

Tema 1

Introduccion a la Informática Industrial

Index

1.1. Introduction

1.2. Design of Digital Systems

1.3. Embedded Systems

1.4. The Microprocessor

1.5 Internal Architecture of a microprocessor

1.5.1 Evolution of the microporcessor architecture.

1.6 Instructions.

1.7. Input / Output

1.8. Microprocessor Evolution

1.9. Developing Tools

1.1. Introducción

◆ **En la Informática Industrial, nos interesan los dispositivos programables.**

- ◆ El comportamiento del circuito viene gobernado por un programa.

◆ **Existen varios tipos de circuitos programables:**

- ◆ Microprocesador
- ◆ Microcontrolador
- ◆ Procesador Digital de Señal (DSP)
- ◆ Circuitos de Lógica Programable
- ◆ Controladores de Logica Programable

◆ **En esta asignatura nos interesan los dos primeros**

1.1. Introducción

◆ Dispositivos Programables

- ◆ El comportamiento de un chip esta gobernado por un programa
- ◆ El mismo chip puede realizar distintas acciones en funcion del programa que este ejecutando.

◆ Un programa es...

- ◆ Un conjunto secuencia de instrucciones que describen, definen o caracterizan la ejecución de una tarea en un microprocesador
- ◆ Las instrucciones del programa se ejecutan secuencialmente.

1.1. Introducción

◆ Un microcontrolador (μ C) es un chip que incluye:

- ◆ Un procesador (μ P, CPU)...
- ◆ ...y mas cosas en el mismo chip:
 - ◆ diferentes tipos y cantidades de memoria
 - ◆ varios interfaces de entrada/salida y perifericos
- ◆ ... Todos ellos conectado a traves de buses unidireccionales o bidireccionales

◆ La principal diferencia entre un μ P y un μ C:

- ◆ El μ C puede incluir en un único chip todos los elementos que se necesita en un sistema digital, consiguiendo así mas integración a un precio menor.

1.2. Diseño con microcontrolador

◆ Diseño con microcontrolador. Varios pasos

- ♦ Primero, debemos conocer el problema a resolver.
- ♦ Existen mucha variedad de chips microcontrolador en el mercado. Debemos elegir el mas adecuado para resolver nuestro problema, teniendo en cuenta lo solicitado:
 - ♦ Microcontroladores con un periférico en particular
 - ♦ Microcontrolador con suficiente capacidad de almacenamiento
 - ♦ Microcontrolador con dimensiones adecuadas, velocidad adecuada,...
- ♦ A continuación, se diseña el PCB con las conexiones adecuadas a otro sistemas.
- ♦ El siguiente paso es el diseño del programa que gobierna el microcontrolador.
 - ♦ Teniendo en cuenta los requisitos solicitados
- ♦ Se continua con la simulación
 - ♦ Se testea que el programa creado resuelva el problema según las especificaciones
- ♦ **Finalmente, se integra el programa y el hardware en un prototipo, y se testea.**

1.3. Sistemas Empotrados

- ◆ **Es habitual construirlos usando dispositivos programables**
 - ◆ microcontroladores, DSPs
- ◆ **Suelen encontrarse “empotrados” en otros sistemas**
- ◆ **Se dice que son sistemas reactivos de tiempo real:**
 - ◆ Reactivos → Reaccionan a eventos externos
 - ◆ Se encuentran continuamente en ejecución, y cuando llega un evento reaccionan a él
 - ◆ Tiempo Real → Su respuesta debe producirse en un plazo limitado
- ◆ **Realizan varias tareas de forma concurrente**

1.4. Instrucciones

- ◆ **Dispositivo Programable. Gobernado por un programa.**
- ◆ **Trabaja realizando un conjunto de pasos individuales**
 - ◆ Cada paso se llama instrucción
 - ◆ Cada elemento de informacion que se maneja se llama dato
 - ◆ Programa: es un conjunto estructurado de instrucciones

Si a un dispositivo programable se le cambia el programa, su funcionalidad cambia

1.4. Instrucciones

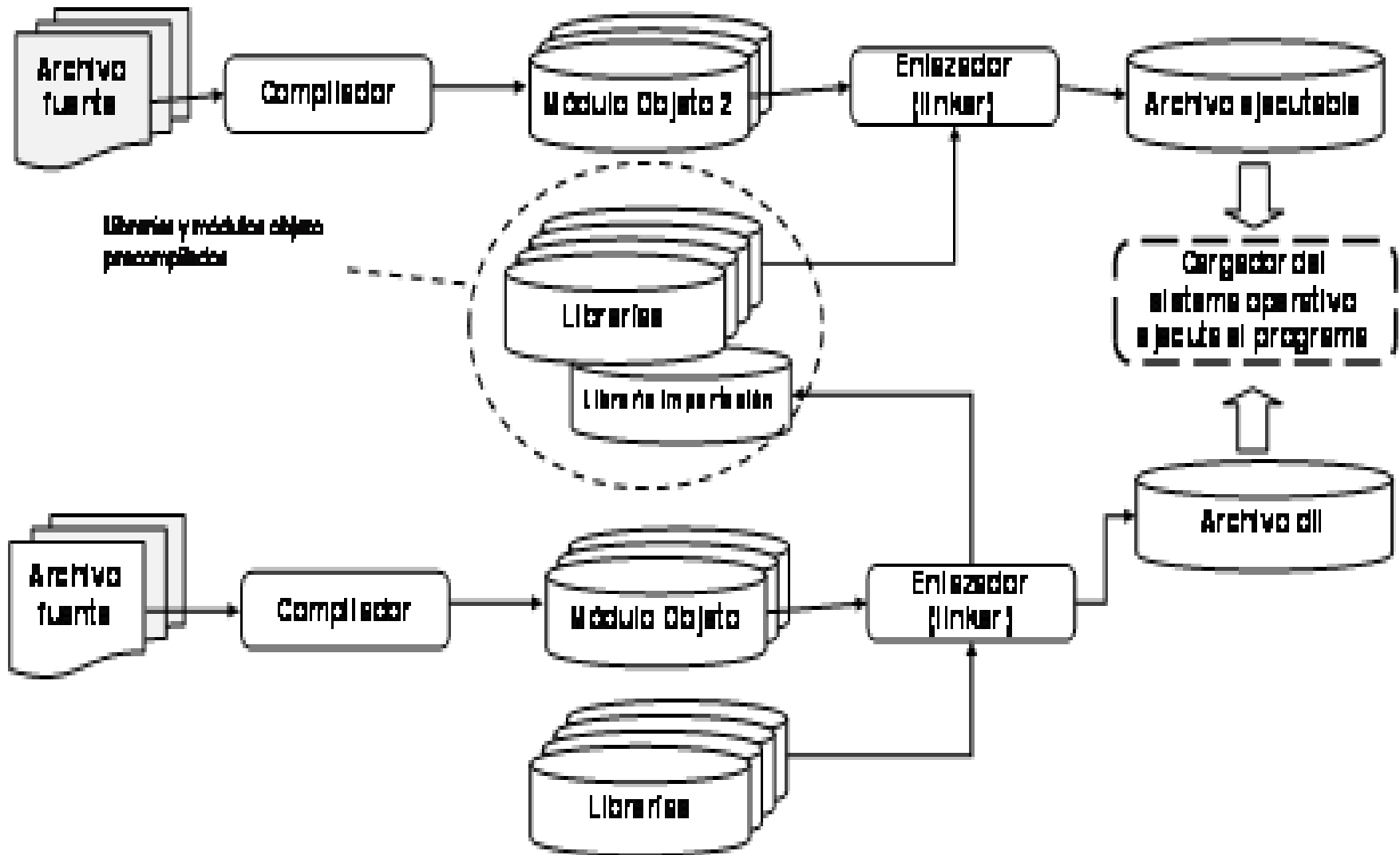
◆ ¿Que es un programa?

