Arquitectura de Computadoras
Modulo 2

Joaquin Labarta
labartajoaquin@gmail.com

Índice

1 Segmentacion

1.1 ¿Que es?

La segmentación de cauce (pipelining) es una forma particularmente efectiva de organizar el hardware de la CPU para realizar más de una operación al mismo tiempo. Consiste en descomponer el proceso de ejecución de las instrucciones en fases o etapas que permitan una ejecución simultánea, explotando el paralelismo entre las instrucciones de un flujo secuencial.

1.2 ¿Como se logra?

Una instruccion cuenta con cinco etapas (no siempre utiliza todas), donde cada una utilizará un ciclo de reloj ejecutandose en algun recurso del sistema. Ni bien acabe una etapa, ya estara ejecutandose la siguiente etapa de la proxima instruccion. Las etapas pueden clasificarse como:

- **Captacion de instruccion (Fetch Instruction, FI)**: Lee la supuesta siguiente instruccion en memoria y la almacena en un *buffer* temporal (Program Counter).
- **Decodificar instruccion (Decode Instruction, DI)**: Determina el codigo de operacion y verifica los operandos haciendo uso del banco de registros.
- **Ejecutar instruccion (Execute, EX)**: Se ejecuta la operacion en la ALU y se almacenan los resultados.
- **Acceso a memoria (Memory Access, MA)**: Se accede a memoria si asi se requiere. Muchas veces no es necesario.
- **Escribir operando (WriteBack, WB)**: Se guarda el resultado de la ALU en un banco de registros de ser necesario.

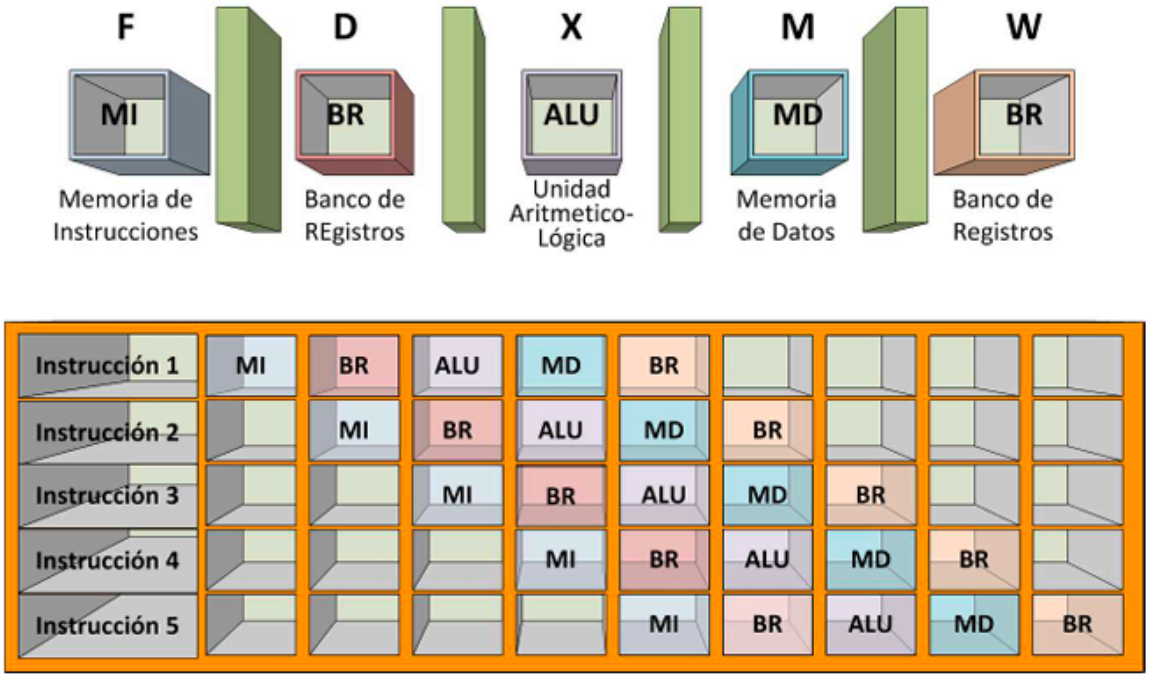


Figura 1: Segmentacion de cauce o pipeline con registros de segmentacion en MIPS

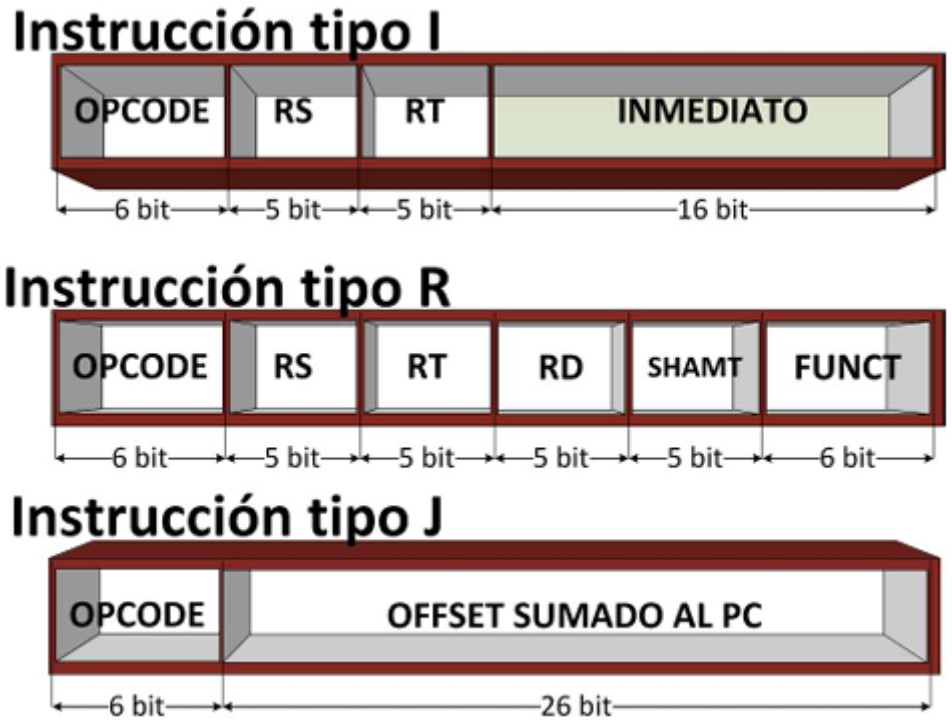


Figura 2: Formato de instrucciones 32 bits

La segmentación es una técnica de mejora de prestaciones a nivel de diseño hardware invisible al programador. Conlleva la necesidad de uniformizar las etapas al tiempo de la más lenta. El diseño de procesadores segmentados tiene gran dependencia del repertorio de instrucciones.

1.3 Ciclos y rendimiento

El máximo rendimiento es completar una instrucción con cada ciclo de reloj. Si K es el número de etapas del cauce:

$$\text{Vel. procesador segmentado} = \text{Vel. secuencial} * K$$

El incremento potencial de la segmentación del cauce es proporcional al número de etapas del cauce. Incrementa la productividad (throughput), pero no reduce el tiempo de ejecución de la instrucción.

