Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Taxonomía de Flynn

SISD	Single Instruction Single Data
MIMD	Multiple Instruction Multiple Data
SIMD	Single Instruction Multiple Data
MISD	Multiple Instruction Single Data

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

PROCESO

PERSPECTIVA GENERAL

Es una tarea que se divide en subtareas para ser ejecutadas cada una “por separado” a fin de obtener el resultado de la tarea principal

PERSPECTIVA DESDE LA OPTICA DE LOS SISTEMAS OPERATIVOS

Es una instancia de un programa en ejecución, incluyendo los valores actuales del contador de programa, los registros y las variables.

Cada proceso tiene su propia CPU virtual. Significa que la CPU real conmuta de un proceso a otro. Se dice que la CPU ejecuta una colección o conjunto de procesos en pseudo paralelo.

Tiene un programa, una entrada, una salida y un estado

Tanenbaum, pagina 84, 85, 86

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

SEGMENTACIÓN DE INSTRUCCIONES

Como su nombre lo sugiere, significa dividir, la tarea de ejecutar una instrucción, en etapas, en el tiempo

Identificar las subtareas en las que es posible dividir la tarea principal de “Ejecutar una instrucción”

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

PIPELINE: es el proceso

Resolver una instrucción, o ejecutar una instrucción, ejecutando secuencialmente cada una de sus etapas en que ha sido dividido, a fin de obtener el resultado de la ejecución de la instrucción

Es equivalente a una cadena de producción en la industria

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Superescalar

Segmentación de Instrucciones

De acuerdo con la definición de Stallings, la segmentación (*pipeling o también encauzamiento*) de instrucciones es similar al uso de una cadena de montaje en una fábrica. Una cadena de montaje saca partido del hecho de que el producto pasa a través de varias etapas de producción.

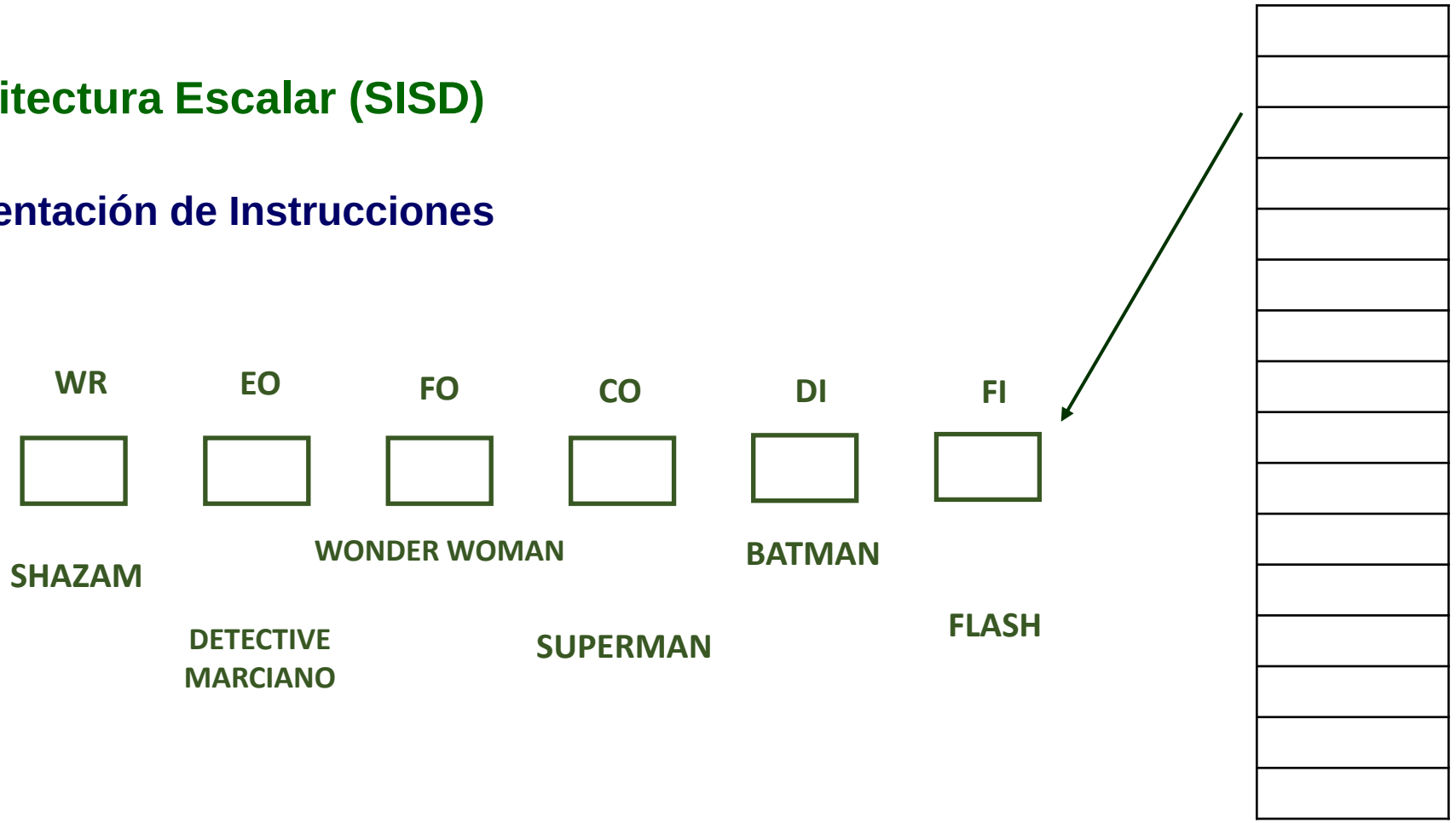
Lo anterior permite optimizar el tiempo de ejecución de las instrucciones, al poder traslapar (o solapar) las etapas por las que pasa el proceso de ejecución.

A continuación se describen las etapas por las que pasa una Instrucción en su proceso de ejecución. pipeline

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Escalar (SISD)

Segmentación de Instrucciones



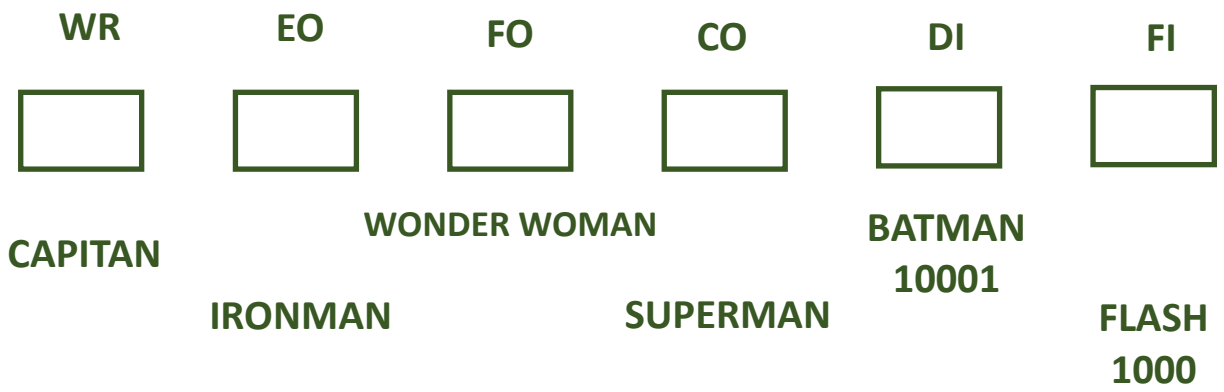
Continuar desde esta lamina en la siguiente sesión 29 de abril de 2021
Con los grupos 3CM11, 3CM13, 3CM14

