Arquitectura de Computadoras

CURSO 2022

Turno:

Clase 8

Resumen clase 8

Procesadores supersegmentados y superescalares

- Evolución de la segmentación : primeras generaciones
- Procesadores supersegmentados
- Procesadores superescalares
- Paralelismo, IPL, MPL
- Procesadores superescalares
- Políticas de emisión
- Renombrado de registros
- Ejecución superscalar
- Ejemplo de procesador superescalar
- Tratamiento de interrupciones

Evolución de la segmentación

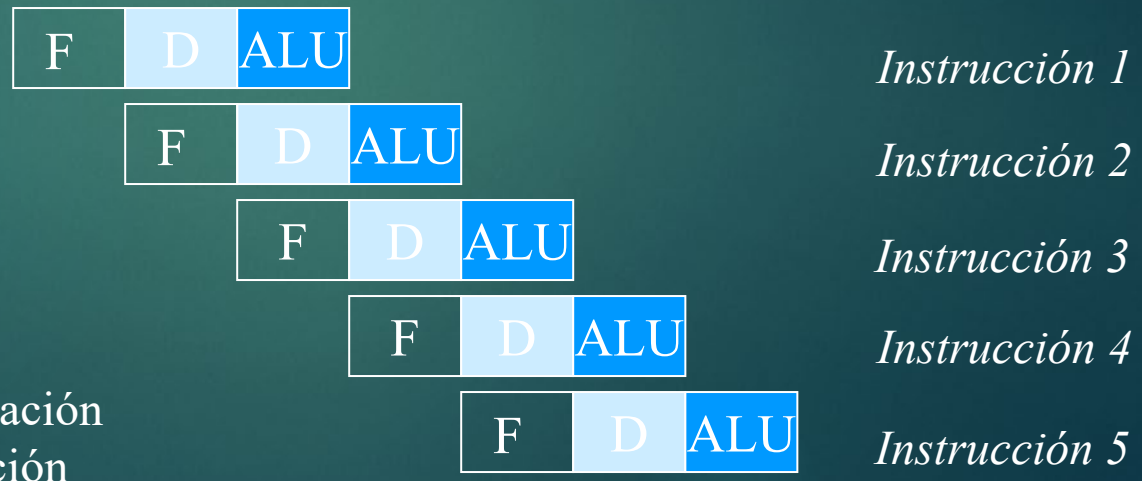
Primera generación – ejecución secuencial

- La primer generación de microprocesadores se caracterizó por un proceso de ejecución completamente secuencial de instrucciones.
- No empezaba una nueva instrucción hasta que se completaba la corriente.
- Ejemplos: i4004,i8008/80, MC6800, MCS6502, Z80, F8 …

Evolución de la segmentación

Segunda generación – ejecución segmentada

- En la segunda generación se empezó a segmentar el cauce, aunque en forma limitada y en muy pocas etapas.
- Nuevas instrucciones podían iniciarse mientras otras estaban en proceso.
- Ejemplos: MC68000, i8086/286, Z8000



Evolución de la segmentación

Tercera generación – ejecución segmentada aumentada

- En la tercera generación la segmentación se profundizó, aumentando significativamente la cantidad de etapas.
- Nuevas instrucciones podían iniciarse mientras otras estaban en proceso.
- Ejemplos: MC68020, i80386, R2000/3000

Evolución de la segmentación

Limitaciones de procesadores segmentados

- En general, un procesador de k etapas tiene una productividad teórica máxima igual a k

  productividad teórica máxima = k

- Pero esa productividad tiene varias limitaciones:
  - La máxima capacidad teórica es 1 instrucción por ciclo (CPI o IPC =1). Es decir, solo puede completar 1 instrucción por ciclo de reloj.
  - Hay 1 solo pipeline para los diferentes tipos (de datos).
  - Los atascos en el cauce producen burbujas innecesarias que reducen la productividad.

Evolución de la segmentación

Rendimiento de procesadores escalares

- El rendimiento de un procesador escalar segmentado depende de:
  - IPC: número de instrucciones por ciclo
  - f: Frecuencia del reloj (T= 1 / f)
  - k: cantidad de etapas del cauce
- El tiempo de ejecución es igual a:

CPU time= N° instr. X IPC x T

donde: T= tiempo de ciclo

- Para estos procesadores, en el mejor de los casos IPC=1, y solo se puede mejorar este tiempo si se aumenta f.

Evolución de la segmentación

Limitaciones en procesadores escalares segmentados

- Las limitaciones en el rendimiento de un procesador escalar segmentado se originan en 2 aspectos:

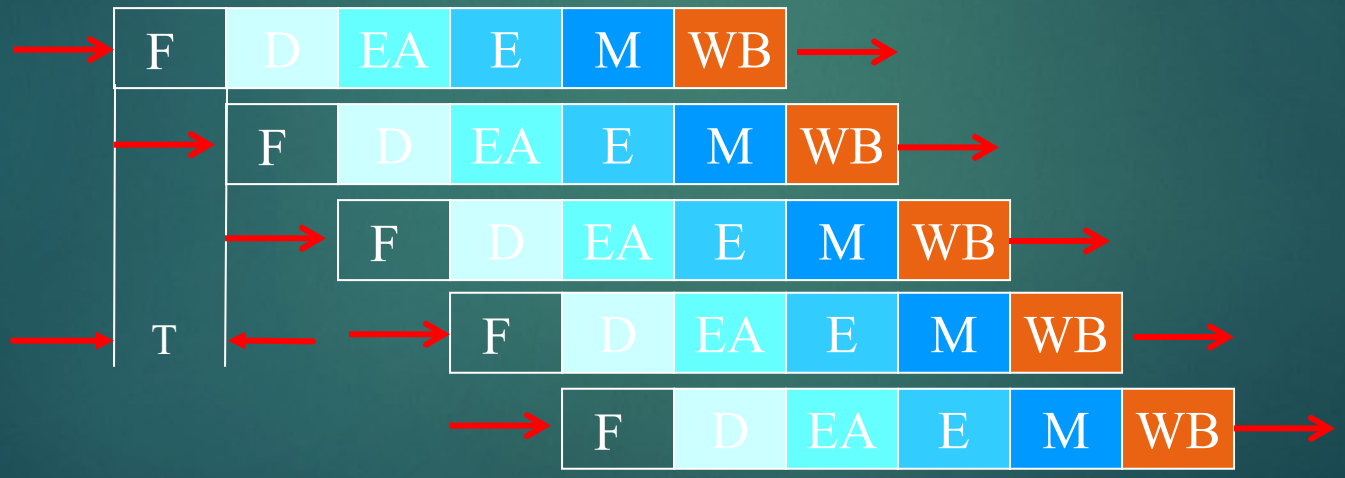
1. El IPC es igual a 1, en el mejor de los casos.
2. Los posibles conflictos que pueden producir atascos en el cauce debidos a:
   - Dependencias de datos: hay 3 tipos de dependencias de datos: Verdadera, de salida, y antidependencia.
   - Penalizaciones en saltos.
   - Conflictos por uso de recursos.

Evolución de la segmentación

Limitaciones en procesadores escalares segmentados

En la figura siguiente se puede apreciar el proceso de ejecución de una secuencia de instrucciones. En cada ciclo T se ingresa una nueva instrucción, y también se resuelve solo una.

F: búsqueda
D: decodificación
EA: dirección efectiva
E: ejecución
M: memoria
WB: post escritura



Existen 2 técnicas para aumentar la performance usando segmentación.

