# Estructura y Función del Procesador

*Daniel Araya Román*

Un procesador incluye tanto registros visibles por el usuario, como registros de control/estado. Los primeros se pueden referenciar de manera implícita o explícita, en instrucciones máquina.

Pueden ser de uso general o tener utilidades especiales. Los seguntos se usan en conjunto a la unidad de control (UC), para controlar el funcionamiento del procesador. Por ejemplo el contador de programa (PC). **La palabra de estado del programa** *(program word status, PSW)*. Continendo bits de estado y de condición, siendo estos el resultado más reciente de la operación aritmética realizada. Como también bits de inidicador, para saber si el procesador está en estado supervisor o usuario.

Los procesadores también utiliza una técnica conocida como segmentación de instrucciones *(instruction pipelining)*, que permite que el procesador dividir en un cauce el ciclo de instrucción, realizando varias etapas de manera paralela. No obstante los saltos condicionales complican la segmentación. Como el uso de cauces segmentados.

## Organización del Procesador

- **Captación de instrucción.** El procesador lee una instrucción de memoria (registro, caché, o memoria principal).
- **Decodificación de instrucción.** La instrucción se decodifica para determinar que acción debe de realizar.
- **Captación de operandos.** La instrucción puede exigir leer datos de la memoria o un módulo de entrada/salida.
- **Procesamiento de datos.** La ejecución de una instrucción puede exigir una operación aritmética o lógica.
- **Escritura de operando.** Los resultados de la operación se escriben en memoria o en un módulo de entrada/salida.

### Etapas del Ciclo de Instrucción

*Instruction Fetch, Instruction Decode, Operands Fetch, Execution, Operand Write.*

Para que el procesador pueda realizar todas estas etapas, debe de tener una memoria interna donde almacenar datos de manera temporal. Recordar la última instrucción ejecutada, para que pueda saber donde debe de buscar la siguiente instrucción. En la siguiente figura se indican los caminos de transferencia de datos y de la lógica de control. Todo esto conectado con el bus interno del procesador. Este es necesario para poder transferir datos entre los diversos registros y la ALU. Ya que esta sólo opera con datos de la memoria interna del procesador.

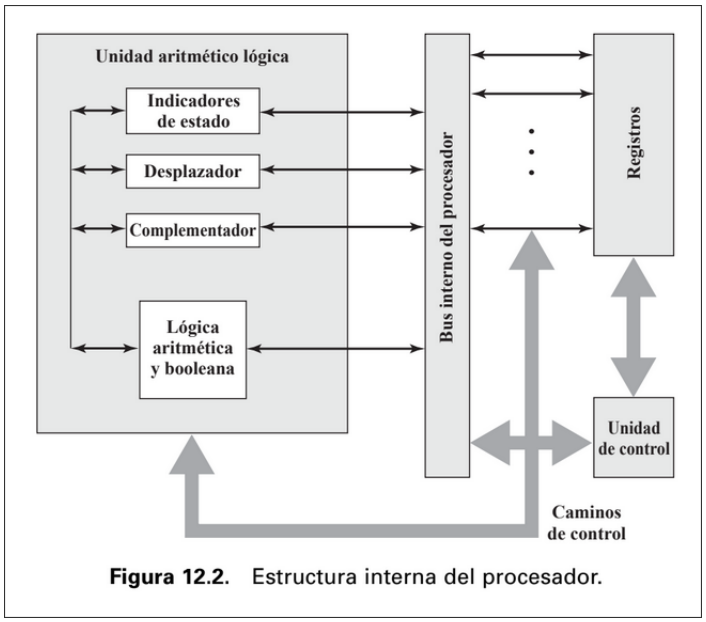


**Figura 12.2.** Estructura interna del procesador.

Figure 1: Estructura Interna de un Procesador. [1]

## Organización de los Registros

Existen dos tipos principales de registros del procesador.

