Procesadores: Arquitecturas y Tecnologías

Andrés Djordjalian <andres@indicart.com.ar>
Indicart Carteles Electrónicos y Facultad de Ingeniería, UBA
Para el Simposio Argentino de Sistemas Embebidos (SASE 2010)
Marzo de 2010

Temario

Una introducción a la arquitectura de computadoras

…orientada hacia micros para sistemas embebidos,

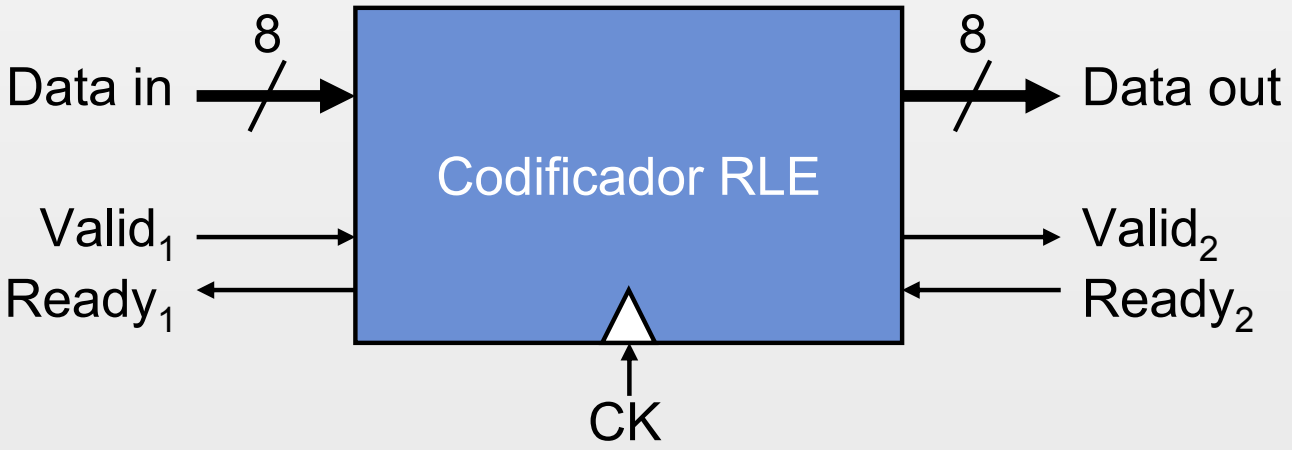
…con algo sobre la historia y los mercados actuales,

…un repaso a las tecnologías de fabricación de sistemas digitales,

…y algunos ejemplos.

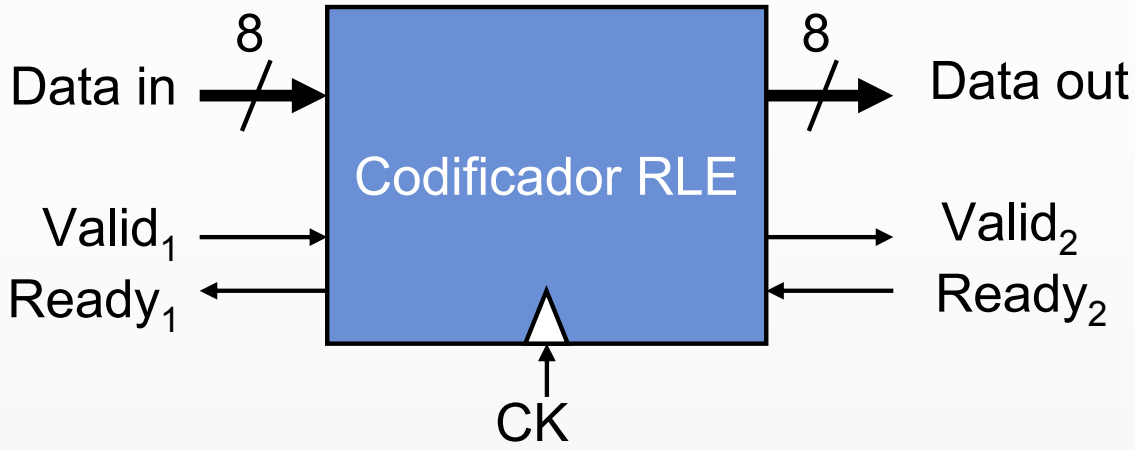
Problema a Resolver

- **Diseñar un circuito que codifique una cadena de bytes a código RLE**
  - Codificar en RLE (*run-length encoding*) significa convertir las secuencias de un mismo número en la tupla:
    <código de escape, número, cantidad de repeticiones>
    - Ej.: **abccbaaaaaba → abccb/a5ba**
  - Es particularmente útil para comprimir gráficos
    - Se usa en el formato PCX



(La interface es como la de una memoria FIFO)

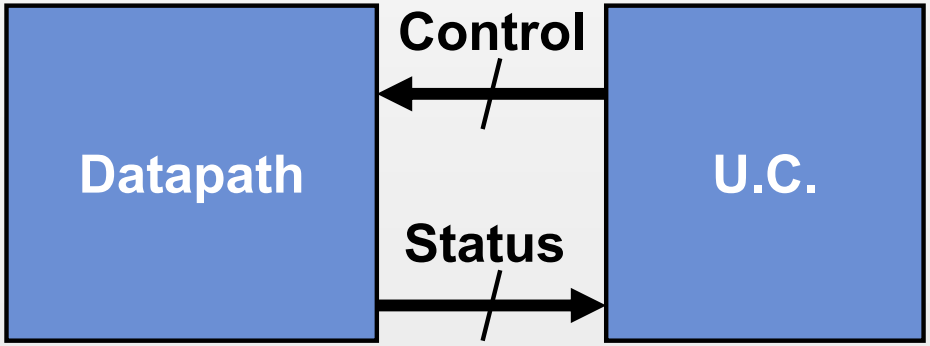
Problema a Resolver



- **¿Va a ser un circuito combinacional o secuencial?**
- **Que sea un secuencial sincrónico**
  - Son más fáciles de diseñar
    - En particular, de validar que la temporización sea correcta
  - Son escalables
  - Por eso, la electrónica digital está orientada hacia sincrónicos
    - Los componentes, el software EDA, etc.
    - Dejemos las técnicas asincrónicas (que son diversas) para casos especiales, más avanzados.
- **Más o menos, ¿qué cantidad de estados necesita tener este secuencial?**

