Universidad
Europea de Madrid

LAUREATE INTERNATIONAL UNIVERSITIES

# UNIDAD DE CONTROL SEGMENTADA

# PLANIFICACIÓN DE PIPES ESTÁTICOS

# Índice

## Presentación

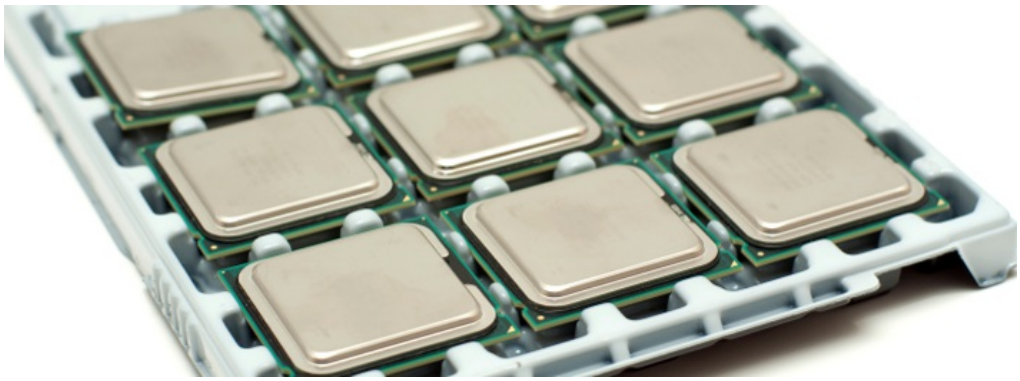
Ya hemos visto que el trabajo en cadena es una forma práctica de paralelizar tareas. Los procesadores segmentados o pipelines paralelizan la ejecución de instrucciones a nivel interno del procesador, separando la ejecución en varias fases o etapas, ejecutando cada una de ellas en un componente hardware independiente.

En este tema aprenderemos que existen unidades de control segmentadas más complejas, en las que la ejecución de una tarea no es siempre secuencial y para completar una tarea es necesaria una retroalimentación a fases anteriores hasta completar la ejecución de una tarea.

Los arquitectos de computadores diseñan unidades de control de control segmentadas, pero también deben decidir cuál es la forma más óptima de trabajar de esa unidad segmentada.

En este tema profundizaremos en la planificación de pipelines estáticos y aprenderemos a aplicar algoritmos que nos permitan decidir cómo programar un pipeline.

La programación de una unidad de control consiste en definir cada cuánto tiempo debemos introducir nuevos datos en la cadena de ejecución, para maximizar el rendimiento del computador.



## Tabla de reservas

Para representar el flujo de ejecución de una tarea compleja en un pipeline unifunción estático, utilizaremos una tabla de reservas (TR).

En una tabla de reservas se representa la ejecución de una sola tarea, indicando qué fases están en ejecución en cada ciclo de reloj y representándose tantos ciclos de reloj como sean necesarios para completar una tarea.

En la siguiente tabla se muestra una tabla de reservas genérica:

- En las filas se representan las fases o etapas del pipeline.
- En las columnas se representa la línea de tiempo medido en ciclos de reloj.
- Cuando una fase está en ejecución en un ciclo concreto, se marca en la celda correspondiente con una X.

| | C0 | C1 | C2 | ... | ... | Cm |
|---|---|---|---|---|---|---|
| Fase 1 | X | | X | | | |
| Fase 2 | | X | | | | |
| ... | | | | | | |
| Fase n | | | | | | X |

Todas las tareas que ejecute esta unidad segmentada deben seguir el mismo esquema (pipeline unifunción) y no puede haber más de una X en una casilla (pipeline estático), es decir, un componente hardware por sí solo no puede estar realizando dos tareas a la vez con distintos datos.

## Pipeline lineal

El caso más sencillo de unidad segmentada es un pipeline secuencial donde, para ejecutar una instrucción, las tareas se ejecutan en orden desde la primera fase hasta la última.