- **Registros Visibles por el Usuario.** Permiten al programador de lenguaje máquina o ensamblador minimizar las referencias a memoria principal. Por medio de la optimización del uso de los mismos.
- **Registros de Control y Estado.** Son utilizados por la **unidad de control**, para controlar el funcionamiento del procesador y por programas privilegiados del sistema operativo. Como motivo de control de ejecución de programas de usuario.

## Registros Visibles por el Usuario

Se pueden clasificar en las distintas categorías.

- **Registros de uso general.** Se pueden ser asignados por el programador a diversas funciones. En algunos casos pueden utilizarse para funciones de direccionamiento. En otros casos hay una separación parcial entre los registros de datos y de direcciones.
- **Registros de datos.** Pueden usarse únicamente para contener datos y no pueden emplear en el cálculo de la dirección de un operando.
- **Registros de direcciones.** Pueden ser de uso general, o pueden estar dedicados a un modo de direccionamiento particular. Por ejemplo.
  - **Registros de segmento.** En una máquina con direccionamiento segmentado, este registro contiene la dirección de la base del segmento. Pueden existir múltiples registros.

  - **Registros índice.** Se usan para direccionamiento indexado. Pueden ser autoindexados.
  - **Registros de pila.** Normalmente hay un registro dedicado que apunta a la cabecera de la pila. Permitiendo el direccionamiento implícito. Por lo que las instrucciones de pila no necesitan contener un operando explícito referente a ella.

- **Registros de códigos de condición.** Tambien conocidos como **banderas**. Son bits fijados por el hardware del procesador como resultado de una operación aritmética o lógica. Se agrupan en uno o más registros y normalmente forman parte de un registro de control. **El programador no puede alterarlos.**

Algunos procesadores como el Itanium (IA-64), o procesadores MIPS, no utilizan códigos de condición en lo absoluto.

**Tabla 12.1.** Códigos de condición.

| Ventajas | Inconvenientes |
|---|---|
| 1. Dado que los códigos de condición son fijados por instrucciones aritméticas y de movimiento de datos normales, deberían reducir el número de instrucciones COMPARE y TEST necesarias. | 1. Los códigos de condición añaden complejidad tanto al hardware como al software. Los bits de códigos de condición se modifican con frecuencia de distintas maneras por instrucciones distintas, complicando la vida tanto al microprogramador como al escritor de compiladores. |
| 2. Las instrucciones condicionales tales como BRANCH se simplifican comparadas con las instrucciones compuestas tales como TEST AND BRANCH. | 2. Los códigos de condición son irregulares; normalmente no forman parte del camino de datos principal, necesitando conexiones hardware adicionales. |
| 3. Los códigos de condición facilitan los saltos múltiples. Por ejemplo, una instrucción TEST puede venir seguida de dos saltos, uno si menor o igual que cero y otro si mayor que cero. | 3. Frecuentemente, las máquinas con códigos de condición tienen que añadir de todos modos instrucciones especiales «sin códigos de condición» para situaciones especiales, tales como comprobación de bits, control de bucles y operaciones atómicas con semáforos. |
| | 4. En una implementación segmentada, los códigos de condición requieren una sincronización especial para evitar conflictos. |

Figure 2: Códigos de Condición. [1]

## Registros de Control y de Estado

La mayoría de estos registros, no son visibles por el usuario. Naturalmente distintas máquinas tendrán distintas organizaciones de registros y usarán distinta terminología. Algunos de estos registros son.

- **Contador de Programa (Program Counter,PC).** Contiene la dirección de la instrucción a captar.
- **Registro de Instrucción (Instruction Register, IR).** Contiene la instrucción más recientemente captada.
- **Registro de dirección de Memoria (Memory Address Register, MAR).** Contiene la dirección de una posición de memoria.
- **Registro de datos de Memoria (Memory Data Register, MDR).** Contiene un dato a escribir en memoria o el dato más recientemente leído de memoria.

Es necesario un mecanismo de almacenamiento intermedio el cual se dé salida a los bits que van a ser transferidos por el bus del sistema y se almacenen temporalmente los bits leídos del bus de datos.

