EC 21-22: unidad de control

Un ordenador arquitectura Von-Neumann consta de tres bloques:

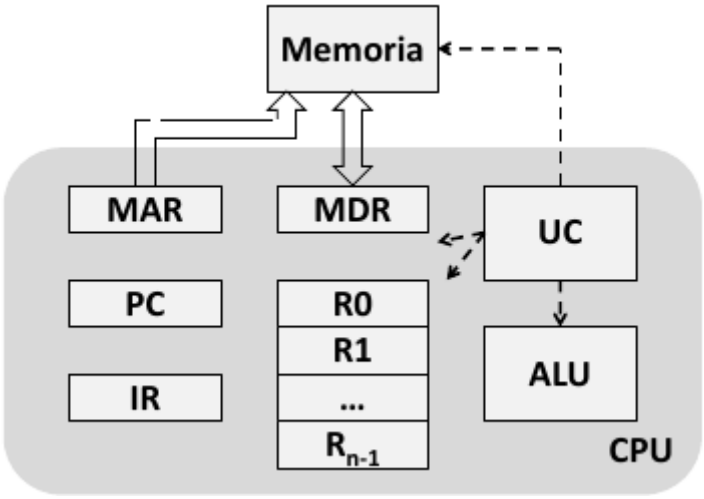
- CPU o procesador
- Memoria principal
    - datos+instrucciones
- Unidades E/S
    - +memoria masiva

En este tema estudiamos la CPU, que puede entenderse como una unidad constituida por la unidad de procesamiento o camino de datos; y la unidad de control.

1. Camino de datos

Unidad de procesamiento y unidad de control

- **Unidad de procesamiento:** comprende los siguientes elementos hardware:
    - Unidades funcionales (ALU, despl., multipl...)
    - Registros:
        - registros de uso general (RPG)
        - registro de estado
        - contador del 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:** interpreta y controla la ejecución de las instrucciones leídas de la memoria principal, en dos fases:
    - Secuenciamiento de las instrucciones
        - La UC lee de MP la instrucción apuntada por PC, IR <- M[PC]*
        - Determina la dirección de la instrucción siguiente y la carga en PC*
    - Ejecución/Interpretación de la instrucción en IR
        - La UC reconoce el tipo de instrucción
        - Manda señales necesarias para tomar los operandos necesarios y dirigirlos a las unidades funcionales adecuadas de la unidad de proceso
        - Manda las señales necesarias para realizar la operación
        - Manda las señales necesarias para enviar los resultados a su destino



Ejemplo: ejecución de Add A, R0*

M[pos A] + R0 y almacenamos en R0

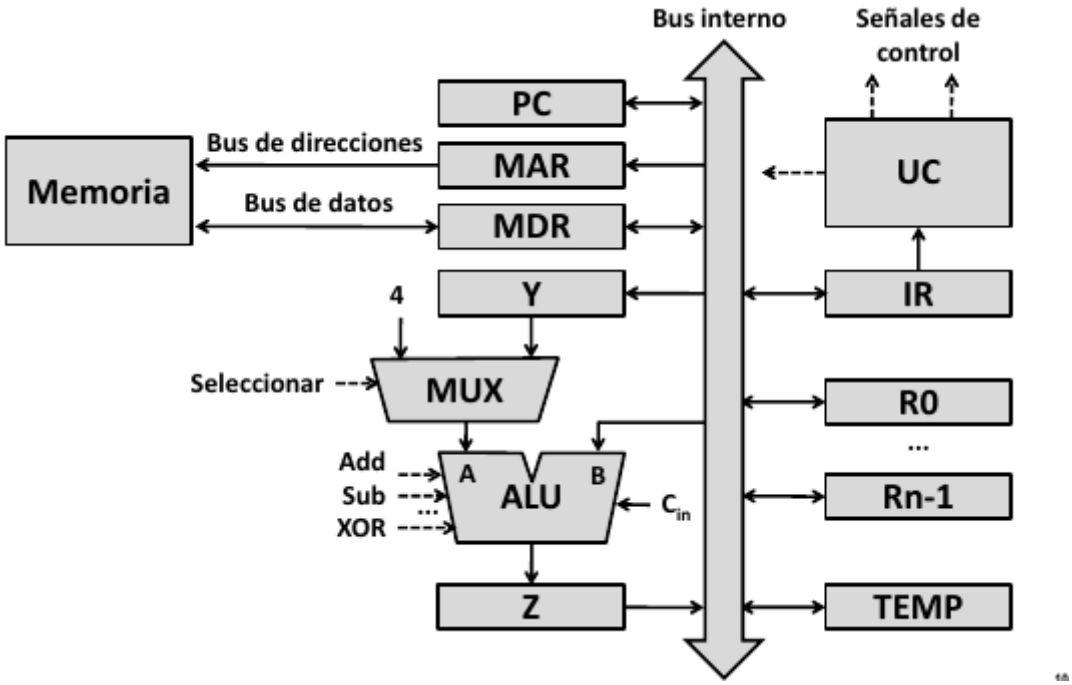
- Digamos pos A = 100
- Perdemos R0
- Arquitectura R/M