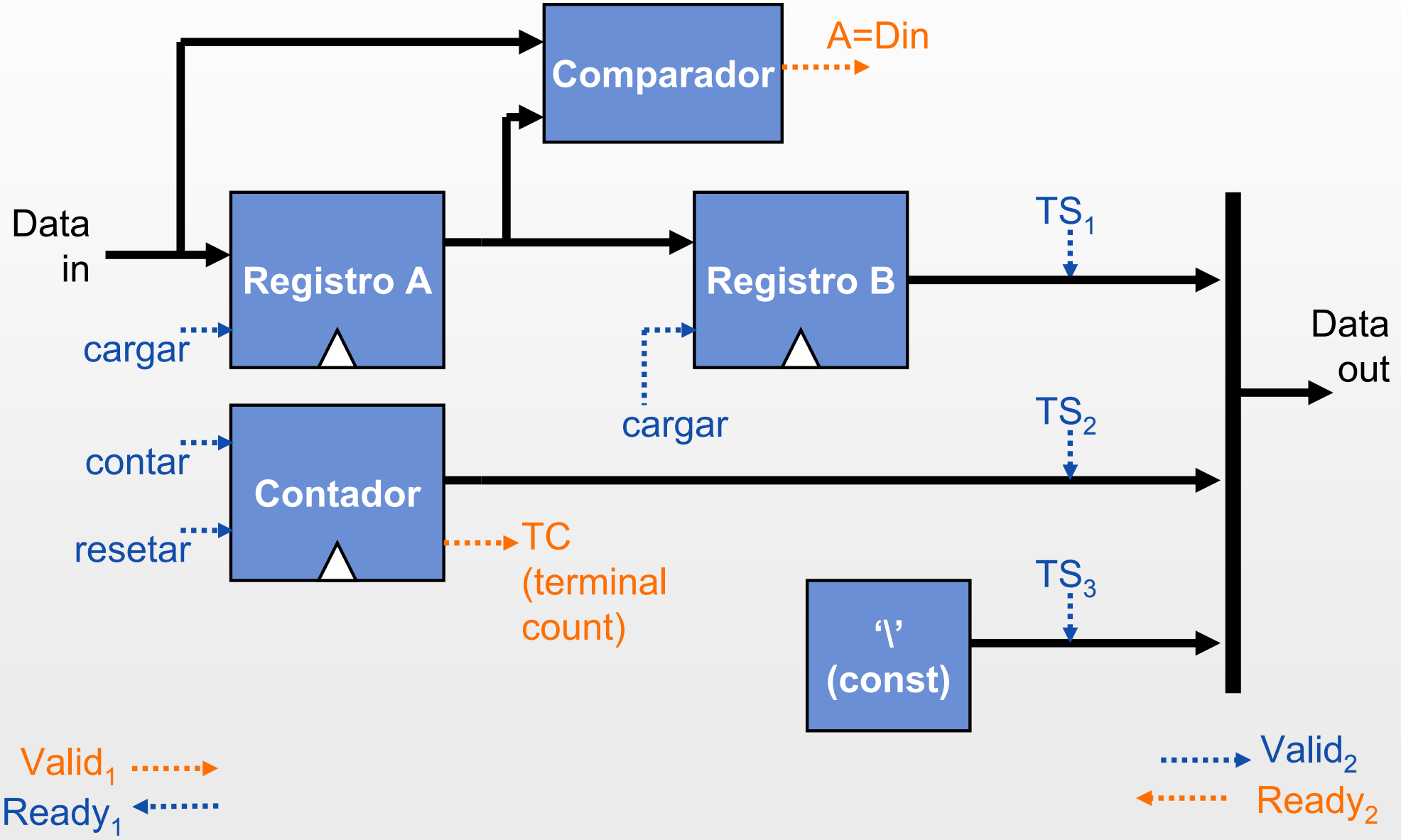
Máquina de estados + Datapath

- **Para encarar estos problemas, se suele separar en dos subsistemas:**
  - Una *ruta de datos* (o ***datapath***), compuesta por los combinacionales y registros que se repiten n (en este caso, 8) veces porque trabajan sobre cada bit.
  - Una máquina de estados (o FSM) que controla el datapath
    - Frecuentemente se le dice ***unidad de control***.
    - (FSM = *Finite-State Machine)*



- **Implementándolo así, la FSM de un codificador RLE necesita no más que un número manejable de estados**

Datapath para un codificador RLE



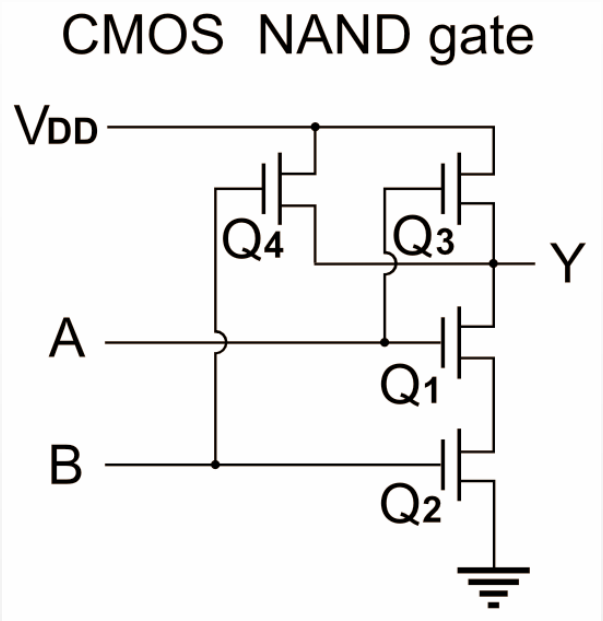
Cómo no implimentaríamos hoy ese codificador

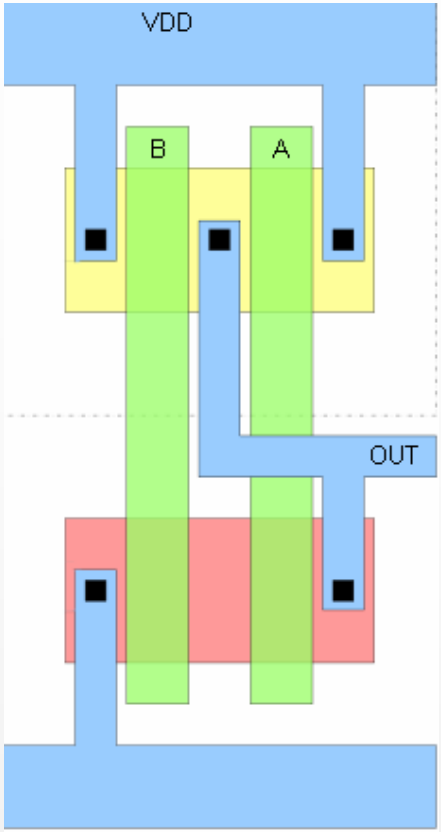
- **Difícilmente nos interese implementarlo con componentes MSI…**
  - Componentes MSI son CIs de compuertas, contadores, etc., por ejemplo de series 74LSxx, 74HCxx, 74Fxx, CD4xxx, etc.
- **…porque no sería tan económico como las alternativas**
  - Debido, entre otros factores, a los costos de interconexión
- **…y porque no sería tan rápido y/o de bajo consumo como las alternativas**
  - Debido, también, a las interconexiones
    - Porque salir de un integrado y entrar en otro implica que hayan capacitancias parásitas
      - Estas bajan la velocidad y aumentan el consumo
  - Si no necesitáramos velocidad y/o bajo consumo, seguramente preferiríamos usar un micro

