Unidad de Control

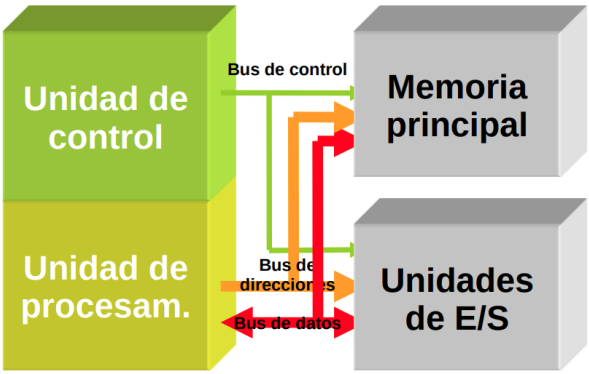
Estructura de Computadores. Tema 3.

Camino de Datos

Introducción

Un computador con arquitectura Von Neumann consta de tres bloques fundamentales:

- CPU o procesador
- Memoria principal (datos + instrucciones)
- Unidades de E/S (y memoria masiva)



Vamos a estudiar la **CPU**, que puede entenderse como una unidad constituida por:

- **Unidad de procesamiento** o camino de datos (*datapath*)
- **Unidad de control**

Unidad de procesamiento

La unidad de procesamiento comprende **elementos hardware** como:

- **Unidades funcionales** (ALU, desplazador, multiplicador,...)
- **Registros**

  registros de uso general (varios GPR); registro de estado contador de programa (PC); puntero de pila (SP); registro de instrucción (IR); registro de dato de memoria (MDR / MBR); registro de dirección de memoria (MAR)
- **Multiplexores**
- **Buses internos**

  ...

Unidad de control

La unidad de control **interpreta y controla la ejecución de instrucciones** leídas de la memoria principal, en dos fases;

1. **Secuenciamiento de las instrucciones**
   a. La UC lee de MP la instrucción apuntada por PC (`IR ← M[PC]`)
   b. Determina la dirección de la instrucción siguiente y carga en PC
2. **Ejecución/interpretación de las instrucciones en IR**
   a. La UC reconoce el tipo de instrucción
   b. Manda las señales necesarias para tomar los operandos necesarios y dirigirlos a las unidades funcionales adecuada de la unidad de proceso
   c. Manda las señales necesarias para realizar la operación
   d. Manda las señales necesarias para enviar los resultados a su destino.

Unidad de procesamiento con un bus

Características

- Componentes interconectados mediante **un bus en común**
- **MDR**: dos entradas, dos salidas
- **MAR**: unidireccional (procesador → memoria)
- **MUX**: selecciona la entrada A de la ALU

    - La constante 4 se usa para incrementar el contador del programa
- La **UC** genera señales internas y externas (memoria)

Transferir de un registro a otro

Cada registro usa dos señales de control:

- **Load**: Carga en paralelo
- **Enable**: Habilitación de salida (buffer triestado)

Ejemplo. Instrucción `R4 ← R1` equivale a `Enable R1, Load R4`

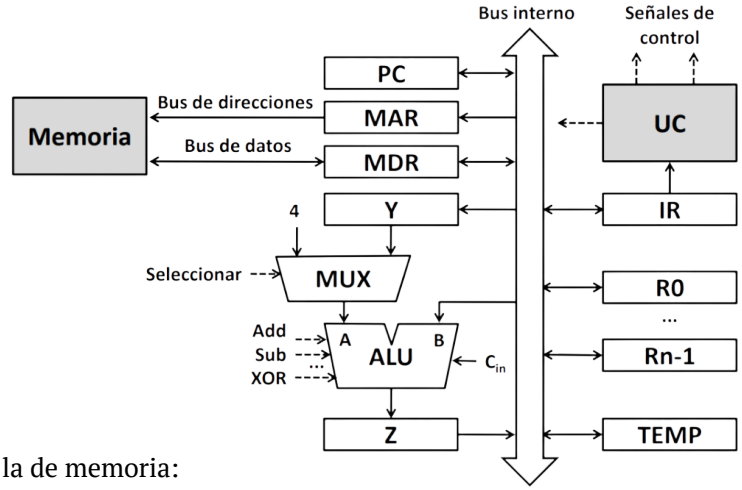
Realizar operación aritmética o lógica

Para ello se usa la la **ALU**, un circuito combinacional sin memoria.

Ejemplo. Instrucción `R3 ← R1 + R2`

1. `Enable R1, Load Y`
2. `Enable R2, Select Y, Add, Load Z`
3. `Enable Z, Load R3`

Cada línea equivale a 1 ciclo de reloj.