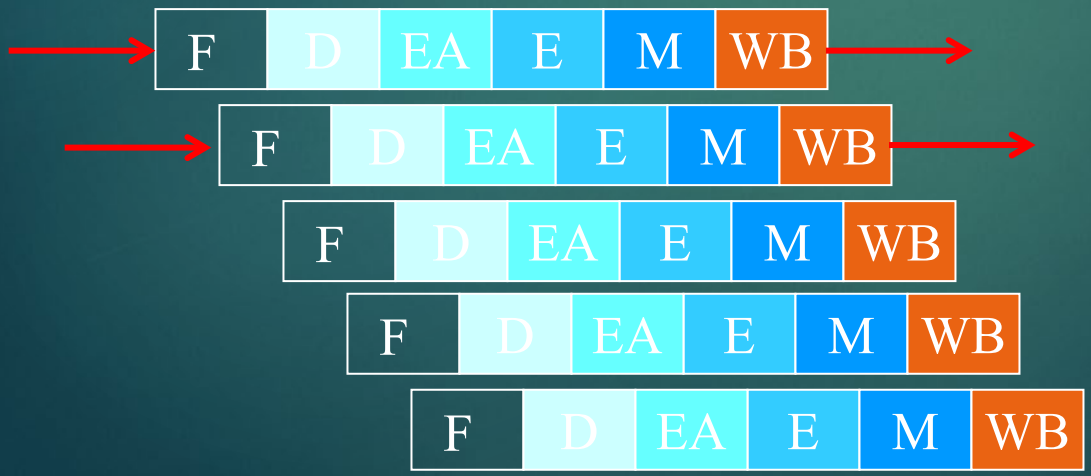
Procesadores supersegmentados y superescalares

1) Procesadores supersegmentados

En la ejecución supersegmentada de instrucciones cada ciclo se divide en fracciones más chicas, en las que se inician nuevas instrucciones.

Por ejemplo, un procesador supersegmentado de grado 2 acepta una nueva instrucción en cada semiciclo de reloj.

F: búsqueda
D: decodificación
EA: dirección efectiva
E: ejecución
M: memoria
WB: post escritura



Procesadores supersegmentados y superescalares

Procesadores supersegmentados

- La ejecución supersegmentada consiste en subdividir cada segmento en partes más pequeñas.
- Como muchas operaciones no necesitan todo un ciclo de reloj, se puede hacer más de una tarea en ese ciclo, subdividiendo el ciclo de reloj en sub-intervalos, lo que es equivalente a usar una mayor frecuencia de reloj (menor ciclo de reloj).
- El tiempo para las instrucciones individuales no varía, pero aumenta el grado del paralelismo de la máquina temporal.
- El pipeline se hace más profundo, pero está limitado por la tecnología (es decir, por la frecuencia máxima).

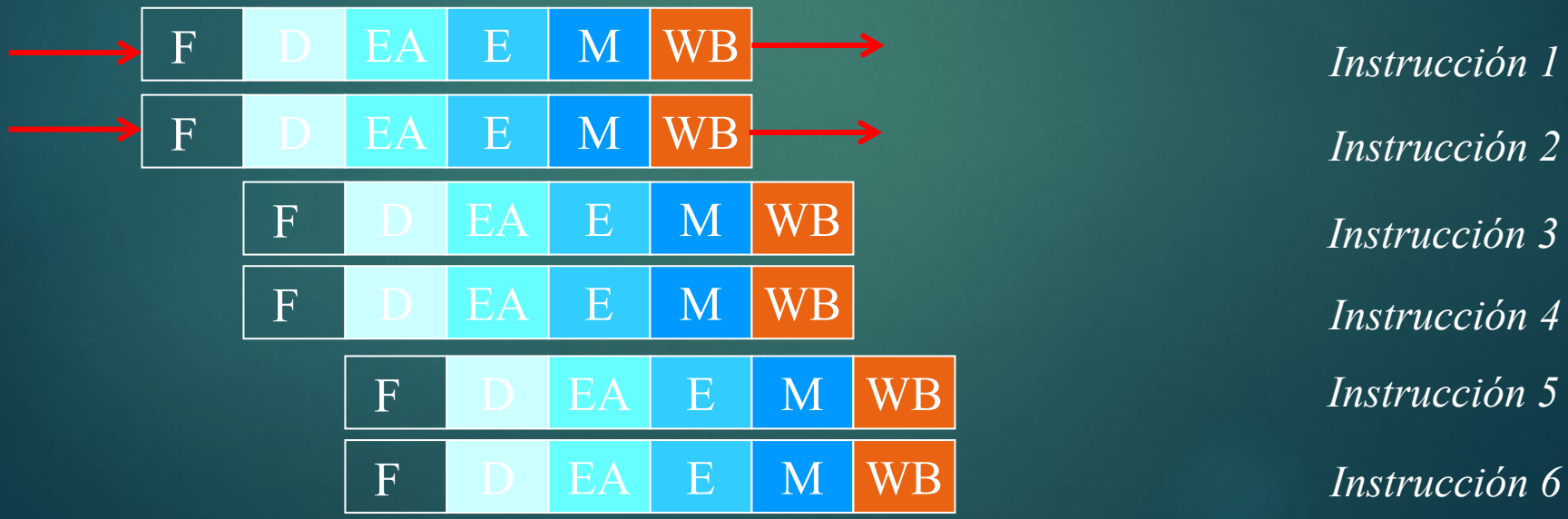
Procesadores supersegmentados y superescalares

2) Procesadores superescalares

En la ejecución superescalar de instrucciones, en cada ciclo se inician más de 1 instrucción.

Por ejemplo, un procesador superescalar de grado 2 inicia 2 nuevas instrucciones en cada ciclo de reloj.

F: búsqueda
D: decodificación
EA: dirección efectiva
E: ejecución
M: memoria
WB: post escritura



Procesadores supersegmentados y superescalares

Procesadores superescalares

- La aproximación superescalar se basa en poder ejecutar varias instrucciones en diferentes cauces de manera independiente y concurrente.
- Algunos ejemplos de procesadores superescalares son: MC68040, i80486, MC88110, i80860,PA-RISC, Sparc, R6000.