Monociclo: en cada ciclo de reloj, se desarrolla una instruccion, terminada esa instrucción se ejecuta en el mismo tiempo la siguiente instrucción, y así sucesivamente. Suponiendo que en cada ciclo de reloj completo se realiza en un ciclo de instrucción.

Multiciclo: en vez de asegurar que cada instrucción se haga en un solo ciclo de instrucción, se aumenta la frecuencia y se hace que una instrucción se realice en múltiplos ciclos de instrucción. La frecuencia de los procesadores cambia, pero no cambia el tiempo de ejecución.

1.4 Registros de segmentacion

Los registros de segmentacion separan estrictamente las etapas del hardware. Esto se hace con la finalidad de que los ciclos de reloj sean siempre los minimos para cada segmento del programa, guardando la informacion en bits de la etapa anterior. Estos registros, al estar controlados por ciclos de reloj, no necesitan un controlador externo que administre el arbitraje de los buses internos.

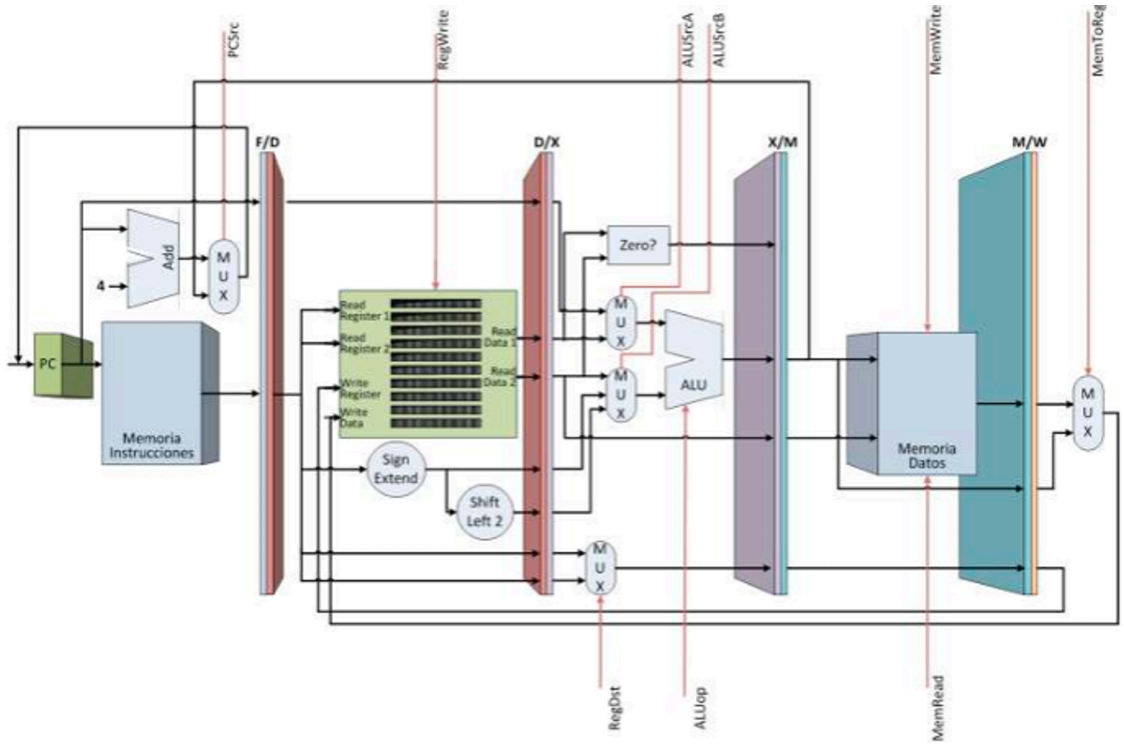


Figura 3: Ruta de datos con registros de segmentacion

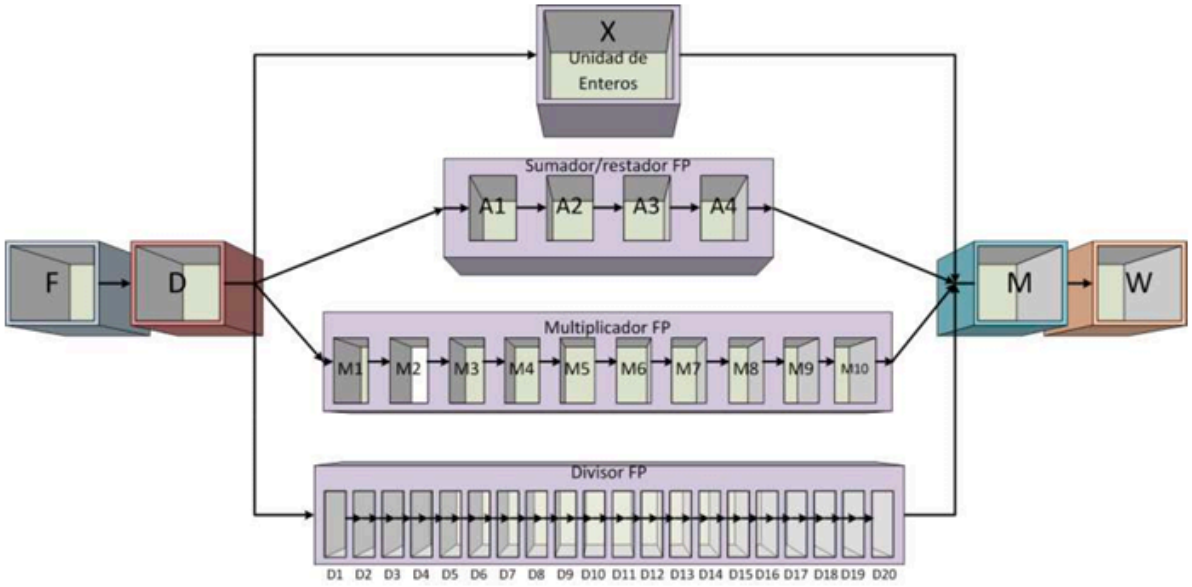


Figura 4: Segmentacion de cauce en MIPS

1.5 Detalles importantes

1. Las direcciones comunmente son de 32 bits, osea 4 bytes.
2. Solo se acepta un modo de direccionamiento, REG - REG. Es indirecto y con desplazamiento. Está prohibido el acceso a memoria.
3. Los registros de almancenamiento interno (IR, MAR, MBR, etc) siguen estando, pero ocultos al programador. Trabajan de igual forma que en Von Neumann pero no se los manipula.

2 Atascos (stalls)

Existen varios tipos de atascos que pueden llegar a hacer que la maquina no cumple con el funcionamiento deseado.

2.1 Estructurales

Son provocados por conflicto de recursos. Usualmente se toma un ciclo de parada. Posibles soluciones:

- Duplicar recursos del hardware.
- Separar memoria de datos e instrucciones.
- Turnar acceso a registros, haciendo escritura en medio ciclo y lectura en el otro medio ciclo.