Pasos básicos de la UC:

- PC apunta a posición donde se almacena la instrucción
- **Captación:**
  - MAR ← PC
  - read
  - PC++
  - T acc
  - MDR ← bus
  - IR ← MDR
- **Decodificación:** se separan los campos de instrucción
  - Codop: ADD mem+reg → reg
  - Dato1: 100 direccionamiento directo, leer M[100]
  - Dato2: 0 direccionamiento registro, llegar R0 a ALU
- **Operando:**
  - MAR ← 100
  - read
  - T acc'
  - MDR ← bus
  - ALU in1 ← MDR
- **Ejecución:**
  - ALU in2 ← R0
  - add
  - T alu
- **Almacenamiento:**
  - R0 ← ALU out

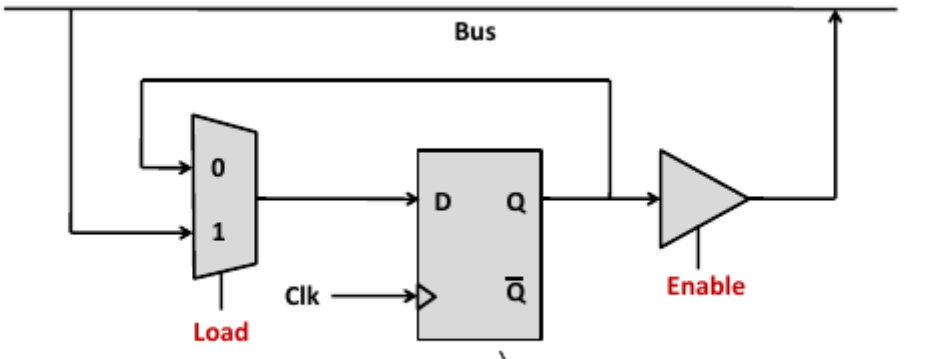
Unidad de procesamiento con un bus

- Componentes interconectados mediante bus común
  - MDR: dos entradas, dos salidas
  - MAR: unidireccional (procesador a memoria)
  - R0 a Rn-1: RPG, SP, índices…
  - Y, Z, TEMP: registros transparentes al programador, almacén temporal interno para operaciones
  - MUX: selecciona entrada A de la ALU. La cte 4 se usa para incrementar el contador del programa
- UC genera señales internas y externas (memoria)



Una instrucción puede ser ejecutada mediante una o más de las siguientes operaciones:

- **Transferir de un registro a otro:** cada registro usa dos señales de control:
  - Load (carga en paralelo)
  - Enable (habilitación de salida, buffer triestado)



- Ej.: R4 ← R1: Enable R1, Load R4

Cada bit de un registro

- **Realizar operación aritmético-lógica y almacenarla en un registro:**
  - ALU: registro combinacional sin memoria
  - Resultado almacenado temporalmente en Z

- Ej.: **R3 ← R1 + R2**
  1. Enable R1, Load Y
  2. Enable R2, Select Y, Add, Load Z
  3. Enable Z, Load R3

Cada línea: 1 ciclo de reloj

Esta transferencia no puede realizarse en el paso 2 ¿por qué?