- Representación gráfica de un pipeline secuencial:

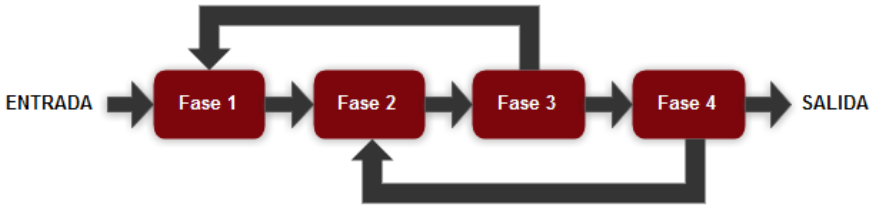

- Tabla de reservas de ejecución secuencial en un pipeline de 4 fases:

| | C0 | C1 | C2 | C3 |
|---|---|---|---|---|
| Fase 1 | X | | | |
| Fase 2 | | X | | |
| Fase 3 | | | X | |
| Fase 4 | | | | X |

En cada ciclo se ejecuta una sola fase. Al primer ciclo se le denomina ciclo cero (C0) y a los siguientes se les va nombrando en orden ascendente: C1, C2, C3,...

## Retroalimentación

A continuación, puedes encontrar una representación gráfica de un pipeline de 4 fases con retroalimentación:




En este pipeline es posible que, al llegar a la fase 3 o a la fase 4, sea necesario volver a pasar una fase anterior para completar la tarea. Incluso es posible que, al completar una fase, la salida sea enviada a dos fases distintas para que operen con los datos en paralelo. Por ejemplo, desde la fase 3 se puede enviar la salida a las fases 1 y 4.

Estas alternativas hacen que las tablas de reservas que se pueden generar a partir de un pipeline sean infinitas, pero hay que tener en cuenta que el pipeline se diseña para un fin específico (unifunción) y eso implica que el orden que seguirán los datos será único para todas las tareas que se ejecuten en ese pipeline.

Seguidamente, encontrarás un ejemplo de tabla de reservas asociada al pipeline de más arriba. Esta tabla de reservas representa la ejecución de una tarea genérica y todas las tareas ejecutadas en este pipeline seguirán este patrón. Cabe destacar que en el ciclo C3, hay dos fases ejecutando datos en paralelo en las fases 1 y 4.

En detalle Tabla de reservas

La tabla de reservas siempre debe respetar el flujo de datos, comenzando por la fase 1 (entrada) y terminando en la fase 4 (salida).

**Tabla de reservas**

| | C0 | C1 | C2 | C3 | C4 | C5 | C6 |
|---|---|---|---|---|---|---|---|
| Fase 1 | X | | | X | | | |
| Fase 2 | | X | | | X | | |
| Fase 3 | | | X | | | X | |
| Fase 4 | | | | X | | | X |

## Definiciones

Para poder afrontar la planificación de pipelines, es necesario conocer algunos conceptos relacionados con unidades segmentadas y medidas de rendimiento de una unidad segmentada.

En la tabla dinámica de más abajo se muestran algunos conceptos.

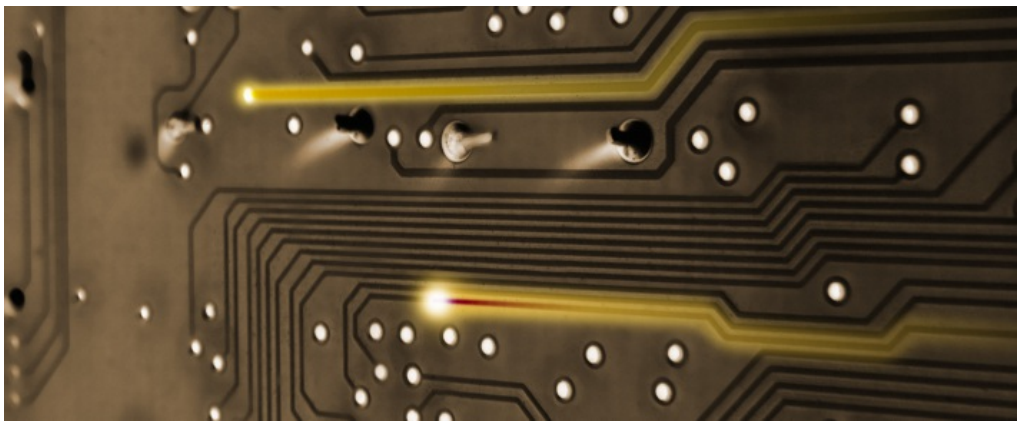
| | |
|---|---|
| Iniciación | Introducción de datos en el pipeline para que este ejecute la tarea para la que fue programada con esos datos de entrada |
| Colisión | Se produce cuando dos tareas intentan utilizar la misma fase (el mismo recurso hardware) para la realización de dos operaciones distintas.<br><br>Es decir, cuando se intenta incluir dos marcas (X) en la misma casilla de la tabla de reservas de un pipeline estático. |
| Tiempo de cálculo | Ciclos que se necesitan para completar una tarea. En la TR, el número de columnas coincide con el tiempo de cálculo (contando el ciclo C0). |
| Latencia | Ciclos que hay que esperar desde que se inicia una tarea hasta iniciar otra tarea (tiempo entre iniciaciones). |
| Latencia prohibida | Es aquella latencia que provoca colisión (desde la última iniciación). |