**Ejemplo de organización de registros**

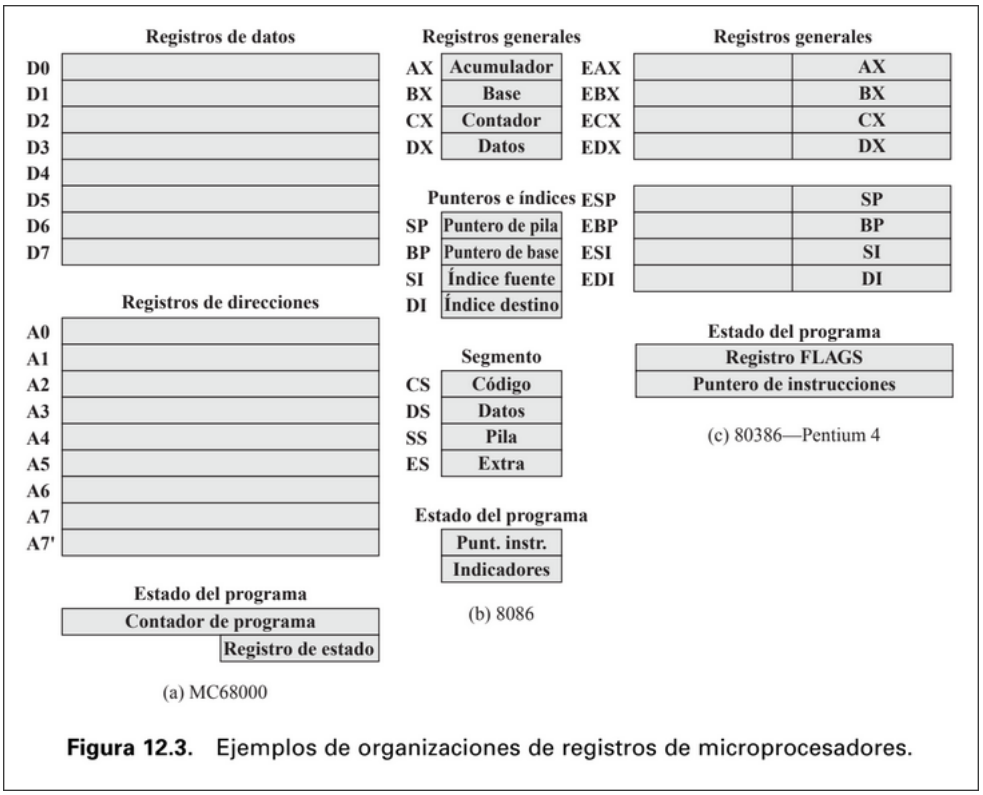


**Figura 12.3.** Ejemplos de organizaciones de registros de microprocesadores.

Figure 3: Organización de Registros. [1]

## Ciclo de Instrucción

El ciclo de instrucción se puede dividir en distintos subciclos. Captación, ejecución y interrupción.

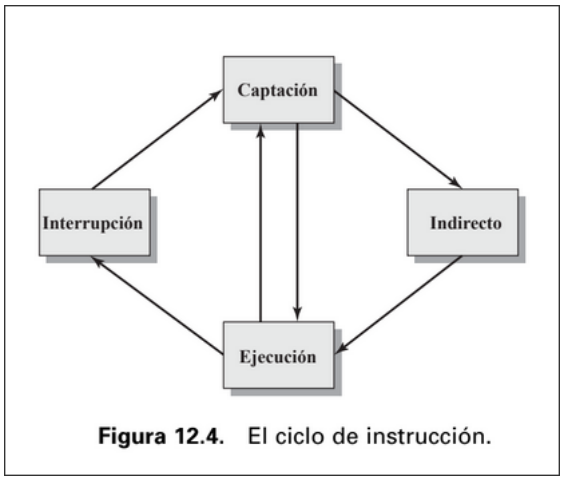


**Figura 12.4.** El ciclo de instrucción.

Figure 4: Ciclo de Instrucción. [1]

### Ciclo de captación

Se lee una instrucción de memoria. El PC contiene la dirección de la siguiente instrucción a captar. Esta se transfiere a MAR, y se coloca 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 MDR y luego se lleva al IR.

