UAH

Introducción

590005
Arquitectura e Ingeniería de Computadores

Objetivo

- Conceptos de Arquitectura de Computadores
- Repaso Von-Neumann
  - Fases, Rendimiento
  - Cálculos Rendimiento
- Perspectivas de aceleración
- Conceptos de paralelismo y concurrencia
- Arquitecturas de Computadores para la explotación del paralelismo

UAH

Arquitectura de Computadores

- Los atributos de computador tal cual los ve un programador.
  - (Amdahl, Blaaw, and Brooks, 1964)
- El término ‘arquitectura’ se pretende que cubra los **tres** aspectos del diseño de un computador: repertorio de instrucciones, organización y hardware (Hennessy-Patterson, 1996 & 2017)
- Def. Clásica: ISA=Instruction Set Architecture
- Evolution
  - ISA
  - +Implementación=
    - Organización
    - Hardware

UAH

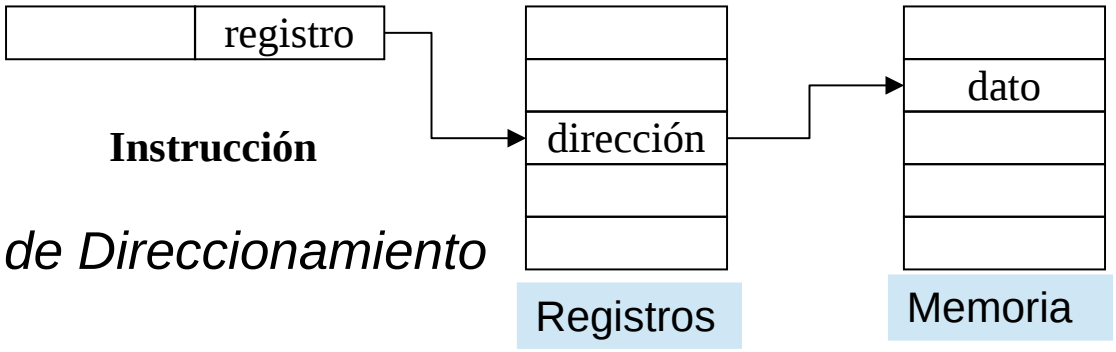
Visión Jerárquica de un computador

UAH

Arquitectura del Repertorio de Instrucciones (ISA)

Codificación

Op-code Rx Ry
10111 1010
0011

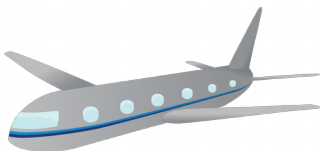


Modos de Direccionamiento

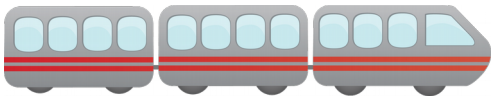
Tendencias Diseño

RISC

CISC



```
andi    R1,#0
ld      R3, N(r0)
bucle:
        st      R3, cont(r0)
        ld      R2, cont(r0)
        str     R2, sum(r0)
        add     R1, sum(r0)
        subi    R3, #1
        bnz     bucle
str     R1, res(r0)
```

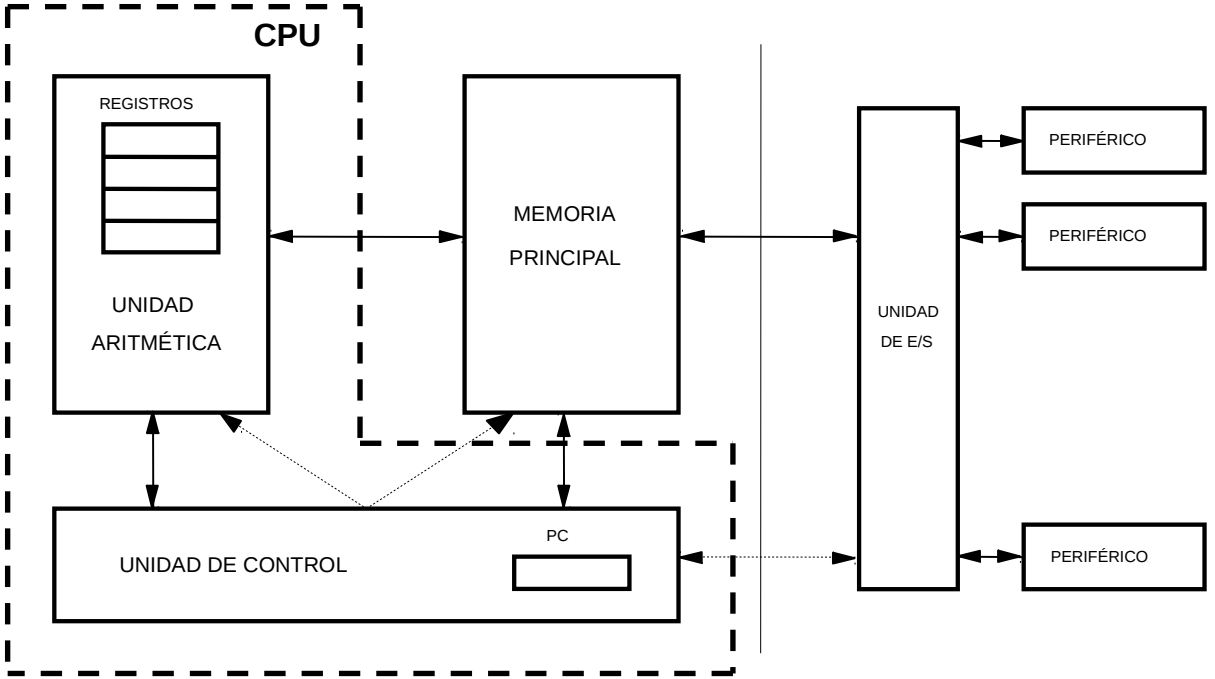


```
MOV #600, R0
MOV #prtbuf, R1
MOV #76, R2
START
  MOVB (R0)+, (R1)+
  DEC R2
  BNE START
HALT
```

UAH

Organización o Microarquitectura

- Aspectos de diseño del HW de alto nivel
- Unidades funcionales
- Memoria
- Su interconexión



Organización o MArq

- Dos procesadores con el mismo ISA (Inst. Set Arch.) pueden tener diferentes implementaciones
  - Ej: AMD Opteron e Intel Core i7
  - Ambos implementan 80x86 ISA
  - Muy distinta organización
    - Diferente cauce de segmentación
    - Diferente sistema memoria cache
  - Diseño de la CPU (aritmética, lógica de saltos etc.)
- Tendencia de múltiples procesadores por microprocesador
  - CORE / MULTICORE (multiples procesadores)
  - Término CPU se va usando menos vs. multicore

Hardware

- Detalles del diseño de bajo nivel
  - Diseño lógico
  - Tecnología de sus circuitos integrados (ICs)
- Dos procesadores con el mismo ISA y organizaciones casi idénticas pueden tener muy diferente implementación hardware
  - Ej: Intel Core i7 y Intel Xeon E7 (Xeon más eficiente Servidores)
    - Diferentes frecuencias de reloj
    - Distintos sistemas de memoria
- Tecnologías cambiantes influyen en la evolución de los computadores
  - Arquitecto debe tenerlas en cuenta para un diseño perdurable