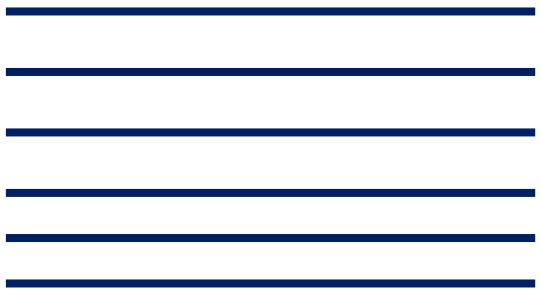
Stallings, página 501-502

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Escalar (SISD)

Segmentación de Instrucciones





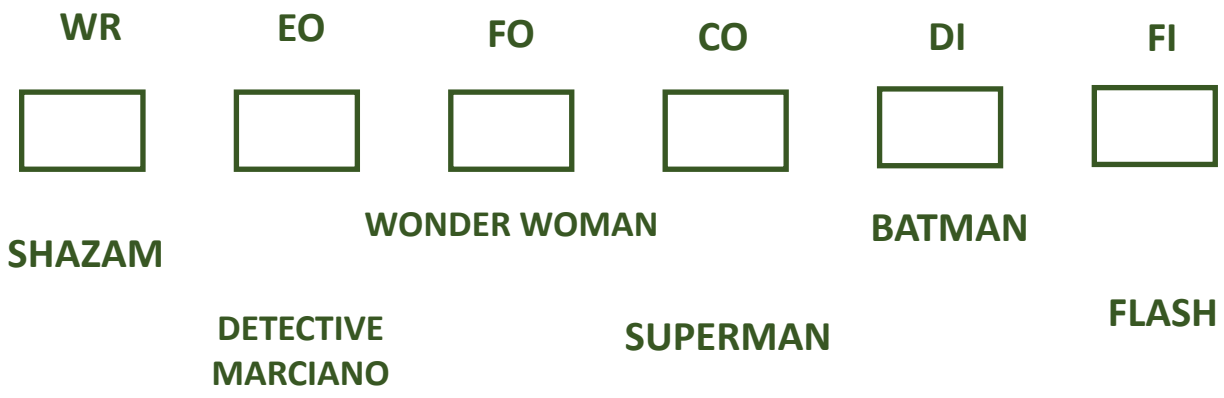
inFlag
outFlag

Stallings, página 501-502

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

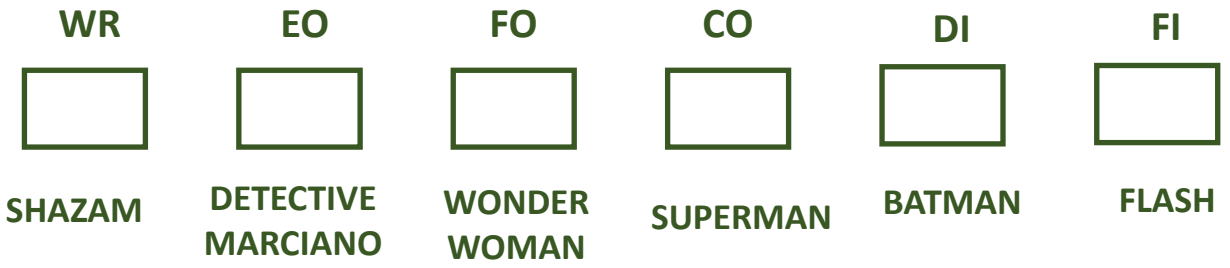
Arquitectura Escalar (SISD): Arquitectura escalar