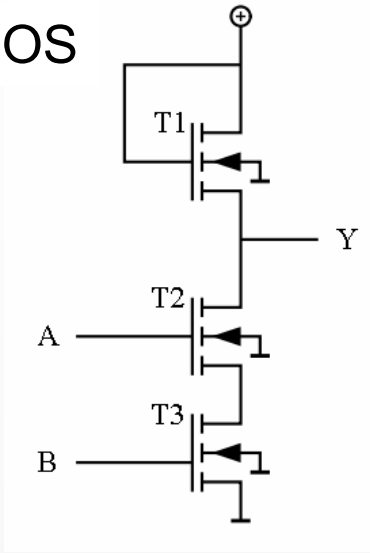
Cómo sí implimentaríamos hoy ese codificador

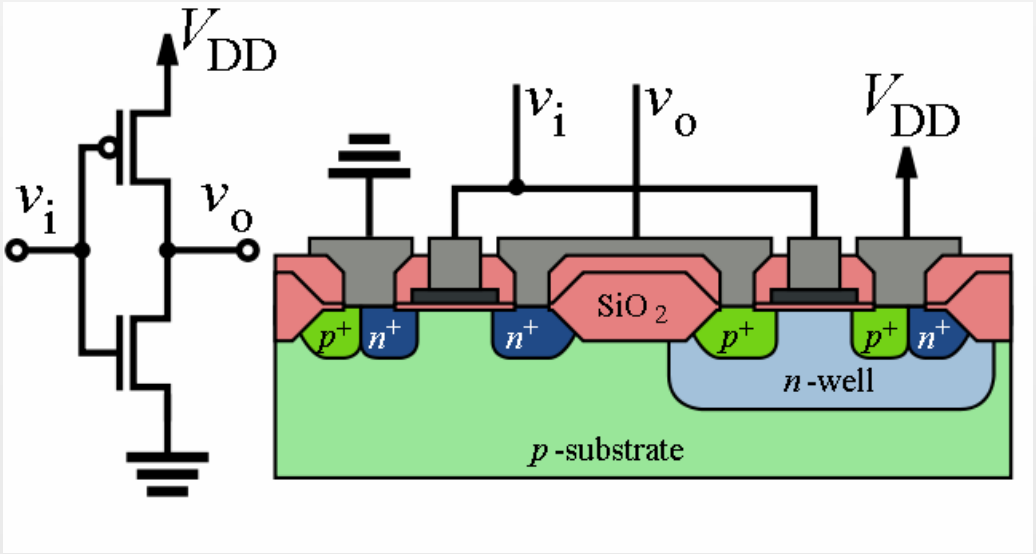
- **Usaríamos una FPGA o un CPLD**
  - FPGAs = *Field-Programmable Gate Array*
    - (o "arreglo de compuertas programable 'in situ'")
  - CPLDs = *Complex Programmable Logic Device*
    - (o "dispositivo lógico programable complejo")
- **…o lo implementaríamos en un circuito integrado *full custom CMOS***
  - *Full custom* = que se diseña todo "a medida"
  - CMOS = Es la tecnología de implementación de CI más utilizada en la electrónica digital
    - …gracias a su alta densidad y bajo consumo
- **…o usaríamos un *Structured ASIC* o un *Gate Array***
  - Esto es un término medio entre las dos opciones anteriores
  - Significa fabricar un circuito integrado (generalmente CMOS), pero haciendo "a medida" únicamente algunas capas de metal (o sea, interconexión), utilizando el resto prefabricado
    - Las capas prefabricadas pueden proveer un arreglo de compuertas o celdas más complejas

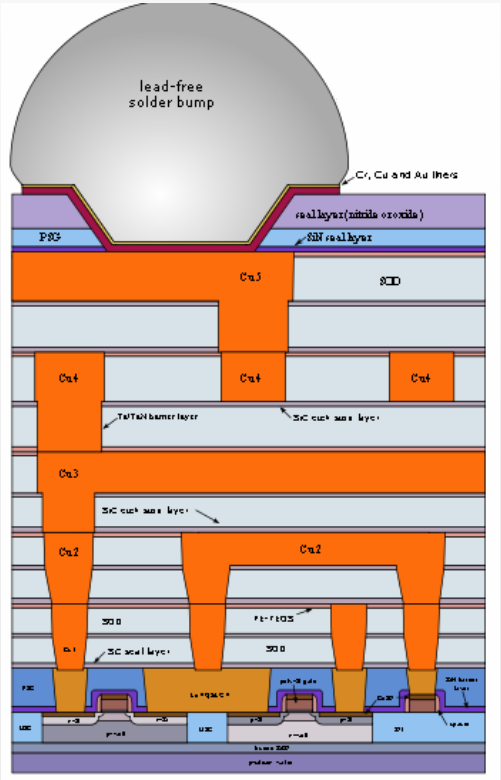
CMOS







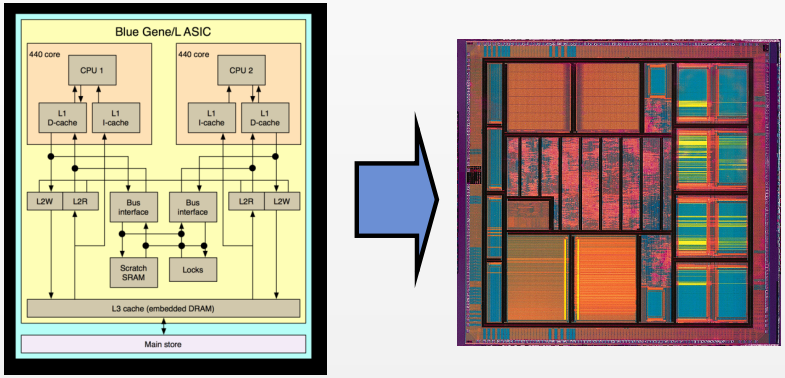




Fuente: [Wikimedia Commons]

Sistema en un Chip (SoC)

- **En un chip, se pueden conectar entre sí distintos bloques prediseñados, como si fueran componentes que se interconectan en un circuito impreso**
  - A esos bloques se los llama ***cores*** (núcleos) o **IP** (*intellectual-property,* o propiedad intelectual)
- **Se forma así un SoC (*System on Chip*)**
- **El la manera típica de diseñar un ASIC complejo**
  - ASIC = *Application-Specific IC*