Cargar posición de memoria en registro

Para ello se siguen los siguientes pasos:

I. Transferir **dirección a MAR**
II. Activar **lectura de memoria**
III. Almacenar **dato** leído **en MDR**

Por lo tanto la temporización interna debe coordinarse con la de memoria:

- La lectura de memoria puede requerir varios ciclos de reloj.
- El procesador debe esperar la activación de señal de finalización de ciclo de memoria.

Ejemplo. Instrucción `Load (R1) ➡ R2`

1. `Enable R1, Load MAR`
2. `Comenzar lectura`
3. `Esperar fin de ciclo de memoria, Load MDR desde memoria`
4. `Enable MDR hacia bus interno, Load R2`

Almacenar registro en posición de memoria

Pasos

I. Transferir dirección a MAR
II. Transferir dato a escribir a MDR
III. Activar escritura de memoria

La temporización interna debe coordinarse con la de la memoria

- La escritura puede requerir varios ciclos de reloj.
- El procesador debe esperar la activación de señal de finalización de ciclo de memoria.

Ejemplo. Instrucción `Store R2 ➡ (R1)`

1. `Enable R1, Load MAR`
2. `Enable R2, Load MDR desde bus interno`
3. `Comenzar escritura`
4. `Esperar fin de ciclo de memoria`

Ejemplos de instrucciones

Ejecución de una instrucción completa. `Add (R3) R1`

captación:
1. `Enable PC, Load MAR, Select 4, Sumar, Enable Z`
2. `Comenzar lectura, Enable Z, Load PC, Load Y`
3. `Esperar fin de ciclo de memoria, Load MDR desde mem.`
4. `Enable MDR hacia bus interno, Load IR`
5. `Decodificar instrucción`

ejecución:
6. `Enable R3, Load MAR`
7. `Comenzar lectura, Enable R1, Load Y`
8. `Esperar fin de ciclo de memoria, Load MDR desde mem.`
9. `Enable MDR hacia bus interno, Select Y, Sumar, Load Z`
10. `Enable Z, Load R1, Saltar a captación`

Ejecución de una instrucción de salto. `Jmp desplazamiento`

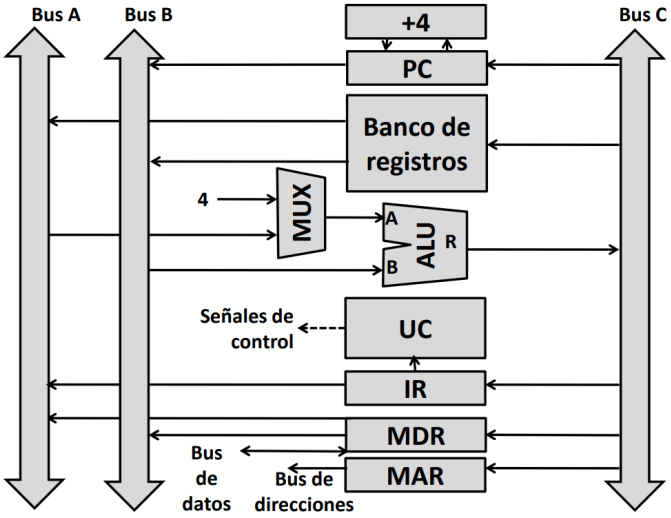
captación:
1. `Enable PC, Load MAR, Select 4, Sumar, Enable Z`
2. `Comenzar lectura, Enable Z, Load PC, Load Y`
3. `Esperar fin de ciclo de memoria, Load MDR desde mem.`
4. `Enable MDR hacia bus interno, Load IR`
5. `Decodificar instrucción`

ejecución:
6. `Enable Campo desplazamiento en IR, Sumar, Load Z`
7. `Enable Z, Load PC, Saltar a captación`

Unidad de procesamiento con buses múltiples

Características

- **Banco de registros** con tres puertos:
  - Dos registros pueden poner sus contenidos en los buses A y B. *(en el mismo ciclo)*
  - Un dato del bus C puede cargarse en un registro.
- **ALU.** Sin registros auxiliares, directamente con los buses.
- **Unidad de incremento** (+4).



Ejecución de una instrucción completa

Instrucción **`R6 ← R4 + R5`**

1. `Enable PC, R=B, Load MAR`
2. `Comenzar lectura, Incrementar PC`
3. `Esperar fin de ciclo de memoria, Load MDR desde mem.`
4. `Enable MDR hacia B, R=B, Load IR`
5. `Decodificar instrucción`
6. `Enable R4 hacia A, Enable R5 hacia B, Select A, Sumar, Load R6, Saltar a captación`

Unidades de Control Cableadas y Microprogramadas

Unidad de Control

Señales de entrada a la UC:

- Señal de reloj
- Instrucción actual (codop, campos de direccionamiento,...)