Segmentación de Instrucciones



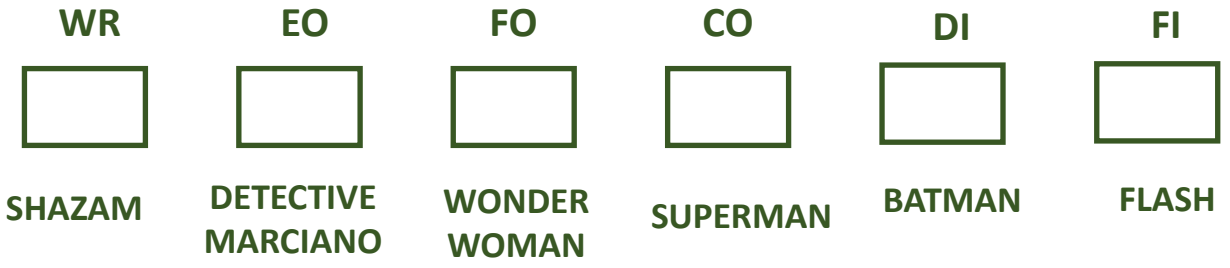
Stallings, página 501-502

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos



Arquitectura Superescalar

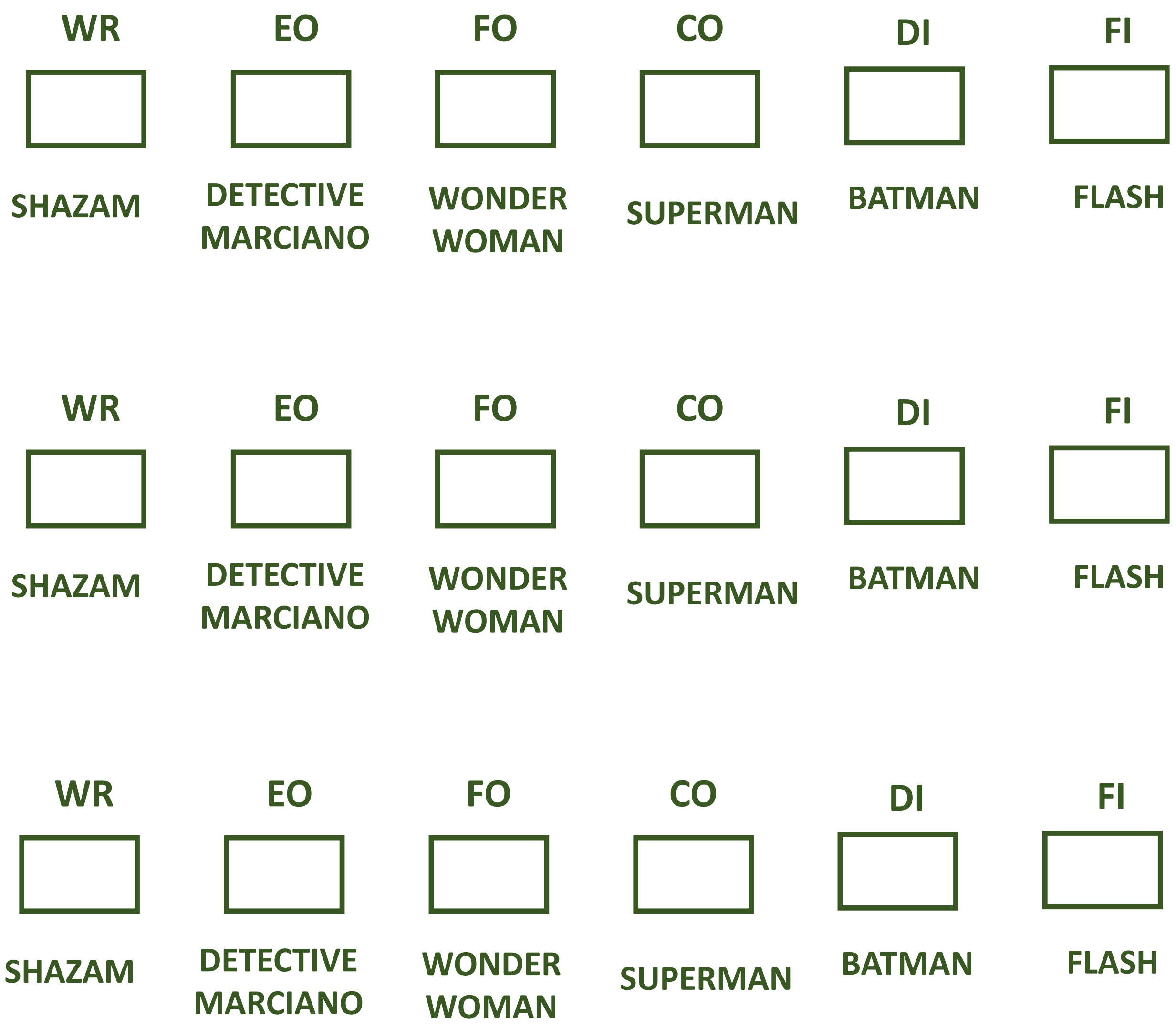
MIMD



Dos o más líneas de Pipeline ejecutándose en paralelo es superescalar

Stallings, página 501-502

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos



Arquitectura Superescalar

MIMD

Dos o más líneas de Pipeline ejecutándose en paralelo es superescalar

Stallings, página 501-502

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Superescalar

Segmentación de Instrucciones

Captar instrucción (Fetch Instruction, **FI**).- Leer la supuesta siguiente instrucción instrucción en el buffer de instrucción.

Decodificar instrucción (Decode instrucction, **DI**).- Determinar el código de operación y los campos de operandos.

Calcular operandos (Calculate Operands, **CO**).- Calcula la dirección efectiva de cada operando fuente. Pa ello se usa alguno de los modos de direccionamientos vistos previamente.

Stallings, página 501-502

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Superescalar

Segmentación de Instrucciones

Captar operandos (Fetch Operands, **FO**).- Cargar en Buffer de ALU, los operandos.

Ejecutar Instrucción (Execute Instrucction, **EI**).- Realizar la operación indicada y almacenar el resultado, si lo hay en la posición del operando destino.

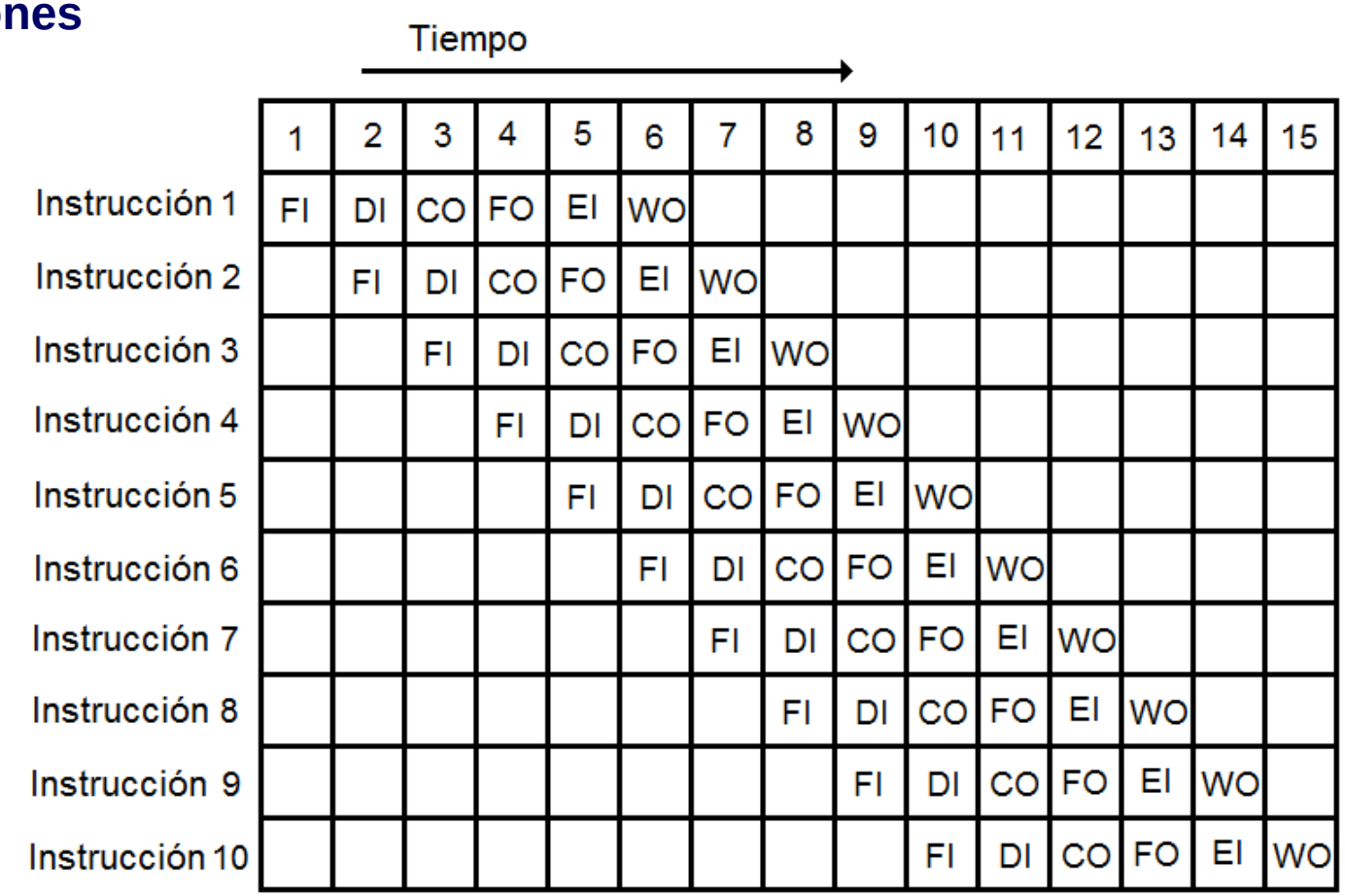
Escribir operando (Write Operand, **WO**).- Almacenar el resultado en memoria

Stallings, página 501-502

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Superescalar

Segmentación de Instrucciones



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Instrucción 1	FI	DI	CO	FO	EI	WO									
Instrucción 2		FI	DI	CO	FO	EI	WO								
Instrucción 3			FI	DI	CO	FO	EI	WO							
Instrucción 4				FI	DI	CO	FO	EI	WO						
Instrucción 5					FI	DI	CO	FO	EI	WO					
Instrucción 6						FI	DI	CO	FO	EI	WO				
Instrucción 7							FI	DI	CO	FO	EI	WO			
Instrucción 8								FI	DI	CO	FO	EI	WO		
Instrucción 9									FI	DI	CO	FO	EI	WO	
Instrucción 10										FI	DI	CO	FO	EI	WO

Stallings, página 501-502

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Superescalar

Segmentación de Instrucciones

Una instrucción es dividida en etapas y cada etapa es ejecutada como en una cadena de producción

Tiempo →

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Instrucción 1	FI	DI	CO	FO	EI	WO									
Instrucción 2		FI	DI	CO	FO	EI	WO								
Instrucción 3			FI	DI	CO	FO	EI	WO							
Instrucción 4				FI	DI	CO	FO	EI	WO						
Instrucción 5					FI	DI	CO	FO	EI	WO					
Instrucción 6						FI	DI	CO	FO	EI	WO				
Instrucción 7							FI	DI	CO	FO	EI	WO			
Instrucción 8								FI	DI	CO	FO	EI	WO		
Instrucción 9									FI	DI	CO	FO	EI	WO	
Instrucción 10										FI	DI	CO	FO	EI	WO

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Escalar

Se refiere a la arquitectura del procesador

A scalar processor is a processor that is based on a single-issue architecture, which means that only a single instruction is run at a time

Lascu., Octavian, and et. al. (2016). International Technical Support Organization. IBM z13 Technical Guide. (May 2016). Redbooks, ibm.com/redbooks. z System. In partnership with IBM Academy of Technology

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Superescalar

Se refiere a la arquitectura del procesador

A superscalar processor allows concurrent (parallel) execution of instructions by adding more resources to the microprocessor in multiple pipelines, each working on its own set of instructions to create parallelism

Lascu., Octavian, and et. al. (2016). International Technical Support Organization. IBM z13 Technical Guide. (May 2016). Redbooks, ibm.com/redbooks. z System. In partnership with IBM Academy of Technology.