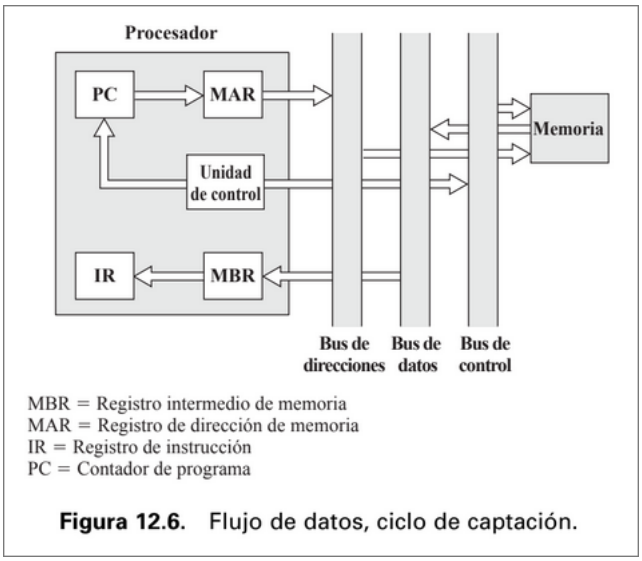


Figure 5: Ciclo de Captación. [1]

### Ciclo indirecto

La ejecución de una instrucción puede involucrar a uno o más operandos de memoria, cada uno de estos requiere un acceso a memoria. Si se usa el direccionamiento indirecto, serán necesarios dos accesos a memoria adicionales.

Después que una instrucción es captada, es examinada para determinar si incluye algún direccionamiento indirecto. Si es así, se captan los operandos usando el direccionamiento indirecto. De igual manera se puede procesar una interrupción antes de la captación de la siguiente instrucción.

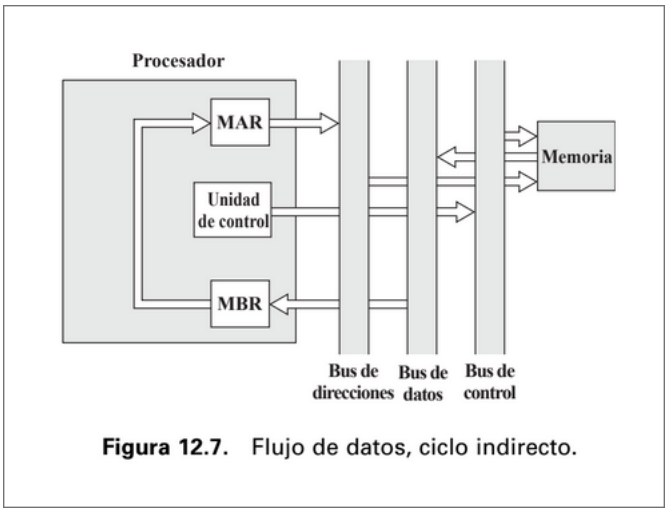


Figure 6: Ciclo Indirecto. [1]

Una vez se concluye el ciclo de captación, la unidad de control examina el contenido de IR para determinar si contiene un campo de operando que use direccionamiento indirecto. Si es así, se produce un ciclo indirecto. Los $n$ bits, más significativos del de MDR, que contienen la dirección de memoria, se transfieren al MAR.

### Ciclo de interrupción

Como los ciclos de captación e indirecto, el ciclo de interrupción es simple y predecible. El contenido actual del PC se guarda para que el procesador pueda reanudar su actividad después de la interrupción. Así el contenido de PC se transfiere a MDR para ser escrito en memoria. La dirección de memoria especial reservada para este propósito se carga en MAR desde la unidad de control. PC se carga con la dirección de la rutina de servicio de interrupción. Y el siguiente ciclo de instrucción comienza captando la instrucción oportuna.