- ◆ Es una secuencia de instrucciones que describen, definen o caracterizan la ejecución de una tarea en un microprocesador

◆ ¿Qué es una instrucción?

- ◆ Es el comando elemental de la CPU

1.4 Instrucciones



1.4 Instrucciones.

◆ Codificando instrucciones

- ◆ Las instrucciones se almacenan en la memoria de programa.
- ◆ Para el uP tienen formato binario(conjunto de '1' y '0' → código máquina)

◆ ¿Cómo se generan los códigos binarios?

- ◆ Mediante un ensamblador, si se parte de lenguaje ensamblador
 - ◆ Lenguaje ensamblador: conjunto de mnemotécnicos asociados a las instrucciones y a los recursos de la máquina.
- ◆ Mediante un compilador, partiendo de un lenguaje de alto nivel
 - ◆ Genera código máquina desde un lenguaje de alto nivel (por ejemplo, C)

1.4 Instrucciones

◆ Lenguaje ensamblador.

- ◆ Cada procesador define su propio lenguaje.
 - ◆ Operaciones: Sólo las que permite el procesador.
 - ◆ ADD, MOV,...
 - ◆ Datos: Registros de propósito general o posiciones de memoria
 - ◆ R1,R2,SP,...
 - ◆ Complejo, pero rápido.

1.4 Instrucciones

◆ Formato de instrucción

- ◆ Código de Operación (Opcode) + [Operando Fuente] + [Operando Destino]

1.4. Instrucciones

◆ Código de Operación (Opcode)

◆ Instrucciones de transferencia

◆ Para copiar información:

- ◆ entre registros,
- ◆ entre registros y posiciones de memoria
- ◆ entre posiciones de memoria.

◆ Instrucciones Aritméticas

◆ Realizan operaciones aritméticas simples: sumas, restas, ...

◆ Instrucciones Lógicas

◆ Realizan operaciones lógicas sobre los operandos: AND, OR, XOR, etc..

1.4. Instrucciones

◆ Código de Operación (Opcode)

◆ Instrucciones de manipulación de bits

- ◆ Modifican un único bit de el operando

◆ Instrucciones de desplazamiento

- ◆ Desplazan el contenido de un registro en una dirección

◆ Instrucciones de Control de programa

- ◆ Cambia el flujo de la ejecución del programa (por defecto, secuencial)

1.4 Instrucciones

◆ Acceso a los operandos

- ◆ Dirección efectiva: localización del operando
- ◆ Modos de direccionamiento: Formas diferentes de llegar a la dirección efectiva
 - ◆ Proporcionan mayor potencia al dispositivo
 - ◆ Simplifican la programación, cuando se usa un lenguaje ensamblador.

1.4. Instrucciones

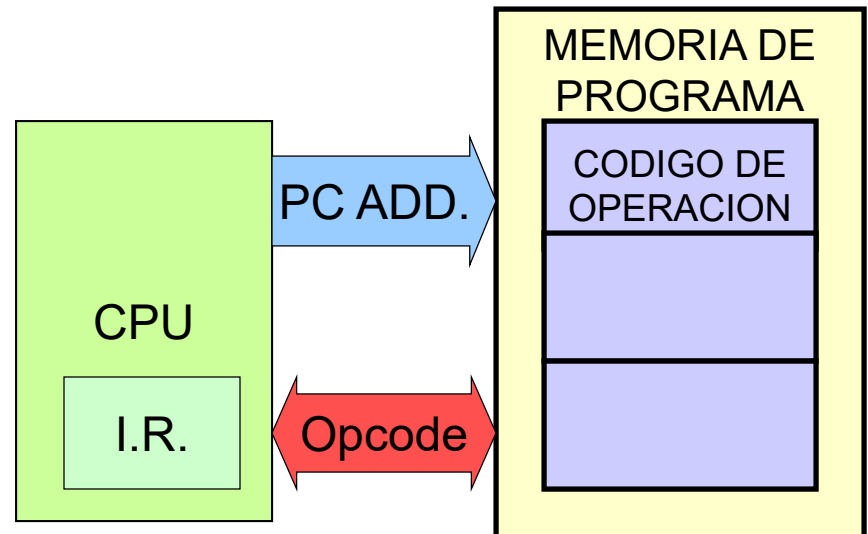
◆ Posibles modos de direccionamiento

- ◆ Implícito
- ◆ Directo a Registro
- ◆ Inmediato
- ◆ Directo a Memoria
- ◆ Indirecto
- ◆ Indirecto con offset
- ◆ Indirecto con Índice
- ◆ Indirecto con pre-indexado
- ◆ Indirecto con post-indexado

1.4. Instrucciones

◆ Fases en la ejecución de una instrucción

- ◆ Se obtiene la instrucción a ejecutar (código de operación).(Fetch)
- ◆ Se calcula la dirección efectiva de los operandos
- ◆ Se buscan los operandos
- ◆ Se ejecuta la operación
- ◆ Se almacena el resultado
- ◆ Se prepara para la próxima instrucción.



1.5. El microprocesador

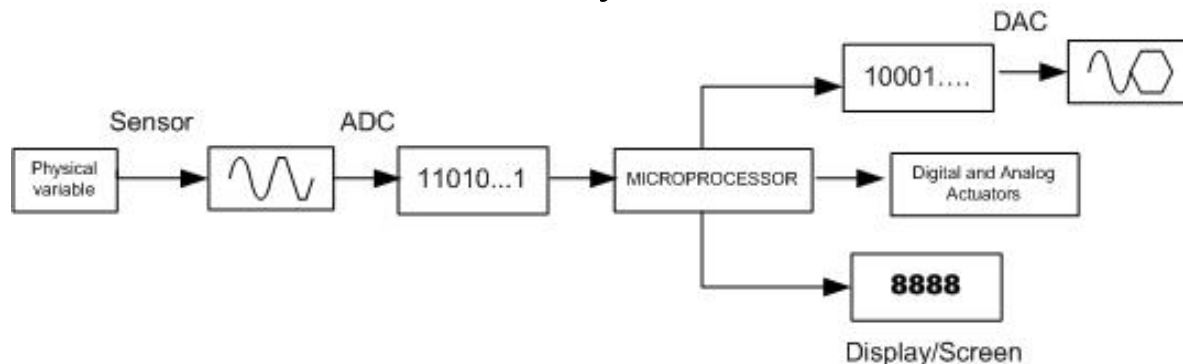
◆ Dispositivo Programable: trabaja realizando un conjunto de pasos individuales