2.2 Dependencia de datos:

Condición en la que los operandos fuente o destino de una instrucción no están disponibles en el momento en que se necesitan en una etapa determinada del cauce. Existen 3 tipos:

- **Read and Write** (**RAW** o dependencia verdadera): Se produce en la etapa de Decode al ver que el dato solicitado todavia no fue escrito. La solucion por software consiste en agregar una instruccion NOP y «ganar» un ciclo de reloj mas, o tambien, siempre esta la opcion de reordenar el codigo. La solucion por hardware es habilitar el forwarding.
- **Write and Read** (**WAR** o dependencia de salida): Una instruccion modifica un dato antes que otra anterior que lo tiene que leer lo lea.
- **Write and Write** (**WAW** o antidependencia): Esta situacion solo se da si se deja que las instrucciones adelanten unas a otras. Se produce porque un dato se actualizo luego de una instruccion posterior.

  Forwarding: Consiste en pasar directamente el resultado obtenido con una instrucción a las instrucciones que lo necesitan como operando. Si el dato necesario está disponible a la salida de la ALU (Xi) se lleva a la entrada de la etapa correspondiente (Xi+1) sin esperar a la escritura (Mi o Wi). Fácil de implementar si se identifican todos los adelantamientos y se comunican a los registros de segmentación correspondientes.

2.3 Dependencia de control:

Ocurren cuando se ejecuta una instruccion que no debia ejecutarse. Los saltos son detectados en la etapa de Decode. Siempre tendremos una penalizacion por salto de un ciclo, no hay salida.

- Saltos **incondicionales**: en la etapa de Decode, se ve que hay un salto en la instruccion y lo ejecuta, desechando la instruccion que se habia empezado a ejecutar luego.
- Saltos **condicionales**: en la etapa de Ejecucion se da un salto en una instruccion anterior en caso de que una operacion se concrete. Esto trae un problema ya que el programa va a seguir procesando instrucciones hasta que esta condicion se concrete. Perdemos tantos ciclos como tarde la condicion en ejecutarse.

2.3.1 Prediccion de saltos

Se pueden usar varias tecnicas para predecir si un salto va a producirse o no.

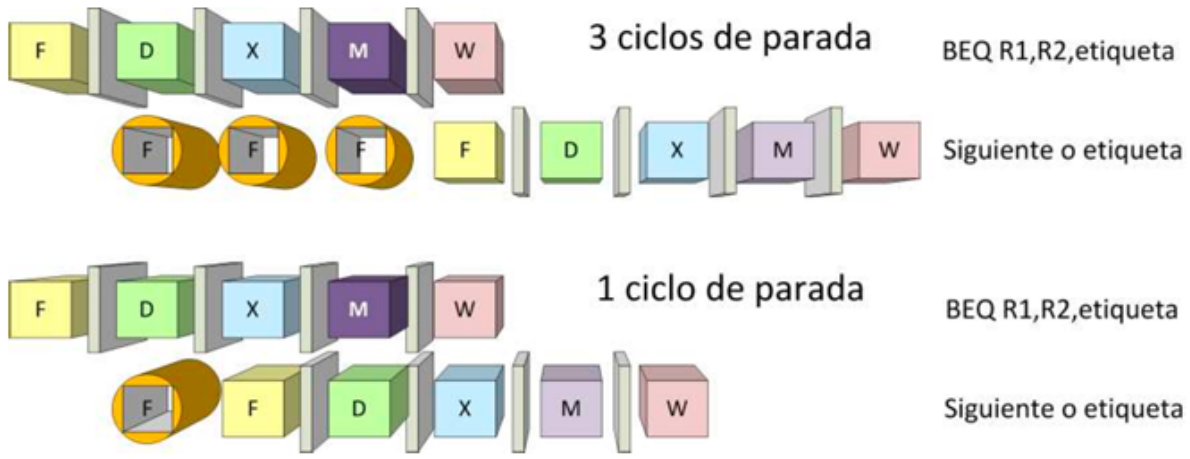


Figura 5: Ciclos por atasco

Estaticas:

- Predecir que nunca se salta (1 ciclo de penalizacion)
- Predecir que siempre se salta

Dinamicas:

- Conmutador saltar/no saltar
- Tabla historica de saltos (BTB)

2.3.1.1 Tabla historica de saltos (BTB)

Tambien llamada Branch Target Buffer, asociado a la etapa de Fetch, guarda un bit de historia que dice si el salto fue tomado o no la última vez. Dado que en un bit solo almacena el estado del salto anterior, su actualización es muy violenta, cambiando la predicción de un salto solo por un comportamiento puntual. Este metodo conlleva un error de dos casos: la primera y la ultima.

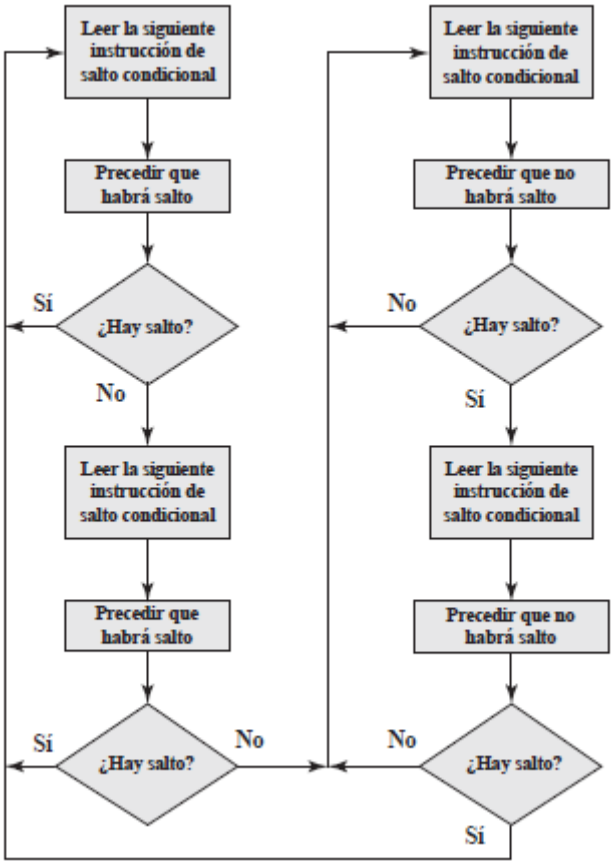


Figura 6: Diagrama de flujo de la prediccion de saltos

2.3.1.2 Salto retardado (Delay Slot)

Es una tecnica por software que consiste en ejecutar SIEMPRE la siguiente instruccion luego del salto. Se usa con el concepto de trabajo util (useful work).

