Estructura y Funciones de la CPU

- Organización del procesador
- Registros
- Ciclo de instrucción
- El procesador Pentium
- El procesador PowerPC

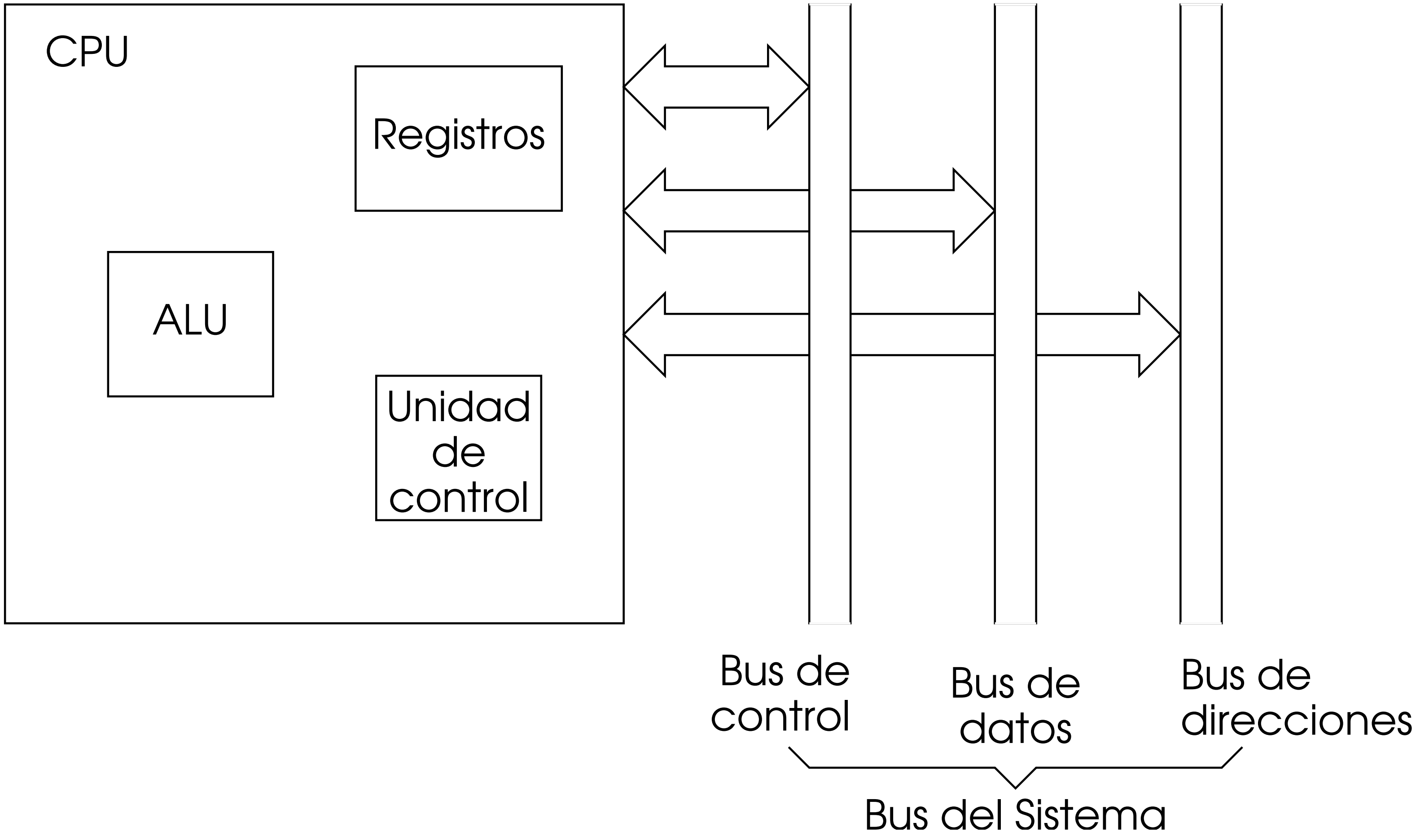
Organización del Procesador

Los objetivos del CPU son:

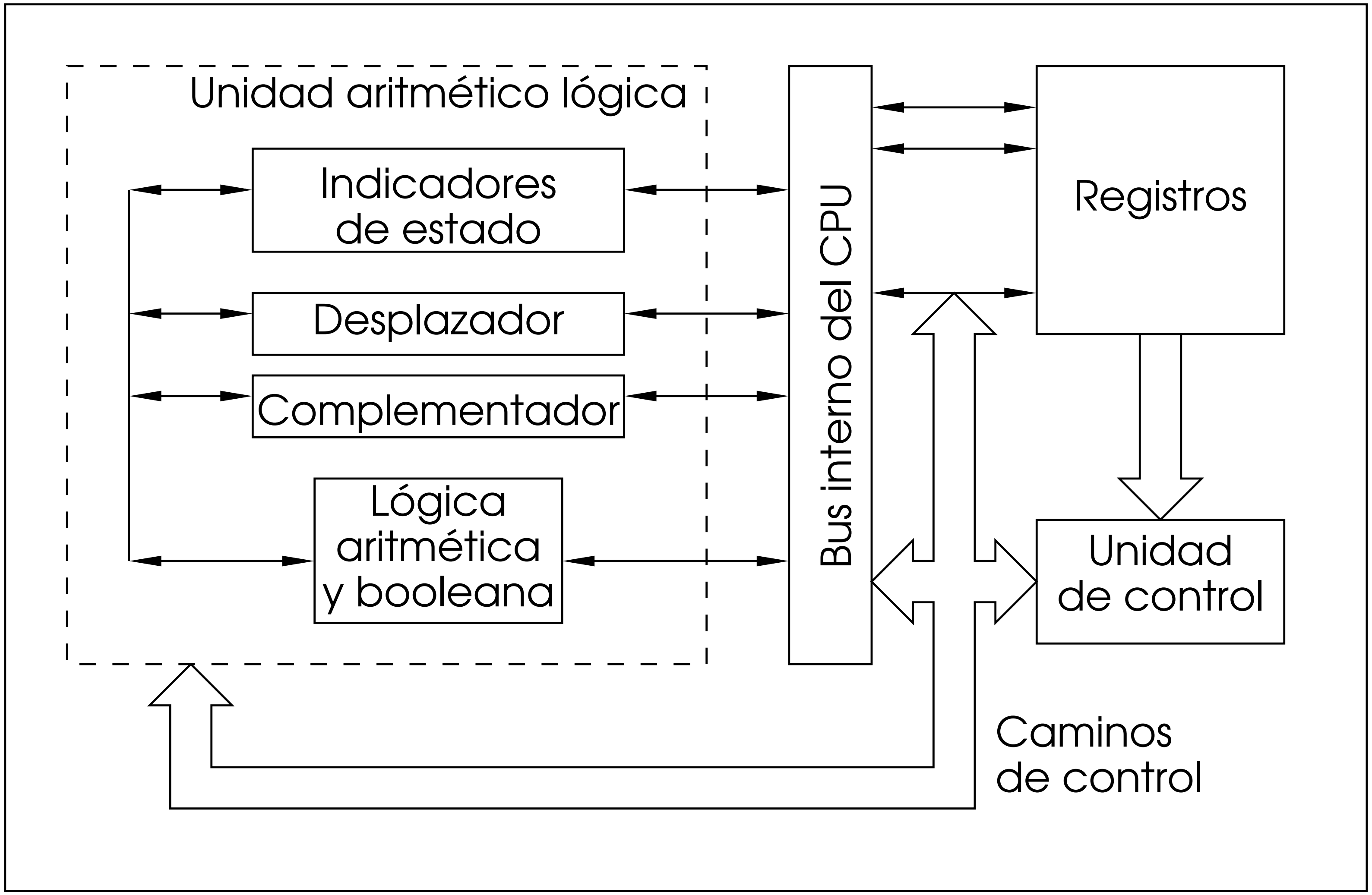
- *Captar instrucciones*
- *Interpretar instrucciones*
- *Captar datos*
- *Procesar datos*
- *Escribir datos*

Al analizar las estructuras del CPU y el bus del sistema con la estructura interna del CPU, se observa la similitud entre ambas.

El CPU y el bus del sistema



Estructura interna del CPU



Organización de los registros

- *Registros visibles al usuario*. Ayudan al programador de lenguaje de máquina o ensamblador a minimizar las referencias a memoria principal.
- *Registros de control y de estado*. Empleados por la unidad de control para manejar el funcionamiento del CPU y por programas privilegiados del S.O. para controlar la ejecución de los mismos.