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Superescalar

Supersegmentación

La segmentación aprovecha el paralelismo potencial entre las instrucciones denominado *paralelismo a nivel de instrucciones (instruction-level paralelism* ILP) [Patterson-Hennessy, pag. 391].

Existen dos estrategias básicas para incrementar el potencial del ILP. En otras palabras, optimizar el tiempo de ejecución y en consecuencia incrementar la velocidad del procesador.

La primera es incrementar la profundidad del segmentado (*pipeline*) para solapar la ejecución de una o más etapas de algunas instrucciones.

El mejor ejemplo de estas instrucciones es la multiplicación, que para ser ejecutada se requiere más de un ciclo de reloj.

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Superescalar

Supersegmentación

Como se describió previamente, la segmentación, *pipeline o también encauzamiento*, consiste en ejecutar de manera "seudo-independiente" cada etapa de la instrucción. Implica ejecutar dos o más instrucciones en forma "seudo-paralela", teniendo en cuenta que en cada periodo de tiempo, este puede ser el ciclo de reloj, se ejecutan etapas distintas de las instrucciones en proceso.

Si bien, en la segmentación, hay más de una instrucción en proceso, etapas distintas se están ejecutando en el mismo ciclo de reloj. O en otras palabras, en el mismo ciclo de reloj, no hay dos etapas similares ejecutándose.

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Superescalar

Supersegmentación

Lo anterior, además, implica que cada etapa es ejecutada por un módulo de hardware independiente que incorpora entradas de habilitación, que le permite esperar para recibir la etapa correspondiente de la siguiente instrucción, hasta que el módulo de la siguiente etapa haya terminado su tarea.

En las etapas del proceso de la instrucción, puede existir alguna etapa que consuma mayor tiempo de ejecución.

Por ejemplo la etapa de *Ejecución de la instrucción* (Execute Instrucction, **EI**), en la que (A) Se realiza la operación indicada y (B) se almacena el resultado, en caso de que lo haya, en la posición del operando destino.

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Superescalar

Supersegmentación

Además, algunas operaciones consumen más de un ciclo re reloj (más tiempo), por ejemplo, la multiplicación.

En este caso, la etapa EI puede ser dividida en sub-etapas e implementar segmentado a nivel de esta etapa.

Segmentación a nivel de instrucción

Instruction-level Pipeline

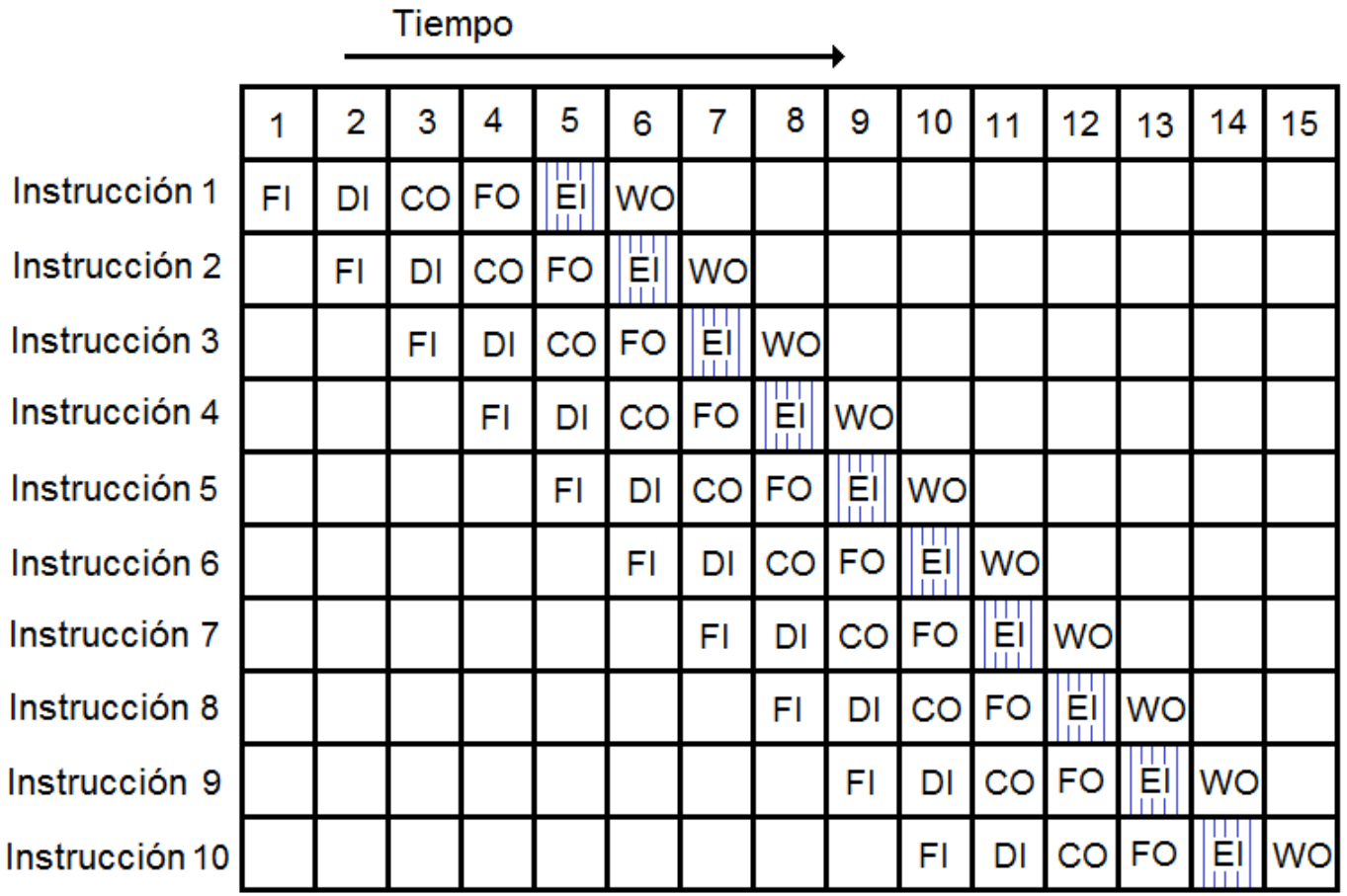
Paralelismo a nivel de instrucción

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Superescalar

Supersegmentación

Super-Segmentacion significa que, en pipeline una etapa es dividida en sub-etapas, y estas sub-etapas son ejecutadas, a su vez, como en una cadena de producción



