Microprogramación

Agustín Fernández, Josep Llosa, Fermín Sánchez

Estructura de Computadors II
Departament d'Arquitectura de Computadors
Facultat d'Informàtica de Barcelona

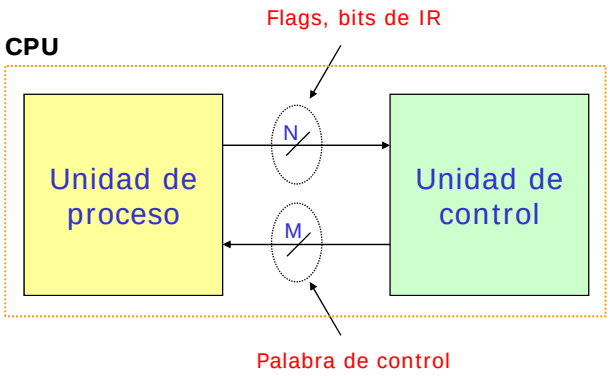
Índice

- Introducción
- Segmentación
- Procesadores RISC
- Pentium 4

Introducción

- Visión vertical en niveles de un computador: Ubicación de la microprogramación
- Diseño cableado de un procesador
  - Unidad de proceso: elementos básicos
  - Unidad de control: sistema secuencial
- Diseño de un procesador
  - Instrucciones de LM
  - Modos de direccionamiento
  - Registros visibles desde el LM
  - Formato de las instrucciones
- Unidad de Control cableada versus microprogramada

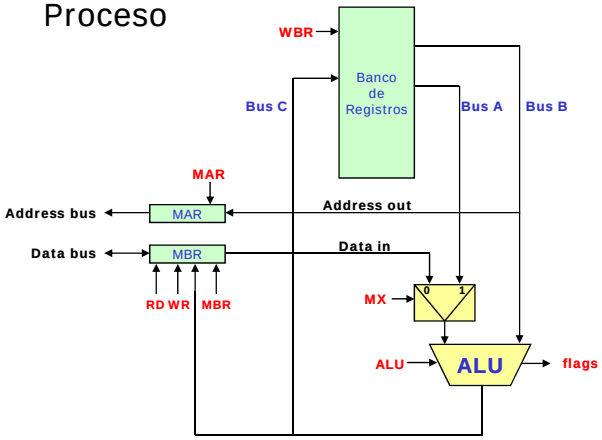
Introducción



Introducción

Unidad de Proceso

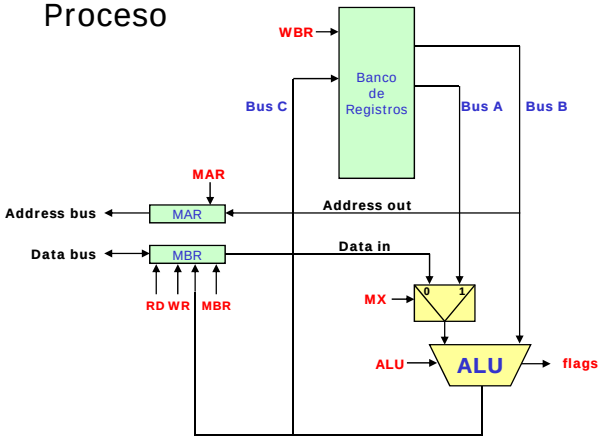
¿Cómo ejecutar addl %eax, (%esi)?



Introducción

Unidad de Proceso

¿Cómo ejecutar addl %eax, (%esi)?