Registros visibles al usuario

Son aquellos que pueden ser referenciados por medio del lenguaje máquina que ejecuta el CPU. Se tienen los siguientes tipos de registros:

- *De uso general*. El programador los asigna a diversas funciones, puede contener el operando para cualquier código de operación. Pueden tener algunas restricciones para ser totalmente generales, como la existencia de registros específicos para operaciones en punto flotante.
- *De datos*. Son usados únicamente para contener datos y no se pueden emplearse en el cálculo de una dirección de operando.

- *De dirección*. Son registros parcialmente generales y pueden usarse para algún modo de direccionamientos, tales como:
  - *Punteros de segmento*. En máquinas con direccionamiento segmentado.
  - *Registros índice*. Direccionamiento indexado, y pueden ser autoindexados.
  - *Puntero de pila*. Apuntador a la cabecera de la pila.
- *Códigos de condición*. Es parcialmente visible al usuario, el cual contiene *códigos de condición* (indicadores o “flags”). Los códigos de condición son puestos por el hardware del CPU como resultado de una operación.

Registros de control y de estado

Por lo general no son visibles al usuario. Se requieren por los menos los siguiente cuatro registros esenciales para la ejecución de una instrucción:

- *Contador de programa (“Program Counter”, PC)*. Contiene la dirección de la instrucción a captar.
- *Registro de instrucción (“Instruction Register”, IR)*. Contiene la instrucción captada más recientemente.
- *Registro de dirección de memoria (“Memory Address Register”,MAR)*. Contiene la dirección de una posición de memoria.
- *Registro intermedio de memoria (“Memory Buffer Register”, MBR)*. Contiene la palabra de datos a escribir en memoria o la palabra leída más recientemente.

Palabra de estado del programa (PSW)

Los diseños de CPUs incluyen un registro o un conjunto de registros que contienen información de estado, como códigos de condición e información de estado. Los campos o indicadores comunes son los suguienes:

- *Signo*.
- *Cero*.
- *Acarreo*.
- *Igual*.
- *Desbordamiento*.
- *Interrupciones habilitadas/inhabilitadas*.
- *Supervisor*. Informa si funciona el CPU en modo supervisor (acceso a zonas privilegiadas de memoria o instrucciones) o usuario.