- ◆ Cada paso se llama instrucción
- ◆ Cada elemento de informacion que se maneja se llama dato
- ◆ Programa: es un conjunto estructurado de instrucciones

Si a un dispositivo programable se le cambia el programa, su funcionalidad cambia

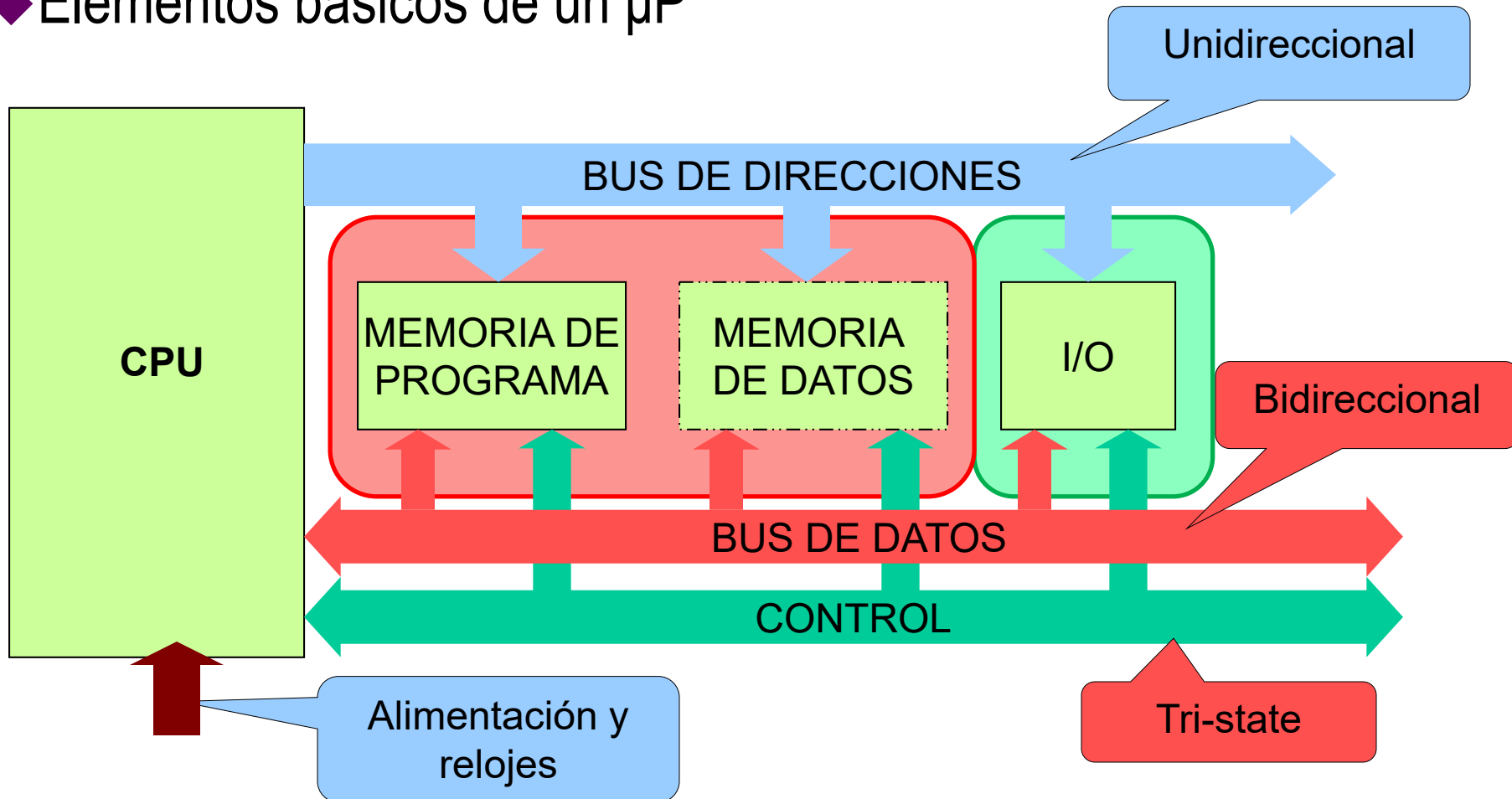
◆ Trabaja con datos digitales.

- ◆ Para que trabaje con señales digitales, deben convertirse éstas mediante conversores ADC y DAC



1.5. El microprocesador

◆ Elementos básicos de un μP



1.5. El Microprocesador

- ◆ El μ P necesita la suficiente memoria para almacenar tanto **instrucciones** como **datos** (variables y constantes) de un programa
 - ◆ Las instrucciones y los datos constantes se almacenan en memoria no volatil (cuyo contenido se mantiene sin alimentación). La memoria de programas solamente puede ser leída o programada (pero no puede ser escrita)
 - ◆ Los datos variables son almacenados en memoria volatil (su contenido se pierde cuando se apaga el sistema). La memoria de datos puede ser leída y escrita

◆ Big endian/little endian

- ◆ Si tenemos que almacenar los datos
0x4A3B2C1D, y a continuación 0x59687786

4A
3B
2C
1D
59
68
77
86
Big Endian (Motorola)

1D
2C
3B
4A
86
77
68
59
Little Endian (Intel)

1.5. El Microprocesador

◆ Tamaño de palabra

- ◆ Es el tamaño en bytes que la CPU considera paa una unidad de datos
 - ◆ 8 bits (Byte)
 - ◆ 16 bits
 - ◆ 32 bits.