Procesadores superescalares

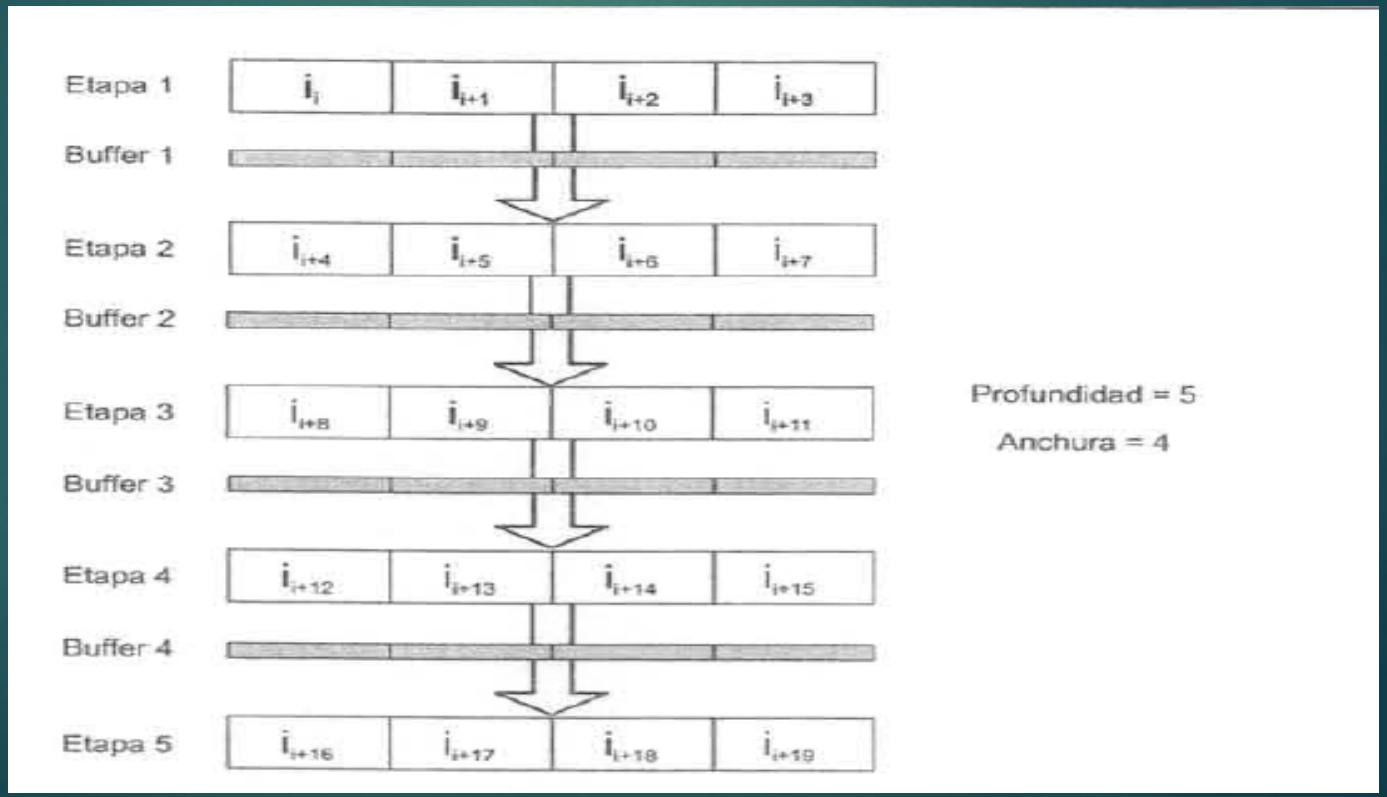
Procesadores superescalares

- Se pueden llevar a cabo (completar) 2 o más instancias de cada etapa de una instrucción simultáneamente.
- Para poder iniciar/ejecutar 2 o más instrucciones simultáneamente, se requiere la duplicación de algunas o todas las partes de la CPU/ALU, por ejemplo:
  - Captación de múltiples instrucciones al mismo tiempo.
  - Ejecución (sumas y multiplicaciones) simultánea.
  - Ejecución de carga/almacenamiento, mientras se lleva a cabo una operación en ALU.
- El grado de paralelismo y, por tanto, la aceleración de la máquina aumentan, ya que se ejecutan más instrucciones en paralelo.

Procesadores superescalares

La aproximación superescalar presenta paralelismo de máquina de 2 tipos: temporal y espacial.

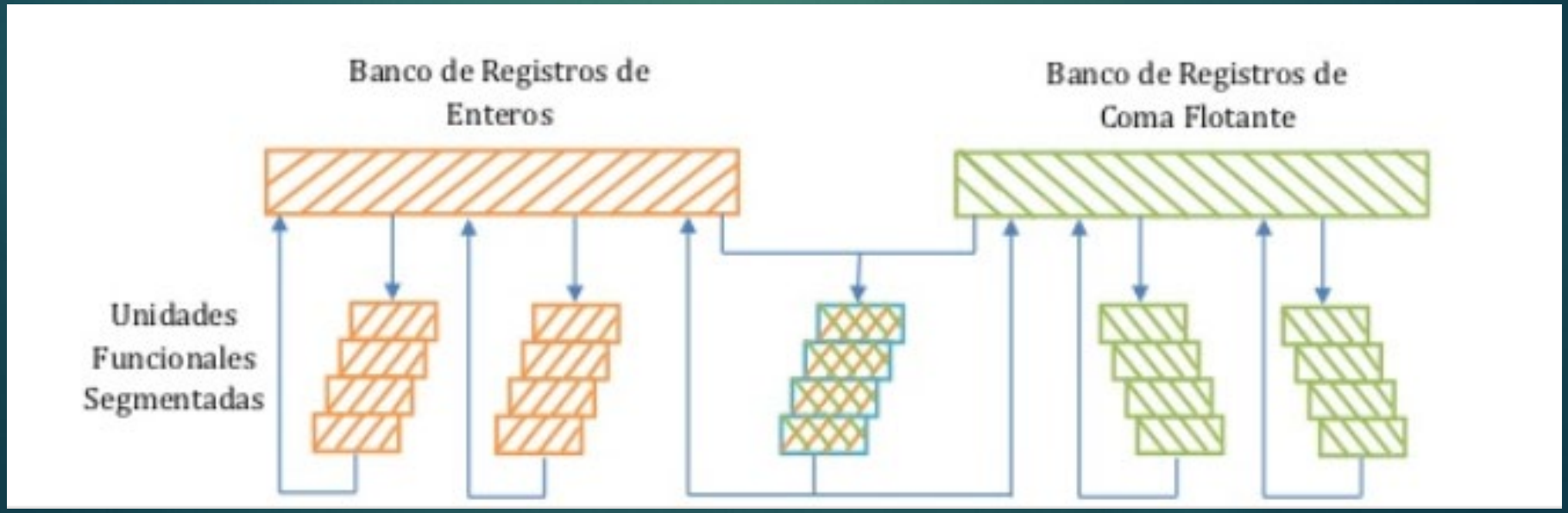
Por ejemplo, una máquina supersegmentada de profundidad 5, y ancho 4 (capacidad de ejecutar hasta 20 instrucciones simultáneamente) tendría la siguiente arquitectura:



Procesadores superescalares

Esquema básico

Un procesador superescalar dispone, básicamente, de multiples unidades funcionales, cada una implementada como un cauce segmentado, que admite la ejecución paralela de varias instrucciones.



Procesadores superescalares

Los conflictos en procesadores superescalares son los mismos que en procesadores segmentados.