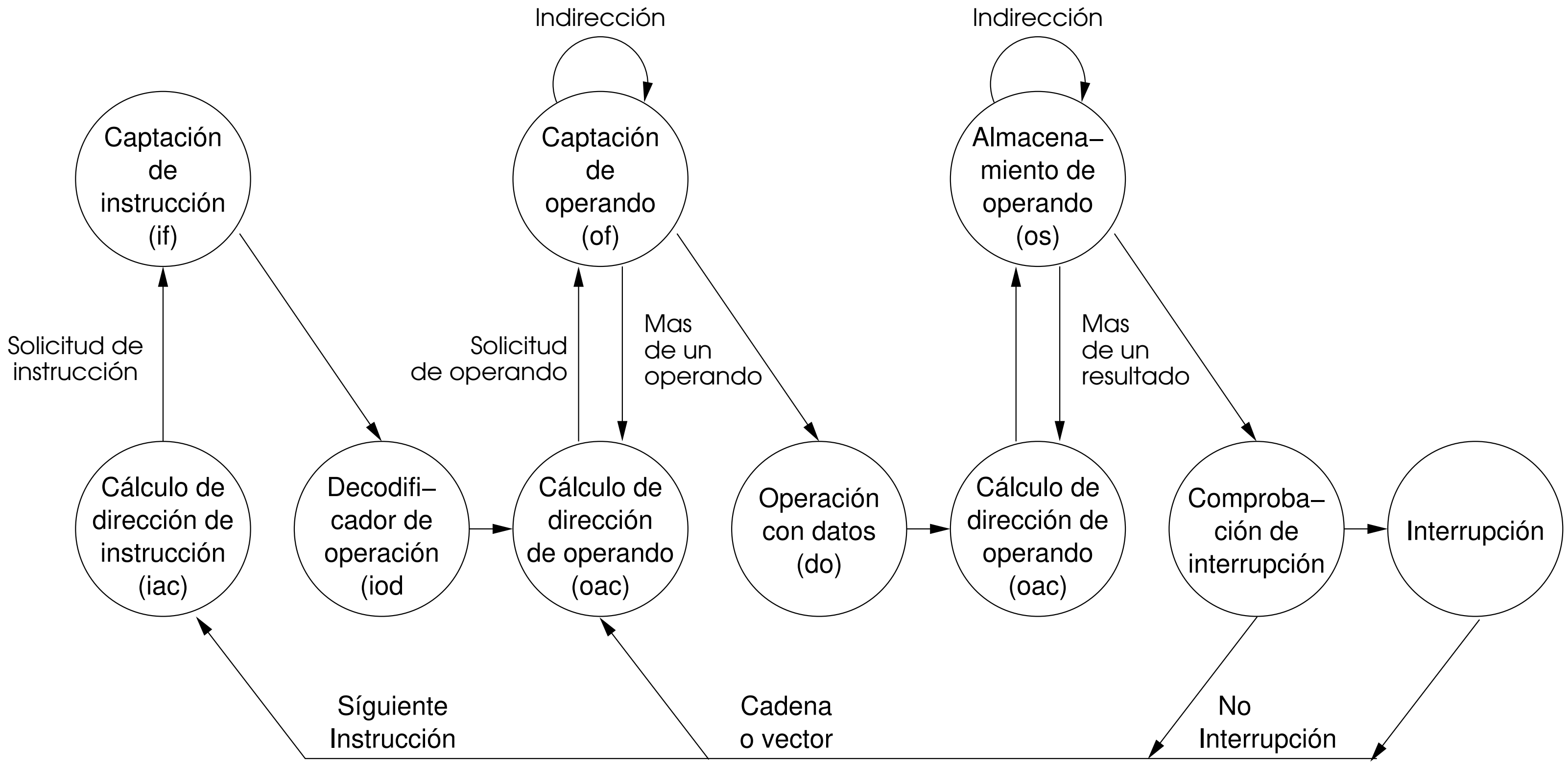
El ciclo de instrucción

El ciclo de instrucción incluye los siguientes subciclos:

- *Captación*. Llevar la siguiente instrucción de memoria al CPU.
- *Ejecución*. Interpretar el código de operación y hacer la operación indicada.
- *Interrupción*. Si están habilitadas las interrupciones y ocurre una interrupción, se debe salvar el proceso actual y atender la interrupción.

Ciclo indirecto

Se presenta cuando se da la captación de direcciones indirectas y es un subciclo de instrucción más.