2.3.2 Otras soluciones hardware

Para que el procesador no este «ocioso» esperando una respuesta, se tienen tecnicas por hardware para tratar estos saltos:

- **Predecir segun el codigo de operacion**: Hay instrucciones con más probabilidades de saltar. La tasa de acierto puede llegar a alcanzar un 75%.
- **Precaptacion del destino**: Se precapta la siguiente instruccion despues del salto y algunas mas, asi, cuando la condicion esté evaluada, el destina ya estaria guardado.
- **Buffer de bucles**: Es una pequeña memoria de gran velocidad gestionada por la etapa de captacion de instrucciones, que contiene secuencialmente, las n instrucciones captadas mas recientemente. Si se va a producir un salto, el hardware comprueba en primer lugar si el destino está en el buffer. En ese caso, la siguiente instruccion se capta del buffer. Su funcionamiento es parecido a una cache, de mucho menor tamaño y mas barato por lo mismo. Es muy eficaz para pequeños saltos y bucles.
- **Flujos multiples**: Varios cauces (uno por cada opción de salto) que precaptan cada salto en diferentes cauces. Se debe utilizar el cauce correcto luego de evaluada la condicion. Tiene ciertas desventajas:
  - Provoca retardos en el acceso al bus y a los registros.
  - Si hay múltiples saltos, se necesita un mayor número de cauces.

3 RISC y CISC

3.1 Historia

Uno de los pioneros en estos procesador fue IBM implementandolos por el 1950, buscaba introducir los siguientes conceptos:

Concepto de familia: Separa la arquitectura de una máquina de su implementación. Se ofrece un conjunto de computadores, con distintas características en cuanto a precio/prestaciones, que presentan al usuario la misma arquitectura. Las diferencias en precio y prestaciones se deben a las distintas implementaciones de la misma arquitectura.

Unidad de microcontrolador programada: Las instrucciones de una computadora son implementadas por un conjunto de microinstrucciones almacenadas en una memoria de control. Cada instruccion era un pequeño programa. La microprogramación facilita la tarea de diseñar e implementar la unidad de control y da soporte al concepto de familia.

Memoria cache: La introducción de memoria caché mejoró el rendimiento del sistema al reducir el tiempo necesario para acceder a los datos más frecuentemente usados, almacenándolos más cerca del procesador.

Memoria RAM: El sistema de memoria dejo de ser grandes gabinetes para hacerse mas pequeña , bajo el costo de infraestructura para hacer una computadora.

Procesadores multiples: Se refieren a sistemas con más de un procesador que trabajan de manera conjunta.

3.2 CISC (Complex Instruction Set Computer)

3.2.1 Finalidad

La finalidad de los procesadores CISC es ejecutar tareas complejas mediante un conjunto de instrucciones robusto y variado. Esto permite que una sola instruccion realice varias operaciones a la vez, simplificando la programación y reduciendo el número de instrucciones necesarias para ejecutar una tarea.

3.2.2 Inconvenientes

- Nivel de lenguaje más complicado por tener mas posibilidad de instrucciones.
- Software más caro que hardware: esta manera de hacer los procesadores físicamente lo que lograba era que el software siempre fuera mucho más caro que el hardware. Con lo cual como el software siempre cambiaba porque se creaban nuevos lenguajes, generando un salto semántico.

Salto semántico: este salto se refiere a las cosas que se querían hacer con descripciones en alto nivel (case, instrucción complicada) y las cosas que el hardware podría implementar dentro de la arquitectura, entonces para tratar de hacer viable estos procesadores, se obligó a tener repertorios e instrucciones cada vez más grandes. Es decir, desde operaciones sencillas hasta operaciones complejas que en realidad eran secuencias de operaciones simples pero agrupadas en una instruccion.

Esto nos llevó a realizar estudios que se fundamentaran en analizar qué operaciones se realizaban con mas frecuencia para establecer cómo seria el funcionamiento del procesador y la interacción con la memoria.

Otro estudio que había que hacer era, ver que operandos se usaban en esas operaciones que queríamos realizar en alto nivel. Con esto teníamos los tipos de operandos y su frecuencia de uso. Esto nos permitiría organizar la memoria y saber cuales son los modos de direccionamiento útiles para la mejor implementación o ejecución de estos programas.

Ademas, habia que analizar como secuenciamos el acceso a los operandos y la ejecución de esas operaciones. La secuencia de las cosas que se van a hacer para poder ejecutar una instrucción nos lleva a mejorar la organización de cómo va a ser el control de las instrucciones y así como si queríamos optimizar la ejecución en el hardware como iba a ser el cauce que tendrían que tener esas instrucciones.

3.2.3 ¿Por que CISC?

La arquitectura CISC presentó dos supuestas ventajas con respecto a sus competencias:

- **Simplificacion de compiladores:** Para cada instruccion compleja se asigna en el compilador un algoritmo escrito en lenguaje maquina, lo que lo hacia eficiente si se busca mas exactitud en la busqueda de operacion y operandos, pero ineficiente ya que se necesita en el compilador tantos algoritmos como posibilidades de instruccion.
- **Esperanza de mayor rapidez:** Como el programa ocupa menos memoria, existe la posibilidad de que al haber menos instrucciones sea menos el trafico de bytes, agilizando el proceso.

Sin embargo, todo esto puede no cumplirse. Debido a que hay mas instrucciones en un CISC, se necesitan codigos de operacion mas largos para identificar que operacion va a realizarse. Tambien, el tener presente programas mas pequeños, puede no asegurar mayor rapidez, debido a que la UC debe hacerse mas compleja, y/o la memoria de control del microprograma ha de hacerse mas grande para acomodar un repertorio de instrucciones mas rico. Cualquiera de los dos factores aumenta el tiempo de ejecucion de las instrucciones sencillas.

3.2.4 Conclusiones sobre estudios

A partir de los estudios realizados, surgen 3 consecuencias con las que se busca optimizar estos procesadores:

1. **Mas registros**: Se demostro que para optimizar la busqueda y recepcion de los datos almacenados, es mas eficiente contar con un amplio banco de registros y que el direccionamiento sea unicamente entre ellos y no el acceso a memoria.
2. **Cause sencillo:** Para optimizar las bifurcaciones en los saltos, se vio que un diseño simple de cauce (sin adelantamiento ni prediccion) es mucho mas ineficiente ya que hay que descartar gran parte de la precaptacion de instrucciones.

3.3 Procesadores RISC

El conjunto de procesadores RISC, cuentan con un tipo de diseño de con las siguientes características fundamentales:

- Instrucciones de tamaño fijo y presentadas en un reducido número de formatos. *Ver figura 2.*
- Solo las instrucciones de carga y almacenamiento acceden a la memoria de datos.
- Gran banco de registros.

El objetivo de diseñar máquinas con esta arquitectura es posibilitar la segmentación y el paralelismo en la ejecución de instrucciones y reducir los accesos a memoria.

	Computador de repertorio complejo de instrucciones (CISC)			Computador de repertorio reducido de instrucciones (RISC)	
Característica	IBM 370/168	VAX 11/780	Intel 80486	SPARC	MIPS R4000
Año de desarrollo	1973	1978	1989	1987	1991
Número de instrucciones	208	303	235	69	94
Tamaño de instrucción (bytes)	2-6	2-57	1-11	4	4
Modos de direccionamiento	4	22	11	1	1
Número de registros de uso general	16	16	8	40-520	32
Tamaño de la memoria de control (Kb)	420	480	246	—	—
Tamaño de caché (KB)	64	64	8	32	128

Figura 7: Caracteristicas RISC y CISC

3.3.1.1 Amplio banco de registros vs cache:

Los registros permiten que la CPU tenga acceso instantáneo a los operandos asegurando una velocidad máxima. La memoria caché, por otro lado, ofrece una manera eficiente de acceder rápidamente a datos que se utilizan con frecuencia, aunque no con la misma inmediatez que los registros, ya que se necesita acceder a memoria para la captacion de los mismos.