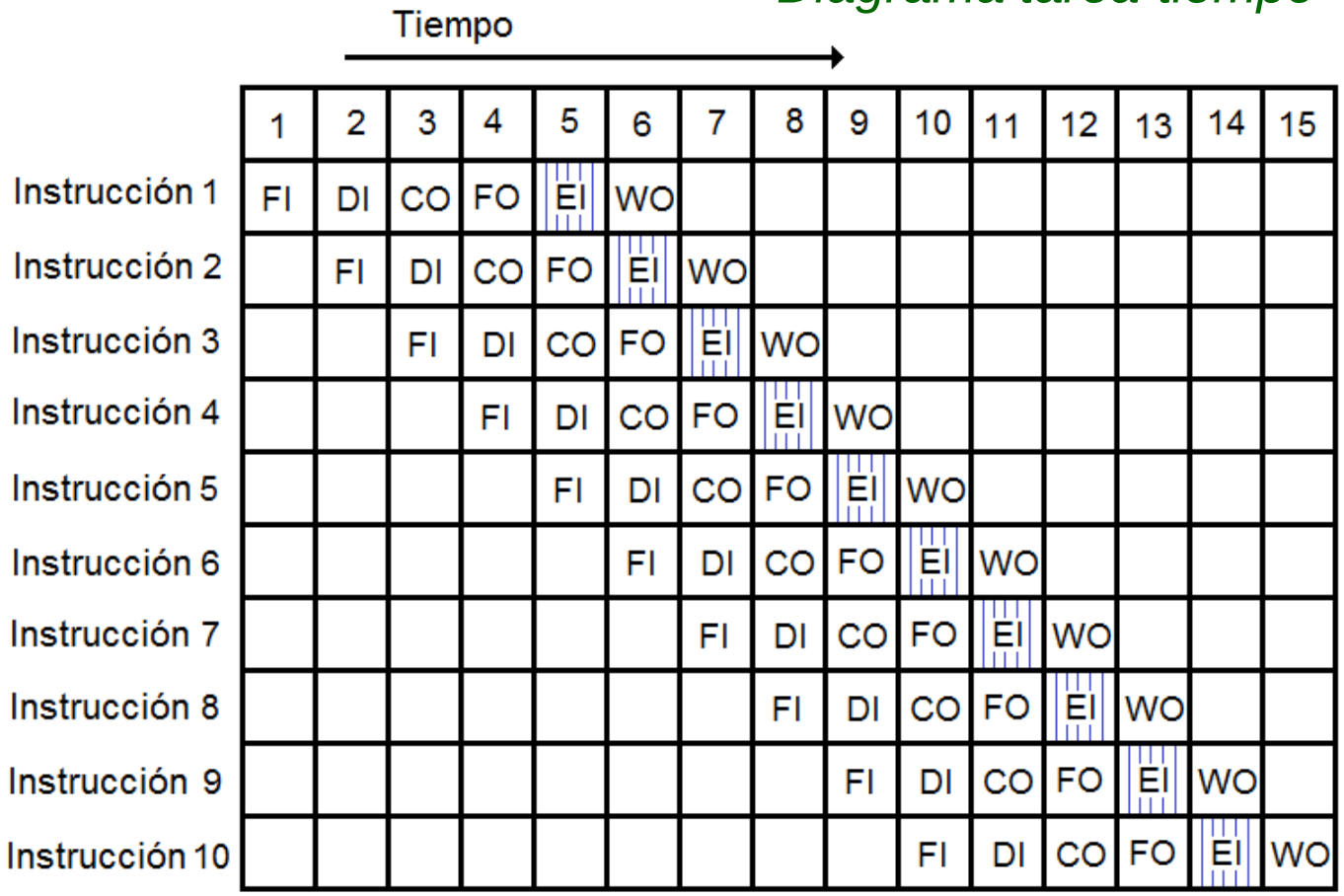
Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Superescalar

A esto es a lo que se le llama **supersegmentación**

Supersegmentación

Diagrama tarea-tiempo



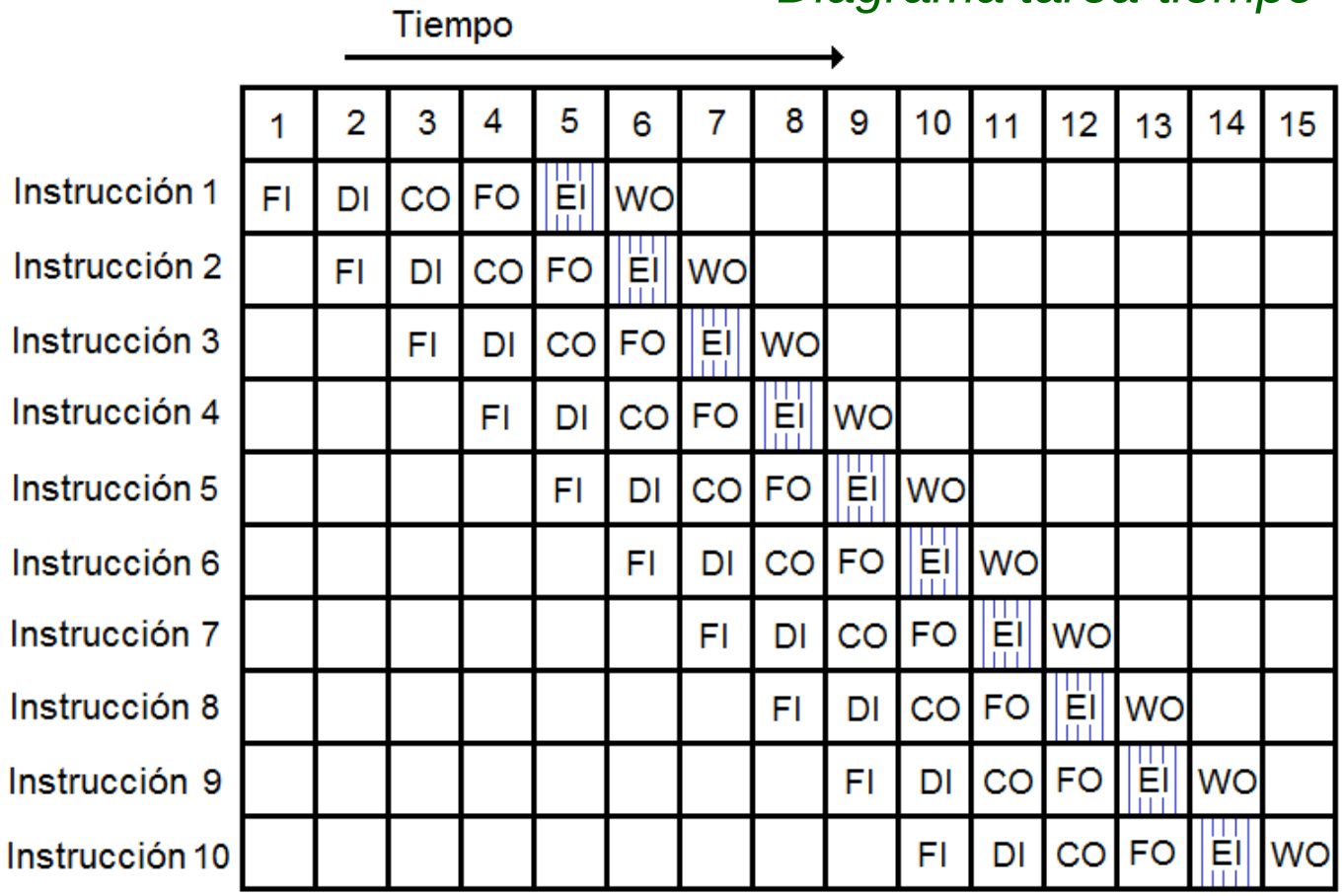
Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Superescalar

Supersegmentación

A esto es a lo que se le llama **superpipeline**

Diagrama tarea-tiempo



Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Superescalar

La segunda estrategia para incrementar el potencial del concepto de *Instruction-Level Paralelism*, consiste en replicar, en el procesador, todos los módulos hardware de un bloque de segmentación para poder procesar múltiples instrucciones segmentadas [Patterson-Hennessy, pag. 391], [cmjsoftwaresynthesis-web.pdf, pag. 3].