## Definiciones II

En la siguiente tabla dinámica puedes encontrar la definición de algunos conceptos más.

| | |
|---|---|
| Velocidad de iniciación | Acrónimo: V<br><br>Definición: Número de iniciaciones por ciclo de reloj<br><br>Fórmula: V = Número de Iniciaciones / Número total de ciclos |
| Latencia media | Acrónimo: LAT<br><br>Definición: Inverso de V. Número medio de ciclos entre dos iniciaciones.<br><br>Fórmula: LAT = 1 / V → V = 1 /LAT |

| | |
|---|---|
| Ocupación media de una etapa | Acrónimo: O (Ei): ocupación de la etapa i.<br><br>Definición: Ocupación media (en porcentaje) que tiene una fase. El rendimiento de un pipeline es óptimo cuando alguna de sus etapas trabaja al 100%. O(Ei) = 100%<br><br>Fórmula: O(Ei) = V x NM (Ei) = NM(Ei) / LAT, donde NM (Ei) es igual al número de marcas de la etapa i en la TR |



## Planificación de pipelines

El objetivo a la hora de programar o planificar un pipeline, es maximizar la ocupación media de las fases o etapas. Para ello, trataremos de minimizar la latencia media (LAT) que transcurre entre cada dos iniciaciones sin generar colisiones con otras tareas en marcha.

La planificación de un pipeline se representa mediante una **secuencia de latencias** que define cada cuántos ciclos se produce una nueva iniciación, tratando de minimizar la LAT sin que se produzcan colisiones.

Una secuencia se representa así:

S = <L1, L2, ..., Ln>

### Ejemplos

| | |
|---|---|
| S1 = <2, 5, 3, 4, 2, ...> | Una vez cargada la primera tarea, la segunda iniciación se produce 2 ciclos después; la tercera, 3 ciclos después; la cuarta, 4 ciclos después, etc. |
| S2 = <1, 3, 1, 3, 1, ...> | Una vez cargada la primera tarea, la segunda iniciación se produce 1 ciclo después; la tercera, 3 ciclos después; la cuarta, 1 ciclo después, etc. |

Si una secuencia es cíclica, también se puede representar mediante un **ciclo de latencias** C = (L1, L2).

| | |
|---|---|
| C1 = (1, 3) | La segunda iniciación se produce 1 ciclo después; la tercera, 3 ciclos después; la cuarta, 1 después,... |

| | |
|---|---|
| C2 = (3) | Todas las iniciaciones se producen cada 3 ciclos. Es un *ciclo constante*. |

## Rendimiento del pipe

Para evaluar el rendimiento de una planificación de un pipeline nos basaremos en la latencia media (LAT).

### ¿Cómo calcular la latencia media (LAT) de una planificación?

Nos basaremos en la premisa de que todas las planificaciones al final terminan inmersas en un ciclo de latencias C que se repite continuamente. La LAT de un ciclo C se calcula como:

> LAT (Ci) = Periodo (Ci) / Número de latencias que componen el ciclo Ci

Donde periodo (Ci) = SUMA (Latencias que componen el ciclo Ci).

### Ejemplos

| | |
|---|---|
| C1 = (1, 3) | LAT(C1) = (1+3) / 2 = 2 |
| C2 = (3) | LAT(C2) = 3 / 1 = 3 |
| C3 = (2, 4, 4) | LAT(C3) = 10 / 3 = 3,33 |

### ¿Cómo calcular una secuencia o ciclo de latencias para analizar su LAT?

Para eso construiremos un diagrama de estados que represente en el tiempo todas las posibles latencias que no generen colisión, y todas las posibles combinaciones de secuencias (o al menos las más factibles de ser óptimas).