3.3.1.2 Ventana de registros

Un conjunto amplio de registros debería reducir el acceso a la memoria, pero su organización es crucial. Las variables locales a cada procedimiento cambian en cada llamado y retorno, lo que complica el uso de registros. La solución es tener varios conjuntos pequeños de registros, cada uno para un procedimiento, con ventanas de registros solapadas para pasar parámetros sin transferir datos. Este diseño utiliza un buffer circular de ventanas para manejar las llamadas recientes, mientras que las más antiguas se guardan en la memoria.

Figura 8: Ventana de registros solapadas

3.3.1.3 Variables globales

- **Opcion 1**: El compilador asigna posiciones de memoria a las variables. Es ineficiente para variables globales a las que se accede frecuentemente.
- **Opcion 2:** Incorporar al procesador un conjunto de registros para variables globales. Se trata de dividir registros de la ventana en curso

3.3.2 Caracteristicas de RISC

- Una instruccion por ciclo.
- Direccionamiento Reg - Reg.
- Modos de direccionamiento sencillos. (reg - reg, con desplazamiento y otros por software)
- Formatos de instruccion sencillos.

3.3.3 Caracteristicas RISC y CISC en comun

Hay algunas caracteristicas que comparten entre las dos arquitecturas, por ejemplo:

- Un unico tamaño de instruccion, tipicamente 4 bytes.
- Pocos modos de direccionamiento, menor a 5.
- No hay operaciones que combinen carga/almacenamiento con calculos aritmeticos.
- No se direcciona mas de un operando de memoria por instruccion.
- Las operaciones de carga/almacenamiento no admiten una alineacion de datos arbitraria.
- No se usa direccionamiento indirecto que requiera acceso a memoria.

La principal diferencia entre estas dos arquitecturas es el formato de la instruccion.

3.3.4 Controversia

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 para separar los efectos del hardware de los del compilador.
- Mayoría de comparaciones con máquinas de “juguete”, no con productos comerciales.
- La mayoría de las máquinas son una mezcla de ambas.

4 Procesadores superescalares

Una implementacion superescalar en un procesador es aquella en la que las instrucciones comunes pueden iniciar su ejecucion simultaneamente y ejecutarse de manera independiente. Esta implementacion se puede usar tanto en procesador RISC como CISC. El enfoque principal de esta implementacion es poder ejecutar varios cauces en simultaneo.

Esta implementacion puede conllevar la duplicacion de recursos para por ejemplos:

- Captar instrucciones en simultaneo
- Ejecutar multiples operaciones en distintas ALU

Por lo tanto, el grado de paralelismo y la aceleracion aumentan significativamente, ya que se ejecutan mas instrucciones en paralelo.

4.1 Enfoque supersegmentado

Enfoque supersegmentado se basa en la utilizacion de sub intervalos de ciclos de reloj para cada instruccion, para asi poder utilizar al maximo los recursos sin dedicarle el tiempo maximo a cada instruccion. Habia que agregar mas relojes y generaba condiciones complejas para controlarlos.

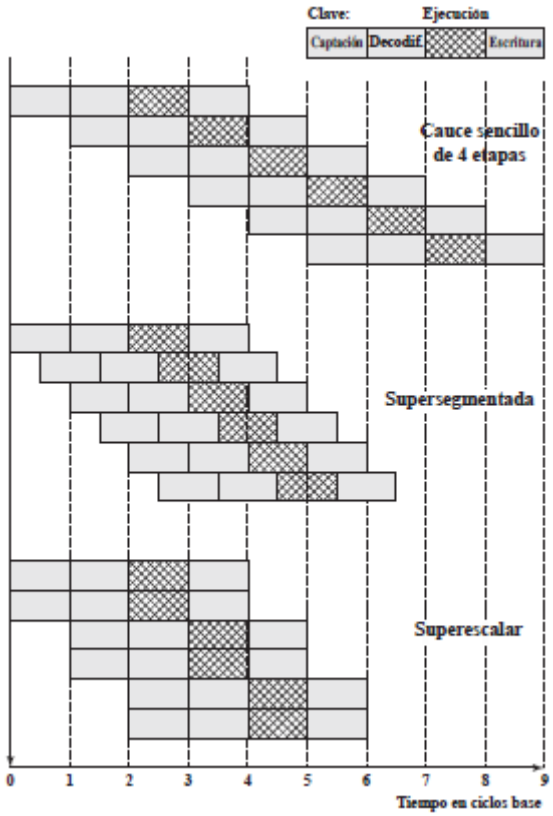


Figura 9: Comparacion de ciclos en distintas implementaciones

4.2 Limitaciones

- Dependencia de datos.
  - Verdadera
  - De salida
  - Antidepencia
- Dependencia relativa al procedimiento (saltos).
- Conflictos en los recursos.

Las dependencias de salida y las antidependecias surgen porque los valores de los registros no pueden reflejar ya la secuencia de valores dictada por el flujo del programa.

4.3 Paralelismo

Existe un paralelismo en las instrucciones cuando estas son independientes y pueden ejecutarse en paralelo solapandose. El paralelismo depende de la *latencia* de una operacion, que determina cuanto retardo causara un atasco en ser resuelto. Otro concepto importante es el paralelismo de la maquina, que es una medida de la capacidad del procesador para sacar partido del paralelismo de instrucciones.

4.4 Envio de instrucciones

Existen distintos protocolos para la emision y envio de instrucciones, hay tres aspectos importantes:

- El orden que se captan
- El orden que se ejecutan
- El orden en que las instrucciones actualizan los contenidos de los registros y la posiciones de la memoria.

Las politicas de emision se pueden separar en tres categorias:

- Emision en orden y finalizacion en orden.

Decodificación		Ejecución			Escritura		Ciclo
I1	I2						1
I3	I4	I1	I2				2
I3	I4	I1					3
	I4			I3	I1	I2	4
I5	I6			I4			5
	I6		I5		I3	I4	6
			I6				7
					I5	I6	8

- Emision en orden y finalizacion desordenada.

Decodificación		Ejecución			Escritura		Ciclo
I1	I2						1
I3	I4	I1	I2				2
	I4	I1		I3	I2		3
I5	I6			I4	I1	I3	4
	I6		I5		I4		5
			I6		I5		6
					I6		7

- Emision desordenada y finalizacion desordenada.

Decodificación		Ventana	Ejecución			Escritura		Ciclo
I1	I2							1
I3	I4	I1, I2	I1	I2				2
I5	I6	I3, I4	I1		I3	I2		3
		I4, I5, I6		I6	I4	I1	I3	4
		I5		I5		I4	I6	5
						I5		6

La emision desordenada es la que menos ciclos tarda por misma cantidad de instrucciones, ya que aprovecha cada ciclo para mandar un dato. La velocidad esta relacionada a la sensacion de mejora percibida por la maquina en base a la cantidad de instrucciones que se pueden hacer en un ciclo.