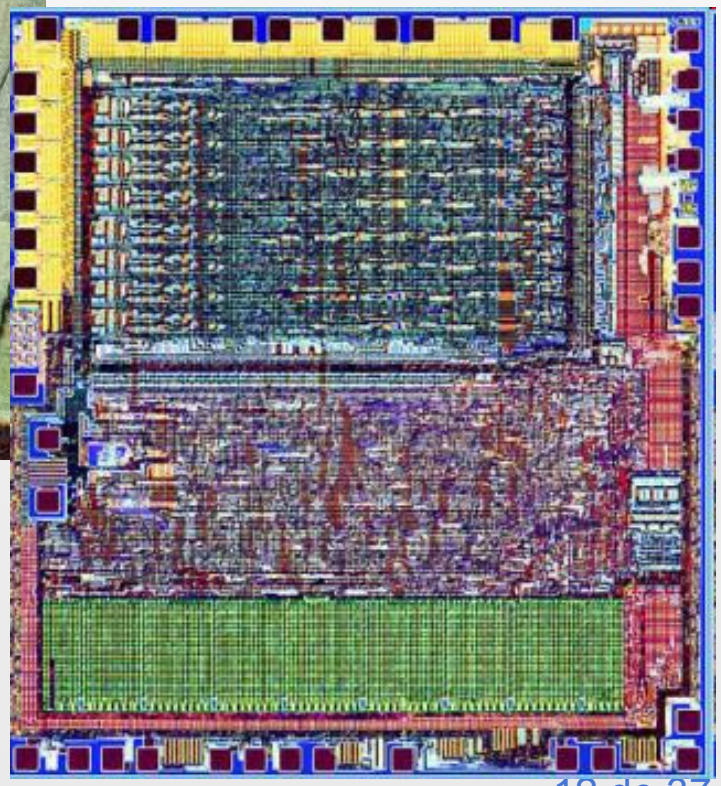


- **Se puede hacer lo mismo con una FPGA**
  - …y se le dice **PSoC** = Programmable System on a Chip
  - Para facilitarlo, algunas FPGA traen (fijo) un procesador
- **La integración normalmente se hace mediante:**
  - Un lenguaje de descripción de hardware (ej. VHDL, Verilog)
  - O una herramienta gráfica
- **Un *core* puede ser un procesador**

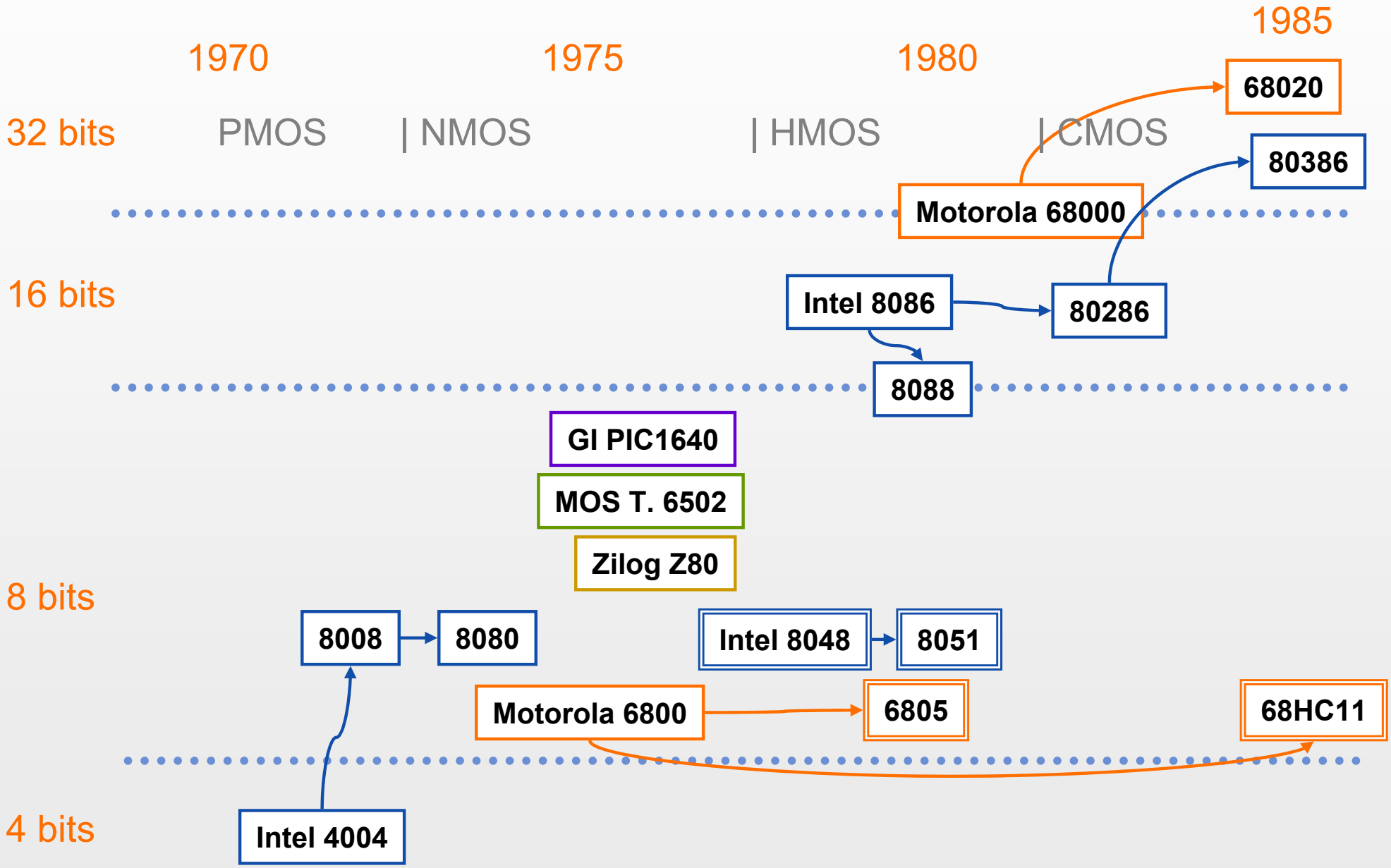
Procesadores

- **Lo explicado hasta ahora puede usarse para diseñar y fabricar procesadores**
- **En ellos, la unidad de control puede ser compleja**
  - En particular si las instrucciones nativas (o sea en Assembly) realizan operaciones complicadas o requieren varios ciclos de máquina
    - Esa era la tendencia hasta los ochentas
- **Estas UCs podían demandar bastante tiempo de diseño y ocupar buena parte del silicio de los primeros microprocesadores**
- **Frecuentemente, para implementar UCs complejas sin perder flexibilidad, se usaba una técnica llamada microprogramación**
  - Consistía en guardar el comportamiento de la UC en una ROM, como si fuera un programa, que era "ejecutado" por un circuito secuencial sencillo y genérico.