### ¿Cómo construir el diagrama de estados?

Para esta tarea contaremos con un sencillo algoritmo que nos ayude a construir el diagrama en unos pocos pasos. Pero antes, definiremos que es un vector de colisión, elemento imprescindible para construir todo el diagrama.

## Vector de colisión

Antes de enunciar el algoritmo para la generación de diagramas de estado, es necesario aprender qué es el **vector de colisión (VC)** y cómo construirlo.

El vector de colisión representa, en un instante dado de la ejecución del pipeline, cuáles son las latencias permitidas y prohibidas partiendo del ciclo en el que nos encontramos.

El vector de colisión es dinámico, es decir, evoluciona en el tiempo cada vez que transcurre un ciclo. En cada ciclo se analiza si es posible añadir nuevos datos al pipe (iniciación) y, si es posible, se presentan dos alternativas: añadir una nueva iniciación o esperar otro ciclo.

Un vector de colisión se representa de la siguiente forma:

$$VC = (V0, V1, ..., Vd-1),$$

Donde:

- V0 = 1 → La Latencia 0 siempre es prohibida en un pipe estático.
- Vi ∈ {0,1} →
  - Si Vi = 0: latencia i permitida.
  - Si Vi = 1: latencia i prohibida.
- d = Tiempo de Cálculo = Nº de columnas en la TR.

El vector de colisión inicial (VCini) se construye analizando directamente la TR por desplazamiento:

La latencia i es permitida si al desplazar las marcas X de la TR i posiciones a la derecha, no se produce ninguna colisión (coincidencia)

La latencia i es prohibida si al desplazar las marcas X de la TR i posiciones a la derecha, se produce alguna colisión.


Construcción del VCini de la tabla de reservas de la sección retroalimentación

**Tabla de reservas**

| | C0 | C1 | C2 | C3 | C4 | C5 | C6 |
|---|---|---|---|---|---|---|---|
| Fase 1 | X | | | X | | | |
| Fase 2 | | X | | | X | | |
| Fase 3 | | | X | | | X | |
| Fase 4 | | | | X | | | X |

6 columnas en

ncias

ibidas: {0, 3}

- Latencias permitidas: {1, 2, 4, 5, 6}
- VCini = (1,0,0,1,0,0)

## Generación del diagrama de estados

Para construir el diagrama de estados, utilizaremos el algoritmo de más abajo. En él:

- Los estados se etiquetarán con el VC correspondiente para ese estado (todos los estados se forman con un vector de *d* componentes).
- Las transiciones implican iniciaciones, y las etiquetas representan la latencia con la que transitamos a un nuevo estado.

### Algoritmo de generación del diagrama de estados

1. Estado inicial = VCini.

2. Para todo estado no procesado, VC(t), y para todo k, tal que el k-ésimo componente de VC = 0:

    a. U = VC(t) desplazado k bits a la izquierda.

    b. Completar U con k ceros a la derecha.

c. U = U OR $VC_{ini}$.

d. U: nuevo estado [VC(t+1)]. Unir VC(t) con VC(t+1) por un arco etiquetado con k.

c. Repetir el paso 2 hasta que no haya estados no procesados.

3. Conectar cada estado con el inicial por un arco etiquetado ≥ d.

Desde el siguiente enlace, puedes acceder a un documento en el que se realiza la construcción del diagrama de estados para el ejemplo anterior, seleccionando la mejor planificación para este pipeline, y calculando la ocupación media de cada etapa para el caso más óptimo.


Ejemplo de planificación de pipelines

## Resumen

La ejecución de tareas que se repiten continuamente en un computador se puede optimizar mediante la construcción de complejas unidades segmentadas. Estos pipelines pueden incluir retroalimentación a fases anteriores ya ejecutadas, lo que hace la ejecución consecutiva de tareas no siempre sea posible.

Es importante saber programar una unidad segmentada, indicando cada cuántos ciclos se puede iniciar una nueva entrada de datos al pipe. Una secuencia de latencias entre cada dos iniciaciones consecutivas define una planificación del pipeline.

En este tema hemos aprendido:

- A interpretar la tabla de reservas de un pipeline.
- A seleccionar la mejor planificación de un pipeline, escogiendo la menor latencia media entre iniciaciones.
- A medir el rendimiento de una configuración de una unidad segmentada mediante el cálculo de la ocupación media de cada etapa.