- Estado de la unidad de proceso
- Señales externas (por ejemplo interrupciones)

Señales de salida de la UC:

- Señales que gobiernan la unidad de procesamiento:
  - Carga de registros
  - Incremento de registros
  - Desplazamiento de registros
  - Selección de entradas de multiplexores
  - Selección de operaciones de la ALU

  ...
- Señales externas (por ejemplo, lectura/escritura en memoria)

Tipos de Unidades de Control

Control fijo o cableado ("hardwired")

- Se emplean métodos de diseño de circuitos digitales secuenciales a partir de **diagramas de estados**.
- El circuito final se obtiene **conectando componentes básicos** como puertas y biestables (a menudo usan PLA, conjunto de puertas AND y OR que se disponen en forma de matriz para hacerla más sencilla).

Control microprogamado

- Todas las señales que se pueden activar simultáneamente se agrupan para formar **palabras de control**, que se almacenan en una memoria de control (normalmente ROM).
- Una instrucción de lenguaje máquina se transforma sistemáticamente en un programa (microprograma) almacenado en la memoria de control.
  - Mayor facilidad de diseño para instrucciones complejas
  - Método estándar en la mayoría de los CISC.

UC cableada

Se diseña mediante **puertas lógicas y biestables** siguiendo uno de los métodos clásicos de diseño de sistemas digitales secuenciales ya conocidos. Algunas características son:

- El **diseño** es **laborioso** y **difícil de modificar** debido a la complejidad de los circuitos.
- Suele ser **más rápida** que la misma UC microprogramada.
- Se utilizan **PLA** (matrices lógicas programables) para llevar a cabo la implementación.

Organización de una UC cableada basada en PLA

Ventajas

- MUY rápido
- Mayor flexibilidad y fiabilidad
- Ahorro de espacio y potencia

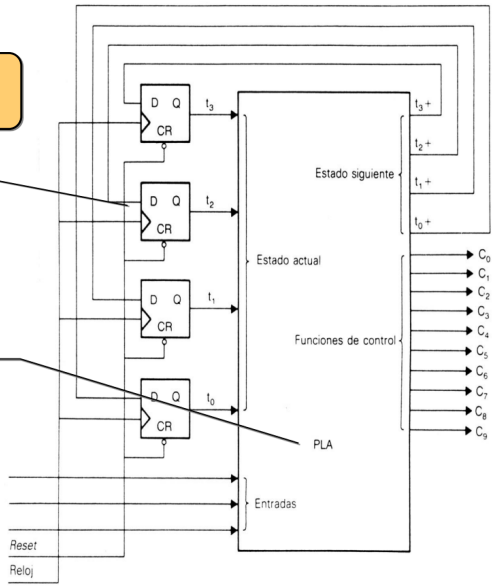
Problema

- Cada cambio implica cambio de cableado

Debido a las modernas técnicas de diseño y a RISC ha tomado nuevo auge la realización de UC cableadas

Los biestables contienen la información relativa al estado en que se encuentra el sistema

La PLA utiliza esta información de estado, junto con las entradas externas, para generar el siguiente estado



Ejemplo

En la implementación de una unidad de control cableada sencilla (ODE) se siguen los siguientes pasos:

1. Definir una máquina de estados finitos

Dado el diagrama de flujo de la UC de ODE, detallamos éste como un conjunto finito de estados y transiciones entre ellos.

2. Describir dicha máquina en el lenguaje de alto nivel

El lenguaje concreto depende del programa que utilicemos para compilar la descripción de la máquina. Estos lenguajes tienen **sentencias para definir**:

- entradas y salidas
- estados y transiciones condicionales e incondicionales

3. Generar la tabla de verdad para la PLA

Según la descripción que hayamos hecho de la máquina de estados...

- podemos usar un programa que use el **modelo Mealy** ➡ salidas dependen de entradas y de estado presente.
- o uno que use el **modelo Moore** ➡ salidas dependen exclusivamente de estado actual.

4. Minimizar la tabla de verdad

Mediante un programa que utiliza algoritmos heurísticos rápidos.

5. Diseñar físicamente la PLA partiendo de la tabla de verdad

Automáticamente ➡ Mediante un programa especial para diseño de layouts de PLA.

Semiautomáticamente ➡ Diseñando mediante un programa de CAD de circuitos VLSI cada una de las celdas que, repetidas convenientemente, forman la PLA.