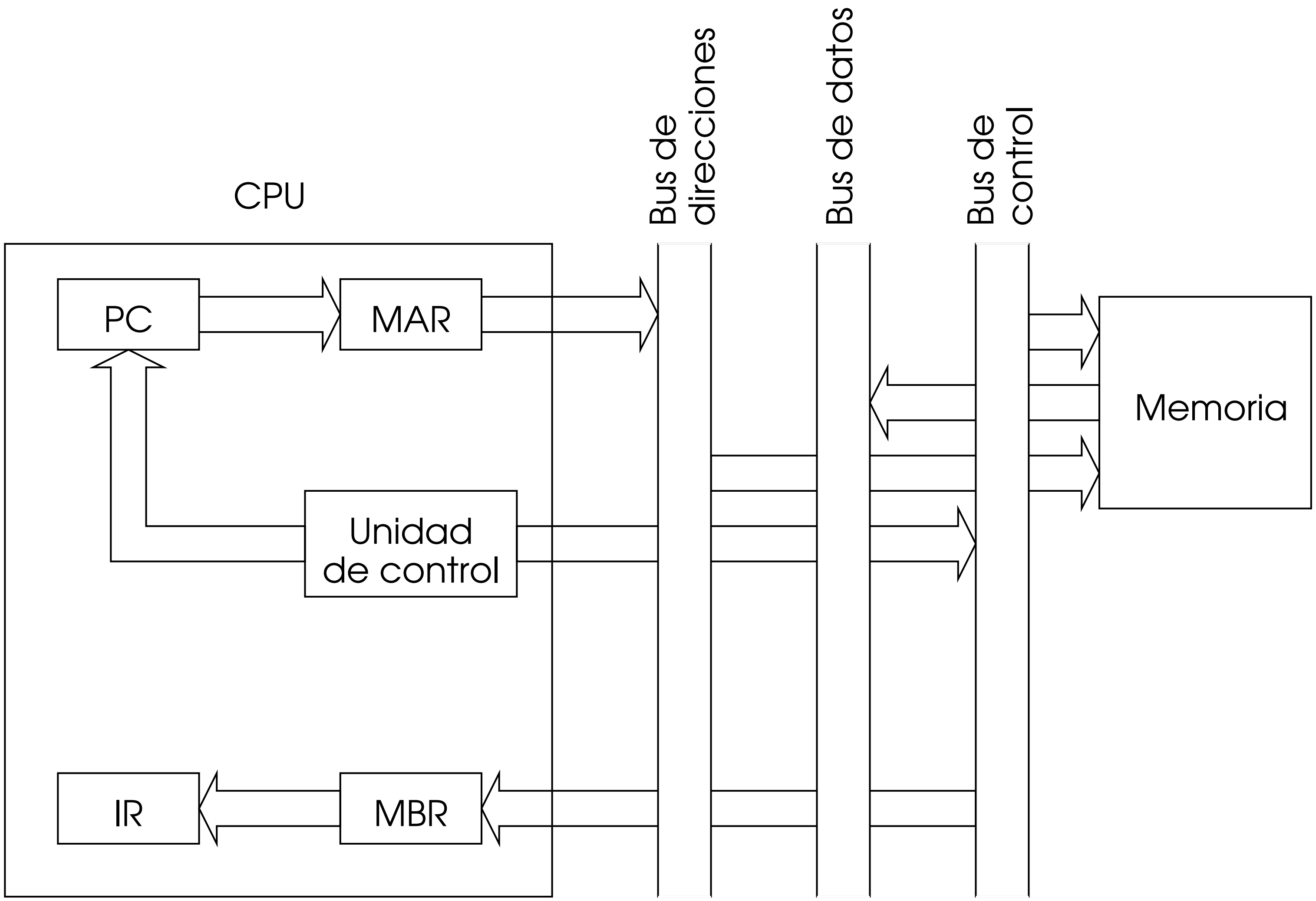


Flujos de datos

En el *ciclo de captación* se lee una instrucción. El PC contiene la dirección de la siguiente instrucción, la cual es llevada a MAR y puesta en el bus de direcciones. La unidad de control solicita una lectura de memoria, y el resultado se pone en el bus de datos, se copia en MBR y se lleva a IR, mientras tanto, PC se incrementa en 1.

Al terminar, la unidad de control examina IR para saber si contiene un campo de operando que use direccionamiento indirecto, en caso de ser este último se da un *ciclo indirecto*, donde los N bits más a la derecha, que contienen la dirección de referencia se transfieren a MAR, con lo que la unidad de control hace una lectura de memoria, para llevar la dirección del operando a MBR.

Flujo de datos, ciclo de captación



Segmentación de instrucciones

Similar al uso de una cadena de montaje en una fábrica, en donde un producto tiene que pasar por varias etapas de prodcción. Un programa puede ser resuelto en varias etapas simultáneamente. A esta forma se le conoce como *segmentación* ("pipelining") en donde por un extremo se aceptan nuevas entradas antes de que algunas entradas aceptadas anteriormente aparezcan como salidas en el otro extremo.

Una aproximación simple, es considerar la subdivisión del procesamiento en dos etapas: captación de instrucción y ejecución de instrucción. Los tiempos en los cuales una instrucción no accede a memoria principal, pueden usarse para captar la siguiente instrucción en paralelo con la ejecución de la actual, lo anterior se conoce como *prebúsqueda o precaptación de instrucción ("instruction prefetch") o solapamiento de la captación ("fetch overlap")*.

Si las etapas de captación y ejecución son de igual duración, el tiempo de ciclo de instrucción se reduce a la mitad. La duplicación de la velocidad de ejecución es poco probable por lo siguiente:

1. El tiempo de ejecución es más largo que el tiempo de captación
2. Una instrucción de bifurcación condicional hace que la dirección de la siguiente instrucción a captar sea desconocida.