4.5 Prediccion de saltos

La estrategia de salto retardado perdio interes, dado que hay que ejecutar por lo menos dos tareas de trabajo util, haciendo dificil la reagrupacion. Para solucionar esto, se volvio a la tecnicas comunes de prediccion, como las estaticas o la historia de saltos.

4.6 Renombramiento de registros

Una solucion tradicional para la solucion a los atascos es la duplicacion de recursos. La tecnica de renombramiento de registros basicamente asigna dinamicamente los valores que necesitan las instrucciones en registros dinamicos no percibidos al programador. Las intrucciones posteriores acceden a ese valor como operando sufren un proceso de renombramiento en cuanto a la referencia de los registros.

R3b:= R3a op R5a (1)
R4a:= R3b + 1 (2)
R3c:= R5a + 1 (3)
R7a:= R3c op R4a (4)

Riesgos:

Sólo quedan los riesgos RAW (dependencia verdadera).
La salida de (1), R3b, es uno de los operandos en (2).
La salida de (3), R3c, es uno de los operandos en (4).

Figura 10: Renombramiento de registros y riesgos

4.7 Ejecucion

El diagrama muestra como es la ejecucion sin importar con que politica se enviaron los datos, ejecutando el envio de instrucciones y la emision dependiendo del hardware disponible y no del orden que traian. Al final del proceso, se ve que se usa una etapa de *reordenacion y entrega* que dictará el orden en que las ejecuciones deberian ser entregadas para no generar dependencias ni errores.

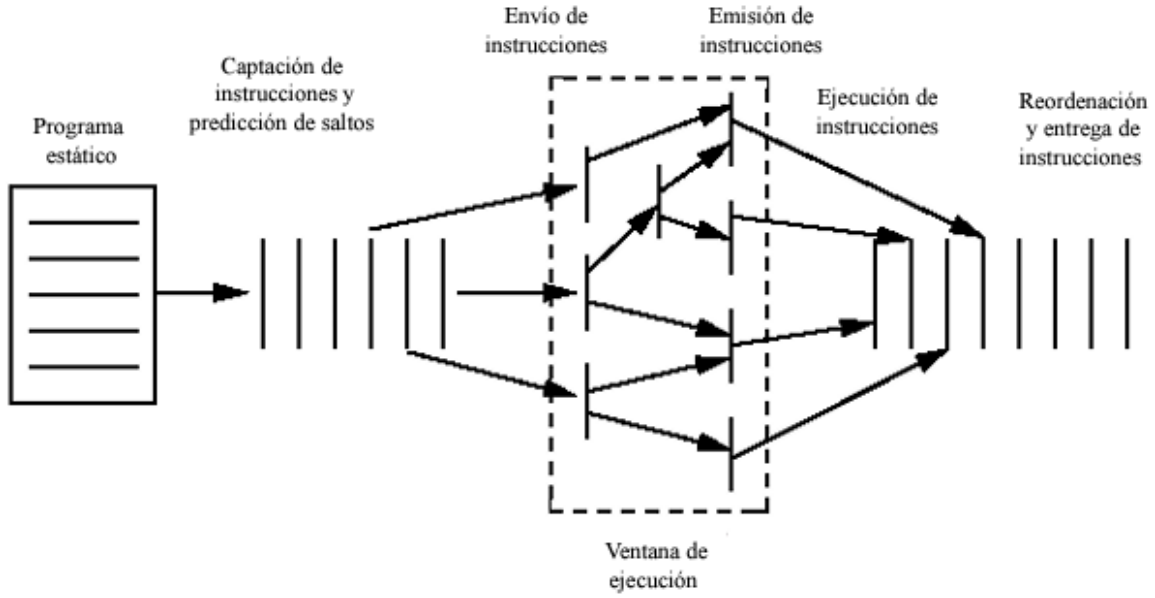


Figura 11: La barra del medio representa el renombramiento, no significa un war o raw, sino que espera a que algun registro este libre (depende la arquitectura).

4.8 Implementacion superescalar

- Estrategias de captacion de instrucciones en simultaneo, ejecutando la decodificacion y prediciendo saltos.
- Logica para determinar dependencias verdaderas entre valores de registros.
- Mecanismos para iniciar o emitir múltiples instrucciones en paralelo.
- Recursos para la ejecución en paralelo de múltiples instrucciones.
- Mecanismos para entregar el estado del procesador en un orden correcto.

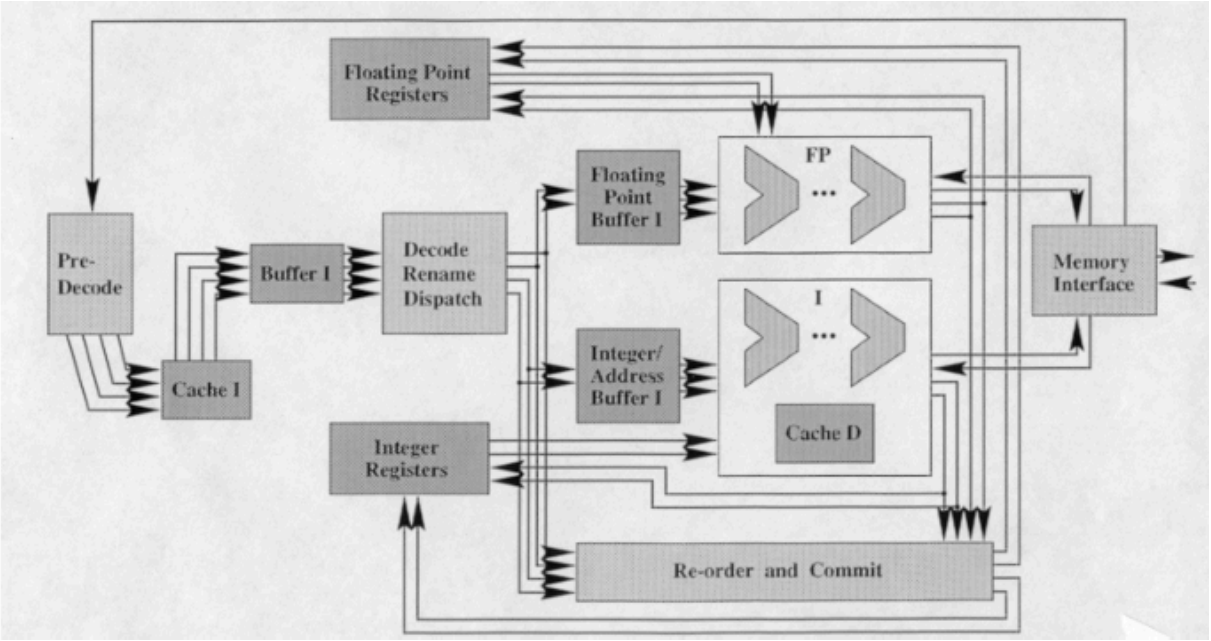


Figura 12: Procesador superescalar

4.9 Consideraciones

En el caso de que ocurra una interrupcion, al finalizarla, debo volver a donde estaba ya que puede haber una instruccion que termino y la otra no. Es mas probable que haya una excepcion (error en instruccion) a que una interrupcion. Por este motivo, el envio de instrucciones se tiene que hacer ordenado, ya que si es desordenado no veriamos donde se produjo el error.