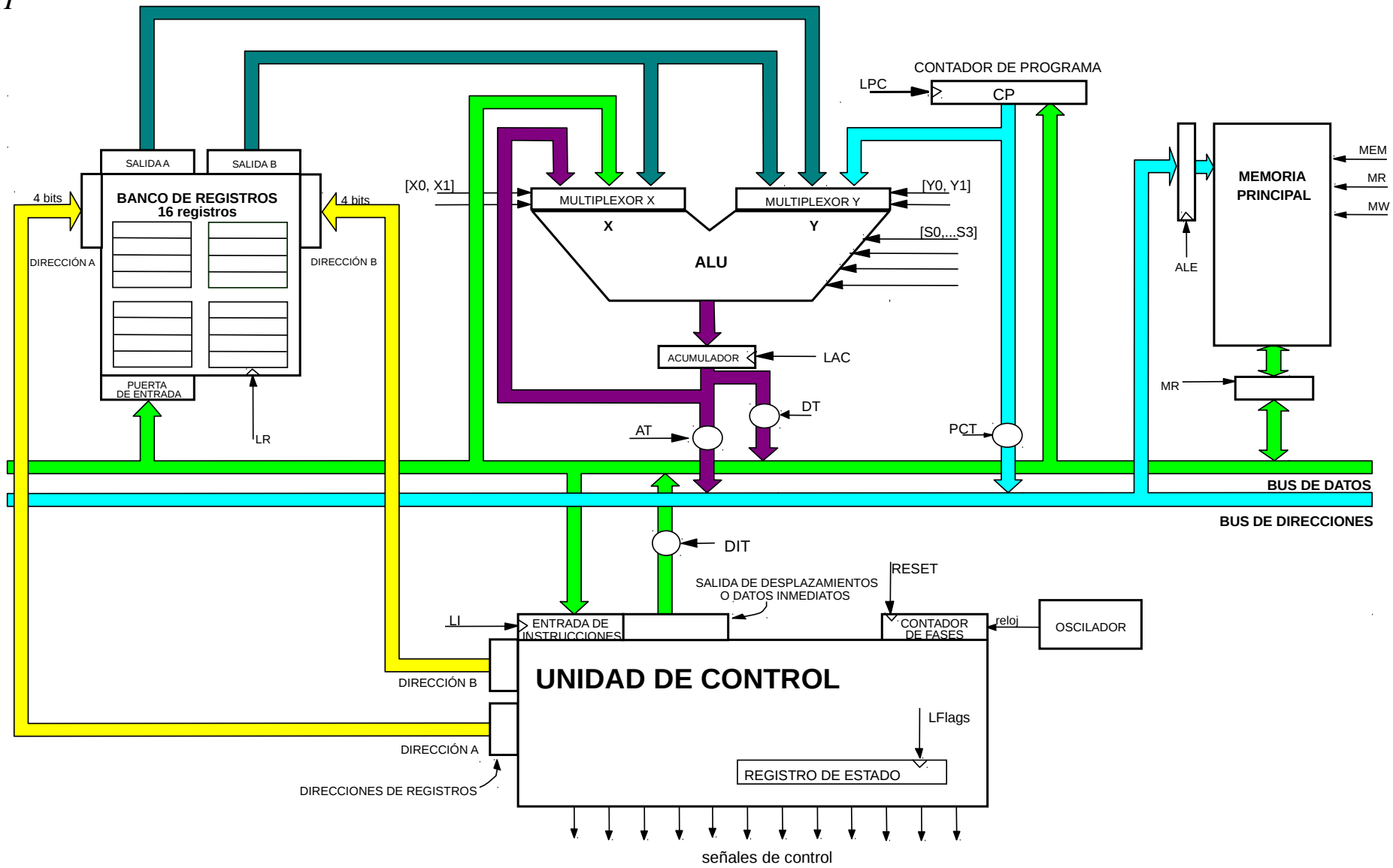
La máquina de Von Neumann

Computador

- Máquina capaz de realizar cómputos
- Ejecuta un programa: PROGRAMABLE
  - Las proto-máquinas “siempre tocaban la misma canción”
- Programa: secuencia de instrucciones
  - El computador ha de ejecutar una tras otra sin fin
- Programa + Datos → Memoria
- CPU: Registros especiales, Unidades funcionales, Registros de Propósito General (GPR) - para operar con los datos -
- Computador básico: la máquina de Von-Neumann