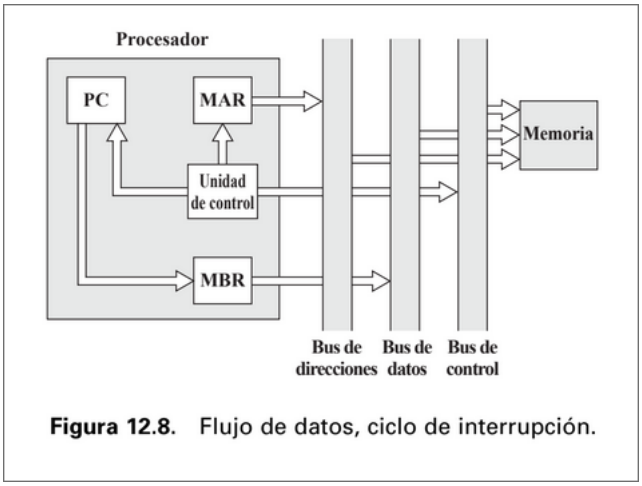


**Figura 12.8.** Flujo de datos, ciclo de interrupción.

Figure 7: Ciclo de Interrupción. [1]

### Ciclo de ejecución

Puede tomar diversas formas, ya que depende de las instrucciones máquina en el IR. Puede implicar transferencias de datos entre registros, lectura o escritura de memoria o entrada/salida, o incluso la utilización de la ALU.

## Diagrama de estados del ciclo de instrucción

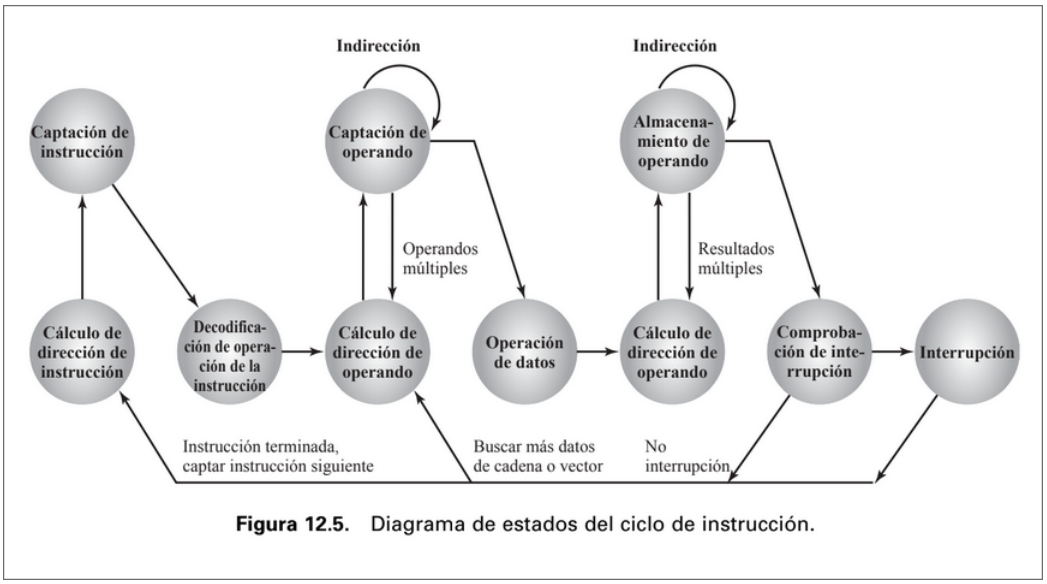


**Figura 12.5.** Diagrama de estados del ciclo de instrucción.

Figure 8: Diagrama de Estados del Ciclo de Instrucción. [1]

## Segmentación de Instrucciones

A medida que los computadores evolucionan, se pueden conseguir mayores prestaciones aprovechando los progresos de la tecnología. Como por ejemplo, uso de múltiples registros, la implementación de una memoria caché, o la segmentación de instrucciones.