Por superescalar nos referimos a una máquina que, como una máquina escalar simple, se ejecuta un flujo de instrucciones no vectorial, pero logra un alto rendimiento mediante la división del trabajo entre las unidades funcionales independientes que operan en paralelo

Richard R. Oehler, Michael W. Blasgen. IBM Research Division
IBM RISC System/6000: Architecture and Performance. June 1991.
IEEE_Micro_1991_-_IBM_6000.pdf

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Superescalar

La segunda estrategia para incrementar el potencial del concepto de *Instruction-Level Paralelism*, consiste en replicar, en el procesador, todos los módulos hardware de un bloque de segmentación para poder procesar múltiples instrucciones segmentadas [Patterson-Hennessy, pag. 391], [cmjsoftwaresynthesis-web.pdf, pag. 3].

Tiempo →

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Instrucción 1	FI	DI	CO	FO	EI	WO									
Instrucción 2		FI	DI	CO	FO	EI	WO								
Instrucción 3			FI	DI	CO	FO	EI	WO							
Instrucción 4				FI	DI	CO	FO	EI	WO						
Instrucción 5					FI	DI	CO	FO	EI	WO					
Instrucción 6						FI	DI	CO	FO	EI	WO				
Instrucción 7							FI	DI	CO	FO	EI	WO			
Instrucción 8								FI	DI	CO	FO	EI	WO		
Instrucción 9									FI	DI	CO	FO	EI	WO	
Instrucción 10										FI	DI	CO	FO	EI	WO

. . . .

Tiempo →

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Instrucción 1	FI	DI	CO	FO	EI	WO									
Instrucción 2		FI	DI	CO	FO	EI	WO								
Instrucción 3			FI	DI	CO	FO	EI	WO							
Instrucción 4				FI	DI	CO	FO	EI	WO						
Instrucción 5					FI	DI	CO	FO	EI	WO					
Instrucción 6						FI	DI	CO	FO	EI	WO				
Instrucción 7							FI	DI	CO	FO	EI	WO			
Instrucción 8								FI	DI	CO	FO	EI	WO		
Instrucción 9									FI	DI	CO	FO	EI	WO	
Instrucción 10										FI	DI	CO	FO	EI	WO

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Superescalar

La segunda estrategia para incrementar el potencial del concepto de *Instruction-Level Paralelism*, consiste en replicar, en el procesador, todos los módulos hardware de un bloque de segmentación para poder procesar múltiples instrucciones segmentadas [Patterson-Hennessy, pag. 391], [cmjsoftwaresynthesis-web.pdf, pag. 3].

A esto es a lo que se le llama "**Arquitectura Superescalar**"

Tiempo →

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Instrucción 1	FI	DI	CO	FO	EI	WO									
Instrucción 2		FI	DI	CO	FO	EI	WO								
Instrucción 3			FI	DI	CO	FO	EI	WO							
Instrucción 4				FI	DI	CO	FO	EI	WO						
Instrucción 5					FI	DI	CO	FO	EI	WO					
Instrucción 6						FI	DI	CO	FO	EI	WO				
Instrucción 7							FI	DI	CO	FO	EI	WO			
Instrucción 8								FI	DI	CO	FO	EI	WO		
Instrucción 9									FI	DI	CO	FO	EI	WO	
Instrucción 10										FI	DI	CO	FO	EI	WO

. . . .

Tiempo →

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Instrucción 1	FI	DI	CO	FO	EI	WO									
Instrucción 2		FI	DI	CO	FO	EI	WO								
Instrucción 3			FI	DI	CO	FO	EI	WO							
Instrucción 4				FI	DI	CO	FO	EI	WO						
Instrucción 5					FI	DI	CO	FO	EI	WO					
Instrucción 6						FI	DI	CO	FO	EI	WO				
Instrucción 7							FI	DI	CO	FO	EI	WO			
Instrucción 8								FI	DI	CO	FO	EI	WO		
Instrucción 9									FI	DI	CO	FO	EI	WO	
Instrucción 10										FI	DI	CO	FO	EI	WO

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Superescalar

La segunda estrategia para incrementar el potencial del concepto de *Instruction-Level Paralelism*, consiste en replicar, en el procesador, todos los módulos hardware de un bloque de segmentación para poder procesar múltiples instrucciones segmentadas [Patterson-Hennessy, pag. 391], [cmjsoftwaresynthesis-web.pdf, pag. 3].

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Superescalar

A esto es a lo que se le llama "**Arquitectura Superescalar**"

Varias instrucciones son ejecutadas en paralelo. Implica el uso de varias unidades de proceso, aunque no necesariamente varios núcleos. Cada Instrucción lleva sus propios datos. [University of Oslo, Department of informatics, Computer architecture, Compendium for INF2270, Philipp Häfliger and Dag Langmyhr, kompendium-inf2270.pdf pag. 83]

Múltiples operaciones son iniciadas y finalizadas simultáneamente

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Superescalar

A esto es a lo que se le llama "**Arquitectura Superescalar**"

Superscalar dispatch multiple execution units or multithreaded execution unit pipelines. [IBM. PowerPC Microprocessor Family. Vector/SIMD Multimedia Extension Technology Programming Environments Manual. Version 2.06. Page 22 of 317, August 22, 2005]

Múltiples operaciones son iniciadas y finalizadas simultáneamente

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Superescalar

A esto es a lo que se le llama "**Arquitectura Superescalar**"

Superscalar processors were introduced even before multi-core and all modern designs belong to this class. It is not quite vector processors. Like vector processors with parallel ALUs, they are actually capable of executing instructions in parallel, but in contrast to vector computers, they are different instructions. Instead of replication of the basic functional units n-times in hardware (e.g. the ALU), superscalar Processors exploit the fact that there already are multiple functional units.. [Philipp Häfliger Dag Langmyhr. Spring 2010. Computer Architecture, Kompendium for INF2270. University Of Oslo Department of Informatics.]

Múltiples operaciones son iniciadas y finalizadas simultáneamente

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Vectorial

William Stallings coloca al tema bajo un contexto sumamente complejo, y talvez ello sugiera o promueva una visión algo pesimista del asunto.

Patterson y Hennessy hacen un planteamiento del tema mucho más didáctico pero bastante simplificado, al grado que pueda llegar a producir una visión menos completa del tema.

Behrooz Parhami se introduce en una discusión menos didáctica, pero buena, y talvez promueva una visión media y menos pesimista.

Se sugiere tomar como marco de referencia el reporte de IBM

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Vectorial

Al iniciarse en un tema desconocido, es frecuente imaginárselo como algo muy complejo.

Sólo hay que tener presente que las cosas muy complejas se integran de otras bastante sencillas.

Sin embargo, en lo que concierne a los conceptos temáticos, vistos hasta el momento, y otros que faltan, su implementación no se agota en un curso de arquitectura como el presente.

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Vectorial

A estas alturas, las máquinas de estado pueden ser un caso. Cuando se desconocen los modelos pueden llegar a parecer muy complejos, pero al tiempo de dominarlas se perciben como algo muy sencillo.

Además, las Máquinas de Estados Finitos, son sumamente útiles en los modelos de las arquitecturas de computadoras.

Por otro lado, los propios diseñadores, que trabajan para empresas lideres en el tema, actualmente se encuentran buscando soluciones a planteamientos aún no resueltos.