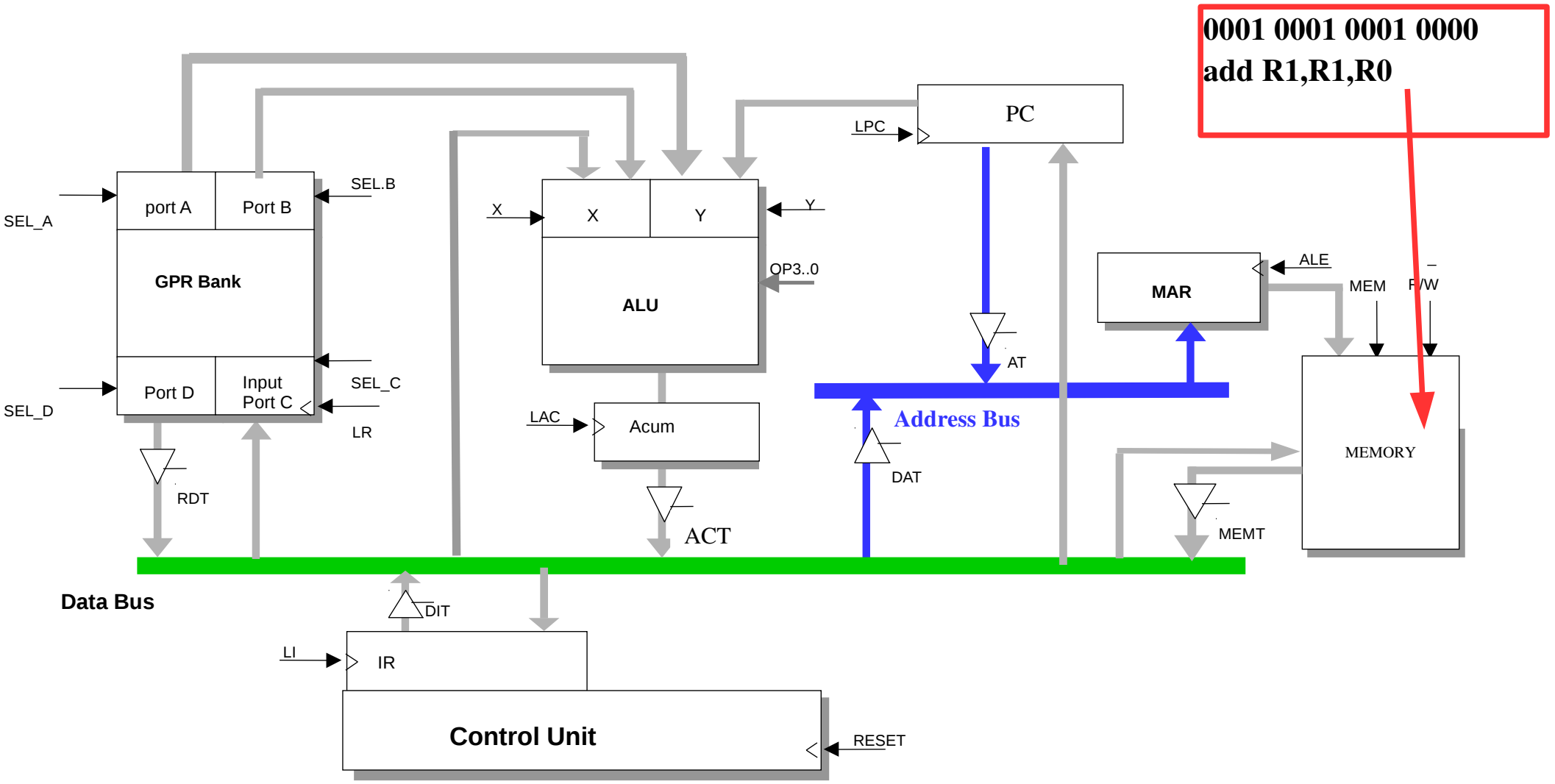
UAH

Von Neumann



UAH

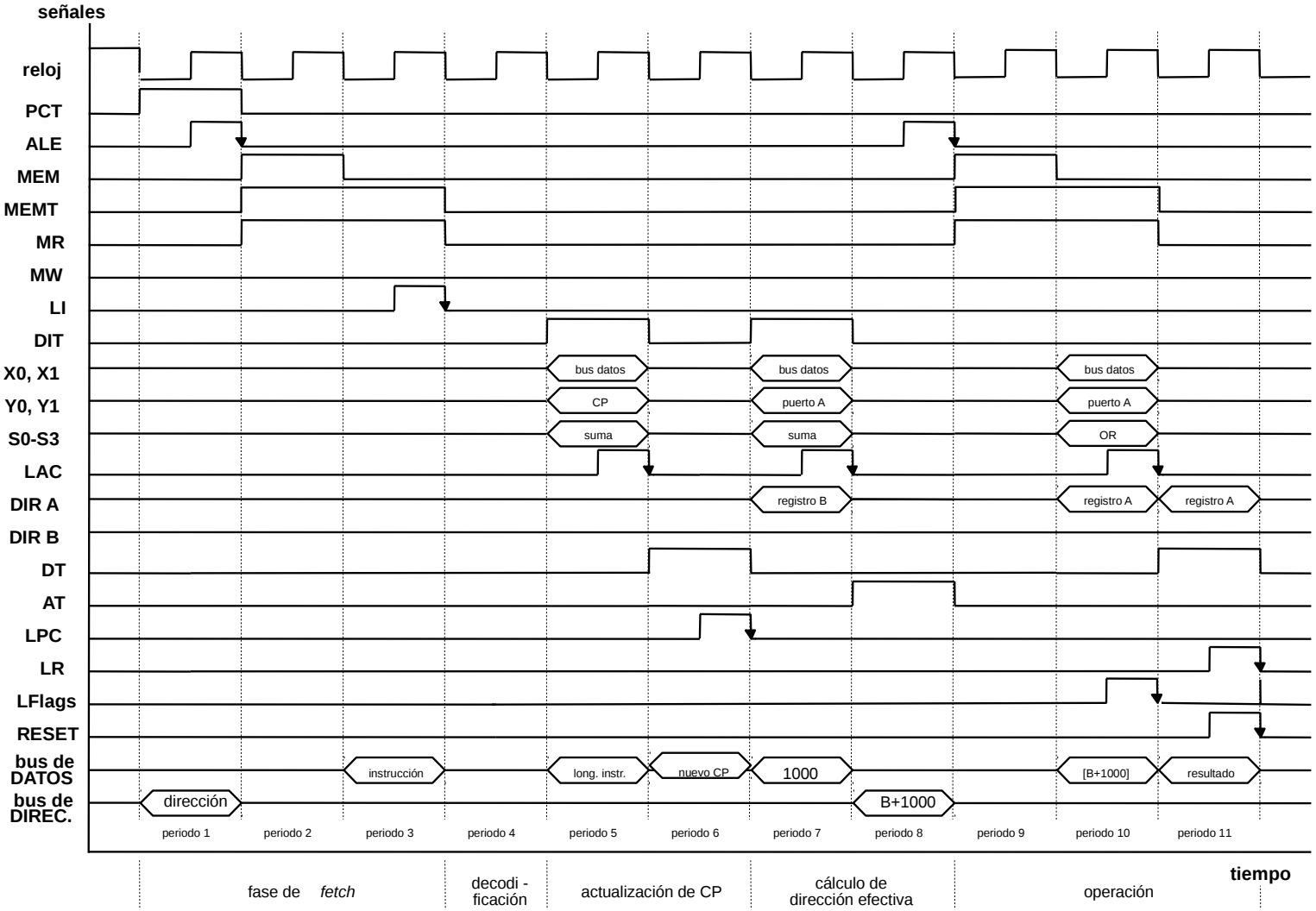
Von Neumann



UAH

Fases

- 1. Búsqueda de la instrucción - Instruction FETCH (IF)
  - RI <---M(PC) .
- 2. Decodificación – Instruction Decoding (ID)
  - Manipulación RI para determinar de qué instrucción se trata
- 3. Incremento PC (IPC)
  - PC ← PC + 4 *** depende organización memoria
- 4. Ejecución (ej: instrucción de proceso en máquina R-R)
  - 4. Búsqueda de operandos -Operands Fetch (OF)
  - 5. Ejecución de la operación (EX)
  - 6. Almacenamiento del resultado (WB)


señales
reloj
PCT
ALE
MEM
MEMT
MR
MW
LI
DIT
X0, X1
Y0, Y1
S0-S3
LAC
DIR A
DIR B
DT
AT
LPC
LR
LFlags
RESET
bus de DATOS
bus de DIREC.
bus datos
CP
suma
bus datos
puerto A
suma
bus datos
puerto A
OR
registro B
registro A
registro A
instrucción
long. instr.
nuevo CP
1000
[B+1000]
resultado
dirección
B+1000
periodo 1
periodo 2
periodo 3
periodo 4
periodo 5
periodo 6
periodo 7
periodo 8
periodo 9
periodo 10
periodo 11
tiempo
fase de fetch
decodi - ficación
actualización de CP
cálculo de dirección efectiva
operación

Rendimiento
Métricas

UAH

Rendimiento

Rendimiento: rapidez con la que el computador puede ejecutar programas

Acabar un trabajo primero (T. de respuesta o de ejecución)
Completar más trabajos durante un período de tiempo

Rendimiento = 1/Tiempo de ejecución

Rendimiento Relativo = Ganancia = Aceleración

Rendimiento de A / Rendimiento de B = n > 1

Una mejora n > 1

UAH

Rendimiento

- Mediciones en tiempo:
  - Cuánto tarda en ejecutar un programa
    - Tiempo de CPU o **Tiempo de Ejecución** (usaremos este)
    - Excluye interacción con I/O
  - Cuánto tarda en responder a una petición
    - Depende de I/O (no lo usaremos)
- Eficiencia (Throughput).
  - Trabajo completado por unidad de tiempo
  - **MIPS**: Millones de Instrucciones Por Segundo
  - Instrucciones Por Ciclo (IPC) --inverso de CPI

UAH

Ecuaciones Rendimiento

T = Tiempo de ejecución de un programa

$T = Cy \cdot Tc$

Cy = Ciclos de reloj consumidos por el programa

Tc = Período del ciclo de reloj en segundos (o ns)

Tc =1/f

f = frecuencia en Hertz (Hz= ciclos/seg), MHz o GHz)

$Cy = I \cdot CPI$

I = Total de instrucciones ejecutadas (recuento dinámico)

CPI = Ciclos por Instrucción.