I0 Add r1, r2

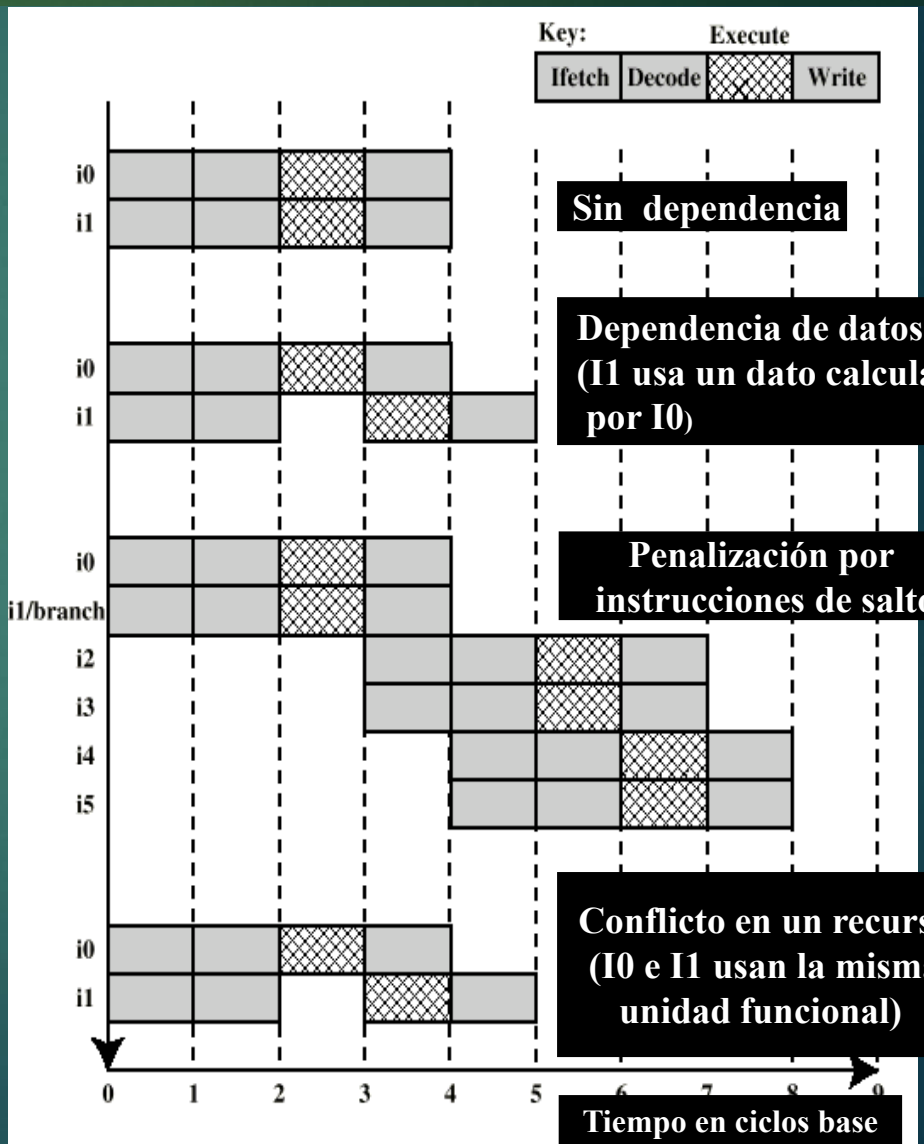
I1 Mov r3, r1

I0 Add r1, r2

I1 Jmp etiqueta

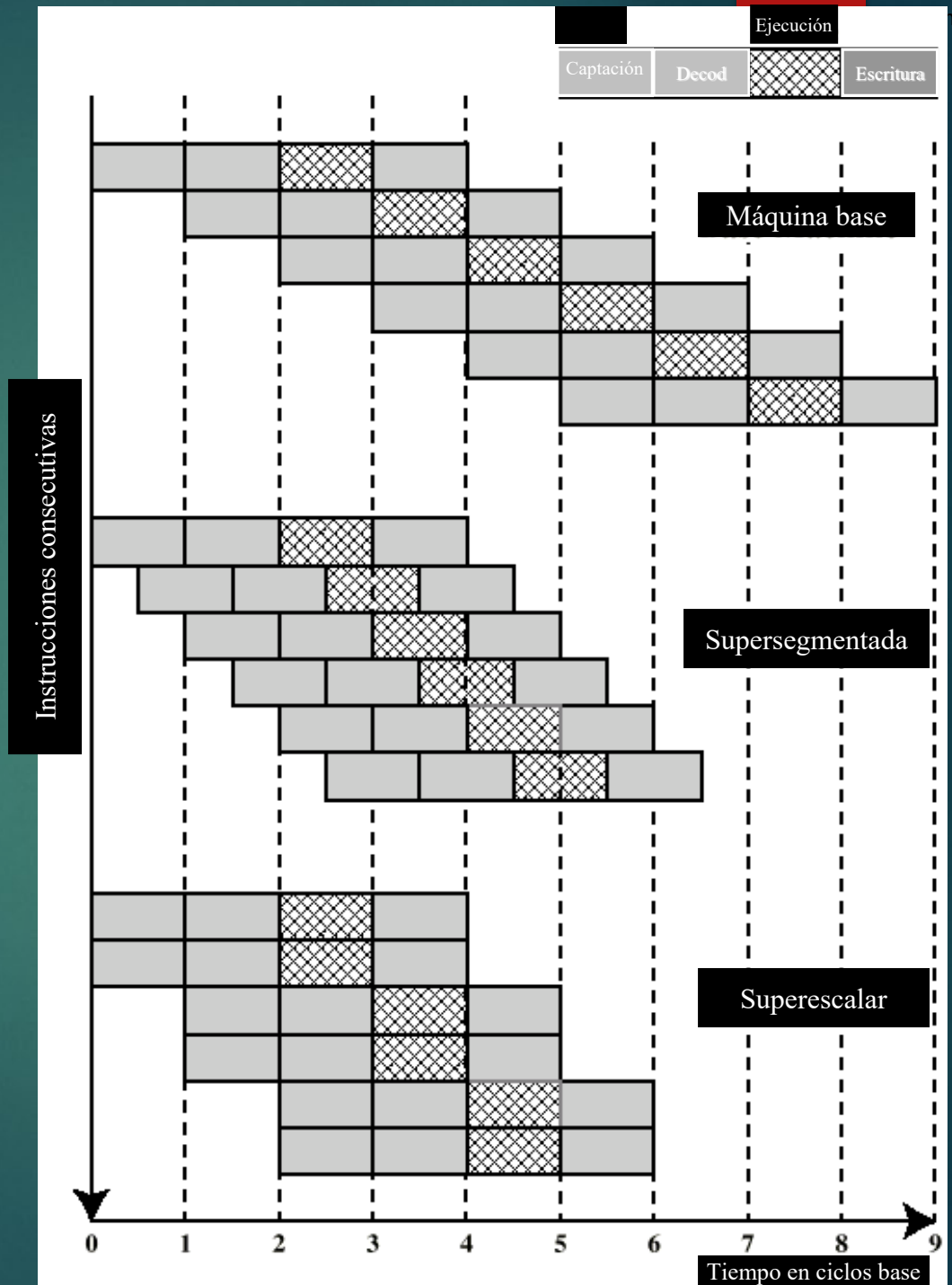
I0 Add r1, r2

I1 Add r4, r3



Procesadores superescalares

El gráfico siguiente muestra una comparación entre las ejecuciones segmentada, superscalar y la supersegmentada.



Paralelismo

- El paralelismo es la capacidad para ejecutar varias tareas en el mismo intervalo de tiempo (en “paralelo”).
- Hay 2 tipos de paralelismo:
  - Paralelismo a nivel de instrucciones: se refiere a las posibilidades que tiene un programa para ejectar instrucciones en paralelo
  - Paralelismo a nivel de máquina: es la capacidad que tiene una máquina para encontrar y ejecutar tareas en paralelo.

Paralelismo

Paralelismo a nivel de programa

- El grado de paralelismo de un programa se mide usando un parámetro conocido como IPL.
- El IPL es la cantidad, en promedio, de instrucciones que pueden ejecutarse en paralelo.
- Las instrucciones pueden ejecutarse en paralelo si son independientes.

```
ADD   R1,R2,R3
SUB   R4,R5,R6
AND   R7,R8,R9
```

Paralelismo de grado 3

```
ADD   R1,R2,R3
SUB   R4,R5,R1
AND   R7,R8,R4
```

Paralelismo de grado 1

Paralelismo

Paralelismo a nivel de máquina

- El grado de paralelismo de una máquina se mide usando un parámetro conocido como MPL, que es la capacidad que tiene una máquina para aprovechar el paralelismo de un programa.
- El MPL es función de:
  - Número de instrucciones captadas por ciclo.
  - Número de unidades funcionales.
  - Mecanismos para localizar y ejecutar instrucciones independientes.

Procesadores superescalares

Tratamiento de las instrucciones

- En los procesadores superescalares el objetivo fundamental es localizar instrucciones que puedan ser introducidas al pipeline y ejecutadas. En este proceso de detección, es necesario distinguir:
  - El orden en que se captan las instrucciones.
  - El orden en que se ejecutan las instrucciones.
  - El orden en que las instrucciones actualizan los registros y las posiciones de memoria.
- Cuanto más sofisticado es el procesador, menos restricciones impone a estos ordenamientos. Incluso puede alterar cualquier ordenamiento, respecto del estrictamente secuencial. La única condición es que el resultado debe ser correcto.

Políticas de emisión

Tratamiento de las instrucciones

- Las políticas de emisión de instrucciones son los protocolos usados para el envío de las instrucciones a las unidades funcionales, es decir, definen la forma en que se captan, ejecutan y finalizan (escriben los resultados) las instrucciones.
- En función del orden en que se captan, ejecutan y terminan las instrucciones, existen 3 políticas de emisión de instrucciones:
  - Emisión y finalización ordenada
  - Emisión ordenada y finalización desordenada
  - Emisión y finalización desordenada

Políticas de emisión

Políticas de emisión de instrucciones

- Para analizar las políticas de emisión de instrucciones, vamos a considerar un procesador superescalar con un cauce de 3 segmentos compuesto por:
  - Búsqueda de instrucción y decodificación: IF+D
  - Ejecución: Ex
  - Escritura de resultados: W

Además, el procesador dispone de:

- 2 unidades de decodificación D1 y D2.
- 3 unidades de ejecución E1, E2 y E3.
- 2 copias de la etapa de escritura W1 y W2.

Políticas de emisión

Políticas de ejecución de instrucciones

- Como tiene 2 unidades de búsqueda y decodificación D1 y D2, 2 instrucciones pueden ser leídas y decodificadas al mismo tiempo.
- Solo se puede leer un nuevo par de instrucciones cuando se liberan ambas unidades D1 y D2.
- Para garantizar la correcta ejecución, los conflictos se resuelven parando la instrucción (“stall”) hasta que el conflicto desaparece.

Políticas de emisión

Políticas de ejecución de instrucciones

Funcionalmente, el procesador superescalar que vamos a usar se puede visualizar de la siguiente manera:

En las filas se van a indicar las instrucciones en proceso, para cada ciclo de reloj.

Políticas de emisión

Políticas de ejecución de instrucciones

Se va a analizar el comportamiento del procesador superescalar para una secuencia de 6 instrucciones con las siguientes características:

- I1 tarda dos ciclos en ejecutarse
- I2 tarda 1 ciclo
- I3 e I4 tardan 1 ciclo y usan la misma unidad funcional (E3)
- I5 e I6 tardan 1 ciclo y usan la misma unidad funcional (E2)
- I5 depende del valor producido por I4

Políticas de emisión

Emisión y finalización ordenada

En esta política, las instrucciones se emiten exactamente como están en el programa (como si fuera "secuencial"), y se escriben los resultados en el mismo orden. El resultado de ejecutar las 6 instrucciones es el siguiente.