Para conseguir una mayor aceleración, el cauce debe tener más etapas, considerar la siguiente descomposición:

- *Captar instrucción (“Fetch Instruction”, FI)*
- *Decodificar instrucción (“Decode Instruction”, DI)*
- *Calcular operandos (“Calculate Operands”, CO)*
- *Captar operandos (“Fetch Operands”, FO)*
- *Ejecutar instrucción (“Execute Instruction”, EI)*
- *Escribir operando (“Write Operand”, WO)*

Con la descomposición anterior, las diversas etapas tienen casi igual duración. Suponiendo que tienen igual duración se puede reducir el tiempo de ejecución de 9 instrucciones de 54 a 14 unidades de tiempo. Así mismo se requiere que no haya conflictos de memoria, por ejemplo las etapas FI, FO y WO que requieren un acceso a memoria.

Los factores que limitan la mejora de prestaciones, son los tiempos desiguales de las etapas, otra es la instrucción de bifurcación condicional, que al igual que las interrupciones, invalidan varias captaciones de instrucción.

La etapa CO puede depender de los contenidos de un registro que podría verse alterado por una instrucción previa que aún esté en el cauce. Podrían ocurrir otros conflictos con registros y con memoria.

Diagrama de tiempos para el cauce de instrucciones

Tiempo →

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
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

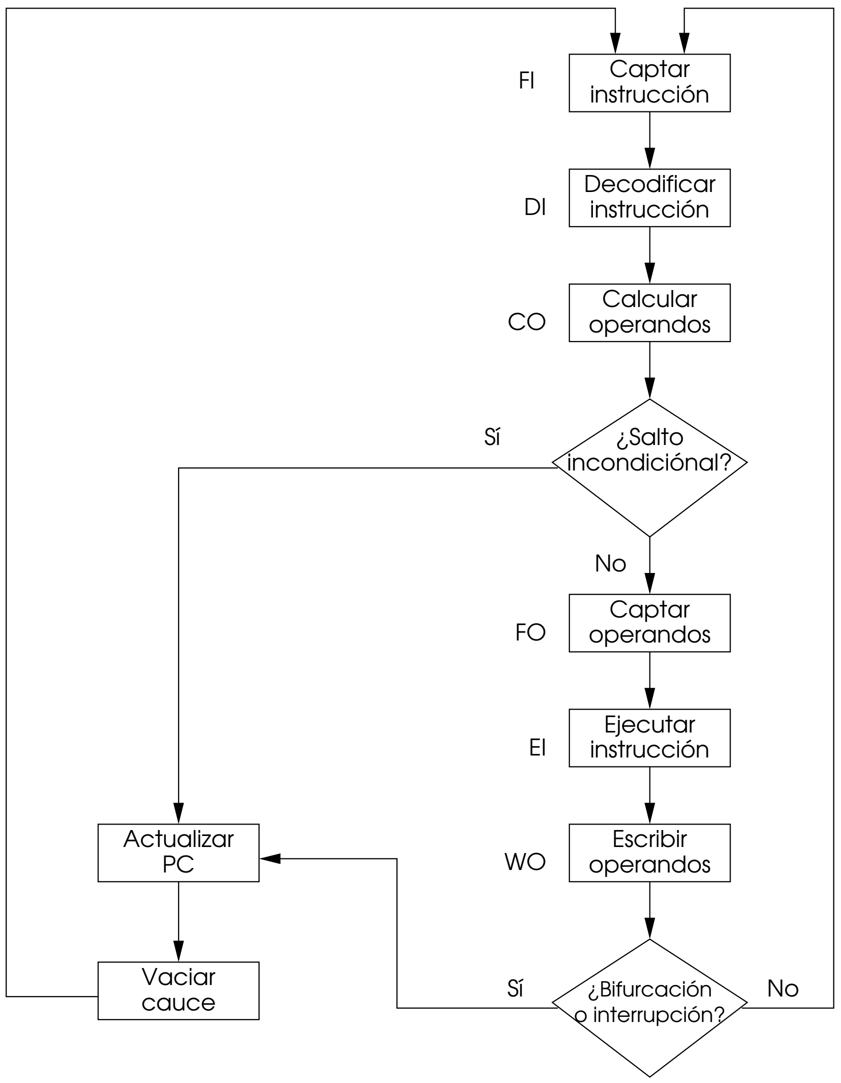
Efecto de una bifurcación condicional

Tiempo

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
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							
Instrucción 5					FI	DI	CO							
Instrucción 6						FI	DI							
Instrucción 7							FI							
Instrucción 15								FI	DI	CO	FO	EI	WO	
Instrucción 16									FI	DI	CO	FO	EI	WO

Penalización debida al salto

Cauce de instrucciones del CPU con seis etapas



Tratamiento de saltos

Uno de los mayores problemas del diseño de un cauce de instrucciones es asegurar un flujo estable de instrucciones en las etapas iniciales del cauce, donde el principal obstáculo es la instrucción de bifurcación condicional, ya que es imposible determinar si el salto se producirá o no, hasta que la instrucción no se ejecute.