Media de ciclos de reloj empleados (mezcla particular)

$MIPS = 10^{-6} \cdot I / T$ m.i.p.s.

Ejemplo 1

Se tiene la siguiente información sobre la mezcla de instrucciones pertenecientes a un repertorio de una máquina que ejecutada un programa Benchmark. Calcular CPI de dicha mezcla

Se debe obtener la media ponderada de 3 cpi

Tipo	**Ciclos que consume**	**%uso**
Aritmética y lógica entera	2	50
Carga/Almacenamiento	4	20
Transferencia de control	2	20
Aritmética Coma Flotante	8	10

Ejemplos:

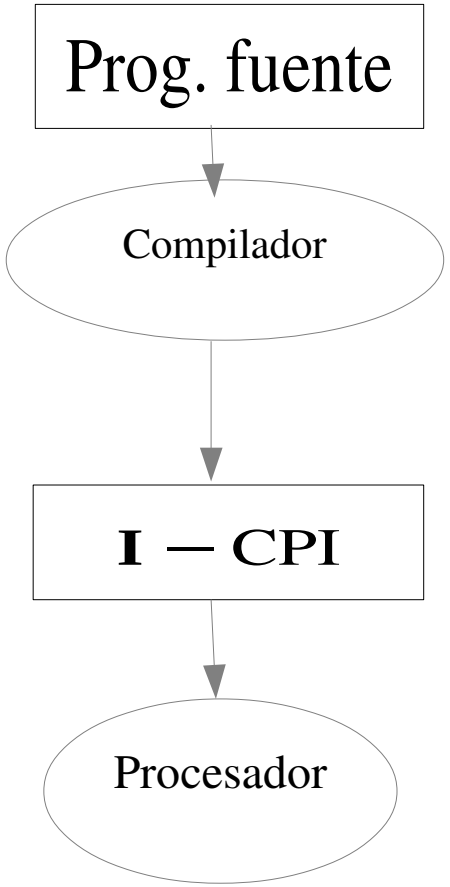
add	R1,R2,R3	R1 ← R2+R3
ld	R1, 16(R4)	R1 ← M(R4+16)
jnez	R5, label1	PC ← gets label1 (address of next instruction)
addf	F0,F5,F3	F0 ← F5+F3 real numbers in FP representation

Ejemplo 2

En un procesador de una frecuencia de 40 MHz. se ejecuta un *benchmark* con la mezcla de instrucciones y ciclos mostrada en la tabla a continuación.
Calcular recuento de instrucciones, CPI, MIPS y Tiempo de ejecución.

Tipo	**NºInstruc.**	**Ciclos**
Aritmética entera	45.000	3
Transferencia de datos	32.000	2
Coma Flotante	15.000	10
Transferencia de control	8.000	2

Sol: 100000 I.; Cy=365000; CPI=3,65; T ~ 9ms.; 10,95MIPS



$$T = I \cdot CPI \cdot T_c$$

Se reduce si

I →
- Genera menos instrucciones máquina
- o/y menos bucles
- OJO: Recuento dinámico, no Estático

CPI →
- Se reduce nº de ciclos de c/instr
- o si la ejecución de las instr. se solapa (*segmentación y ejecución superescalar*)

T_c →
- Se incrementa la frecuencia de reloj
  - Mejorar la Tecnología del CI
  - Reducir el procesamiento por ciclo

Observar que no son factores independientes: mejorar uno puede empeorar otro

Factores y su ámbito de influencia

Tej =	Nº Inst.	CPI	Per. reloj
Programa	*X*		
Compilador	*X*	(X)	
Repertorio inst.	*X*	X	(X)
Organización		X	X
Tecnología			X

Aceleración o Ganancia

- Mide el efecto de una mejora en un computador “A” o “G”
- **Cociente** entre Tiempos de Ejecución antes y después de la mejora de un mismo programa(s)
  - Adimensional
  - > 1 significa que hay mejora
    - T1/T2 es >1 si T2 < T1
    - T2 es menor: se redujo el tiempo de ejecución, hay ganancia
  - El programa/programas se denomina “benchmark”

Ejemplo 3

- Un programa *benchmark* en un procesador se ejecuta en 250 ms de los cuales, 200 ms se emplean en operaciones que manipulan números enteros.
- Se realiza un cambio en la circuitería de este procesador con el propósito de acelerar estas operaciones de enteros.
- Tras hacer estos cambios se comprueba que el mismo programa que se ejecutaba antes en 250 ms ahora tarda 210 ms.
-

¿Cuál es la Ganancia o Aceleración obtenida?

G = 250ms / 210ms = 1,19 o un 19% de aceleración

Ejemplo 3

¿Cuánto se ha acelerado el procesamiento de enteros para obtener esta ganancia?

T1 = 250ms = 200ms + 50 ms T2 = 210ms = ?? + 50ms

El tiempo de proceso de enteros (??) tras la mejora fue de 160ms

La aceleración parcial sobre los enteros es de 200ms/160ms = 1,25

Conclusión: una aceleración de un 25% sobre los enteros permite que el benchmark se acelere un 19% (G)

Aumentar el rendimiento es la cuestión

UAH

Necesidad de aumentar Rendimiento

- ¡Siempre se superan las capacidades de los computadores!
  - Mayor Potencia de Cálculo
  - Saturación de las máquinas (tarde o temprano)
- Aumento lineal en algunos problemas implica un aumento exponencial del cálculo asociado. Ejemplo:
  - Si multiplicar matrices cuadradas de tamaño N requiere N^3 operaciones (N^2 elementos, cada uno requiere N multiplicaciones y N-1 sumas)...
  - Multiplicar matrices de 2N implicarán $(2N)^3 = 8N^3$ multiplicaciones
  - Para matrices de 10N serían $(10N)^3 = 10.000N^3$ multiplicaciones

UAH

Necesidad-..

Ejemplos de aplicaciones ávidas de potencia de cálculo

- Procesos de control en *tiempo real*
  - (centrales nucleares, vehículos auto-guiados, vehículos espaciales, aplicaciones de tráfico reactivas, ...)
- Simulación
  - (modelos biológicos, predicciones meteorológicas, aviónica, ...)
- Otros
  - (inteligencia artificial, minería de datos, demanda psicológica, ...)