◆ Tiempo de ejecución de una instrucción

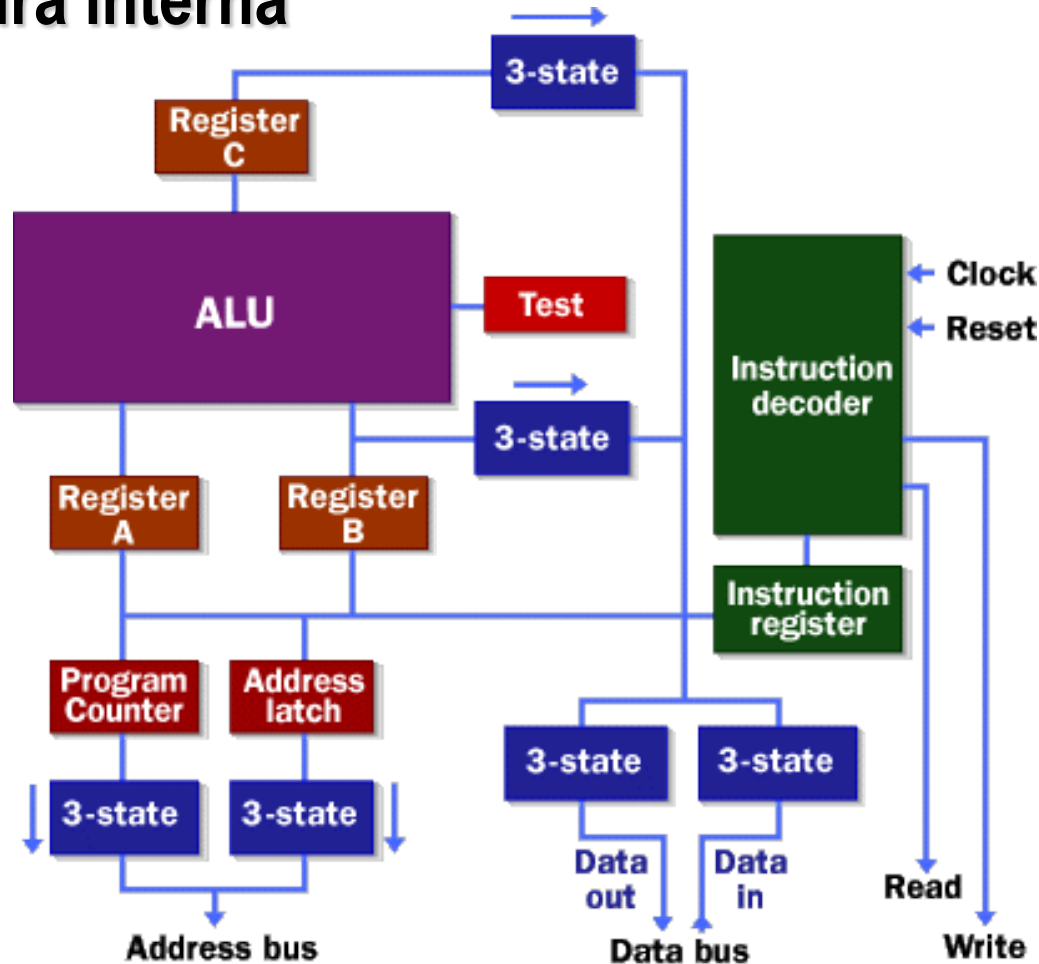
- ◆ Es el tiempo que le lleva al procesador ejecutar una instrucción
 - ◆ Variable.

◆ Fases de ejecución de una instrucción

- ◆ ¿Cómo implementa cada una de las fases?

1.6. Arquitectura Interna de un microprocesador

◆ Arquitectura interna



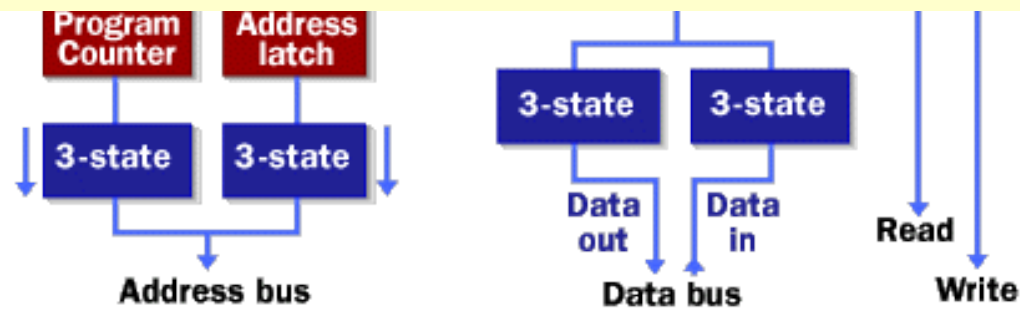
1.6. Arquitectura Interna de un microprocesador

◆ Arquitectura interna



OPERACIONES QUE PUEDE REALIZAR UN uP:

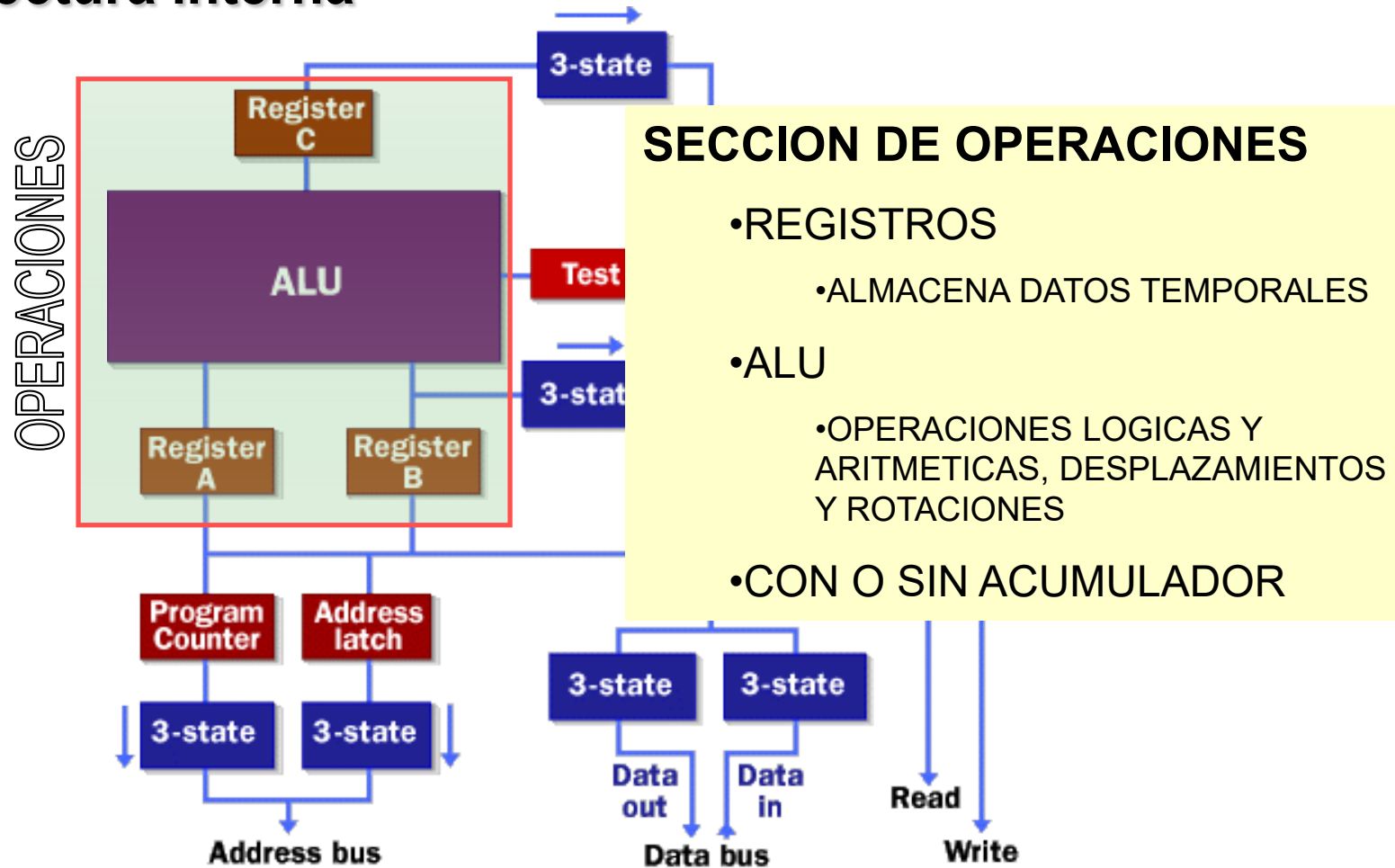
- CARGAR REGISTROS
- LEER O ESCRIBIR EN POSICIONES DE MEMORIA
- REALIZAR OPERACIONES LOGICAS O ARITMÉTICAS
- ROTACIONES Y DESPLAZAMIENTOS