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Vectorial (SIMD)

El concepto de Arquitectura Vectorial tiene su origen, como el nombre lo indica en operaciones vectoriales.

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} x \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

$$c_1 = (a_1) x (b_1)$$

$$c_k = (a_k) x (b_k)$$

Cada resultado c_k es independiente del resultado c_{k+1}

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Vectorial (SIMD)

El concepto de Arquitectura Vectorial tiene su origen, como el nombre lo indica en operaciones vectoriales.

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} x \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

$$c_1 = (a_1) x (b_1)$$

$$c_k = (a_k) x (b_k)$$

ARQUITECTURA VECTORIAL

Se saca ventaja del concepto de superescalar, aunque no necesariamente es superescalar. Una operación con vectores se ordena ejecutar en una sola instrucción, aunque ello implique múltiples flujos de datos.

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

ARQUITECTURA VECTORIAL

Se saca ventaja del concepto de superescalar, aunque no necesariamente es superescalar. Una operación con vectores se ordena ejecutar en una sola instrucción, aunque ello implique múltiples flujos de datos.

ARQUITECTURA VECTORIAL

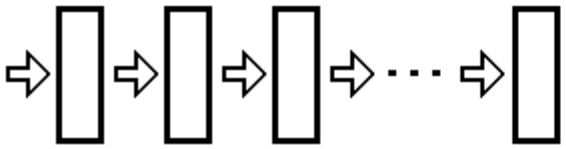
Solo un flujo de instrucciones
Múltiples datos procesándose

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

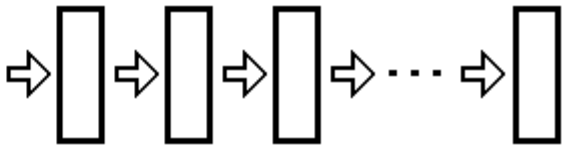
Arquitectura Vectorial

Sin embargo, a nivel de hardware, las operaciones han de realizarse mediante instrucciones y bits.

Una categorización de los computadores paralelos que surgió desde los años 1960, se basa en el número de flujo de instrucciones y de datos. [Patterson y Hennessy, pág. 648].



Flujo de Instrucciones



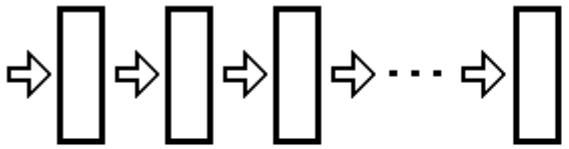
Flujo de Datos

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

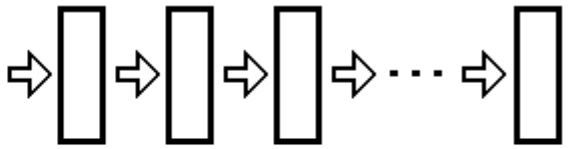
Arquitectura Escalar

Taxonomía de Flynn

SISD: Un monoprocesador convencional tiene un Flujo de Instrucciones y un Flujo de Datos (**S**ingle **I**nstrucction **S**ingle **D**ata stream).



Flujo de Instrucciones



Flujo de Datos

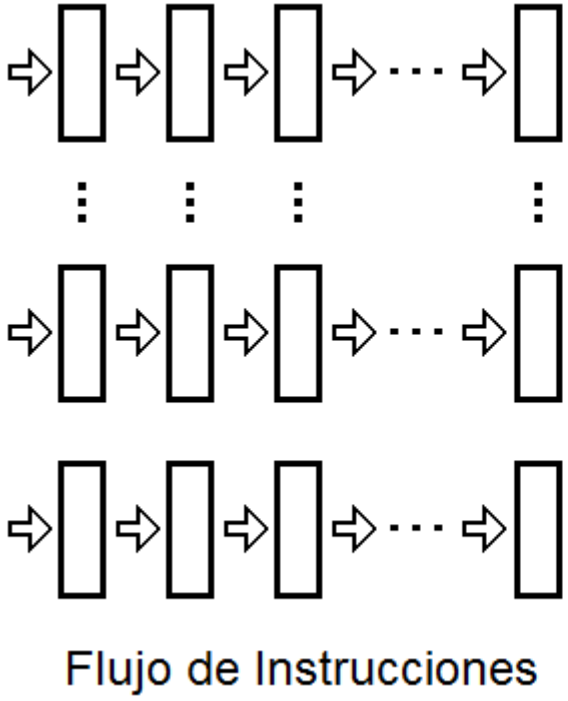
Puede implicar operaciones en paralelo pero sobre un solo dato
Además de Pipeline y/o Superpipeline

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Vectorial

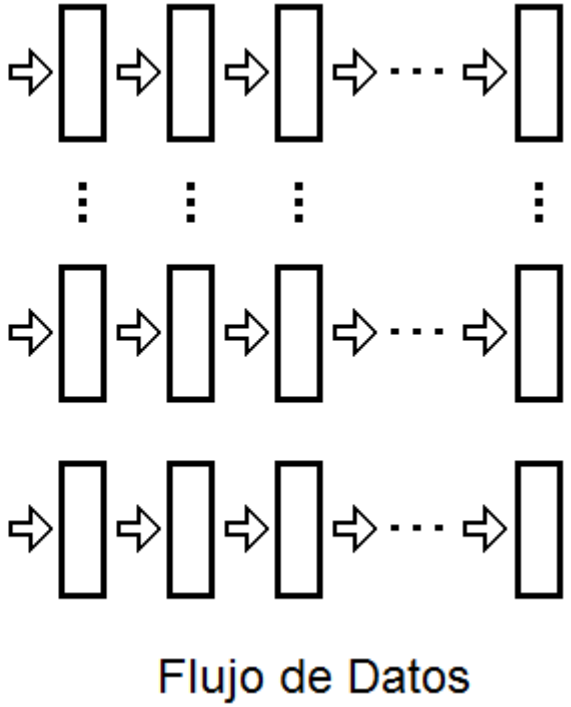
Taxonomía de Flynn

MIMD: En un multiprocesador hay varios Flujos de Instrucciones y varios Flujos de Datos. (**M**ultiple **I**nstrucction **M**ultiple **D**ata stream).



Esto es MIMD

Superescalar



Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Vectorial

MIMD: En un multiprocesador hay varios Flujos de Instrucciones y varios Flujos de Datos. (**M**ultiple **I**nstrucction **M**ultiple **D**ata stream).

Diferencia entre **Superescalar** y **Vectorial**

En Superescalar los resultados de cada operación dependen de la operación anterior. No son independientes.

En Vectorial se saca ventaja de operaciones en que los resultados de cada operación no dependen de la operación anterior. Son independientes. Por ejemplo, operaciones entre vectores.

Sin embargo, en operaciones parciales, el modelo Vectorial puede sacar ventaja de los modelos Segmentado y Superescalar.

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Vectorial

MIMD: En un multiprocesador hay varios Flujos de Instrucciones y varios Flujos de Datos. (**M**ultiple **I**nstrucction **M**ultiple **D**ata stream).

Diferencia entre **Superescalar** y **Vectorial**

En Superescalar se ejecutan varias instrucciones, cada una con sus respectivos datos, mientras que en vectorial se ejecuta una instrucción con múltiples datos [kompendium-inf2270.pdf, pág 83]

Cada procesador se encuentra procesando una instrucción al tiempo que otro procesa otra instrucción (superescalar).

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Vectorial

MIMD: En un multiprocesador hay varios Flujos de Instrucciones y varios Flujos de Datos. (**M**ultiple **I**nstrucction **M**ultiple **D**ata stream).

Diferencia entre **Superescalar** y **Vectorial**

Como se mencionó previamente, el modelo Vectorial, saca ventaja de los modelos Segmentado y Superescalar.