UAH

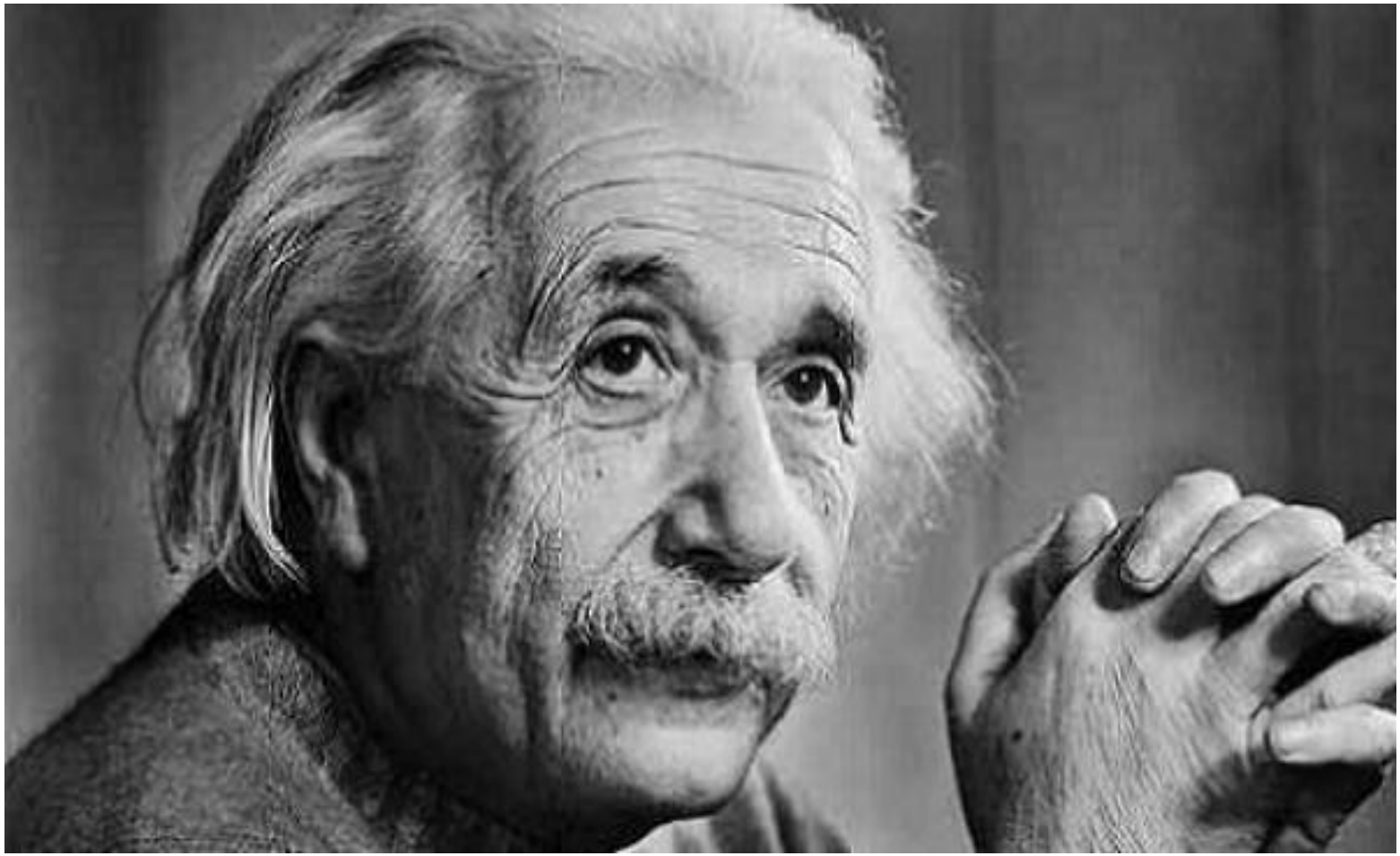
La pregunta es

¿Podemos seguir aumentando las capacidades de una máquina clásica, la máquina von Neumann?

¿De qué factores depende este aumento?

UAH

Los límites físicos del crecimiento



UAH

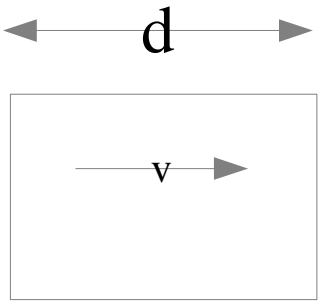
Limitaciones físicas: propagación

Aumentar la FRECUENCIA del Reloj

Límites:

- Velocidad transición 0-1, 1-0 en Si
- Generación excesiva de calor: Disipación
- Espacio ocupado (densidad del CI)

Propagando en una superficie de 3x3cm.



transiciones (1 --> 0) en Si:

d = 3 cm

v = $3x10^7$ m/s (la luz @ $3x10^8$ m/s)

--> t = 1 ns

t = d/v

Limitaciones físicas...

Objetivo: Aumentar la densidad de integración, componentes/cm^3

Límites: ancho de *calle* ya cercano a la longitud de onda

La fabricación de CI tiene procesos de impresión en material fotosensible

Es lo que permite “dibujar” vías de conexión entre componentes. Una calle es una de estas “líneas”.

La separación entre ellas se consigue con una máscara debe poder impresionarse en el proceso fotográfico --> tamaño de longitud onda luz

Limitaciones físicas...

Las Memorias no son tan rápidas como los Elementos de cálculo

--> Cuellos de Botella en el uso de Memoria

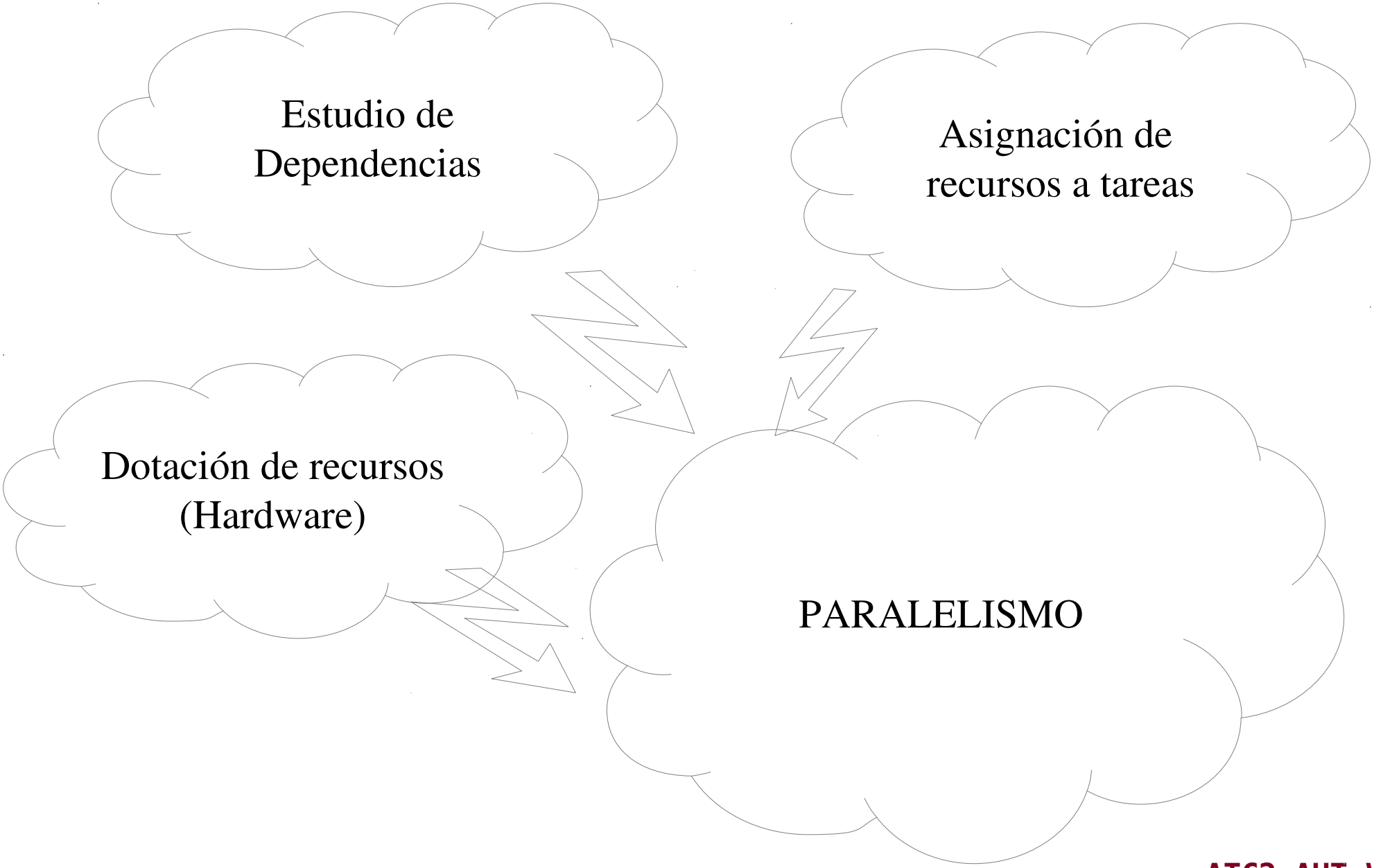
--> Circuitos de Cómputo infrautilizados

¿Y qué alternativas de mejora tenemos?