©2000 How Stuff Works

1.6. Arquitectura Interna de un microprocesador

◆ Arquitectura interna

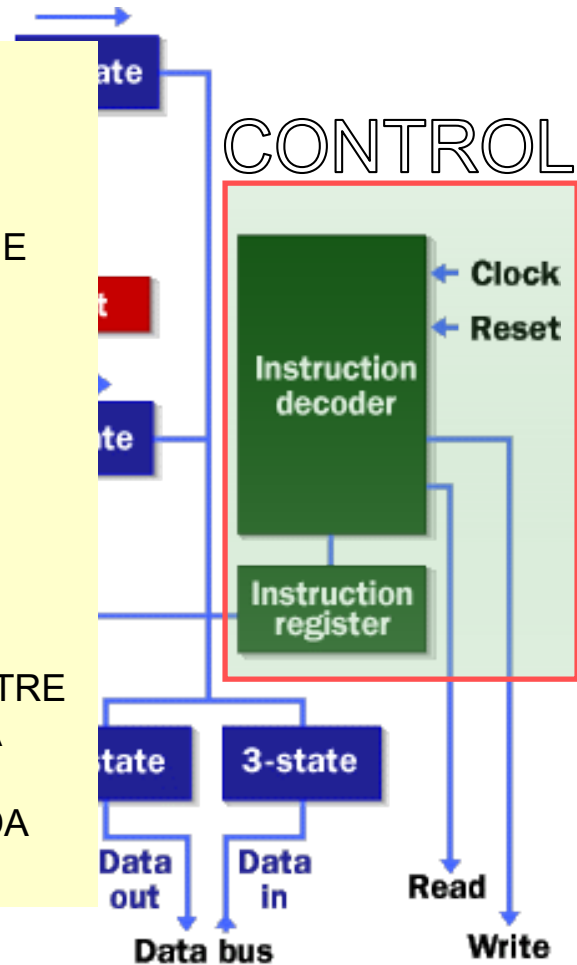


1.6. Arquitectura Interna de un microprocesador

◆ Arquitectura interna

SECCION DE CONTROL

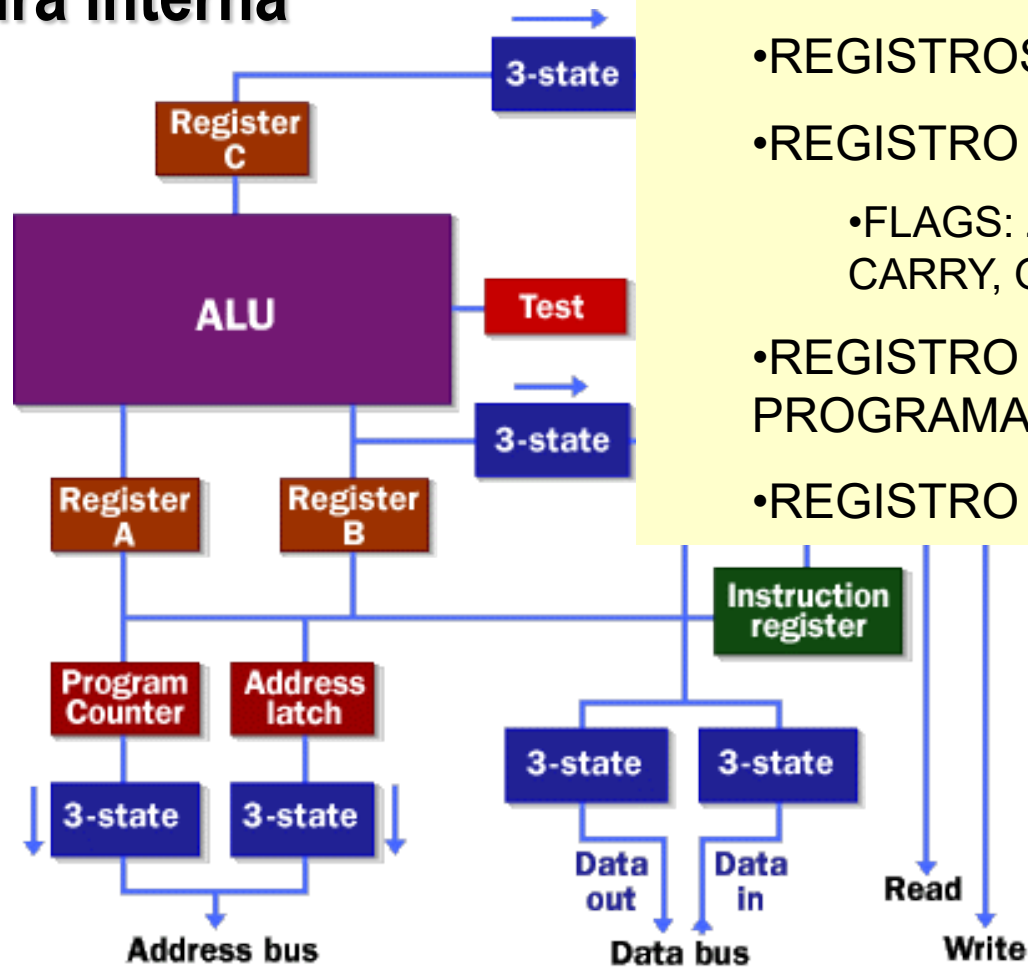
- CARGA INSTRUCCIONES
 - OPCODE. CONJUNTO DE BITS QUE LLEGAN AL REGISTRO DE INSTRUCCIONES
- DECODIFICACION
 - ANALIZA EL OPCODE Y GENERA MICROORDENES.
- GENERA MICROORDENES
 - HABILITA LA TRANSFERENCIA ENTRE ESTOS REGISTROS Y CONFIGURA LAS OPERACIONES DE LA ALU, SEGUN LA OPERACION SOLICITADA EN EL OPCODE.



©2000 How Stuff Works

1.6. Arquitectura Interna de un microprocesador

◆ Arquitectura interna



OTROS REGISTROS

- REGISTROS AUXILIARES
- REGISTRO DE ESTADO
 - FLAGS: ZERO, SIGNO, CARRY, OVERFLOW
- REGISTRO CONTADOR DE PROGRAMA
- REGISTRO DE PILA

1.6. Arquitectura Interna de un microprocesador

◆ Registro de Instrucciones

- ◆ Carga el código de operación desde la memoria de programa.
- ◆ El decodificador genera las micro-ordenes al resto de unidades del procesador para que la operación recibida se pueda ejecutar con éxito

◆ Contador de programa (PC)

- ◆ Contiene la dirección de la memoria de programa donde se encuentra la instrucción que se va a ejecutar a continuación.
- ◆ Tras la ejecución de una instrucción, el Contador de Programa se modifica para apuntar a la siguiente.