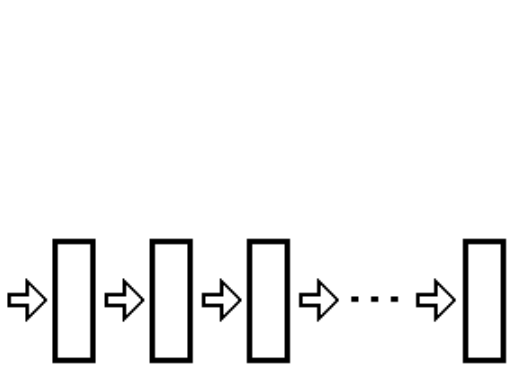
Con el objeto de optimizar hardware, algunos módulos como las ALU utilizan segmentación.

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Vectorial

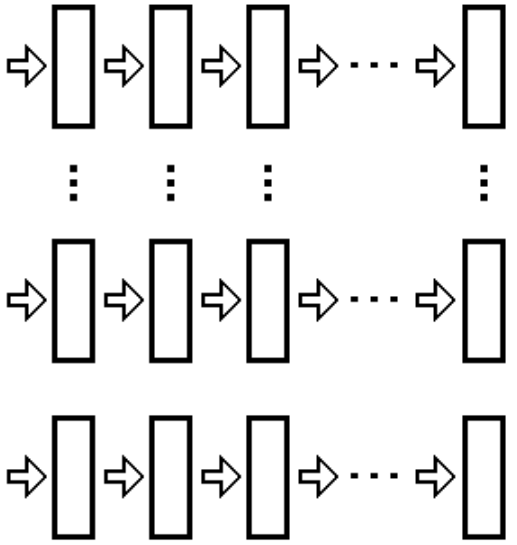
SIMD: Single Instruction Multiple Data.

Por ejemplo, una Instrucción SIMD, podría sumar 64 números enviando 64 flujo de datos a 64 ALUs para hacer 64 sumas en un único ciclo.



Flujo de Instrucciones
Una única instrucción por ciclo

Esto es SIMD Vectorial



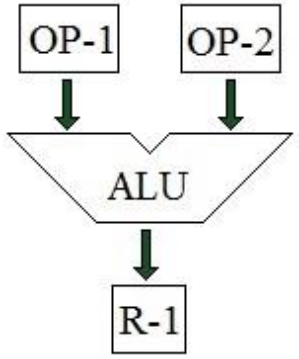
Flujo de Datos
Multiples Datos por ciclo

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Vectorial

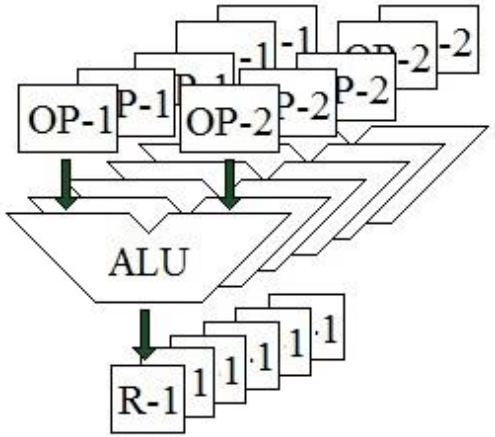
SIMD: Single Instruction Multiple Data.

Por ejemplo, una Instrucción SIMD, podría sumar 64 números enviando 64 flujo de datos a 64 ALUs para hacer 64 sumas en un único ciclo.



Superescalar

Single-Instruction Single-Data
(MIMD)



Vectorial

Single-Instruction Multiple-Data
(SIMD)

Vector Processors: Revised by Krste Asanovic. Department of Electrical Engineering and Computer Science, MIT.

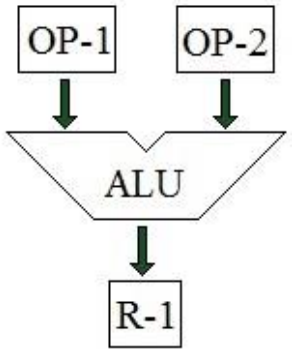
Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Vectorial

SIMD: Single Instruction Multiple Data.

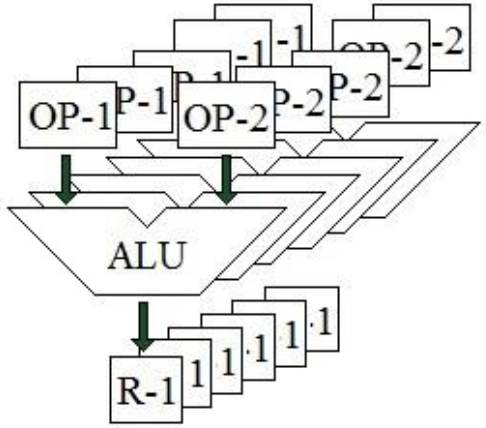
Por ejemplo, una Instrucción SIMD, podría sumar 64 números enviando 64 flujo de datos a 64 ALUs para hacer 64 sumas en un único ciclo.

Puede implicar Pieline y/o Superpipeline



Superescalar

Single-Instruction Single-Data
(MIMD)



Vectorial

Single-Instruction Multiple-Data
(SIMD)

Puede implicar Pieline y/o Superpipeline

IBM RISC System/6000: Architecture and Performance

IEEE_Micro_1991_-_IBM_6000.pdf

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

MISD

Significa **Múltiples Instrucciones, Single Data**. En un principio no se veían posibles aplicaciones de esta clasificación, debido principalmente a que múltiples Instrucciones procesando el mismo dato crea redundancia.

El fenómeno de redundancia puede ser aprovechado para respaldar datos, resultado de la redundancia, en servidores localizados en diferentes regiones geográficas. No tiene aplicaciones en nuestros días, hasta donde se sabe, pero es posible que pronto surjan aplicaciones en minerías de datos.

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Vectorial

SPMD: **S**ingle **P**rogram **M**ultiple **D**ata.

Un procesador vectorial es un procesador que puede operar en un vector entero en una instrucción. El operando con las instrucciones son vectores completas en lugar de un elemento.

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Vectorial

SPMD: **S**ingle **P**rogram **M**ultiple **D**ata.

Un único programa se ejecuta en todos los procesadores introduciendo sentencias condicionales para conseguir que procesadores diferentes ejecuten secciones de código diferentes.

Si tenemos en cuenta que las operaciones se llevan a cabo a nivel de bits, este modelo, en verdad puede caer en los anteriores.

Por esta razón es que SPMD no ha tenido relevancia debido a su dependencia de los otros modelos.

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Vectorial

Aunque se reconoce que la Arquitectura Vectorial es en esencia SIMD, en realidad obtiene ventaja de los otros modelos SISD y MIMD, como estrategia desde la óptica de un diseñador.

La filosofía básica de las arquitecturas vectoriales es recoger los datos de memoria, ponerlos ordenados en un **gran conjunto de registros**, operar sobre ellos utilizando, posiblemente, **otro gran conjunto de registros** y escribir los resultados en memoria.

El elemento clave de las Arquitecturas Vectoriales es el **gran conjunto de registros**, pero sobre todo, que producen **resultados independientes**.

Clasificación de las arquitecturas atendiendo al flujo de instrucciones y datos

Arquitectura Vectorial

Finalmente **MISD**

MISD no tiene aplicaciones en nuestros días, aunque ha habido propuestas para diseñar procesadores MISD para crear redundancia de datos.

Se ha argumentado que los procesadores MISD serían más robustos en cuanto a tolerancia a errores.

Redundancia y Concurrencia ???