Las aproximaciones para el tratamiento de bifurcaciones condicionales son:

- Flujos múltiples
- Precaptar el destino del salto
- Buffer de bucles
- Predicción de saltos
- Salto retardado

Flujos múltiples

Un cauce simple tiene penalización por las instrucciones de bifurcación ya que debe escoger una de las dos instrucciones a captar a continuación y puede hacer la elección equivocada. Una solución simple es duplicar las partes iniciales del cauce y dejar que capte las dos instrucciones, utilizando los dos caminos. Los problemas son los siguientes:

- Hay retardos por conflictos en el acceso a los registros y a la memoria.
- Pueden entrar en cualquiera de los dos flujos instrucciones de bifurcación adicionales antes de que se resuelva la decisión de la bifurcación original.

Precaptar el destino del salto

Al identificar una instrucción de bifurcación condicional, se precapta la instrucción del salto, además de la siguiente a la de bifurcación. Se guarda entonces esta instrucción hasta que se ejecute la instrucción de bifurcación. Si se produce el salto, el destino ya habrá sido precaptado. El IBM 360/91 usa esta aproximación.

Buffer de bucles

Es una memoria pequeña de gran velocidad gestionada por la etapa de captación de instrucción del cauce, que contiene, secuencialmente, las n instrucciones captadas más recientemente. Si se va a producir un salto, el hardware comprueba primeramente si el destino del salto está en el buffer. Se tienen los siguientes efectos:

1. Se anticipa almacenando algunas instrucciones que secuencialmente están despueés de la dirección donde se capta la instrucción actual.
2. En un salto a unas pocas posiciones más allá de la dirección de la instrucción de bifurcación, el destino ya esta en el buffer.
3. Es buena en el tratamiento de bucles, o iteraciones.

Predicción de saltos

Se pueden usar varias técnicas para predecir si un salto se va a producir, se pueden usar las siguientes:

- Nunca se salta.
- Siempre se salta.
- Según el código de operación.
- Conmutador saltar/no saltar.
- Tabla de historia de saltos.

Las primeras tres son estáticas: no dependen de la historia de la ejecución que haya tenido lugar hasta la instrucción de bifurcación condicional.

Las dos últimas aproximaciones son dinámicas: dependen de la historia de la ejecución.

Conmutador saltar/no saltar y Tabla de historia de saltos

Intentan mejorar la exactitud de la predicción registrando la historia de las instruciones de bifurcación condicional. Una forma es asociar a cada una de las instrucciones anteriores uno o más bits que refleje su historia reciente y que sean referenciados como un conmutador saltar/no saltar.