1.6. Arquitectura Interna de un microprocesador

◆ Registro de Estado

- ◆ Contiene flags que proporcionan información del resultado de la última operación realizada: signo, zero, carry, overflow.
- ◆ Permite decisiones condicionadas al resultado de la operación previa que se realizó en el microprocesador.

◆ Registro de pila (Stack pointer, SP)

- ◆ Contiene una dirección de la memoria de datos, que se usa para el almacenamiento temporal de datos.

1.6. Arquitectura Interna de un microprocesador

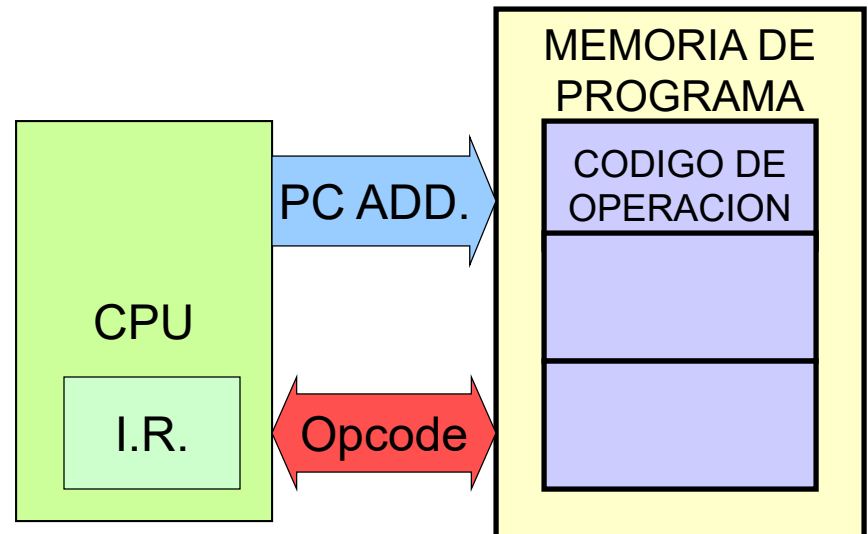
◆ Buses internos

- ◆ Es un bus que permite las comunicaciones entre varios componentes del microprocesador.
- ◆ El numero de lineas de este bus viene determinado por el numero de bits que se pueden procesar en paralelo en la ALU, y determina el tamaño de palabra. Este es uno de los parámetros que permiten clasificar los microprocesadores.
- ◆ Este bus se relaciona con los buses de datos externos al microprocesador a través del buffer del bus de datos.

1.6. Arquitectura Interna de un microprocesador

◆ Fases en la ejecución de una instrucción

- ◆ Se obtiene la instrucción a ejecutar (código de operación).(Fetch)
 - ◆ PC → Bus de direcciones; se obtiene el OP CODE
- ◆ Se calcula la dirección efectiva de los operandos
- ◆ Se buscan los operandos
- ◆ Se ejecuta la operación
- ◆ Se almacena el resultado
- ◆ Se prepara para la próxima instrucción.

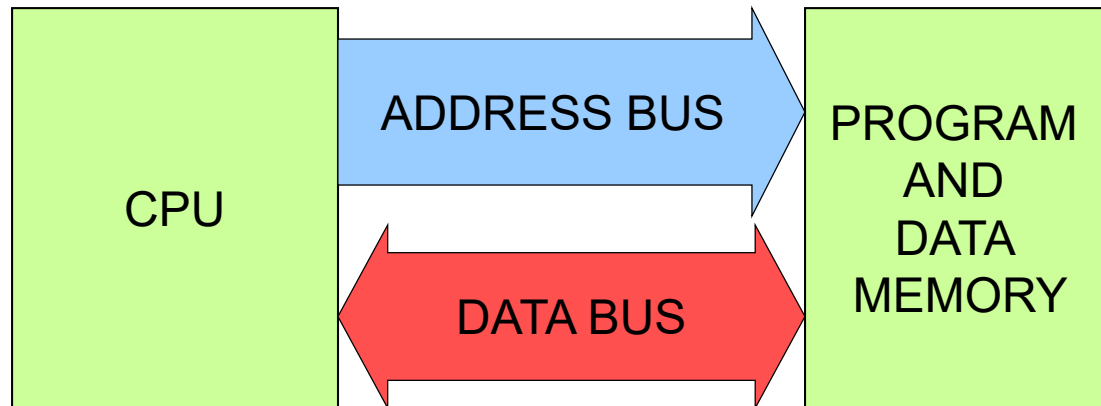


- ◆ Se incrementa el PC

1.6.1. Evolución de arquitectura de los uprocesadores

◆ Arquitectura de Microprocesadores: Von-Neuman

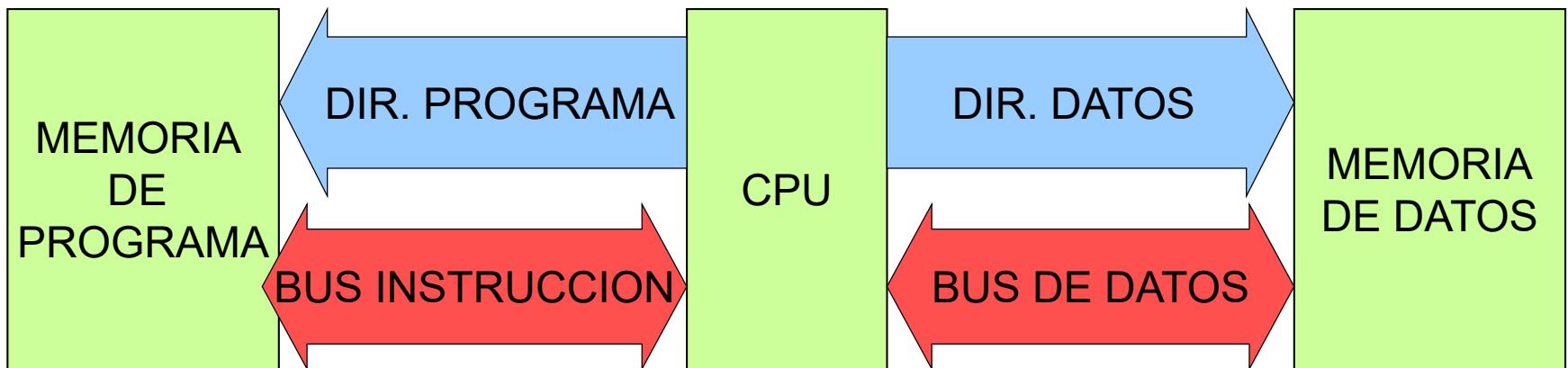
- ◆ Dos buses: Direcciones (Address, **ADD**) y Datos (**D**).
- ◆ Las Instrucciones y los Datos se accede de forma secuencial, a través del mismo bus de datos.