MOS Technology 6502 (año 1975)



Algunos de los Primeros Micros



CMP
United Business Media

Consideration of 8-bit chip families

Vendor	2006 %	2005 %	Vendor	2006 %	2005 %
Atmel AVR	32	33	Rabbit 2000, 3000	11	16
Microchip PIC 18	30	31	Dallas/Maxim 80xx	10	15
Freescale HC05, HC08, HC11	27	25	Philips P80x, P87x, P89x	10	11
Microchip PIC 14/16	26	28	Renesas	10	11
Intel 80xx, '251	21	22	Cypress PSoC	9	11
Microchip PIC 10/12	17	21	Cygnal/SiLabs 80xx	8	11
Zilog Z8, Z80, Z180, eZ80	16	19	NEC K0	5	5
Microchip PIC 17	15	18	Infineon C500	3	6
Atmel 80xx	13	16	National COP8	3	4
TI TMS370, 7000	13	14	Ubicom SX	1	5
STMicro ST6, ST7, ST8	12	11	Other	2	3
Xilinx PicoBlaze	12	14			

EETIMES
Embedded Systems Design

N = 662

53. Which of the following 8-bit chip families would you consider for your next embedded project?

Embedded Systems Conference Silicon Valley

Arquitectura de Computadoras

- **Paremos un minuto para definir algunas cosas:**
- **"Arquitectura" es la descripción de un sistema en un nivel de abstracción alto**
  - Ej., mediante un diagrama de bloques
  - Ej., "**arquitectura de computadoras**", de la cual hay dos tipos:

1) **"Arquitectura de Conjunto de Instrucciones"**
   - *Instruction-Set Architecture,* o **ISA**
   - Es la arquitectura "vista de afuera"
     - Registros, conjunto de instrucciones, esquema de interrupciones, etc.
2) **"Microarquitectura"**
   - Es la arquitectura interna del procesador
     - O sea, cómo está implementada la ISA
   - Por ejemplo, diagramas del datapath del procesador

Tipos de ISA

	Acumulador	Registro-Memoria	*Load-Store*

Código para:
`foo=bar+baz`
	`Load bar`
	`Add baz`
	`Store foo`
		`Load R1,bar`
		`Add R1,baz`
		`Store foo,R1`
			`Load R1,bar`
			`Load R2,baz`
			`Add R3,R1,R2`
			`Store foo,R3`

Algunos de los micros con ISA de este tipo	Prácticamente todos los de 8 bits	Intel 8086, Motorola 68000	Los de 32 bits modernos

Instrucciones en el 8086 y 68000

- **Las dos arquitecturas son register-memory de 2 operandos (como máximo)**
  - No pueden haber dos accesos a memoria en una instrucción.
- **8086:** `ADC AX,baz`
  68000: `ADD.W baz,D0`
- **En el 68000 original, la anterior tardaba 8 ciclos (!)**
  - Suponiendo que la memoria respondía sin esperas
  - Un simple `MOVE.B D0,D1` tardaba **4 ciclos** (!)
- **Tienen muchos modos de direccionamiento**
  - 12 en el 8086, 8 en el 68000
  - Incluyendo cosas como:

```
CMPM.B    (A1)+,(A2)+
CMPA.L    100H(A2,D0.W),A3
```

- **Multiplicación**

```
MULU   foo,D1
```

...70 ciclos (!)

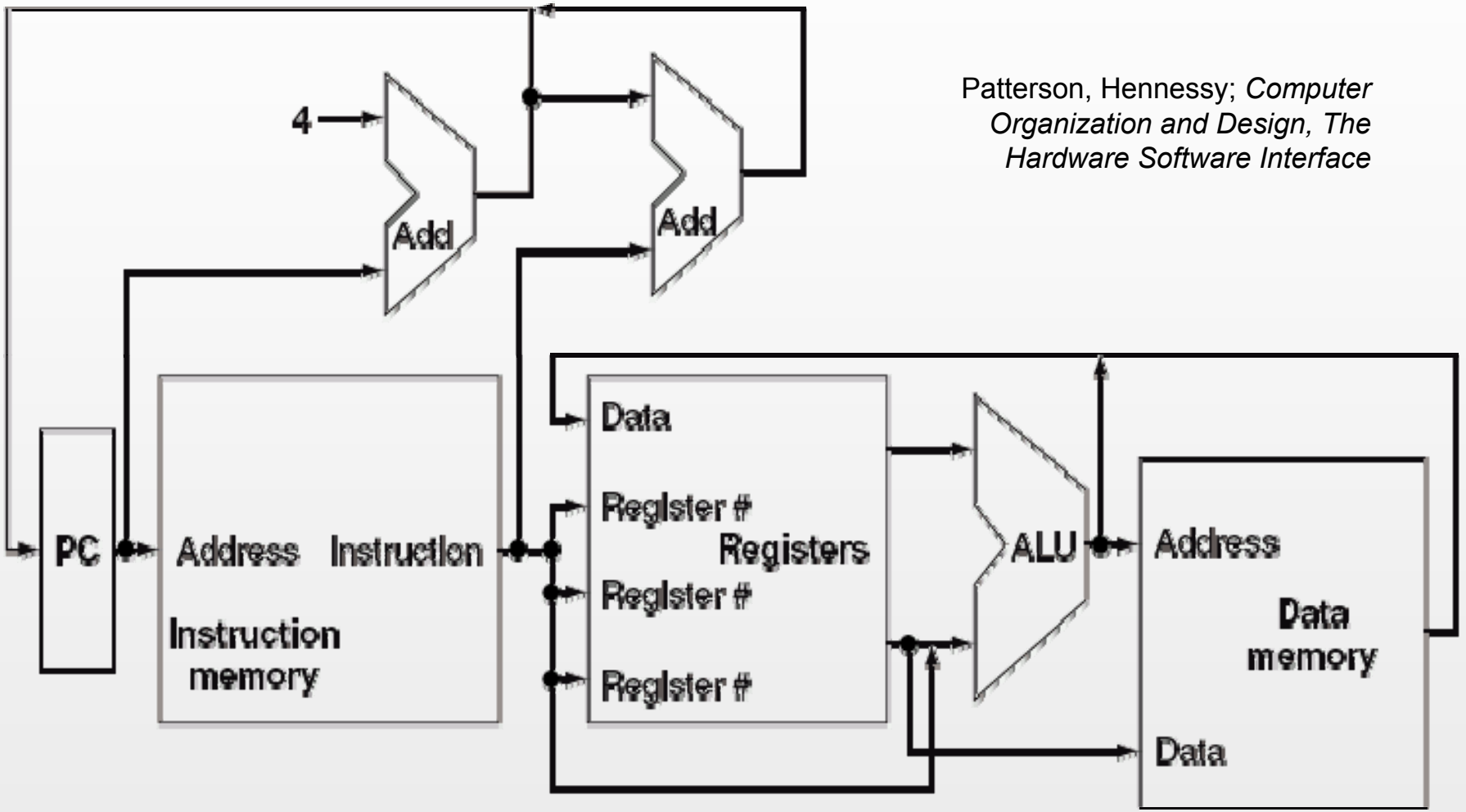
Reduced-Instr.-Set Computer (RISC)