  - Las señales de control de la ALU podrían estar codificadas

- **Cargar posición de memoria en registro:**
  - Transferir dirección a MAR
  - Activar lectura de memoria
  - Almacenar dato leído en MDR (MDR dos entradas, dos salidas)
  - La temporización interna debe coordinarse con la de la memoria
    - La lectura puede requerir varios ciclos de reloj
    - El procesador debe esperar la activación de señal de finalización de ciclo de memoria

Ej.: **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:**
  - Transferir dirección a MAR
  - Transferir dato a escribir a MDR (dos entradas dos salidas)
  - 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

- Ej.: **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

Ejecución de una instrucción completa

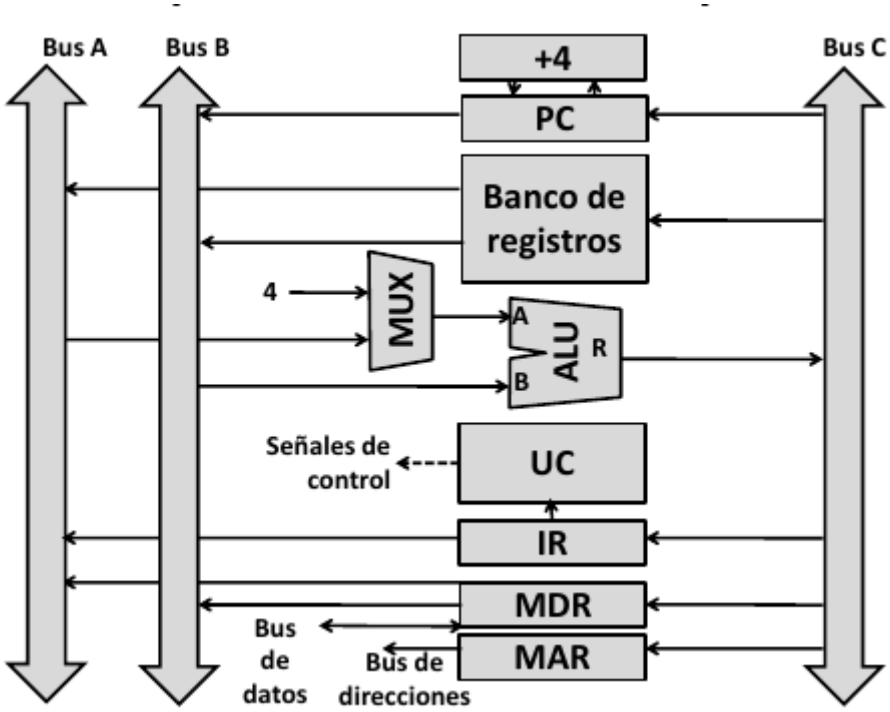
- Captación
  - Enable PC, load MAR, select 4, load Z
  - Comenzar lectura, enable Z, load PC, load Y*
  - Esperar fin ciclo de memoria, load MDR desde memoria
  - Enable MDR hacia bus interno, load IR
- Decodificar instrucción
- Ejecución:
  - Enable R3, load MAR

- Comenzar lectura, enable R1, load Y
- Esperar fin cilo de memoria, load MDR desde memoria
- Enable MDR hacia bus interno, select Y, sumar, load Z
- Enable Z, load R1, saltar a captación

Ejecución de una instrucción de salto

- Captación
  - Enable PC, load MAR, select 4, load Z
  - Comenzar lectura, enable Z, load PC, load Y*
  - Esperar fin ciclo de memoria, load MDR desde memoria
  - Enable MDR hacia bus interno, load IR
- Decodificar instrucción
- Ejecución:
  - Enable campo de desplazamiento en IR, sumar, load Z
  - Enable Z, load PC, saltar a captación

Unidad de procesamiento con múltiples buses



- Banco de registros con tres puertos, en el mismo ciclo...
  - Dos registros pueden poner sus contenidos en los buses A, B
  - Un dato del bus C puede cargarse en un registro
- ALU:
  - No necesita los registros Y,Z
  - Puede pasar A o B directamente a R mediante bus C
- Unidad de incremento (+4): la cte 4 situe siendo útile para incrementar otras direcciones en instrucciones de movimiento múltiple