### Estrategia de segmentación

Piense en una fábrica de montaje de automóviles. Una cadena de montaje saca partido del hecho que el producto pasa a través de distintas etapas de producción. Logrando trabajar sobre los productos en varias etapas simultáneamente. Esto se conoce como **segmentación de cauce (pipelining)**. Hay que entender que una instrucción por naturaleza tiene varias etapas. En el x86, se pueden distinguir 6 etapas.

### Etapas de una instrucción

- **Captación de instrucción (Fetch Instruction, FI).** Leer la supuesta siguiente instrucción en un buffer.
- **Decodificación de instrucción (Decode Instruction, DI).** Decodificar la instrucción y determinar que operación se debe realizar, y los campos de operando.
- **Captación de operandos (Fetch Operands, FO).** Calcular la dirección efectiva de cada operando. Involucrando distinto tipos de direccionamiento.
- **Ejecución de instrucción (Execute Instrucion, EI).** Realizar la operación indicada y almacenar el resultado.
- **Escritura de operandos (Write Operands, WO).** Almacenar el resultado en el lugar apropiado.

### Etapas de una instrucción deseado

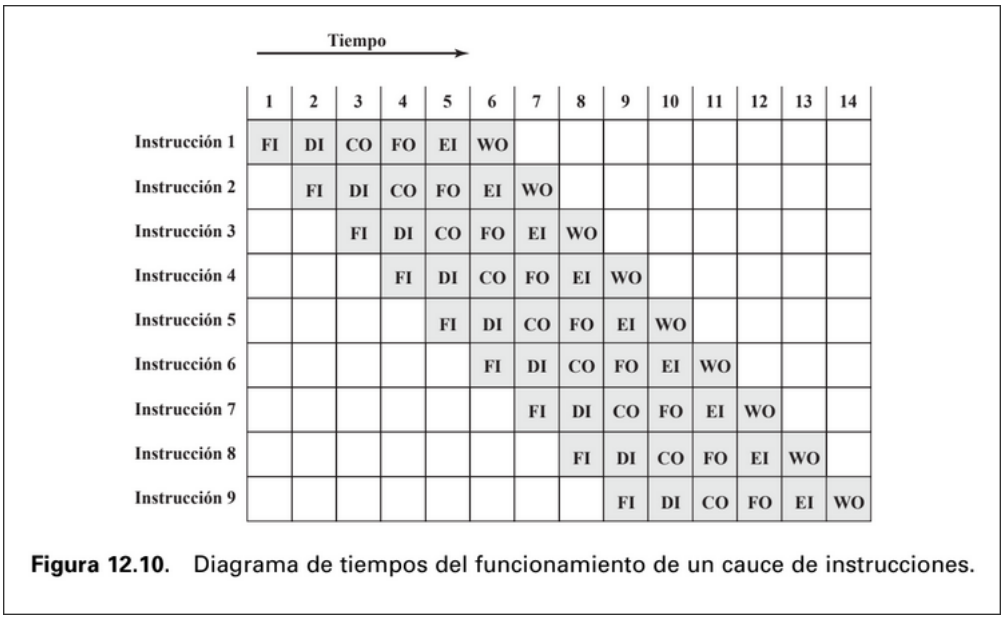


**Figura 12.10.** Diagrama de tiempos del funcionamiento de un cauce de instrucciones.

Figure 9: Etapas de una Instrucción. [1]