MAR ← %esi

RD

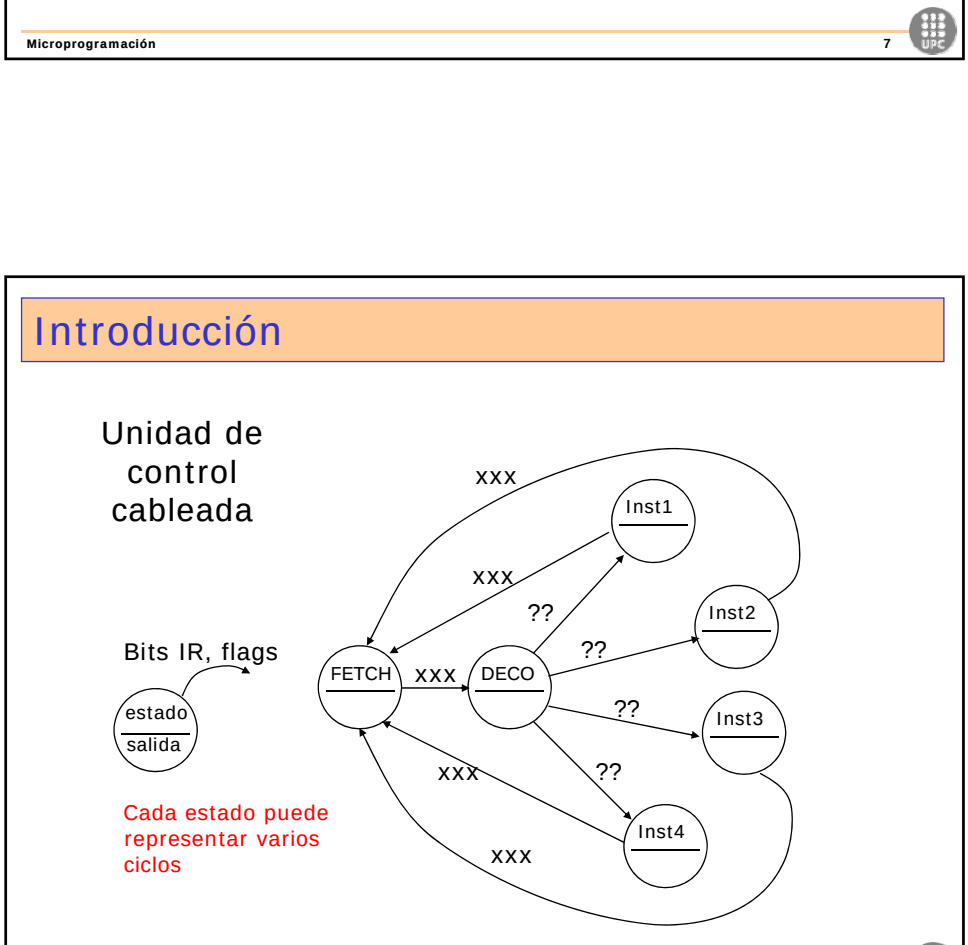
MBR ← MBR + %eax

WR

Introducción

¿Cómo ejecutar addl %eax, (%esi)?

Bus C	WBR	Bus A	Bus B	MX	ALU	MAR	RD	WR	MBR	
X	0	X	%esi	X	X	1	0	0	0	**MAR ← %esi**
										RD
										MBR ← MBR + %eax
										WR



Introducción

Unidad de control cableada

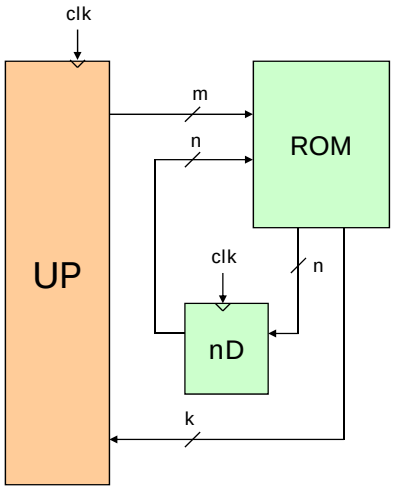
Bits IR, flags

estado / salida

Cada estado puede representar varios ciclos

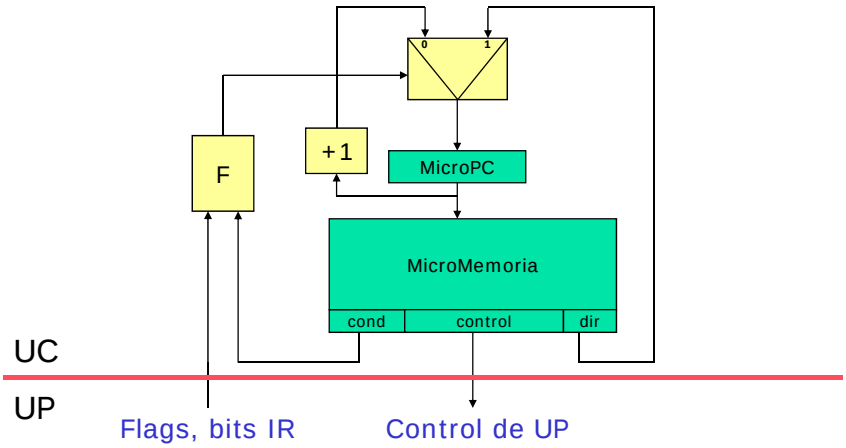
Introducción

- Unidad de control cableada
  - X estados
  - n=$\log_2$ X biestables D
  - m entradas (bits IR, flags)
  - k Salidas (UP)
  - ROM 2^{m+n} . (k+n) bits