Ejecución de una instrucción completa

ej. R6 ← R4+R5

- Captación:
  - Enable PC, R=B, load MAR

    - Comenzar lectura, PC++
    - Esperar fin ciclo de memoria, load MDR desde memoria
    - Enable MDR hacia B, R=B, load IR
- Decodificar instrucción
- Ejecución:
    - Enable R4 hacia A, enable R5 hacia B, select A, sumar, load R6, saltar a captación

2. Unidades de control cableadas y microprogramadas

- **Señales de entrada a la UC:**
    - Señal de reloj
    - Instrucción actual (codop, campos de direccionamiento...)
    - Estado de la unidad de proceso
    - Señales externas (ej. 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 operadores ALU
        - ...
    - Señales externas
        - Ej. lectura/escritura en memoria

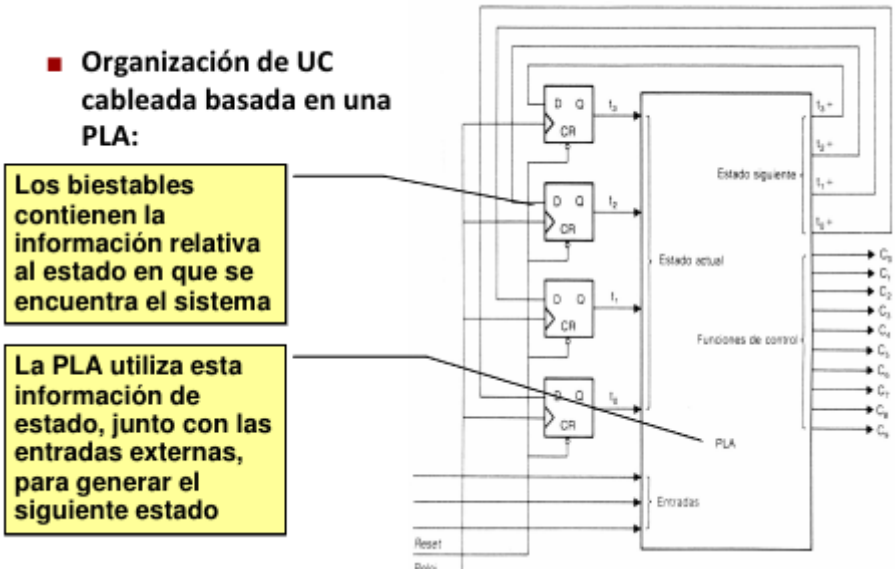
Existen dos formas de diseñar una UC:

- **Control fijo o cableado (hardwired):**
    - Se emplean métodos de diseño de circuitos digitales secuenciales a partir de programas de estados
    - El circuito final se obtiene conectando componentes básicos como puertas y biestables, aunque más a menudo se usan PLA
- **Control microprogramado:**
    - 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 CISC

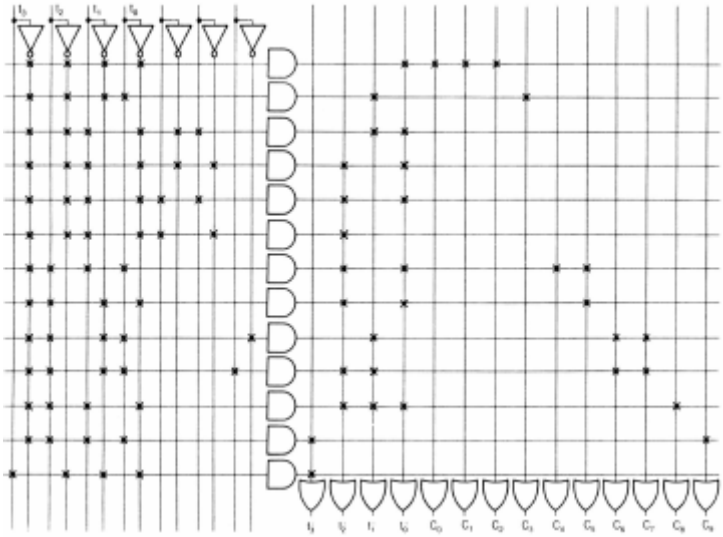
Diseño de una UC cableada

- Se diseña mediante puertas lógicas y biestables siguiendo uno de los métodos clásicos de diseño de sistemas digitales secuenciales
    - 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)
    - **Debido a las *modernas técnicas* de diseño y a RISC, ha tomado nuevo auge la realización de UC cableadas**

