que los parámetros se pasen sin que exista transferencia de datos.

### Ventanas de registro solapadas



### Diagrama de buffer circular



### Variables globales

* El compilador asigna posiciones de memoria a las variables
  + Ineficiente para variables globales a las que se accede frecuentemente
* Incorporar al procesador un conjunto de registros para variables globales
  + Dividir registros de la ventana en curso

### Amplio banco de registros vs cache

| Banco de registros amplio | Cache |
| --- | --- |
| Todos los datos escalares locales | Datos escalares locales recientemente usados |
| Variables individuales | Bloques de memoria |
| Variables globales asignadas por el compilador | Variables locales y globales usadas recientemente |
| Salvaguarda/restauración basadas en la profundidad de anidamiento | Salvaguarda/restauración basadas en el algoritmo de reemplazo |
| Direccionamiento de registro | Direccionamiento de memoria |

## Optimización de uso de registros basada en el compilador

* Supongamos un pequeño número de registros (16 o 32)
* El uso optimizado es responsabilidad del compilador
* Los programas HLL no tienen referencias explícitas a los registros
  + Normalmente - pensado en C - registro int.
* Cada “cantidad” del programa candidata se asigna a un registro simbólico o virtual
* Asignar el número ilimitado de registros simbólicos a un número fijo de registros reales
* Registros simbólicos que no se solapan pueden compartir el registro real
* Si se agotan los registros reales, algunas de las variables se asignan a posiciones de memoria
  + En la optimización se usa “coloreado de grafos”

## Por que CISC?

* Simplificación del compilador?
  + Instrucciones de máquina complejas son difíciles de aprovechar
  + La optimización es más difícil ↓tamaño↑velocidad
* Programas más pequeños?
  + El programa ocupa menos memoria, pero la memoria hoy día es muy barata
  + El número de bits de memoria que ocupa no tiene por qué ser más pequeño al tener menos instrucciones
    - Más instrucciones necesitan códigos de operación más largos
    - Las referencias a registros necesitan menos bits
* Programas mas rápidos?
  + Propensión a usar las instrucciones más sencillas
  + Unidad de control más compleja
  + Memoria de control del microprograma más grande
  + Aumenta el tiempo de ejecución de las instrucciones simples
* No está nada claro que la tendencia hacia CISC fuera la apropiada

## Características del RISC

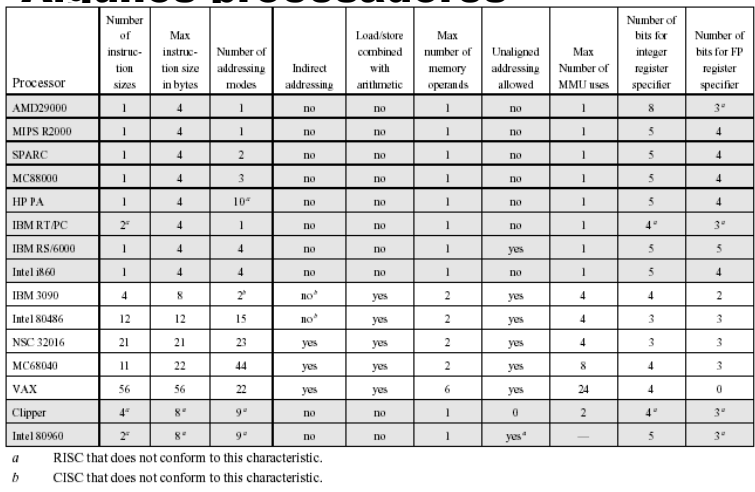
* Una instrucción por ciclo
* Operaciones registro a registro
* Modos de direccionamiento sencillos
* Formatos de instrucción sencillos
* Diseño cableado (sin microcódigo)
* Formato de instrucción fijo
* Mayor tiempo/esfuerzo de compilación

### RISC frente a CISC

No existe una clara barrera diferenciadora.

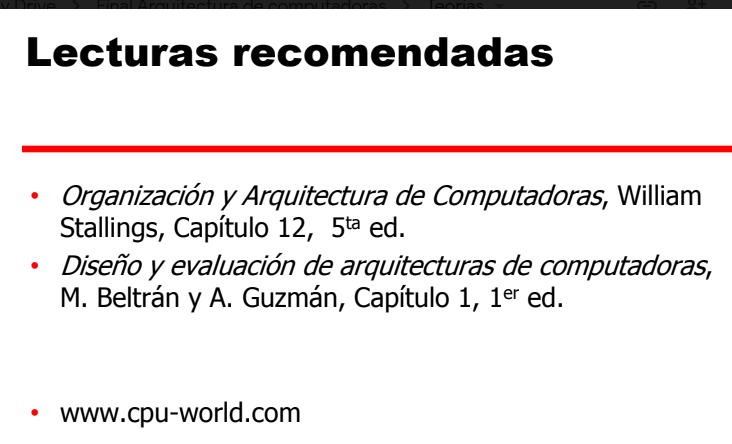
* Muchos diseños incluyen características de ambos criterios (Ej: PowerPC y Pentium II)}

Algunos procesadores:



### Controversia RISC y CISC

* Cuantitativa: Comparación del tamaño de los programas y su velocidad de ejecución
* Cualitativa: Revisión de soporte de lenguajes de alto nivel y uso óptimo de los recursos VLSI
* Problemas de las comparaciones:
  + No existe un par de máquinas RISC y CISC directamente comparables
  + No hay un conjunto de programas de prueba definitivo
  + Dificultad de separar los efectos del hardware de los del compilador
  + Mayoría de comparaciones con máquinas de “juguete”, no con productos comerciales
  + La mayoría de las máqunas son una mezcla de ambas



[Anexo clase 6](https://drive.google.com/file/d/1SVyuYmzUQNeLEnkGUPdz5k-F0vsvOmMa/view?usp=sharing)