Introducción

Unidad de control microprogramada



Introducción

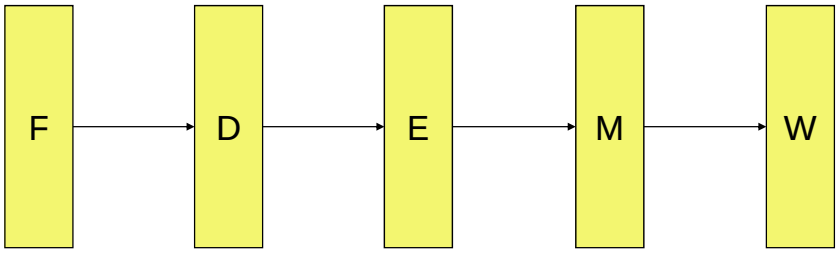
- Unidad de control microprogramada
  - Cada instrucción "compleja" se divide en varios "pasos sencillos" que se ejecutan secuencialmente
    - Fetch
    - Decodificación
    - Ejecución detallada
  - La ejecución de cada "paso" se describe en una palabra de la micromemoria
    - control: bits que controlan directamente los circuitos de la UP
    - dir: dirección de la próxima microinstrucción en caso de que se tenga que romper el secuenciamiento implícito en la micromemoria
    - cond: forma de evaluar los flags y bits del IR para decidir si se produce o no salto

Segmentación

- La ejecución de instrucciones se divide en fases:
  - Fetch (F)
  - Decodificación (D)
  - Ejecución y cálculo de direcciones efectivas (E)
  - Acceso a memoria (M)
  - Escritura (W)
- Cada fase se realiza en uno o varios ciclos (o etapas)
- Cada fase se realiza en una parte específica del procesador
- Varias instrucciones pueden ejecutarse simultáneamente, cada una de ellas en una fase diferente de su ejecución

Segmentación

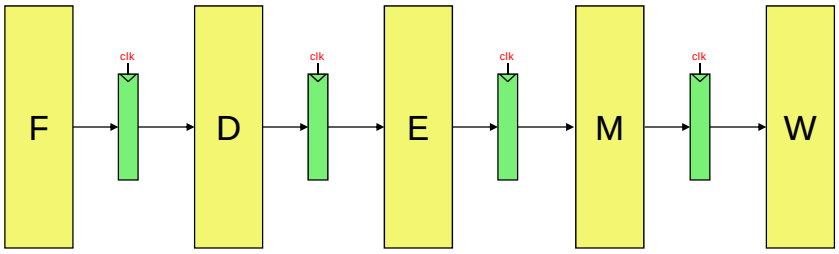
Procesador No-segmentado



Unidad de Fetch | Unidad de Decodificación | ALU | Memoria de datos | Banco de Registros

Segmentación

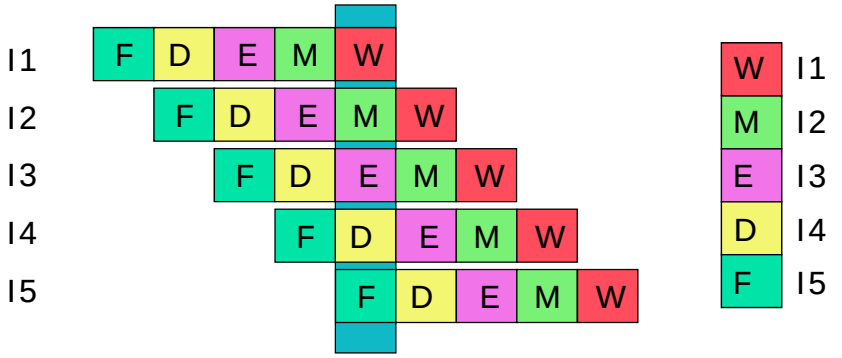
Procesador Segmentado



Unidad de Fetch | Unidad de Decodificación | ALU | Memoria de datos | Banco de Registros