Decodificación		Ejecución			Escritura		Ciclo
D1	D2	E1	E2	E3	W1	W2	
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

Políticas de emisión

Emisión y finalización ordenada

- En el ejemplo anterior se puede ver que:
  - Las instrucciones se captan y decodifican de a 2.
  - Las instrucciones I5 e I6 deben esperar hasta que se desocupen ambos decodificadores D1 y D2.
  - Dado que el orden en que se escriben los resultados debe ser el mismo en que se captan, I2 no puede ser escrita hasta que I1 no se haya completado y esté lista para guardarse (W1 o W2).
  - I4 tiene que esperar que I3 libere E3. Idem I5 e I6.
  - I5 tiene que esperar que se ejecute I4.
  - Se requieren 8 ciclos para ejecutar las 6 instrucciones.

Políticas de emisión

Emisión ordenada y finalización desordenada

En esta política, las instrucciones se emiten exactamente como está en el programa, pero los resultados se escriben en "cualquier orden". Las mismas 6 instrucciones se ejecutan de la siguiente manera.

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

Políticas de emisión

Emisión ordenada y finalización desordenada

- En el ejemplo anterior se puede ver que:
  - Las instrucciones se captan y decodifican de a 2. Por eso, las instrucciones I5 e I6 deben esperar hasta que se desocupen ambos decodificadores D1 y D2.
  - Dado que el orden en que se escriben los resultados puede ser distinto del que se captan, I2 puede ser escrita antes que I1 se haya completado y esté lista para guardarse.
  - Al permitir la finalización desordenada, I3 puede ser ejecutada anticipadamente, ganándose 1 ciclo.
  - Ahora se requieren 7 ciclos para ejecutar las 6 instrucciones.

Políticas de emisión

Emisión ordenada y finalización desordenada - Conclusiones

- Las instrucciones entran a las unidades de ejecución a medida que se van decodificando y hay unidades de ejecución disponibles.
- Las emisiones de las instrucciones están limitadas por los posibles conflictos de recursos, dependencias de datos y de saltos.
- Al permitir la finalización desordenada, aparecen nuevas fuentes de conflictos por dependencia de datos.
- En el siguiente ejemplo se pueden apreciar los nuevos conflictos que aparecen por la política de finalización desordenada.

Políticas de emisión

Emisión ordenada y finalización desordenada - Conflictos

Consideremos el siguiente programa:

R3:= R2 op R5 (I1)
R4:= R3 + 1 (I2)
R3:= R5 + 1 (I3)
R7:= R3 op R4 (I4)

Los riesgos que aparecen en el ejemplo anterior son:

1) RAW (dependencia verdadera)

I1 escribe el resultado en R3, que es un operando de la I2.

2) WAW (dependencia de salida)

I1 escribe el resultado en R3 igual que la I3. Si I3 se escribe antes que I1, entonces la I4 usará un valor incorrecto de R3.

3) WAR (antidependencia)

I3 escribe en R3. Pero I2 necesita el valor previo de R3. I3 no puede escribir R3 hasta que I2 haya leído el valor.

Políticas de emisión

Emisión ordenada y finalización desordenada - Conflictos

- La finalización desordenada requiere detectar estas posibles nuevas dependencias, y evitar que se ejecuten instrucciones en forma desordenada que puedan alterar el resultado del programa.
- Por tal motivo, la lógica de emisión de instrucciones es más compleja que la empleada en máquinas con política de finalización ordenada.
- Con la política de emisión ordenada, el procesador solo puede decodificar instrucciones hasta el punto de dependencia o conflicto. Es decir, no analiza si hay nuevas instrucciones que pueden emitirse sin producir conflicto.

Políticas de emisión

Emisión y finalización desordenada

- Para permitir que el procesador busque más allá del punto de conflicto por nuevas instrucciones, se require usar una política de emisión desordenada.
- La política de emisión desordenada require separar, mediante un buffer, llamado Ventana de instrucciones, las unidades de decodificación y las de ejecución.
- Cada instrucción decodificada se coloca en un buffer intermedio, desde donde se emiten las que no presentan conflictos. Mientras no se llene el buffer, nuevas instrucciones son decodificadas y colocadas para su emisión a las unidades de ejecución. Las instrucciones emitidas no deben tener conflictos por recursos ni por dependencias de datos.

Políticas de emisión

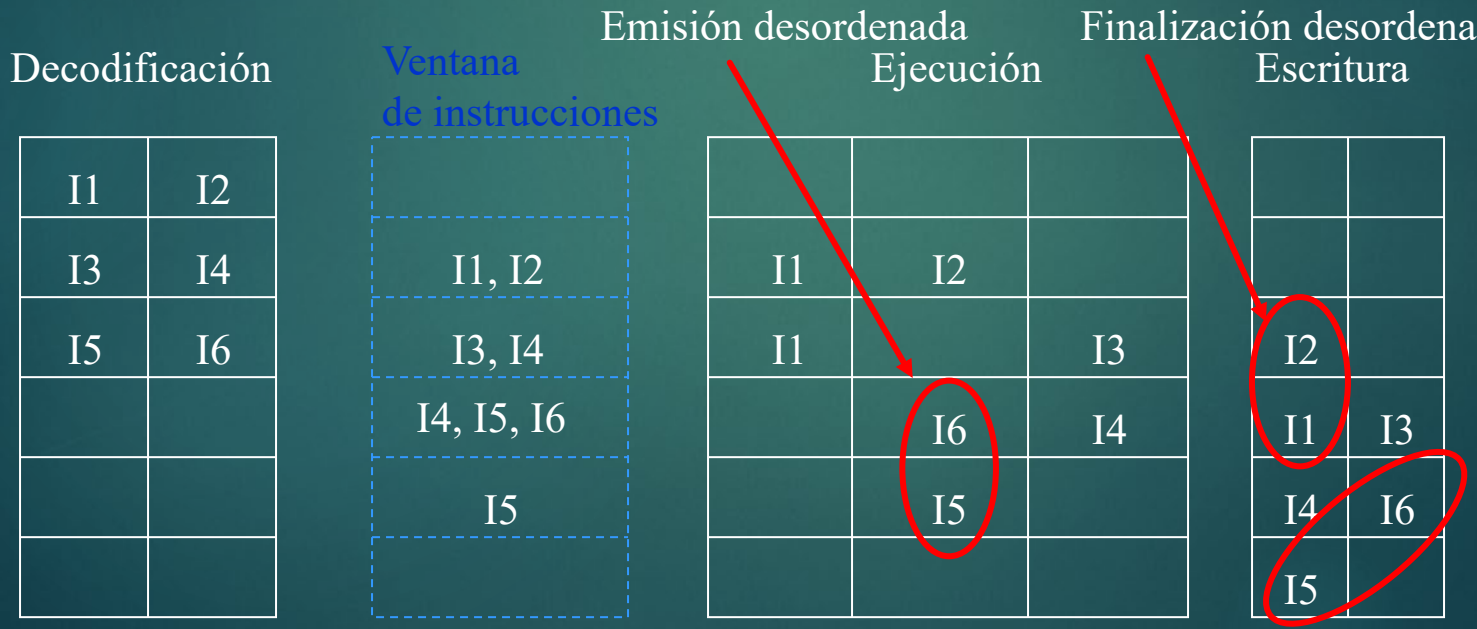
Emisión y finalización desordenada

- La Ventana de instrucciones no es una etapa del cauce, es un buffer que retiene la información necesaria para emitirla.
- En el siguiente ejemplo se muestra el esquema funcional de una máquina con emisión y finalizacion desordenada.

Políticas de emisión

Emisión y finalización desordenada

- Se intercala una buffer (ventana de instrucciones) entra las unidades de decodificación y las de ejecución.
- Las instrucciones se emiten y los resultados se guardan en forma desordenada.