➡ Dando una especificación de cómo han de colocarse (tabla de verdad minimizada).

Unidad de Control microprogramada *(más común)*

Idea

Emplear una **memoria** (**de control**) para **almacenar** las señales de control de los períodos de cada **instrucción**.

Conceptos

- **Microinstrucción**: Cada palabra de la memoria de control
- **Microprograma**: Conjunto ordenado de microinstrucciones cuyas señales de control constituyen el cronograma de una (macro) instrucción del lenguaje máquina.
- **Ejecución de un microprograma**: lectura en cada pulso de reloj de una de las microinstrucciones que lo forman, enviando las señales leídas a la unidad de proceso como señales de control.
- **Microcódigo**: conjunto de los microprogramas de una máquina.

Ventajas

- **Simplicidad conceptual** ➡ La información de control reside en una memoria.
- Se puede incluir (sin dificultades) **instrucciones complejas** de muchos ciclos de duración ➡ El único límite es el tamaño de la memoria de control.
- Las **correcciones, modificaciones y ampliaciones** son **mucho más fáciles** de hacer que en una UC cableada ➡ Sólo hay que cambiar el contenido de algunas posiciones de la memoria de control.
- Permite construir computadores con **varios** juegos (**configuraciones**) **de instrucciones**, cambiando tan solo la memoria de control.

Desventaja

- **Lentitud** frente a cableada (debido a una menor capacidad de expresar paralelismo de las microinstrucciones)

Control Microprogramado

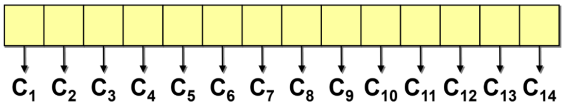
Formato de las Microinstrucciones

Las señales de control que **gobiernan un mismo elemento** del datapath se suelen **agrupar en campos**. Ejemplos:

- Señales triestado que controlan el acceso a un bus
- Señales de operación de la ALU
- Señales de control de la memoria

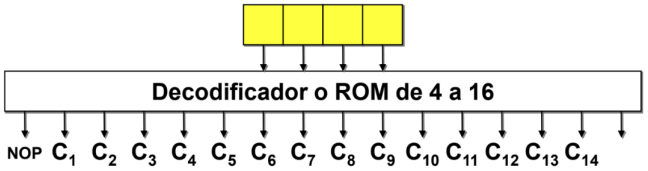
Formato no codificado

Hay **un bit para cada señal** de control de un campo



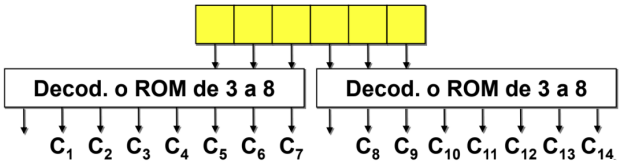
Formato codificado

Para acortar el tamaño de las **microinstrucciones se codifican** todos o alguno de sus campos.



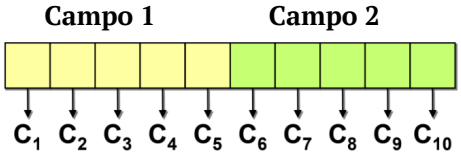
Inconveniente ➡ Hay que incluir decodificadores para extraer la información real.

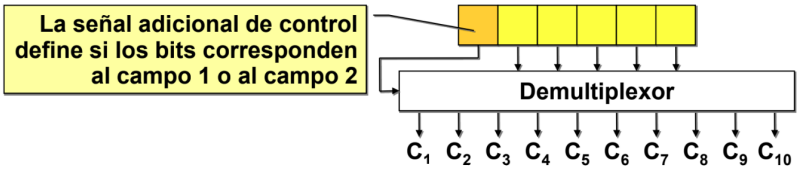
Solución de compromiso ➡



Solapamiento de campos

Si **sólo** unas **pocas señales** de control están **activas** en cada ciclo, **o existen** con frecuencia **señales excluyentes**, que no se pueden activar simultáneamente, se puede **acortar la longitud** de las microinstrucciones soplando campos:





Inconvenientes:
- **Retardo** introducido por el demultiplexor.
- Hace **incompatibles** las operaciones de los **campos** solapados.