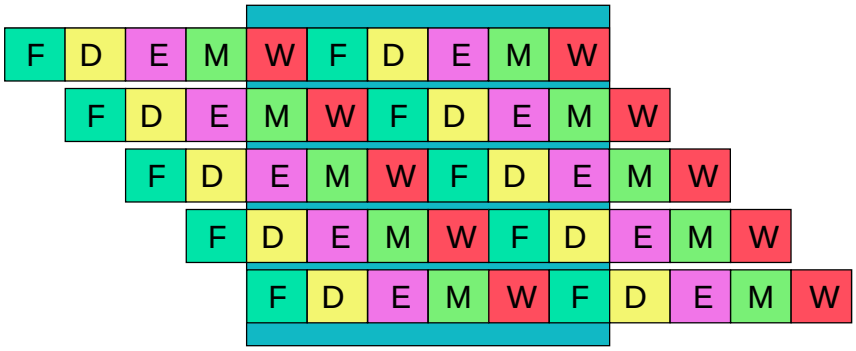
Segmentación

- Ejemplo sencillo: 1 inst.=5 ciclos, 1 ciclo/etapa



Segmentación

- Ejemplo sencillo: 1 instrucción / ciclo



Segmentación

- Para que la segmentación sea posible, las instrucciones deben ser sencillas
- Riesgos: No siempre puede ejecutarse una instrucción sin que la anterior haya finalizado
  - Riesgos estructurales: conflictos de recursos
  - Riesgos por dependencias de datos: la ejecución de una instrucción depende de los resultados de una instrucción anterior
  - Riesgos de control: provocados por los saltos

Procesadores RISC

- RISC (Reduced Instruction Set Computer)
- Pocas instrucciones, sencillas y fácilmente segmentables
- Tipos de instrucciones
  - Acceso a memoria (load, store)
  - Aritmético-lógicas (operan con registros de la CPU)
  - Saltos (condicionales e incondicionales)
- Las instrucciones IA32 son complejas y difíciles de segmentar. Ej: addl $17, 35(%ebx, %edx, 4)

Procesadores RISC

- Ejemplo: un procesador RISC sencillo
  - Ri ← M[Rj]
  - M[Rj] ← Ri
  - Ri ← Ri op Rj
  - Beq Ri, etiqueta
  - Bneq Ri, etiqueta
- Instrucciones sencillas, fáciles de segmentar y rápidas de ejecutar
- ¿Se pueden ejecutar instrucciones IA32 con este juego de instrucciones?

Procesadores RISC

- Ejemplo: addl $17, 35(%ebx, %edx, 4)

```
tmp6 ← GetFactorEscala(IR)
tmp1 ← %edx << tmp6
tmp2 ← tmp1 + %ebx
tmp7 ← GetDesplazamiento(IR)
tmp3 ← tmp2 + tmp7
tmp4 ← M[tmp3]
tmp8 ← GetInmediato(IR)
tmp5 ← tmp8 + tmp4 (activar flags)
M[tmp3] ← tmp5
```


Procesadores RISC

- Ejemplo: call etiq

%esp ← %esp - 4
M[%esp] ← %eip
tmp1 ← GetDesplazamiento(IR)
%eip ← %eip + tmp1

Procesadores RISC

- Ejemplo: ret
- Ejemplo: pushl 100(,%ebx,8)

Hacedlo vosotros

Procesadores RISC

- Ejecución de instrucciones IA32
  - Fetch
    IR <= M[%eip]
    %eip <= %eip + constante (tamaño instrucción)
  - Decodificación
    - Traducción de instrucciones IA32 a microinstrucciones RISC
  - Ejecución
    - Ejecución de las instrucciones RISC de forma segmentada

Pentium 4

- Compatible IA32
- Traduce de IA32 a lenguaje interno pseudo-RISC (micro_operaciones) en tiempo de ejecución
- Segmentado en 20 etapas
- Cada fase dura varios ciclos
- Fase Fetch+decodificación (traducción a micro_operaciones): 4 ciclos
- Varias decenas de micro_operaciones en ejecución simultánea

Ejemplo arquitectura Core