Políticas de emisión

Emisión y finalización desordenada

- En el ejemplo anterior se puede ver que:
  - Las instrucciones se captan y decodifican de a 2, pero ahora la ventana de instrucciones permite, en el ciclo 3, decodificar las instrucciones I5 e I6.
  - Como I6 no tiene dependencias y está lista para ejecutarse, puede ser emitida anticipadamente, incluso antes de I5. La ventana le permite “ver” al procesador más adelante de las instrucciones en conflicto, pudiendo detectar nuevas instrucciones (la I6) que no genera conflicto.
  - Ahora se requieren 6 ciclos para ejecutar las 6 instrucciones.

Renombrado de registros

- Al permitir finalización y/o emisión desordenada surgen los nuevos problemas de dependencia de datos (WAW y WAR).
- Las dependencias de salida y antidependencias surgen porque los valores de los registros pueden no reflejar la secuencia de valores dictada por el flujo del programa (en realidad es un conflicto de almacenamiento: hay instrucciones que compiten por un mismo registro).
- Tal como se vio en clases anteriores, estos nuevos conflictos pueden resolverse con detenciones en la etapa del cauce, pero esa solución produce pérdidas de tiempo que pueden llegar a ser inaceptables.

Renombrado de registros

Supongamos la siguiente secuencia de instrucciones de un programa:

R3:= R3 op R5 (I1)
R4:= R3 + 1 (I2)
R3:= R5 + 1 (I3)
R7:= R3 op R4 (I4)

En la secuencia anterior se pueden observar las 3 dependencias:

RAW entre la I1 y la I2
WAR entre la I2 y la I3
WAW entre la I1 y la I3

Renombrado de registros

- Dado que en definitiva, es un conflicto de recursos (los registros), se puede resolver duplicando los recursos.
- Esta técnica se conoce como Renombrado de Registros.
- Los registros en las instrucciones del programa, son registros “lógicos”. Los registros físicos (es decir, los reales) los asigna el procesador.
- Cada vez que se guarda un dato en un registro el procesador le asigna un nuevo registro físico. Las instrucciones siguientes que referencien a ese registro, deben renombrarse para referenciar el registro físico que tiene el dato correcto.

Renombrado de registros

Consideremos el ejemplo anterior (donde los nombres de registros son referencias “lógicas”):

R3:= R3 op R5 (I1)
R4:= R3 + 1 (I2)
R3:= R5 + 1 (I3)
R7:= R3 op R4 (I4)

El procesador va asignando nuevos registros en cada escritura, quedando las siguientes referencias “físicas”:

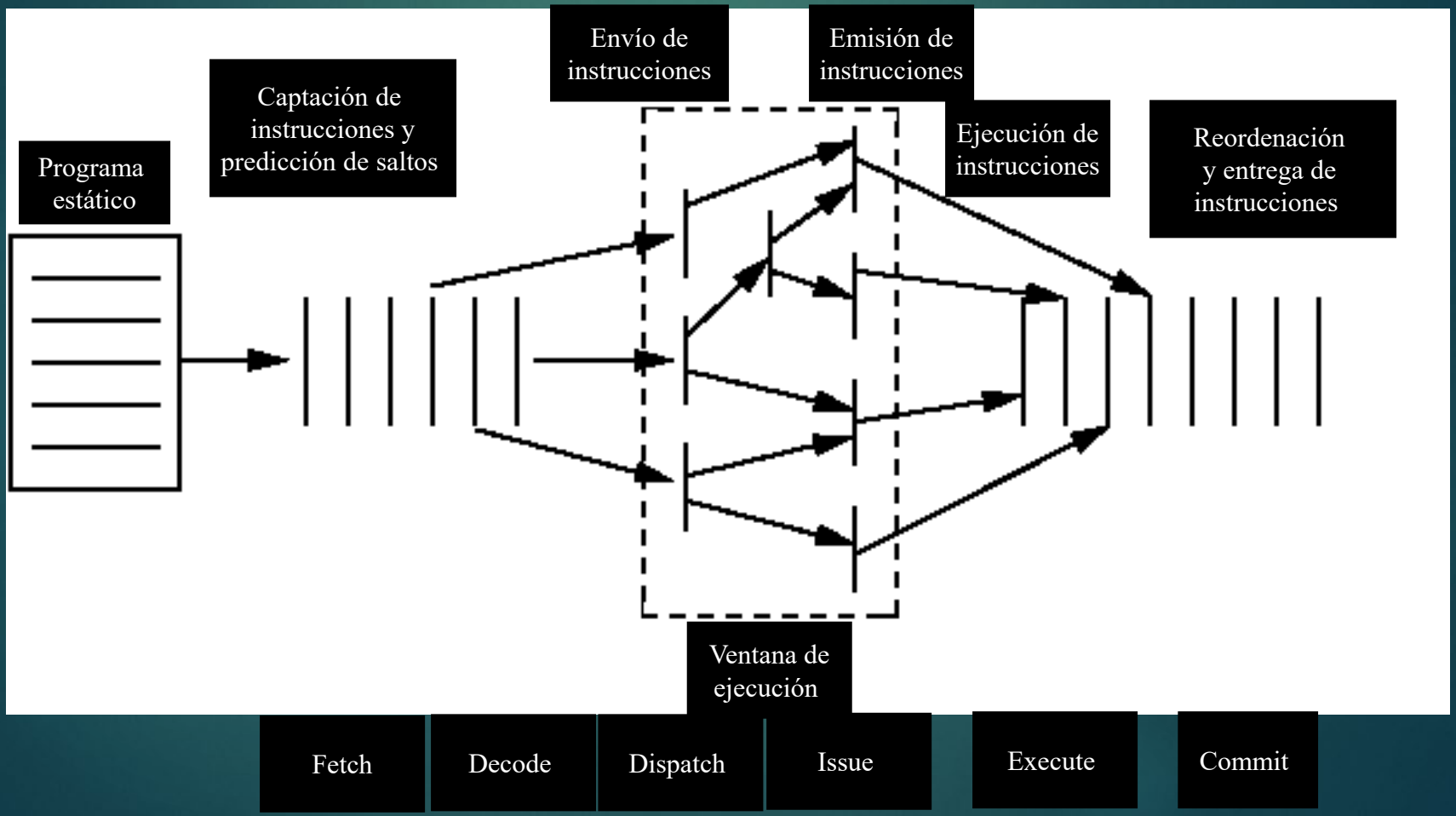
R3b:= R3a op R5a (I1)
R4a:= R3b + 1 (I2)
R3c:= R5a + 1 (I3)
R7a:= R3c op R4a (I4)

Renombrado de registros

- En el ejemplo anterior, en la instrucción I1 se asigna un nuevo registro R3b distinto del usado anteriormente R3a.
- La I2 debe renombrarse para usar R3b (y no R3a), y crea R4a (que no existía previamente).
- La I3 asigna un nuevo registro R3c, distinto del usado anteriormente (R3b).
- La I4 debe renombrarse para usar R3c (y no R3b), y crea R7a (que no existía previamente).
- De esta manera se eliminan las dependencias WAW y WAR (antidependencia y dependencia de salida) quedando únicamente la dependencia verdadera RAW.

Ejecución superescalar

En resumen, el modelo de ejecución superescalar se puede esquematizar de la siguiente manera.



Ejecución superescalar

La imagen anterior representa una ejecución superescalar típica, donde:

- Las instrucciones son captadas a partir del programa estático, y decodificadas para la detección temprana de dependencias y predicción de saltos.
- Las instrucciones, todavía ordenadas, son enviadas a la ventana de ejecución, donde se las redistribuye en función de las dependencias verdaderas, esto es, las que están en condiciones de ser ejecutas o las que necesitan disponer de datos de otras instrucciones.
- Las instrucciones son ejecutadas en un orden basado en la disponibilidad de unidades de ejecución y en las dependencias verdaderas.

Ejecución superescalar

- Las instrucciones ejecutadas son nuevamente reordenadas secuencialmente, y los resultados guardados en el orden que corresponde.
- Este último paso es para:
  - Reordenar el almacenamiento de acuerdo al programa original
  - Eliminar todas las ejecuciones inservibles (aquellas que se hicieron por predicción de salto o ejecución especulativa) y que se deben descartar.
  - Mejorar la respuesta a las interrupciones.
- De esta manera, las instrucciones ejecutadas producen resultados almacenados temporalmente, hasta que se graban los resultados válidos en los registros reales.