**Etapas de una instrucción con penalización de salto**

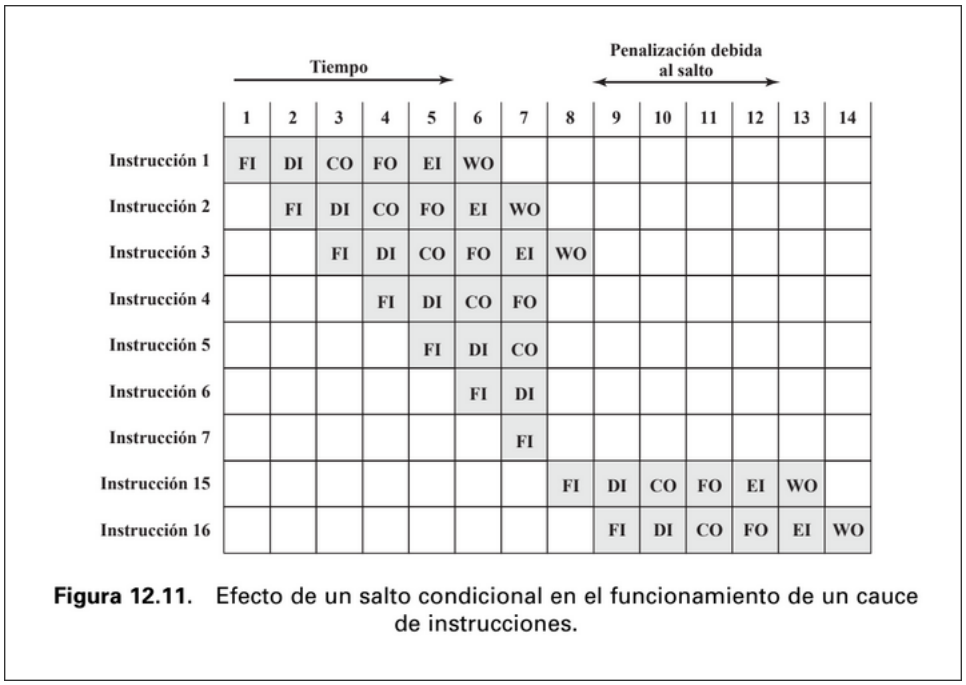


**Figura 12.11.** Efecto de un salto condicional en el funcionamiento de un cauce de instrucciones.

Figure 10: Etapas de una Instrucción con Penalización de Salto. [1]

## Tratamiento de Saltos

Los saltos condicionales son unos de los mayores problemas de diseño de un cauce de instrucciones. Hasta que la instrucción no se ejecuta realmente, es imposible determinar si se debe de saltar o no. Por lo que se han creado aproximaciones para tratar este problema.

### Tipo estático

- **Flujos múltiples.** Se ejecutan ambas ramas del salto, y se descarta el resultado de la rama que no se debe de ejecutar. Esto es muy costoso, ya que se desperdicia tiempo accediendo a memoria o a registros, e incluso en el cauce puede haber instrucciones de salto adicionales.
- **Precaptar el destino del salto.** Precapta la dirección de la instrucción a la que se debe de saltar.
- **Buffer de bucles.** Pensandolo de manera análoga es como una caché, sin embargo está limitada por el espacio. Elimina la penalización de los accesos a memoria, ya que las instrucciones son captadas solamente una vez de memoria. Aparte de que solo puede guardar instrucciones consecutivas.

### Tipo dinámico

- **Predicción de saltos.** Se usa un predictor de saltos, que se basa en el historial de saltos para predecir si se debe de saltar o no. Si se predice correctamente se evita la penalización, pero si se predice incorrectamente se debe de penalizar.
- **Salto retardado.** Las instrucciones de salto tienen lugar después de lo realmente deseado, llenar el cauce.

### Procesamiento de Interrupciones

Es un servicio que se proporciona para apoyar al sistema operativo en cuestión. Permite que la aplicación de un programa sea suspendido temporalmente, para que el sistema operativo pueda realizar una tarea. Para luego ser reanudado.

Hay dos tipos de eventos que suspende la ejecución de flujo de instrucciones en curso. **Interrupciones.** Se generan por una señal del hardware y puede ocurrir en momentos impredecibles durante la ejecución de un programa. Por otra parte las **Excepciones**. Se generan por una señal del software y es provocada por la ejecución de una instrucción.

## References

[1] W. Stallings. *Organización y arquitectura de computadores.* Fuera de colección Out of series. Pearson Educación, 2006.