UAH

Cómo aumentar rendimiento

- Seguir persiguiendo el “techo tecnológico” actual
  - Incremento de la densidad de integración
  - Técnicas de disipación de calor
- Búsqueda de nuevas tecnologías
  - Transistores moleculares
  - Computación cuántica
- Buscar el PARALELISMO
  - “Hacer tareas a la vez”
  - Disponer de Unidades que operen simultáneamente
  - Ver si las tareas son dependientes o no


Estudio de Dependencias
Asignación de recursos a tareas
Dotación de recursos (Hardware)
PARALELISMO

Una definición formal de paralelismo:

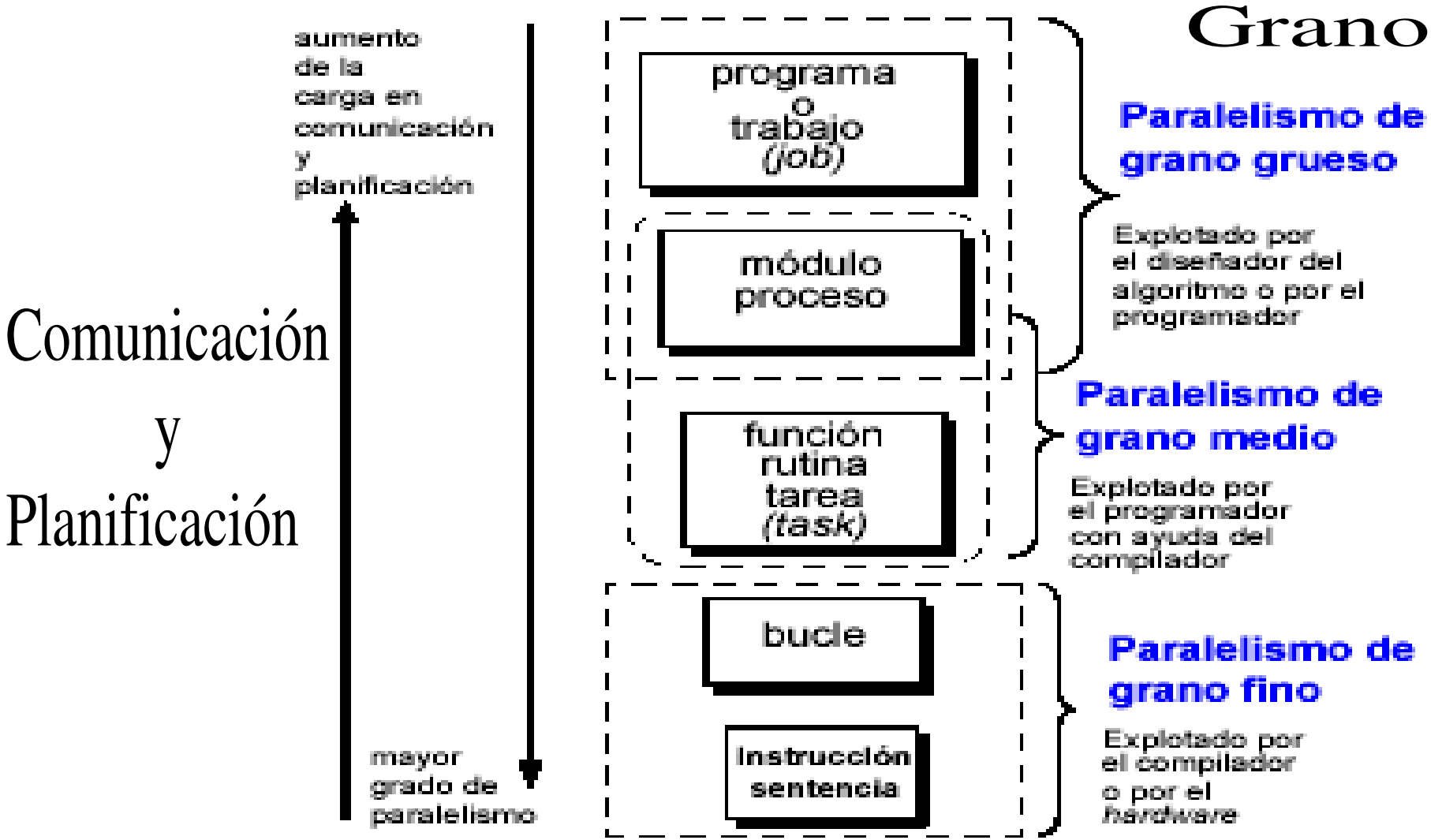
"El procesamiento paralelo es una forma eficaz de procesamiento de información que favorece la explotación de los sucesos concurrentes en el proceso de computación" [Hwang-Briggs]

Conceptos básicos:

Granularidad: tamaño de la tarea computable

Planificación: asignación de los recursos a las tareas

Comunicación: sincronización de tareas independientes


Comunicación
y
Planificación
aumento de la carga en comunicación y planificación
mayor grado de paralelismo
Grano
programa o trabajo (job)
módulo proceso
función rutina tarea (task)
bucle
instrucción sentencia
Paralelismo de grano grueso
Explotado por el diseñador del algoritmo o por el programador
Paralelismo de grano medio
Explotado por el programador con ayuda del compilador
Paralelismo de grano fino
Explotado por el compilador o por el hardware

Tipos de Paralelismo

Paralelismo implícito:

Programas convencionales. Invisible al programador.

El compilador se encarga de “paralelizar” y de asignar los recursos de la máquina

- Ventajas
  - No es necesario reescribir el programa: se aprovecha todo el código secuencial existente
  - El trabajo del programador es fácil (clásico)
- Inconvenientes
  - Gran dependencia del compilador y su nivel de optimización
  - Aprovechamiento del paralelismo de nuevas máquinas y portabilidad del compilador son factores contrapuestos

Tipos de Paralelismo

Paralelismo explícito:

el programador utiliza lenguajes de procesamiento paralelo para desarrollar su sistema

Ventaja

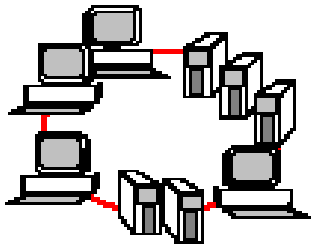
Mejor aprovechamiento del paralelismo de la máquina

Inconveniente

El trabajo del programador es más sofisticado, requiere mayor cualificación

UAH

Tipos de P.: “hardware” de soporte e implementación del paralelismo

nivel			técnicas de implementación
		paralelismo a nivel de procesador	- segmentación - división funcional - procesadores vectoriales
estructuras paralelas		paralelismo en multiprocesadores	- memoria compartida - memoria distribuida
		paralelismo en multicomputadores	- *clusters* - sistemas distribuidos