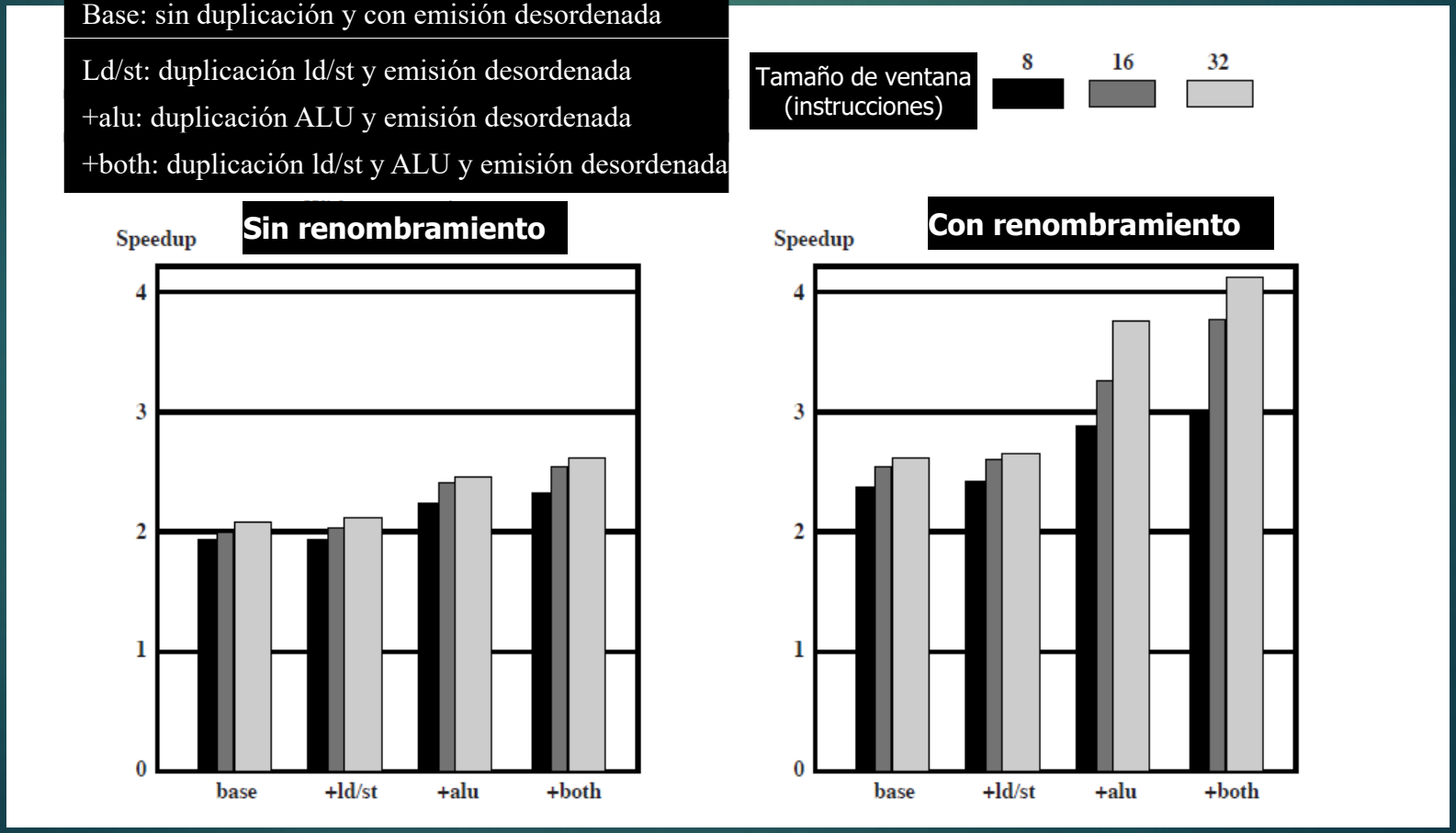
Ejecución superescalar

En resumen, la ejecución superscalar involucra:

- Diferentes estrategias de captación simultánea de múltiples instrucciones.
- Lógica para determinar dependencias verdaderas entre valores de registros y mecanismos para comunicar esos valores.
- 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.

Ejecución superescalar

Se han realizado diferentes estudios para determinar la influencia de algunas técnicas empleadas en procesadores superescalares. Un estudio es el que se muestra a continuación.



Ejecución superescalar

Estudios sobre ejecuciones superescalares

En el estudio anterior, se han comparado las mejoras obtenidas (o "speedup" en el eje vertical) para los siguientes casos:

- 2 situaciones: sin (gráfico de la izquierda), y con (gráfico de la derecha), renombrado de registros .
- 3 tamaños de ventanas de instrucciones: 8, 16 y 32 instrucciones.
- 4 tipos de máquinas:
  - Máquina base: sin duplicación de unidades funcionales
  - Máquina base + ld/st: duplicando las unidades de acceso a la memoria
  - Máquina base + ALU: duplicando las ALU
  - Máquina base + ld/st + ALU: duplicando las unidades de acceso a la memoria y las ALU

Ejecución superescalar

Conclusiones

- En la máquina sin renombrado de registros se observa que:
  - Duplicar las unidades funcionales de acceso a la memoria y las ALU (barras “both”) tiene poco impacto en la aceleración obtenida (el eje vertical pasa de 2 a 2.5).
  - Disponer de una Ventana de instrucciones mas grande (8 a 32) no modifica significativamente los resultados.
- En la máquina con renombrado de registros se observa que:
  - Duplicar las ALU y también las unidades de acceso a la memoria mejoran fuertemente la aceleración obtenida (el je vertical pasa de 2.5 a 4).
  - El tamaño de la Ventana de instrucciones tiene un impacto significativo cuando cambia de 8 a 16 instrucciones.

Ejecución superescalar

Conclusiones

- La gran diferencia entre los resultados con y sin renombrado de registros se debe a un elemento fundamental en el proceso de ejecución de instrucciones:
  - En las máquinas sin renombrado de registros, todos los conflictos por las dependencias de datos (RAW, WAR, WAW) se resuelven mediante detenciones del cauce (“stall”).
  - En las máquinas con renombrado de registro se eliminan los conflictos por dependencias de salida y antidependencia (WAW, WAR) quedando únicamente la dependencia directa o verdadera (RAW).

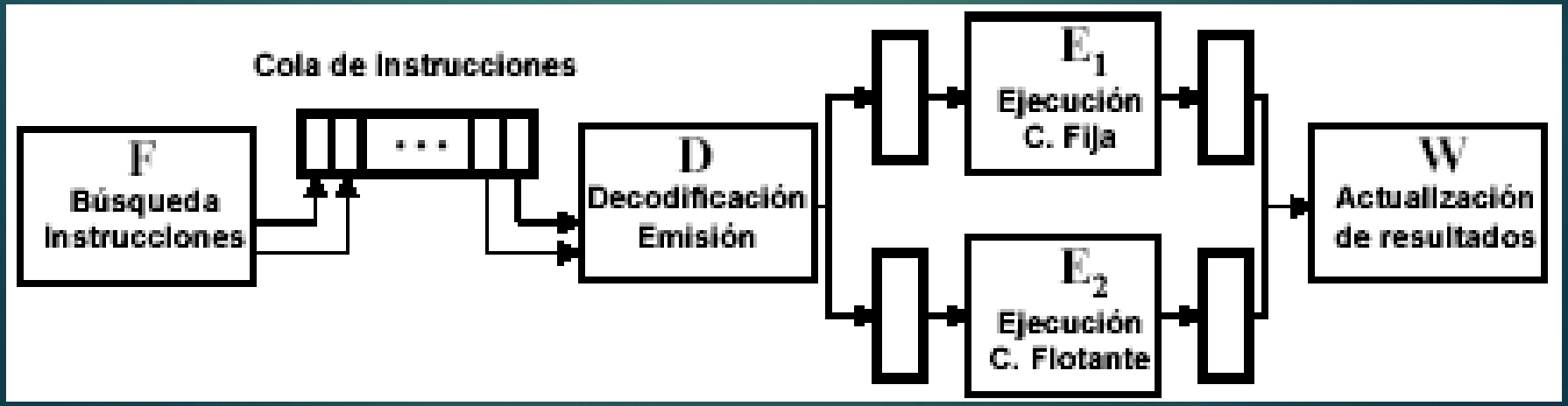
Ejemplo procesador superescalar

Un esquema de un procesador superescalar se muestra en la figura siguiente.