1.6.1. Evolución de arquitectura de los uprocesadores

◆ Arquitectura de Microprocesadores: Harvard

- ◆ Dos buses de direcciones, un **bus de Datos** y un **bus de Instrucciones**.
- ◆ Lee la siguiente instrucción a ejecutar y accede a un dato **al mismo tiempo**.



1.6.1. Evolución de arquitectura de los uprocesadores

◆ Pipelining (Segmentación)

- ◆ Metodología para el diseño electrónico de procesadores basado en el solapamiento en la ejecución de instrucciones diferentes.
- ◆ En cada etapa, se desarrolla una parte de la instrucción.
- ◆ Un procesador con N etapas puede ejecutar N instrucciones al mismo tiempo, incrementando N veces su eficiencia.
- ◆ La eficiencia se incrementa únicamente reorganizando temporalmente la ejecución de las instrucciones, no se modifican/incrementan elementos de la CPU.
- ◆ Por otro lado, el diseño de la Unidad de Control se complica para conseguir esa eficiencia.

1.6.1. Evolución de arquitectura de los uprocesadores

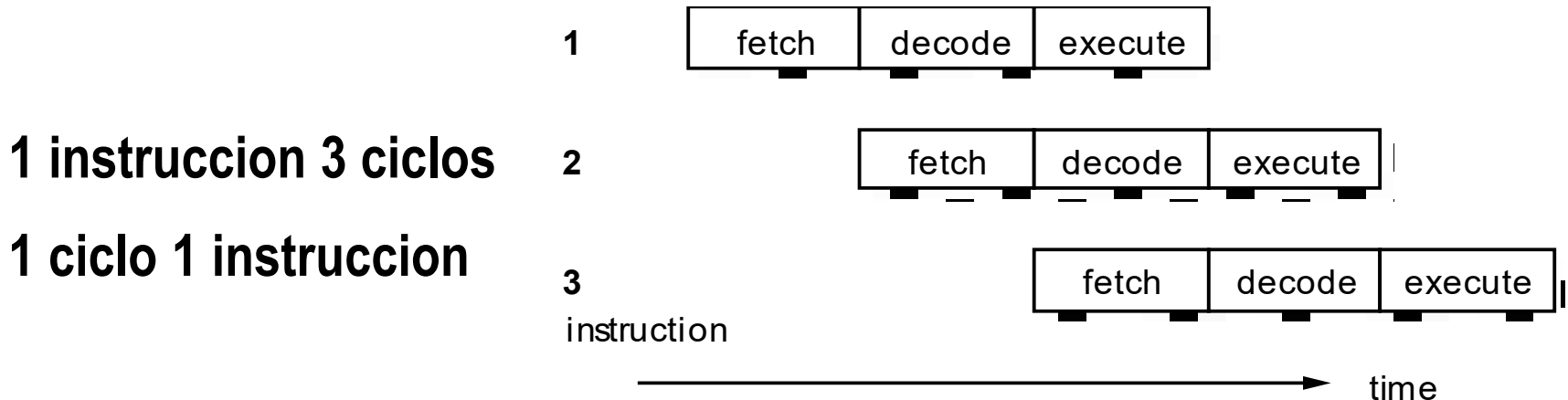
◆ Pipelining (Segmentación)

- ♦ La arquitectura con pipeline más usada en los noventa tenía 5 etapas: **fetch** (IF), **decode** (ID), **operand read** (MEM), **execution** (EX) and **operand write** back (WB).
- ♦ En la actualidad, los procesadores tienen un mayor numero de etapas, como por ejemplo, 20 en un Pentium IV.

1.6.1. Evolución de arquitectura de los uprocesadores

◆ Pipeline de 3 etapas: Riesgos

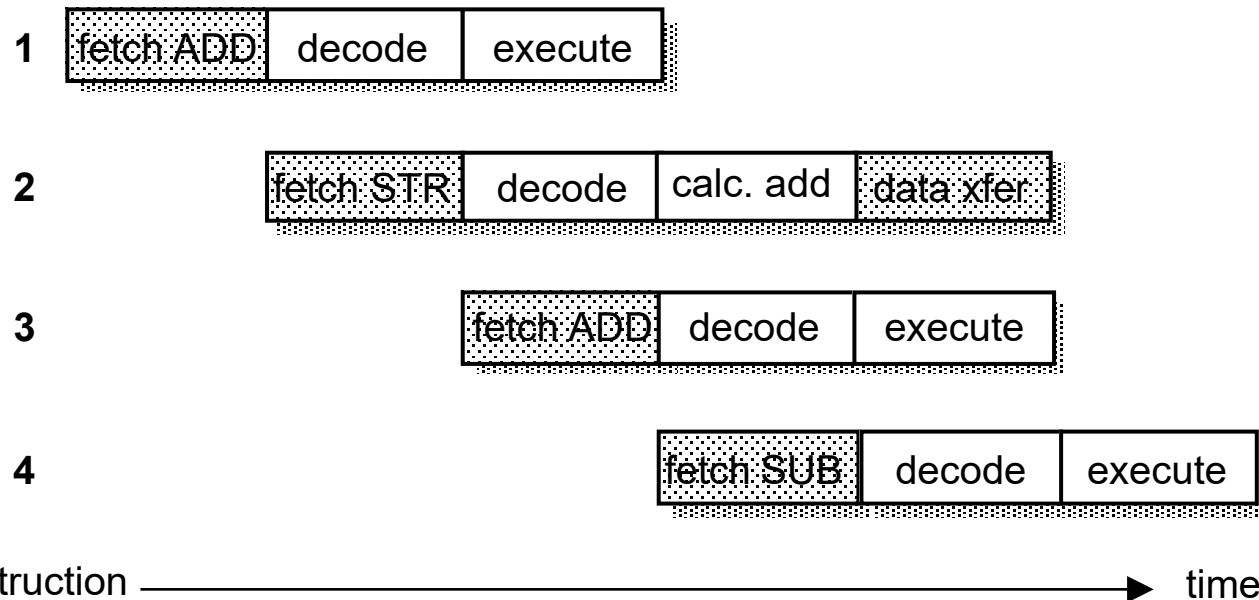
- ◆ **Fetch:** La instrucción es transferida desde la memoria de programa y almacenado en la unidad de control
- ◆ **Decode:** La instrucción es decodificada
- ◆ **Execute:** Resto de operaciones en la ALU, transferencias y almacenamientos