Taxonomía de Flynn

SISD (Simple Instruction Simple Data): flujo de instrucciones único que trabaja sobre un flujo de datos único (arquitectura clásica, superescalares)

SIMD (Simple Instruction Multiple Data): flujo de instrucciones único que trabaja sobre un flujo de datos múltiple (computadores matriciales)

MISD (Multiple Instruction Simple Data): flujo de instrucciones múltiple que trabaja sobre un flujo de datos único (sin realizaciones prácticas)

MIMD (Multiple Instruction Multiple Data): flujo de instrucciones múltiple que trabaja sobre un flujo de datos múltiple (multiprocesadores)

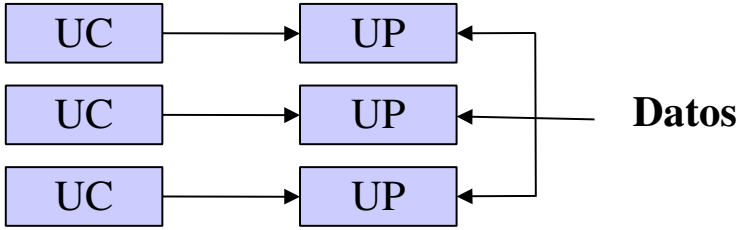
Tipos de Paralelismo – Taxonomía de Flynn

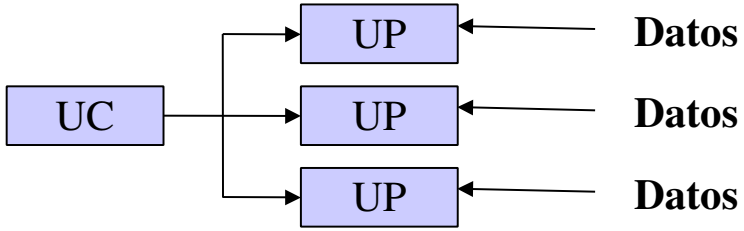
SISD

todas máquinas serie (von Neumann)

MISD

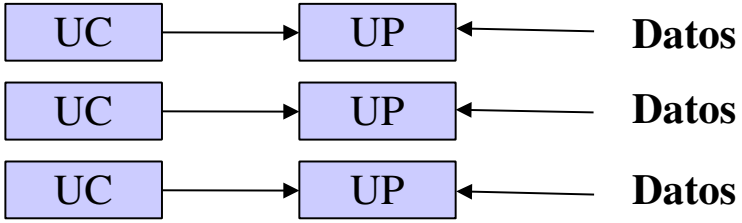
????