La cpu invalida instrucciones de a pares, entonces para evitar que suceda el caso de que una instruccion termino y otra se haya abortado, se repiten las dos instrucciones hayan o no terminado. Al invalidarse una instruccion por excepcion, se aborta la instruccion creando un gestor que me indica donde esta la excepcion y la instruccion comienza de nuevo.

Si queremos trabajar con interrupciones tenemos que adoptar un ordenamiento inicial (post decode) a pesar de haber mandado instrucciones ordenadas o desordenadas, ya que necesito un medio para saber donde retomar en caso de una excepcion o interrupcion y recomenzar. El controlador es el que limpia el error para volver a los registros originales.

5 Interconexion de buses

Un bus es un camino de comunicacion entre dos o mas dispositivos. Una caracteristica clave de un bus es que se trata de un medio de transmision compartido consituido por varios caminos de comunicacion o lineas representadas por 0 o 1.

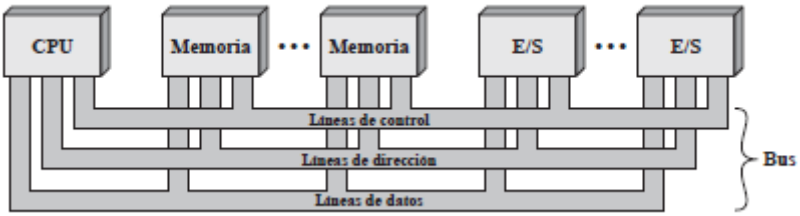


Figura 13: Esquema de interconexion

Fisicamente, el bus es un conjunto de conductores electricos paralelos, que son lineas de metal grabadas en una tarjeta.

Todas las unidades que conforman una computadora tienen que estar interconectadas entre si para lograr una maxima comunicacion. Existen distintos tipos de interconexiones para distintos tipos de unidades:

- Memoria
- E/S
- Procesador

5.1 Memoria

- Recibe y entrega datos.
- Recibe direcciones (ubicación de trabajo).
- Recibe señales de control
  - Leer
  - Escribir
  - Temporizar

5.2 E/S

E/S es funcionalmente similar a la memoria

- Recibe y entrega datos del/al procesador
  - Envía y recibe datos al/del periférico
- Recibe direcciones (ubicación del periférico)
- Recibe señales de control del procesador
  - Envía señales de control al periférico
- Envía señales de control al procesador
  - Interrupción

5.3 Procesador

- Lee instrucciones y datos.
- Escribe datos (los procesados).
- Envía señales de control a otras unidades.
- Recibe (y utiliza) señales de interrupción.

5.4 Bus de datos

El bus de datos esta conformado por varias lineas de comunicacion con otros modulos del sistema y su labor principal es transmitir datos. La anchura es la cantidad de lineas que contiene el bus de datos y determinará cuantos bits en simultaneo pueden transmitirse. Recordar que las lineas solo transmiten de a un bit.

5.5 Bus de direcciones

El bus de direcciones esta conformado por varias lineas de comunicacion con otros modulos del sistema, en donde cada linea se encargara de llevar las direcciones donde haya que llevar los datos que esta transmitiendo el bus de datos. El ancho del bus de direcciones determina la capacidad maxima de memoria en el sistema.

5.6 Bus de control

El bus de control esta conformado por lineas de control que se encargaran de «organizar» el uso del bus de datos y de direcciones de los distintos modulos del sistema. Ademas, por el bus seran transmitidas las señales de temporizacion.

5.7 Uso del bus

Si un modulo desea enviar un dato a otro debe hacer dos cosas:

1. Obtener el uso del bus mediante una peticion.
2. Transferir la señal o dato a traves del bus.

Conectar varios modulos al bus del sistema puede traer retardos de propagacion, debido al tiempo que necesitan los dispositivos para conectarse entre si. La mayoria de los sistemas optan por usar buses multiples, ya que permite pasar mayor cantidad de datos en simultaneo, ahorrando tiempo. Para esto se necesita una organizacion jerarquica.