- Técnicas de diseño por computador (CAD) para circuitos VLSI (compiladores de silicio)
    - Resuelven automáticamente la mayor parte de las dificultades del diseño de la lógica cableada
    - Generan directamente las máscaras de fabricación de circuitos VLSI a partir de descripciones del comportamiento funcional del circuito de un lenguaje de alto nivel



Organización de UC cableada basada en PLA:



Ventajas:

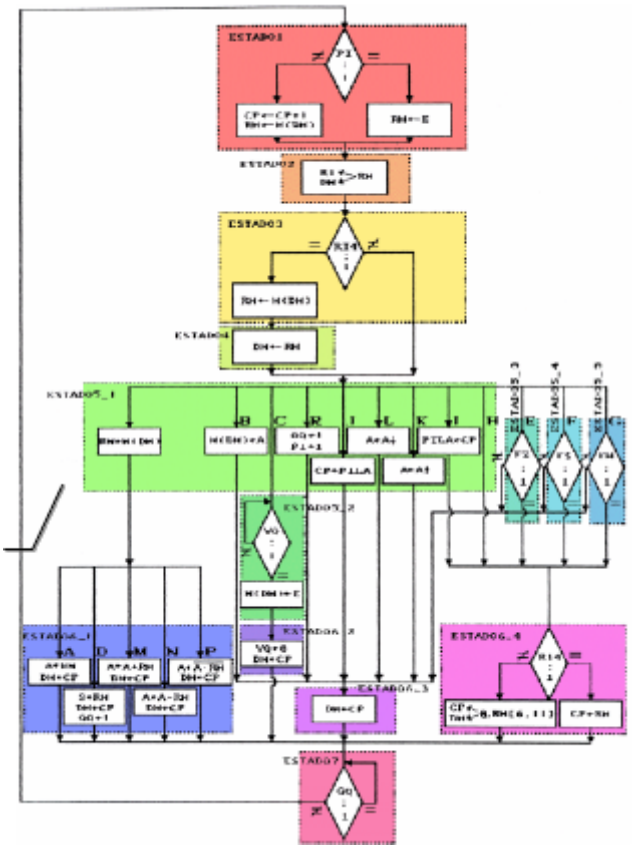
- Minimización del esfuerzo de diseño
- Mayor flexibilidad y fiabilidad
- Ahorro de espacio y potencia

Ejemplo de UC cableada

Implementación de una unidad de control cableada sencilla (ODE). Pasos a seguir para llegar al diseño físico:

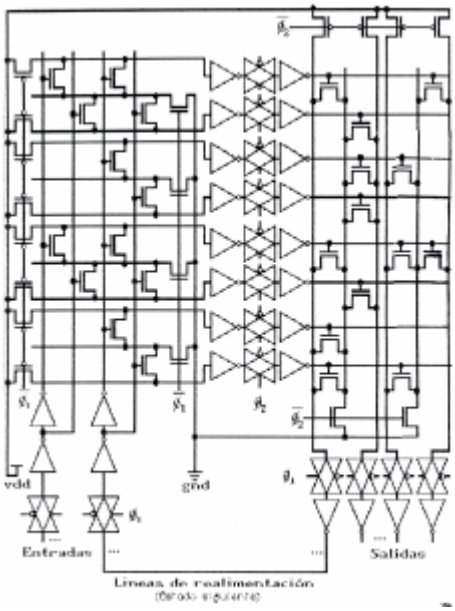
1. Definir una máquina de estados finitos
    1. Dado el diagrama de flujo de la UC de ODE, detallamos este como un conjunto finito de estados y transiciones entre ellos