SIMD

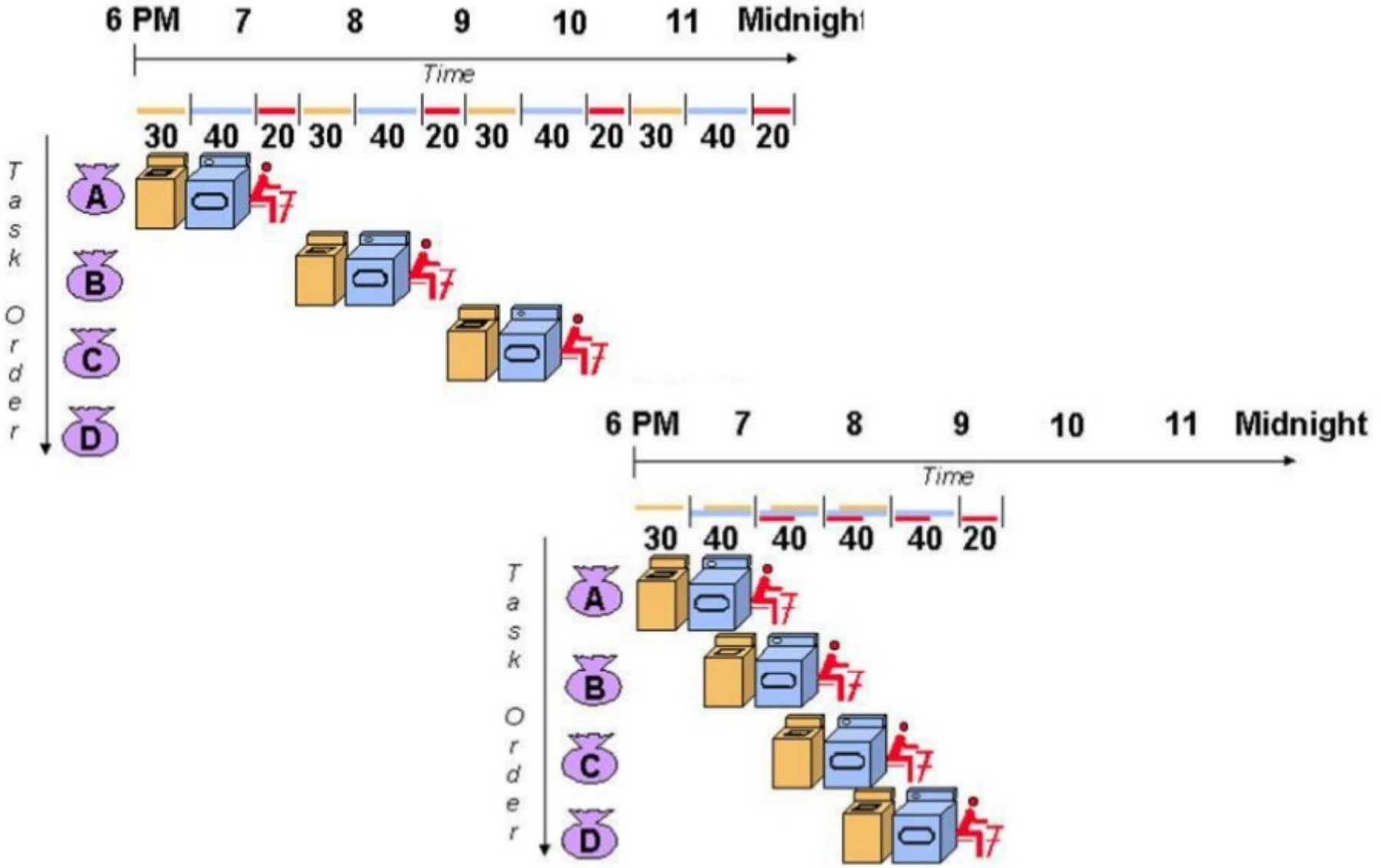
Cray II, Cray Y-MP, ICL-DAP



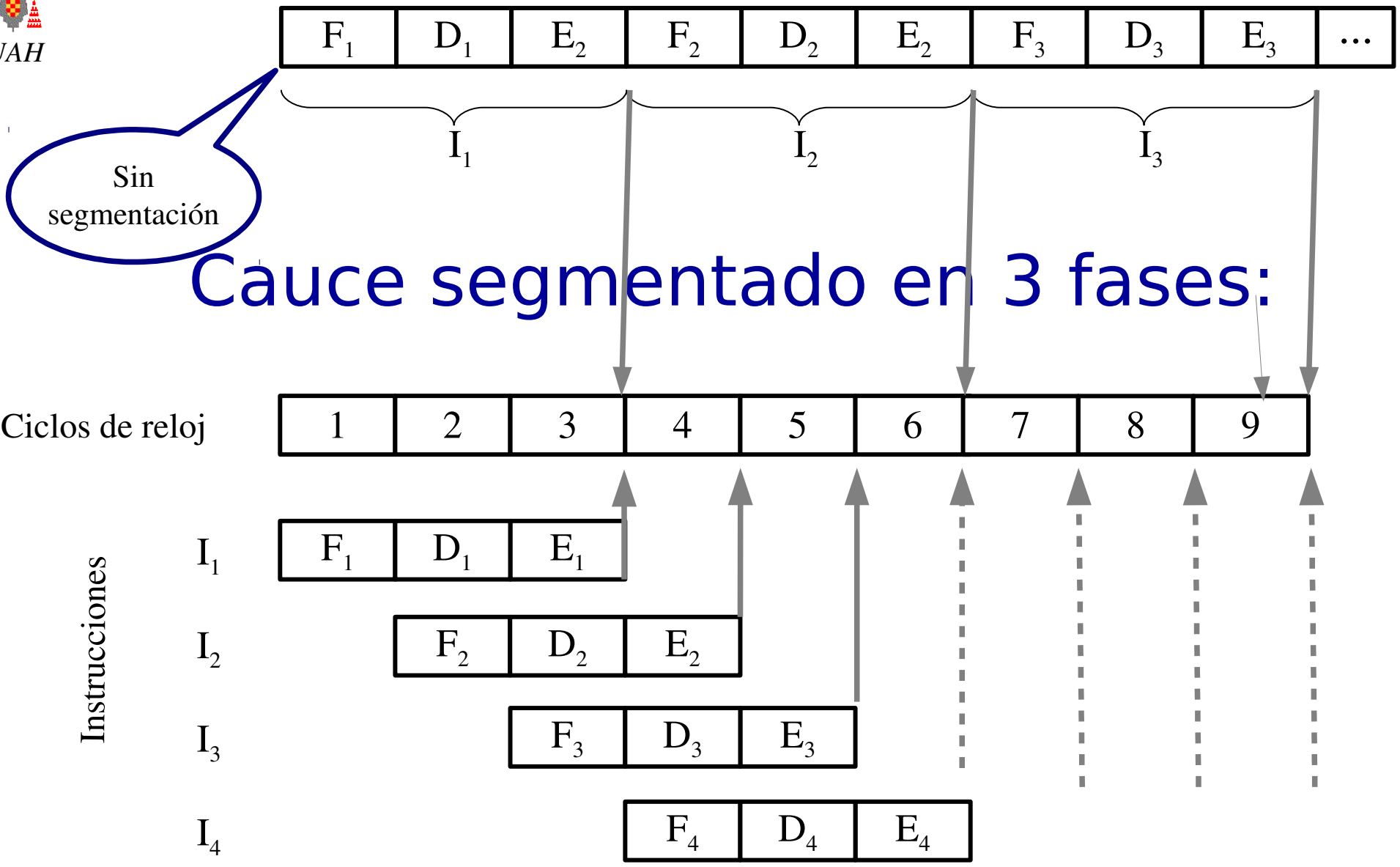
MIMD

Encore, Multimax, nCUBE, Cray T3E, Intel iPSC

SEGMENTACIÓN
Pipelining


6 PM
7
8
9
10
11
Midnight
Time
30 40 20 30 40 20 30 40 20 30 40 20
Task Order
A
B
C
D
6 PM
7
8
9
10
11
Midnight
Time
30 40 40 40 40 20
Task Order
A
B
C
D

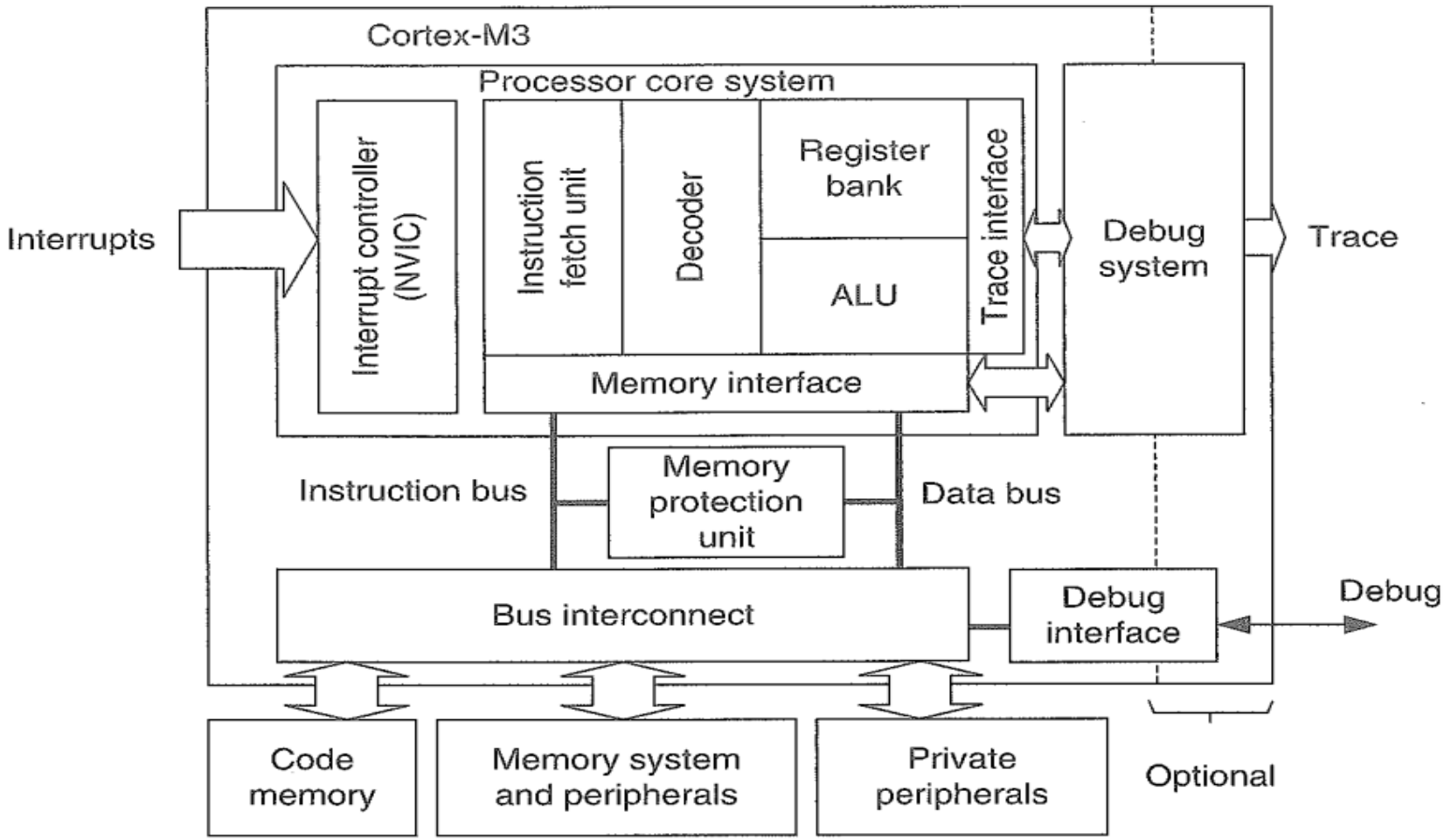
UAH



Cauce segmentado en 3 fases:

UAH

Cortex M3 Pipeline



UAH

Arquitectura Harvard