- **En los 70s, en IBM primero, y en las universidades de Stanford y Berkeley después, se empezó a cuestionar esta tendencia de sumarle complejidad a los conjuntos de instrucciones**
- **Sus argumentos:**
  - Gracias a los compiladores, las instrucciones sofisticadas dejan de ser útiles
  - Eliminándolas, se puede optimizar las que sí lo son
  - Se ahorra mucho tiempo de diseño
  - Se libera superficie de silicio ocupada por la unidad de control, la que puede usarse para mejorar el datapath
    - Por ej, para ponerle un mejor multiplicador
- **Nace el estilo RISC para el diseño de arquitecturas de computadoras**
  - …que, desde entonces, tiene enorme influencia en la disciplina

El Estilo RISC

- **Características típicas (no excluyentes)**
  - Arquitectura tipo **load-store**
    - Las operaciones lógicas y aritméticas operan sólo sobre registros. Las únicas instrucciones que acceden a memoria son para transferir su contenido desde o hacia un registro
  - Conjunto de instrucciones **reducido**
  - Las intrucciones tienen **largo uniforme** (ej. 32 bits), y todas pueden ser ejecutadas en **1 ciclo**
    - …suponiendo que la memoria responde lo suficientemente rápido
- **Se habla de "arquitecturas RISC" versus "arquitecturas CISC"**
  - La mayoría de las arquitecturas planteadas desde entonces son RISC
  - En muchas de las que no (ej. PC), se traducen internamente las instrucciones CISC a un código RISC que es ejecutado por un núcleo RISC
- **Una de las ventajas de RISC es que es óptimo para *pipelining***
  - Pero primero veamos un típico datapath RISC

Datapath RISC



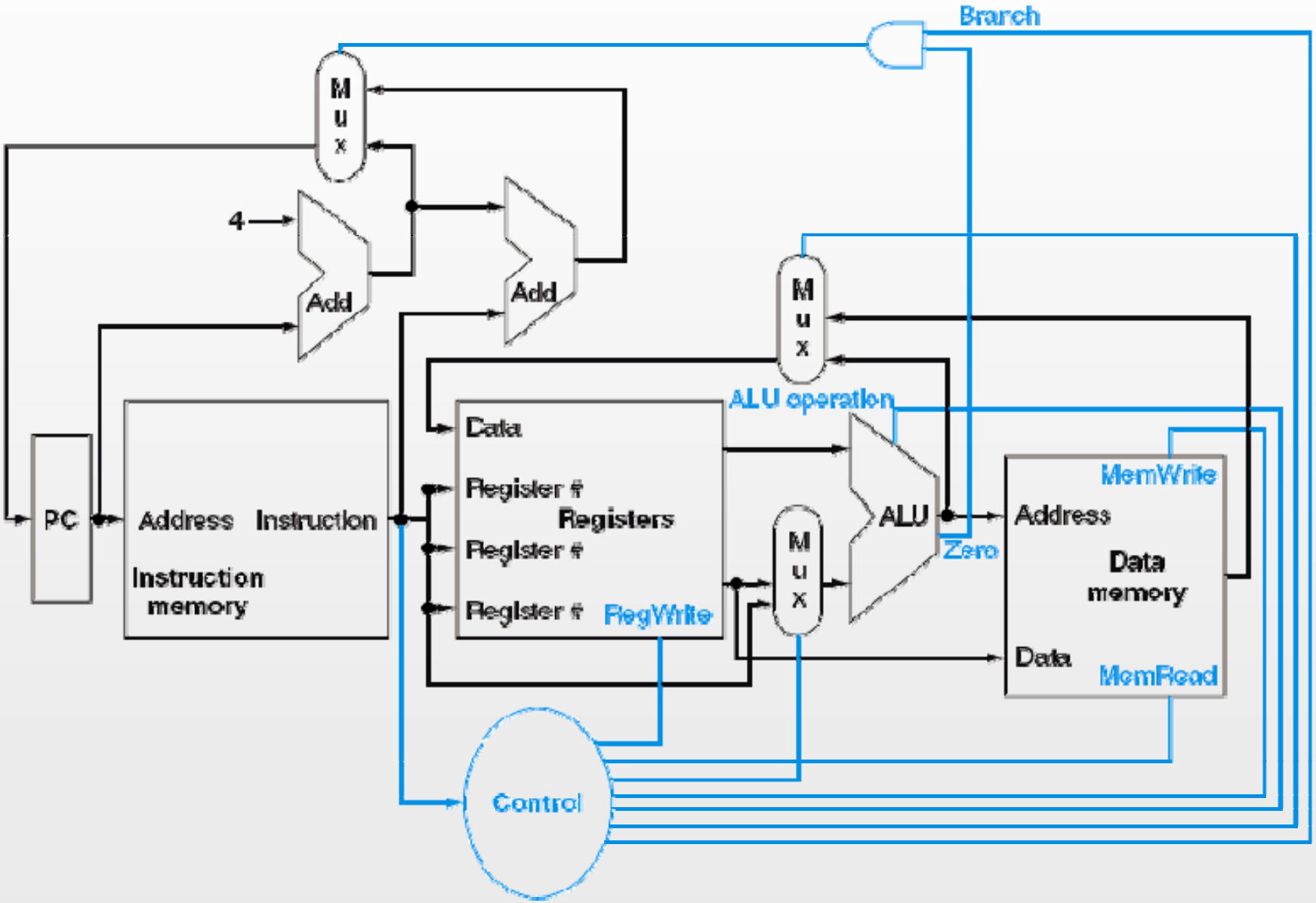
Patterson, Hennessy; *Computer Organization and Design, The Hardware Software Interface*

Ejemplo de una Instrucción con 3 Operandos:

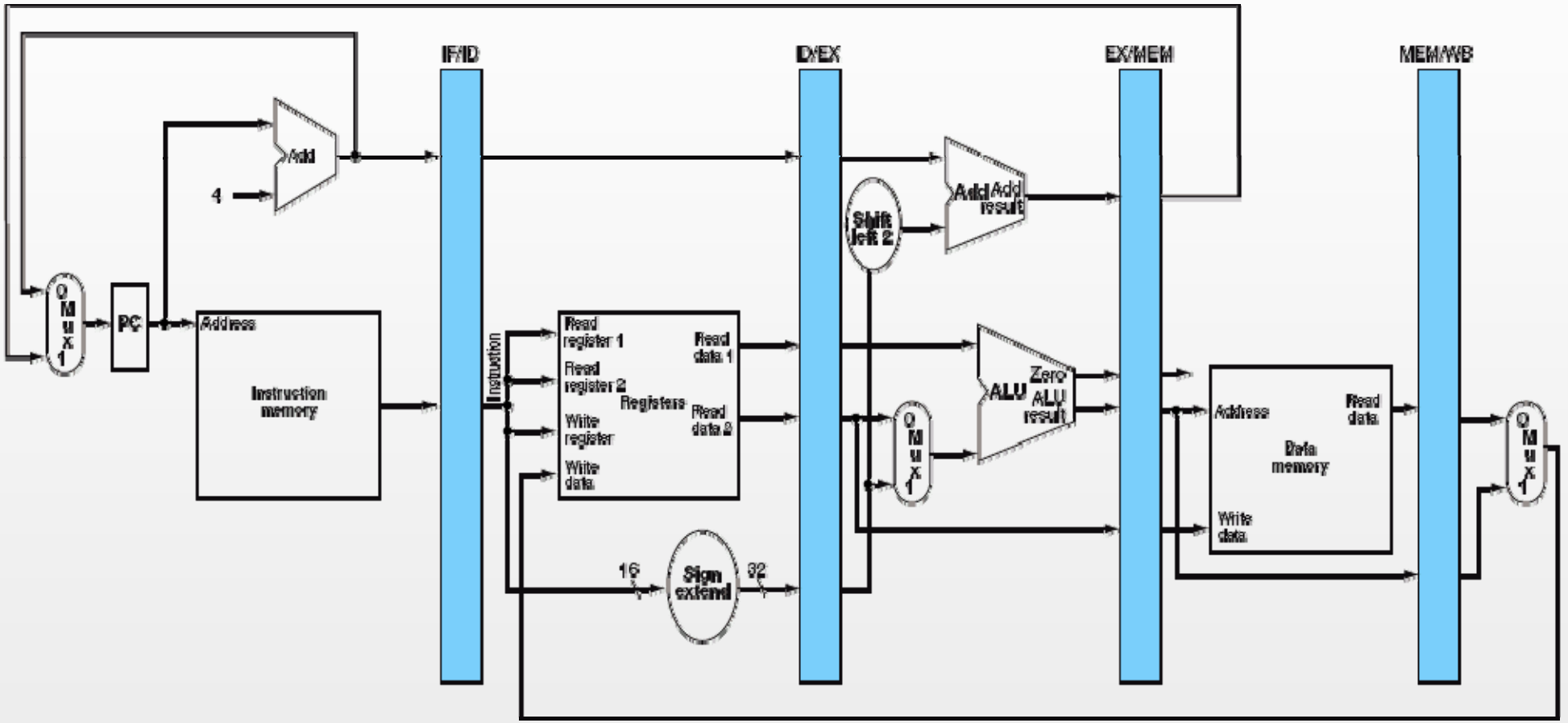
01001 … 010	0	0	0	0	1	1	0	0	0	1	0	1	0	0	0	1	1	0	0 … 1

Opcode, etc. **Reg. destino** **Registros fuente**

Datapath RISC + Unidad de Control



Segmentación (*Pipelining*)



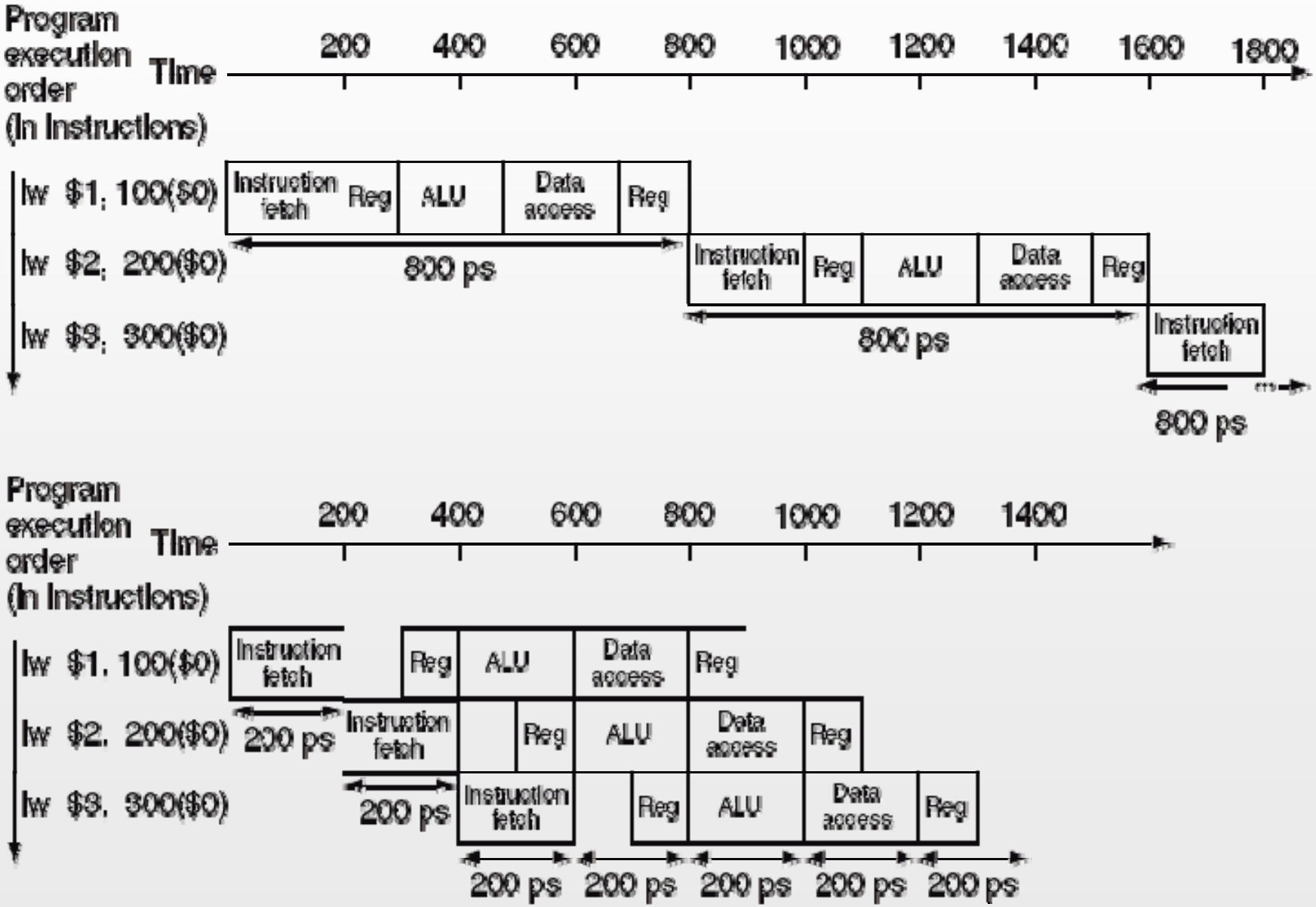
Ejemplos de código Assembly:

```
LDR R1,[R0]
LDR R3,[R2]
ADD R1,R1,#200
```

```
CMP R5,R6
BEQ SonIguales
ADD R5,R5,R4
```