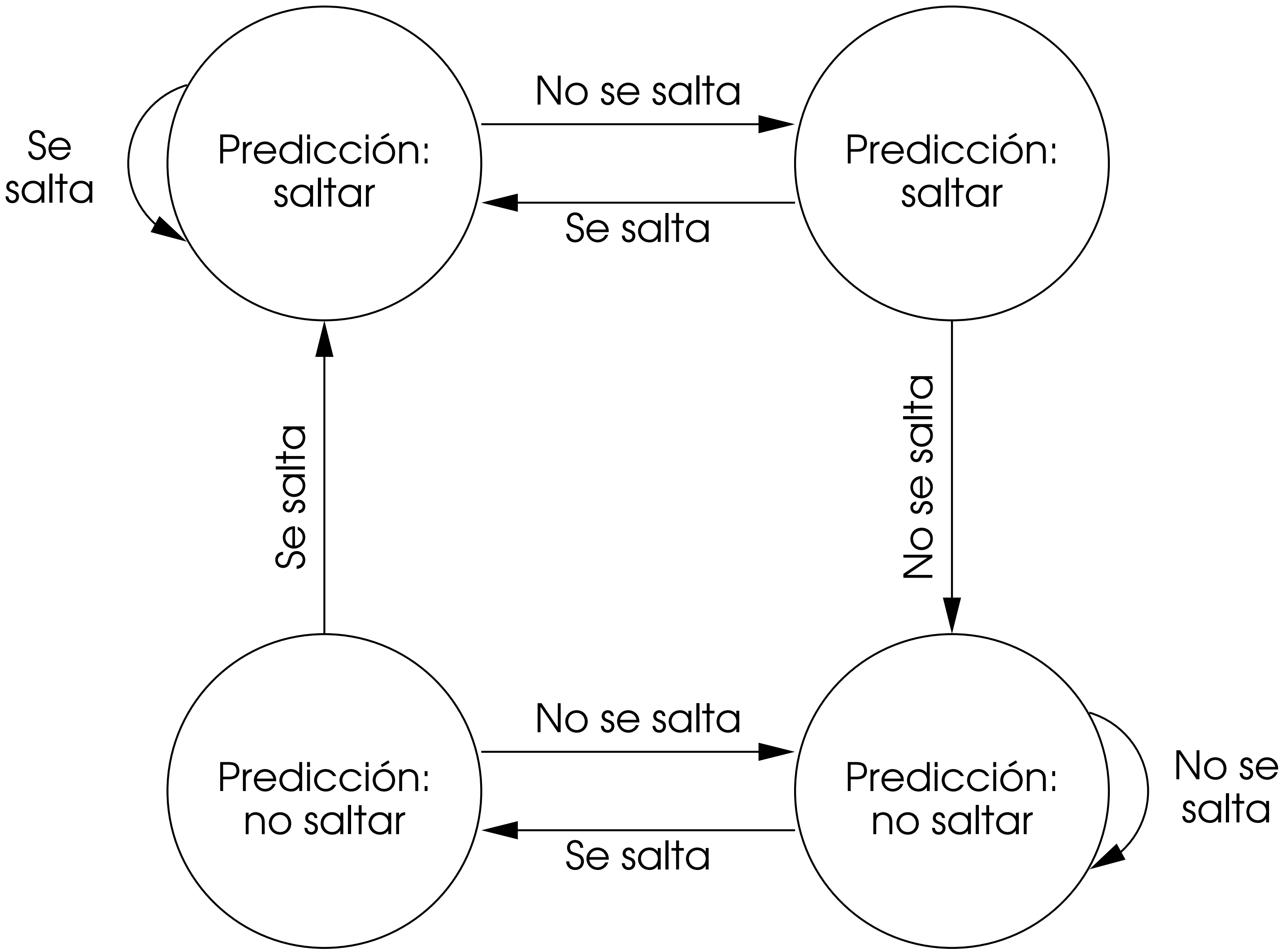
Las formas de asociación de los bits son:

- Asociarlos a cualquier instrucción de bifurcacin que esté en cache, cuando la instrucción en cache es reemplazada se pierde su historia.
- Mantener una pequeña tabla de las instrucciones de bifurcacin ejecutadas recientemente, con uno o ms bits, donde el procesador podría acceder la tabla asociativamente, como un cache, o usando los bits de orden inferior de la dirección de la instrucción de bifurcación.

Con un único bit, lo que se puede registrar es si la última ejecución de una instrucción dio lugar a un salto o no. En las instrucciones de bifurcación condicional casi siempre da lugar a un salto, con lo que ocurre un error en la predicción dos veces en cada uso del bucle: una vez cunado se entra y otra al salir.

Si se usan dos bits, se pueden emplear para registrar el resultado de las últimas veces que se ejecutó la instrucción asociada, o para registrar el estado de alguna otra forma. El proceso de decisión puede representarse por medio de una máquina de estados finitos con cuatro estados.

Diagrama de estados de la predicción de saltos



Conmutador saltar/no saltar y Tabla de historia de saltos

El incoveniente es que si la decisión que se toma es efectuar el salto, la instrucción destino no se capta hasta que su dirección, que sea decodificada. Se puede lograr una mayor eficiencia si la instrucción captada puede iniciarse tan pronto como se toma la decisión de salto, para lo que se requiere guardar más información, en lo que se conoce como un buffer de destino de saltos, o tabla de historia de saltos (“branch history table”, BHT).

La tabla de historia de saltos es una pequeña memoria cache asociada con la etapa de captación de instrucción del cauce.