1.6.1. Evolución de arquitectura de los uprocesadores

◆ Riesgo de instrucciones Multi-ciclo

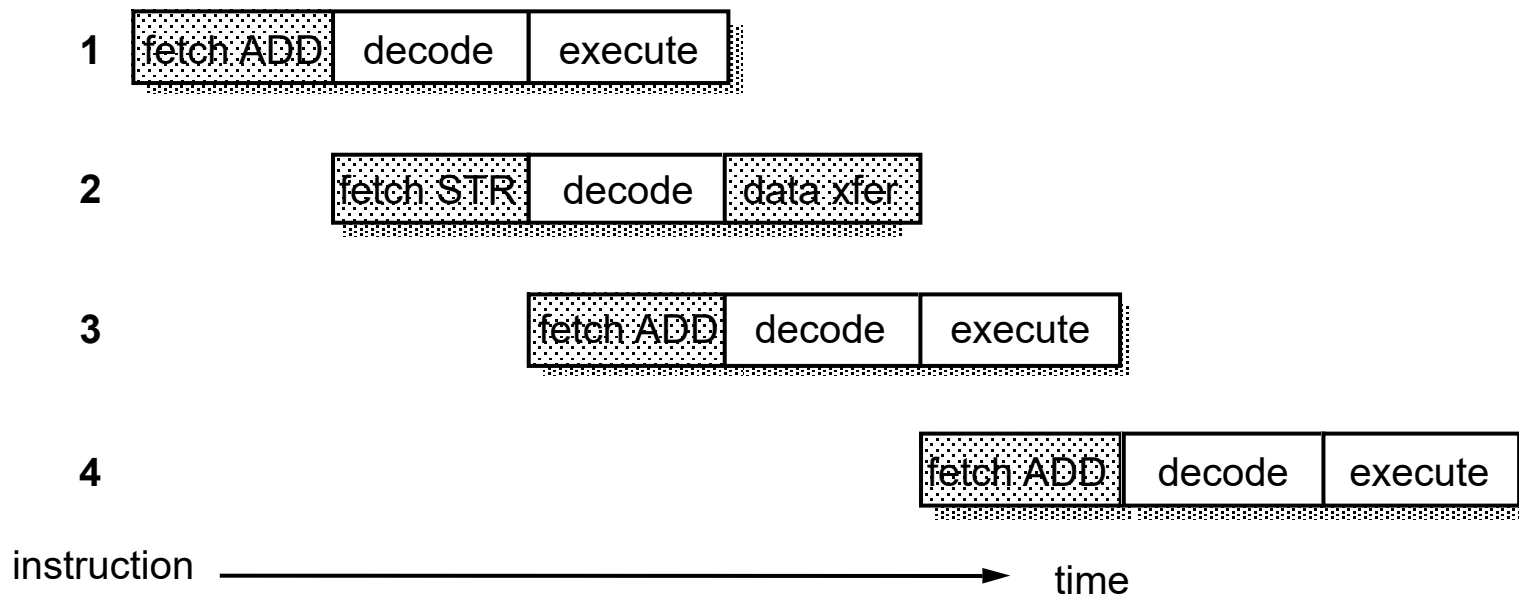
- La instrucción 2 es compleja y necesita mas ciclos que una instrucción estándar. El uP debe ser capaz de manejar estas instrucciones



1.6.1. Evolución de arquitectura de los uprocesadores

◆ Riesgo Estructural

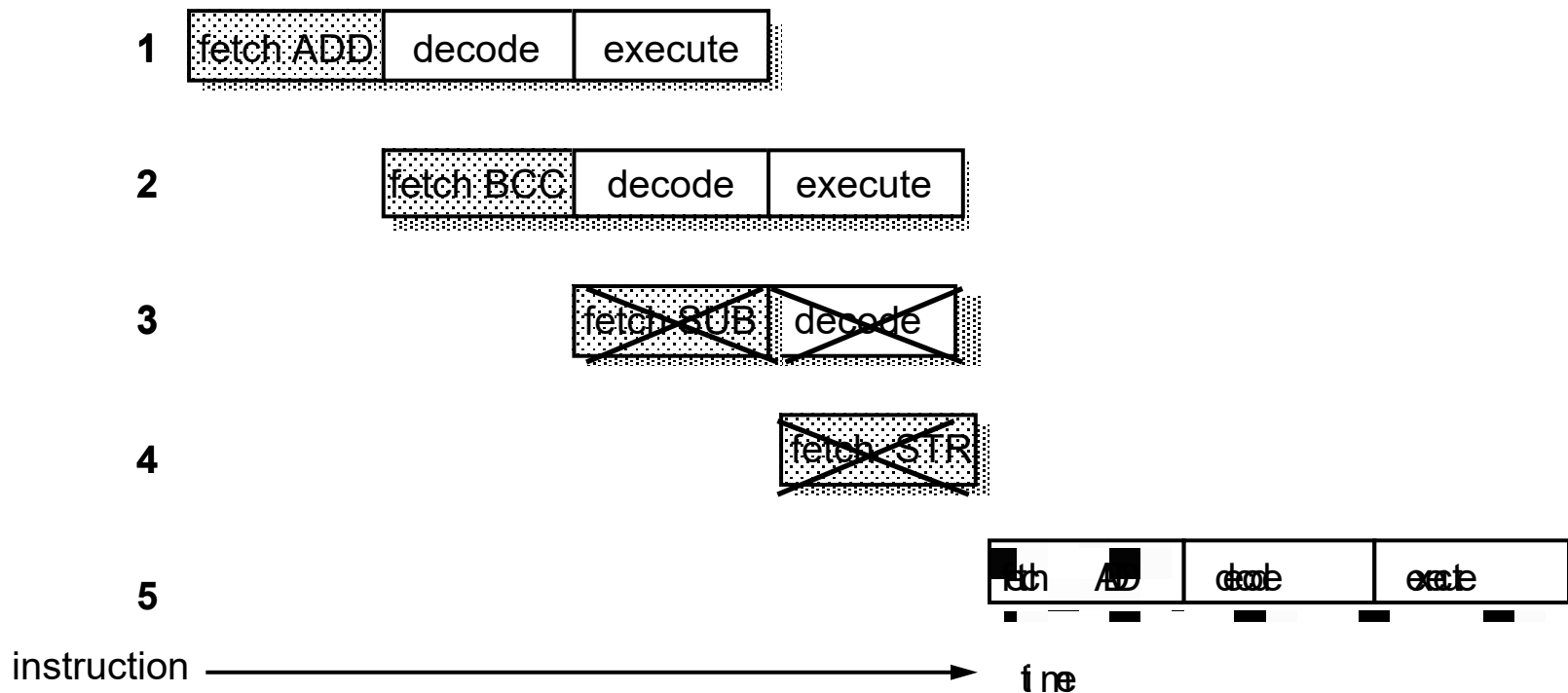
- ♦ En un mismo momento, dos instrucciones utilizan el mismo recurso hardware



1.6.1. Evolución de arquitectura de los uprocesadores

◆ Riesgo de Control

- Si hay un salto condicional en una instrucción en ejecución, es posible que fases de instrucciones posteriores deban ser eliminadas



1.6.1. Evolución de arquitectura de los uprocesadores

◆ Procesadores Super-escalares

- ♦ Metodología de diseño electrónico basado en solapar la ejecución de diferentes instrucciones usando unidades funcionales repetidas en la CPU.
- ♦ Procesadores con un nivel escalar L tienen L unidades funcionales repetidas que pueden ejecutar L instrucciones al mismo tiempo.