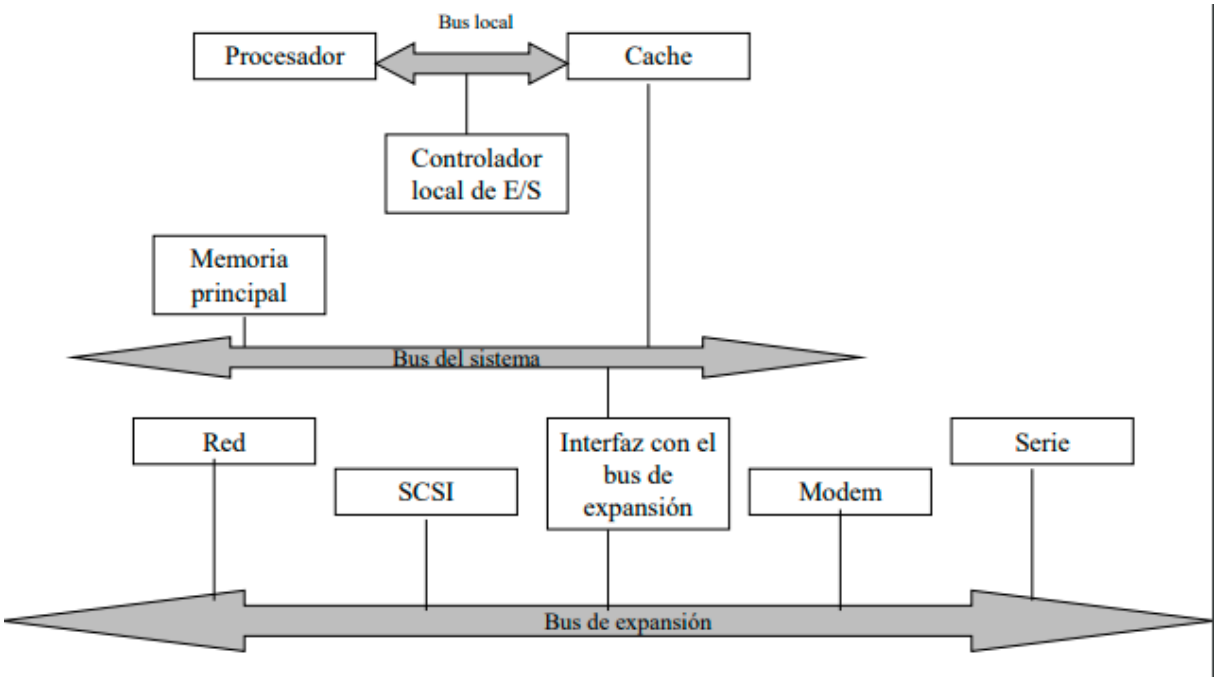


Figura 14: Arquitectura del bus tradicional

5.7.1 Tipos de buses

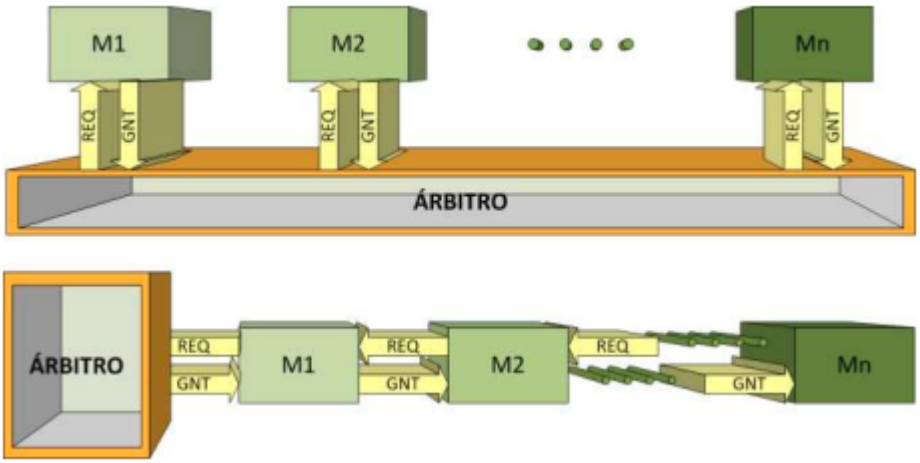
Las lineas de bus se pueden dividir en dedicadas o multiplexadas.

- **Dedicada:** Una linea de bus dedicada esta permanentemente asignada a una funcion o un subconjunto fisico de componentes de la maquina. Por ejemplo la separacion de lineas para la transmicion de datos y de direcciones.
- **Multiplexado:** Consta del mismo conjunto de lineas para transmitir datos y direcciones, y una aparte de control. La ventaja es tener menos lineas pero la circuiteria es mas compleja para cada modulo.

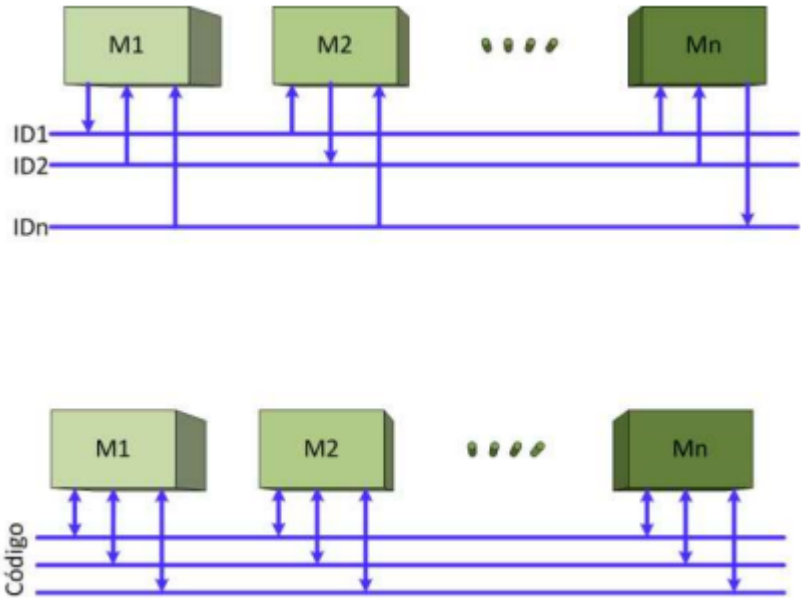
5.7.2 Arbitraje

Si mas de un modulo necesita el acceso al bus, se requiere algun metodo de arbitraje para organizar y que los datos lleguen correctamente a destino.

- **Centralizado:** consta de un arbitro responsable de asignar tiempos en el bus. Puede estar separado o ser parte del procesador.



- **Distribuido:** no existe un controlador central, ya que cada modulo dispone de logica para controlar el acceso y los modulos actuan conjuntamente para compartir el bus. Aqui, cada modulo puede responder al pedido.



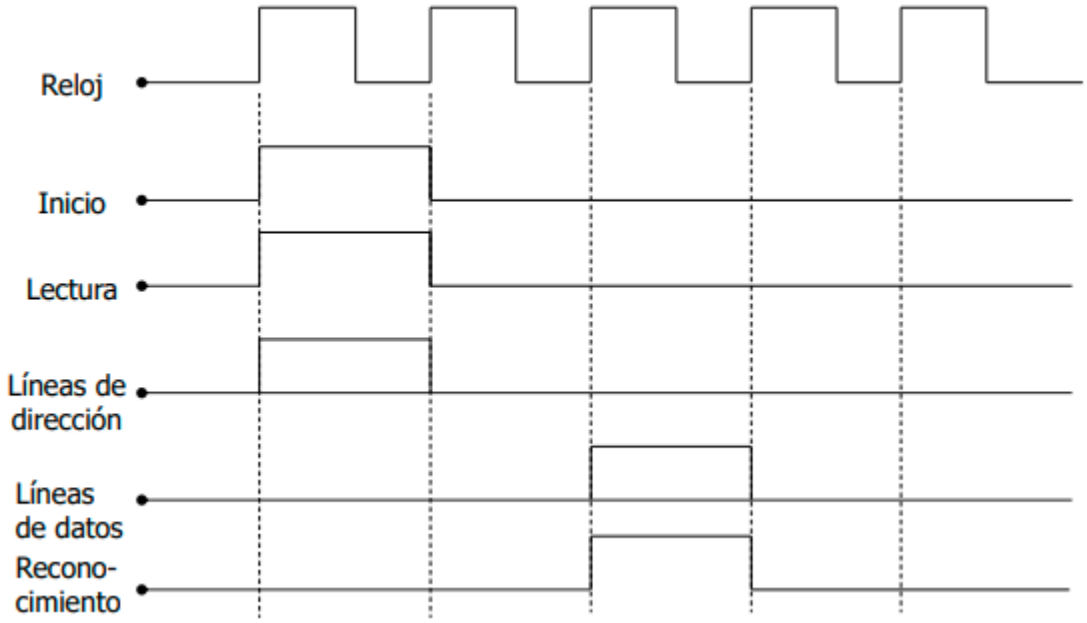
En ambos metodos, es necesaria la designacion de un maestro del bus, para que inicie la transferencia con otro modulo que actua como esclavo.

5.7.3 Temporizacion

La temporización en los buses de comunicación puede ser síncrona o asíncrona.

Temporización Síncrona: Utiliza un reloj para coordinar los eventos, con intervalos regulares que definen ciclos de reloj. Todos los dispositivos del bus sincronizan sus acciones al inicio de cada ciclo de reloj. Es más fácil de implementar y verificar, pero requiere que todos los disposi-

tivos operen a la misma frecuencia, lo que limita la flexibilidad y el aprovechamiento de mejoras en el rendimiento de los dispositivos.



Temporización Asíncrona: No depende de un reloj, sino de la ocurrencia de eventos previos. Esto permite una mayor flexibilidad, ya que dispositivos rápidos y lentos pueden compartir el bus, utilizando tecnologías tanto antiguas como modernas. Aunque más compleja de implementar, facilita el uso de una variedad de dispositivos con diferentes velocidades.