La unidad de búsqueda de instrucciones (F) busca y carga en un buffer (cola de instrucciones), las instrucciones leídas.

La unidad de decodificación/emisión puede tomar 2 instrucciones de la cola y decodificarlas.

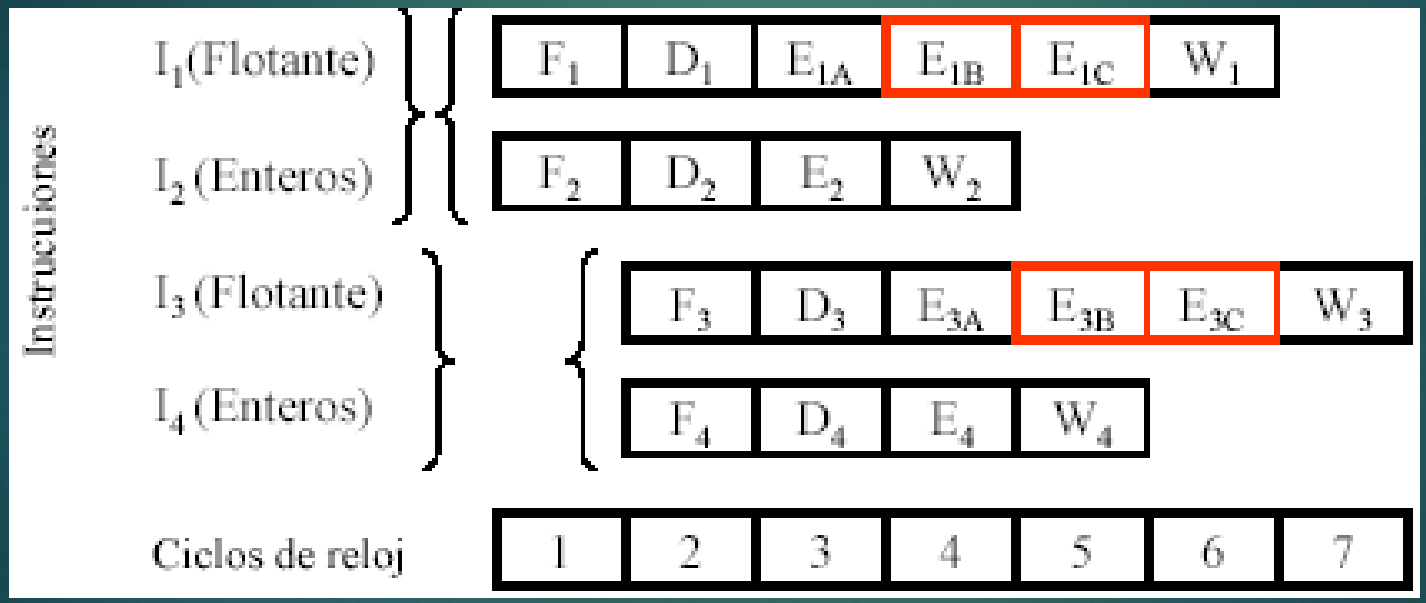
Hay 2 unidades de ejecución (ALU), una de punto fijo y la otra de punto flotante.



Ejemplo procesador superescalar

Si las 2 instrucciones que capta, una es de enteros y la otra de coma flotante (segmentada), y no existen riesgos, entonces pueden emitirse y ejecutarse a la vez.

En el diagrama siguiente, entran 2 instrucciones por ciclo:



Es responsabilidad del compilador generar el código apropiado para el máximo aprovechamiento del procesador supersescalar.

Interrupciones en procesadores superescalares

- En los procesadores superescalares el tratamiento de interrupciones internas o excepciones (rupturas por ejecución de instrucciones) y externas (rupturas por eventos externos) es más complicado que en los procesadores convencionales.
- La razón es que en el momento en que se produce una interrupción hay varias instrucciones en ejecución (y además, desordenadas).
- El problema que se plantea es si producida una excepción en una instrucción dada I2, en qué estado quedaron las instrucciones anteriores (por ejemplo I1) y posteriores (por ejemplo I3) dado que con la política de ejecución desordenada pudieron haber quedado en distintos estados.

Interrupciones en procesadores superescalares

Interrupciones internas o excepciones

Existen 2 estrategias para el tratamiento de excepciones:

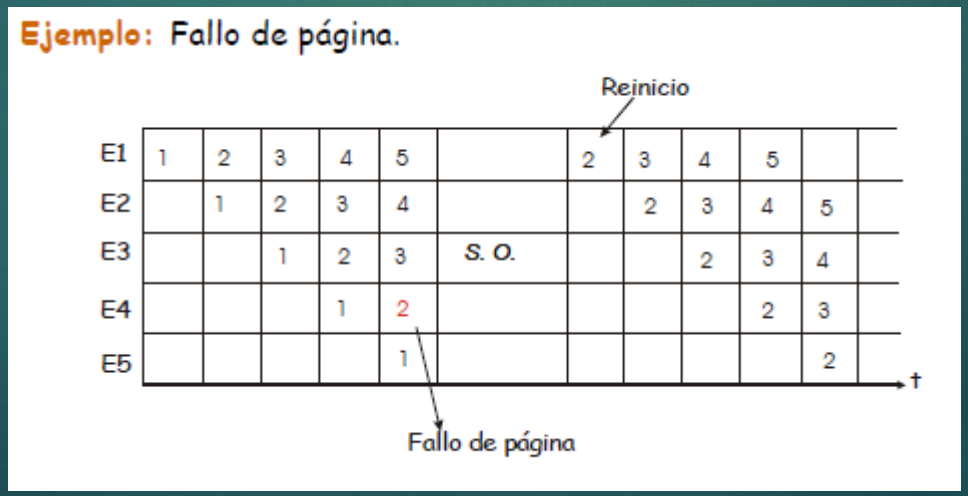
- Excepciones precisas: las instrucciones que preceden a la que produjo la excepción se completan, y las que le suceden se reinician. Es decir, el comportamiento es "idéntico" al que tendría la misma computadora no segmentada
- Excepciones imprecisas: no se respetan las condiciones exactas de la máquina no segmentada.

Interrupciones en procesadores superescalares

Interrupciones internas o excepciones

- Para garantizar un estado consistente (preciso) se requiere:
  - Instrucciones anteriores terminan correctamente
  - La que origina la excepción y siguientes se abortan
  - Tras la rutina de tratamiento se comienza por la que originó la excepción

Es decir:



Interrupciones en procesadores superescalares

Interrupciones internas o excepciones

- Para que las interrupciones sean precisas los resultados se deben almacenar en el orden en que aparecen las instrucciones.
- Para ello, se requiere retardar las escrituras de los resultados de tal manera que queden en el orden en el que estaba el programa secuencial.

Interrupciones en procesadores superescalares

Interrupciones externas

- Las interrupciones externas pueden tener distinta grado de "precisión", dado que no necesariamente tienen que ver con las instrucciones en ejecución (por ejemplo operaciones de I/O).
  - Interrupciones imprecisas: se completa lo que estaba en ejecución.
  - Interrupciones "algo" imprecisas: el software (la rutina de servicio de la interrupción) resuelve algunas inconsistencias.
  - Interrupciones precisas: solo se completan las instrucciones previas a la interrupción, para lo cual se requiere implementar la escritura con buffer de salida. La unidad de emisión deja de emitir, se cancela la cola, todas las instrucciones pendientes se completan, y comienza el procesamiento de la interrupción.

Interrupciones en procesadores superescalares

Conclusiones

En general se tiene:

- En los sistemas con finalización desordenada:
  - Excepciones imprecisas
  - Mayor performance
- En los sistemas con finalización ordenada (es decir, con buffer de reordenamiento):
  - Excepciones precisas
  - Menor performance

Referencias

- Organización y Arquitectura de Computadoras, William Stallings, Capítulo 13 de 5ta edición ó Capítulo 14 de 7ma edición.