Micro-programación horizontal

- Ninguna o **escasa codificación**
- ✔ Capacidad para expresar un **alto grado de paralelismo** en las microoperaciones a ejecutar (simultáneamente)
- ✖ **Microinstrucciones largas**

Microprogramación vertical

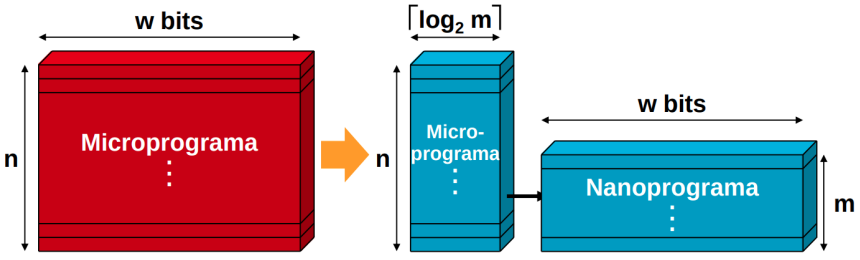
- **Mucha codificación**
- ✔ **Microinstrucciones cortas**
- ✖ **Escasa** capacidad para expresar **paralelismo** (la longitud del programa se ve incrementada)

Nanoprogramación

Objetivo

Reducir el **tamaño** de la **memoria** de control.

Esto implica una memoria a dos niveles: **memoria de control** y **nanomemoria**.



Microprograma original con n µinstrucciones de w bits. Tamaño = n·w

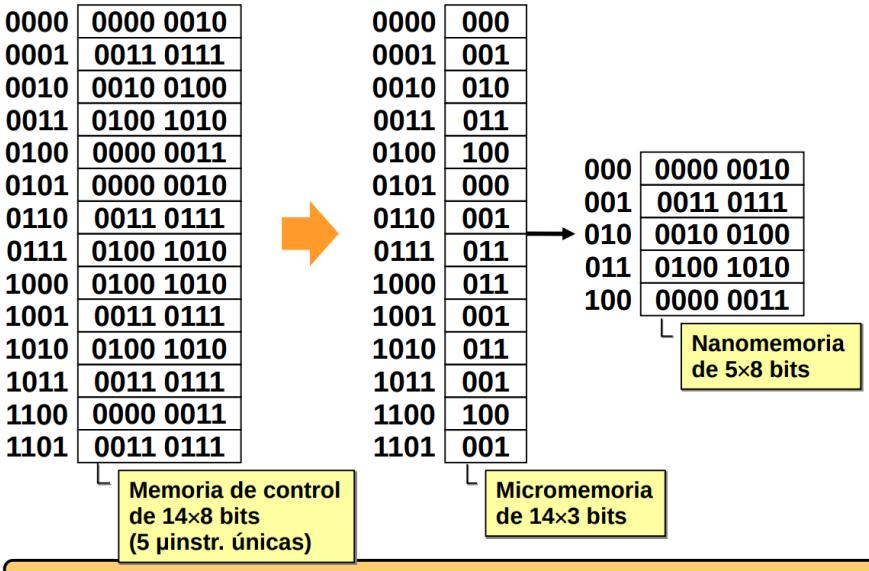
m<<n µinstrucciones únicas de 2^w posibles

Se reemplaza cada µinstrucción por su direc-ción en la nanomemoria Tamaño: n·⌈log2 m⌉

Contiene las m µinstrucciones diferentes (cada una se incluye una sola vez). Tamaño: m·w

Ahorro de memoria: n·w – (n·⌈log2 m⌉+m·w)

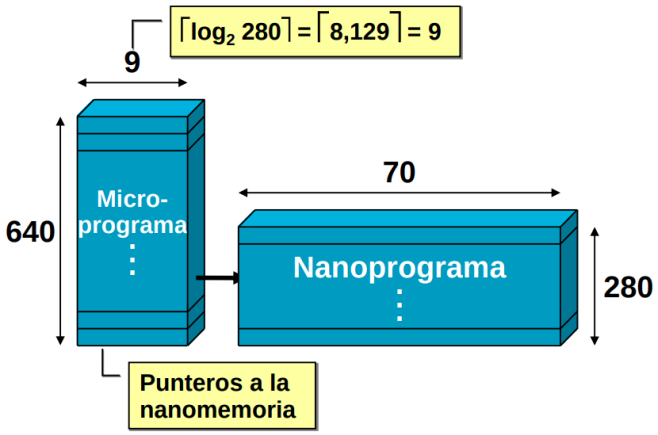
Ejemplo 1



Ahorro de memoria: $14 \cdot 8 - (14 \cdot \lceil \log_2 5 \rceil + 5 \cdot 8) = 112 - 82 = 30$ bits (27%)

Ejemplo 2

- **Estructura de la UC del Motorola 68000**
  - 640 microinstrucciones, de las cuales 280 son únicas



Ahorro de memoria: $640 \cdot 70 - (640 \cdot 9 + 280 \cdot 70) = 44800 - 25360 = 19440$ bits (43%)