Modelo Mealy: salidas dependen de entradas y estado presente



2. Describir dicha máquina en un lenguaje de alto nivel
    1. El lenguaje concreto depende del programa que utilicemos para "compilar" la descripción de la máquina. Estos lenguajes tienen sentencias para definir:
        1. Entradas y salidas
        2. Estados y transiciones condicionales entre dos estados
3. Generar la tabla de verdad para la PLA
    1. Según la descripción que hayamos hecho de la máquina de estados...
        1. Podemos usar un programa que use el modelo Mealy (salidas dependen de entradas y estado presente)
        2. O uno que use el modelo Moore (salidas dependen exclusivamente del estado presente)
4. Minimizar la tabla de verdad
    1. Mediante un programa que utiliza algoritmos heurísticos rápidos
5. Diseñar físicamente la PLA partiendo de la tabla de verdad
    1. Automáticamente, mediante un programa específico
    2. Semiautomáticamente
        1. Diseñando un programa CAD de circuitos VLSI cada una de las celdas que, repetidas convenientemente, forman la PLA
        2. Dando una especificación de cómo han de colocarse (tabla de verdad minimizada)

- **Esquema simplificado de la PLA usada para la UC de ODE**
  - CMOS de dos fases de reloj
    - No hacen falta biestables de estado siguiente
    - Φ1=1 ⍰ se leen las entradas y se precarga el plano AND
    - Φ2=1 ⍰ se evalúa el plano AND y se precarga el plano OR
    - Φ2=0 ⍰ se evalúa el plano OR

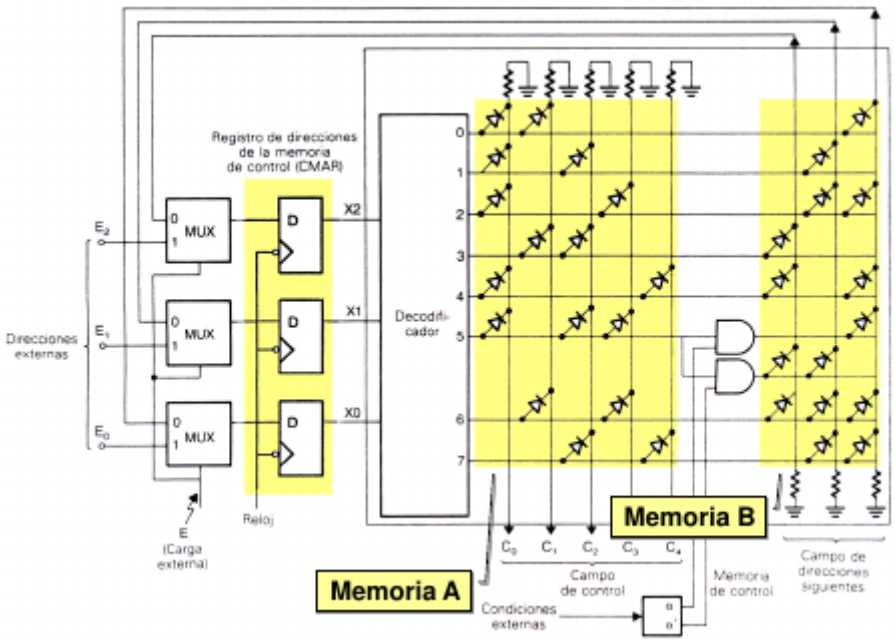


Concepto de una UC microprogramada

La idea: emplear una memoria de control para almacenar las señales de control de los períodos de cada instrucción

Origen histórico:

- Maurice Wilkes en los años 50 propone el siguiente esquema:
  - Dos memorias A, B, construidas con matrices de diodos
    - Las señales de control se encuentran almacenadas en la memoria A
    - La memoria B contiene la dirección de la siguiente microinstrucción
  - Se permiten microbifurcaciones condicionales, mediante un biestable y un decodificador que selecciona entre dos direcciones de la matriz B



Definiciones

- **Microinstrucción:** cada palabra de la memoria de control
- **Microprograma:** conjunto ordenado de microinstrucciones cuyas señales de control constituyen el cronograma de una macroinstrucción del lenguaje máquina
- **Ejecución de un microprograma:** lectura de cada pulso de reloj de una de las microinstrucciones que lo forman, enviando las señales

- **Microcódigo:** conjunto de los microprogramas de una máquina

Ventajas y desventajas

Ventajas:

- Simplicidad conceptual: la información de control reside en una memoria
- Se pueden incluir sin dificultad 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
    - No hay que rediseñar toda la unidad, sino cambiar únicamente el contenido de algunas posiciones de la memoria de control
- Permite construir computadores con varios juegos de instrucciones, cambiando el contenido de la memoria de control (si es RAM permite emular otros computadores)

Desventaja:

- Más lenta que la cableada, debido a una menor capacidad de expresar paralelismo entre las microinstrucciones

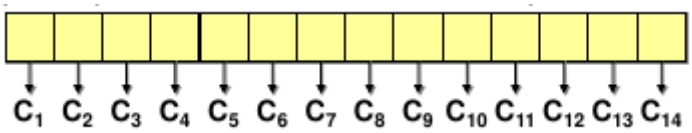
3. Control microprogramado

Formato de las microinstrucciones

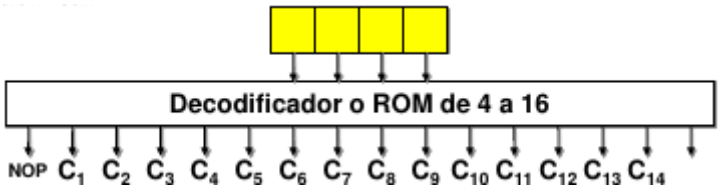
Las señales de control que gobiernan un mismo elemento del datapath se suelen agrupar en campos. Por ejemplo:

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

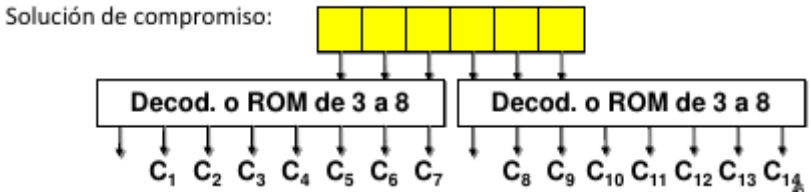
Formato no codificado: hay un bit para cada señal de control de un campo



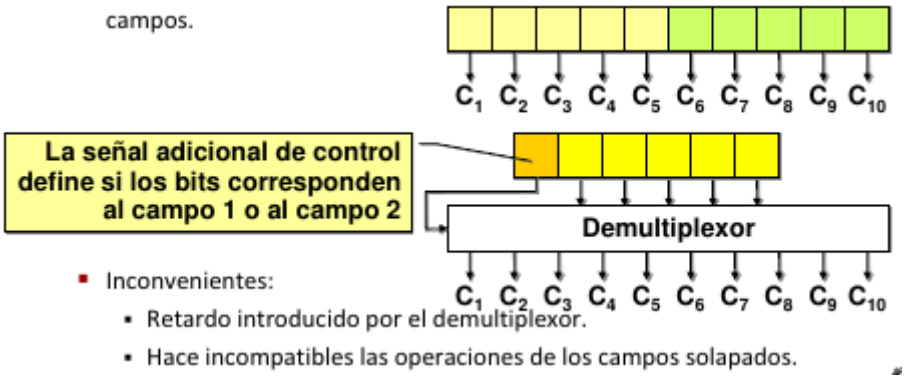
Formato codificado: para acortar el tamaño de las microinstrucciones se codifican todos o alguno de sus campos. El inconveniente es que hay que incluir un decodificador para extraer luego la información



El compromiso:



Solapamiento de campos: si solo unas pocas señales de control están activas en cada ciclo, o existen con frecuencia de señales excluyentes (no se pueden activar simultáneamente), se puede acortar la longitud de las microinstrucciones solapando campos



Microprogramación vertical y horizontal

- **Horizontal:** ninguna o escasa codificación
    - Pro: capacidad para expresar un alto grado de paralelismo en las microoperaciones a ejecutar (simultáneamente)
    - Contra: microinstrucciones largas
- **Vertical:** mucha codificación
    - Pro: microinstrucciones cortas
    - Contral: escasa capacidad para expresar paralelismo (la longitud del programa es mayor)

Nanoprogramación

Secuenciamiento de las microinstrucciones