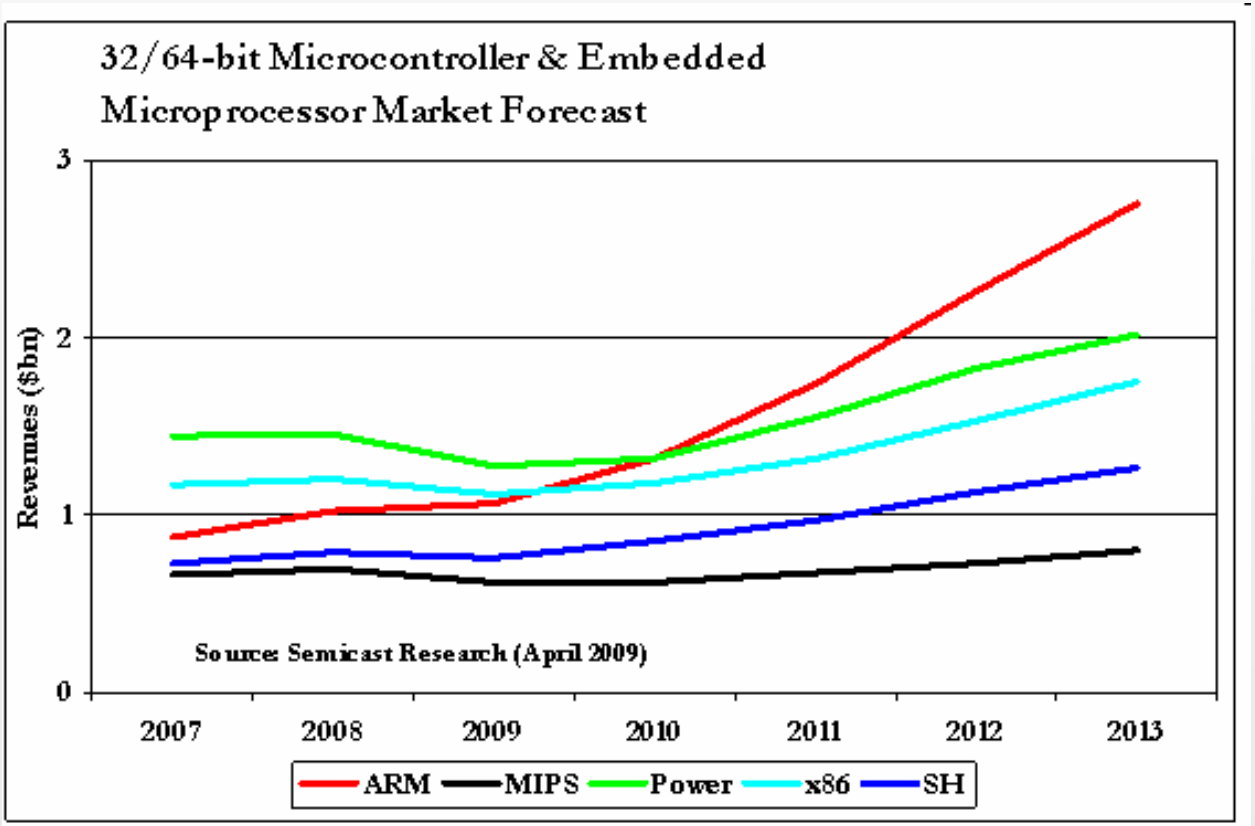
```
ADD R5,R6,R7
SUB R4,R4,R5
```

Segmentación (*Pipelining*)



ARM

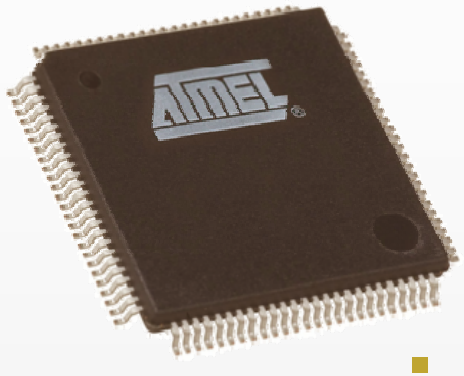
- Acorn era "la Apple británica". En los 80s creó su propio procesador RISC para sus computadoras
- Acorn, Apple y VLSI Technology forman ARM en 1990
- El modelo de negocios es ofrecer IP, para que licencien:
  - Fabricantes de **microcontroladores** como NXP, Atmel, ST, TI, etc.
  - O quienes necesitan micros en sus ASICs
    - Ej., fabricantes de **teléfonos celulares**



(Semicast Research)

Dos Núcleos (*Cores*) ARM

- **ARM 7TDMI**



  - Arquitectura RISC
    - 32 bits
    - Lanzado en 1995
    - Pipeline de 3 etapas
    - ISA ARM (32 bits) y Thumb (16 bits)
    - 0,95 DMIPS/MHz (con el set ARM)
    - 0,28 mW/MHz (con proceso TSMC 0.18G)

- **ARM Cortex M3**

  - Arquitectura RISC
    - 32 bits
    - Lanzado en 2004
    - Pipeline de 3 etapas con *branch speculation*
    - ISA Thumb 2 (16/32 bits)
    - 1,25 DMIPS/MHz
    - 0,19 mW/MHz (con proceso TSMC 0.18G)

- **Un 8051 procesa a unos 0,1 DMIPS/MHz**

CMP
United Business Media

2006 State of Embedded Market Survey

Consideration of 32-bit chip families

Vendor	2006 %	2005 %	Vendor	2006 %	2005 %
Intel '386, '486, Pentium, Celeron	31	34	AMD Alchemy (MIPS)	9	8
AMD '386, '486, Athlon, Opteron, Geode	24	20	Cirrus Logic EP73xx, EP93xx (ARM)	8	-
Intel PXA, IXP, XScale (ARM)	21	18	Actel ProASIC 3 (ARM)	7	-
Atmel AT91xx (ARM)	20	-	Renesas SuperH, H8SX, M32C, M32R	7	9
Freescale PowerPC 5xx, 6xx,	20	16	STMicro ST20	7	7
Freescale PowerPC 7xx, 8xx	19	20	AMCC PowerPC 4xx	5	7
Freescale 68K, ColdFire	19	23	SPARC (any)	4	5
IBM PowerPC 4xx, 7xx	19	23	Transmeta (x86)	4	6
Xilinx MicroBlaze, PowerPC 405	18	19	Fujitsu FR series	3	4
Freescale PowerQUICC	16	16	IDT 32xxx	2	-
Freescale DragonBall MX (ARM)	13	15	P.A. Semi 13xx, 16xx (PowerPC)	2	-
Altera Nios, Nios II	10	13	Other	5	6
Intel Itanium	10	9			

- Indicates not asked in 2005

EETIMES
Embedded Systems Design

55. Which of the following 32-bit chip families would you consider for your next embedded project?

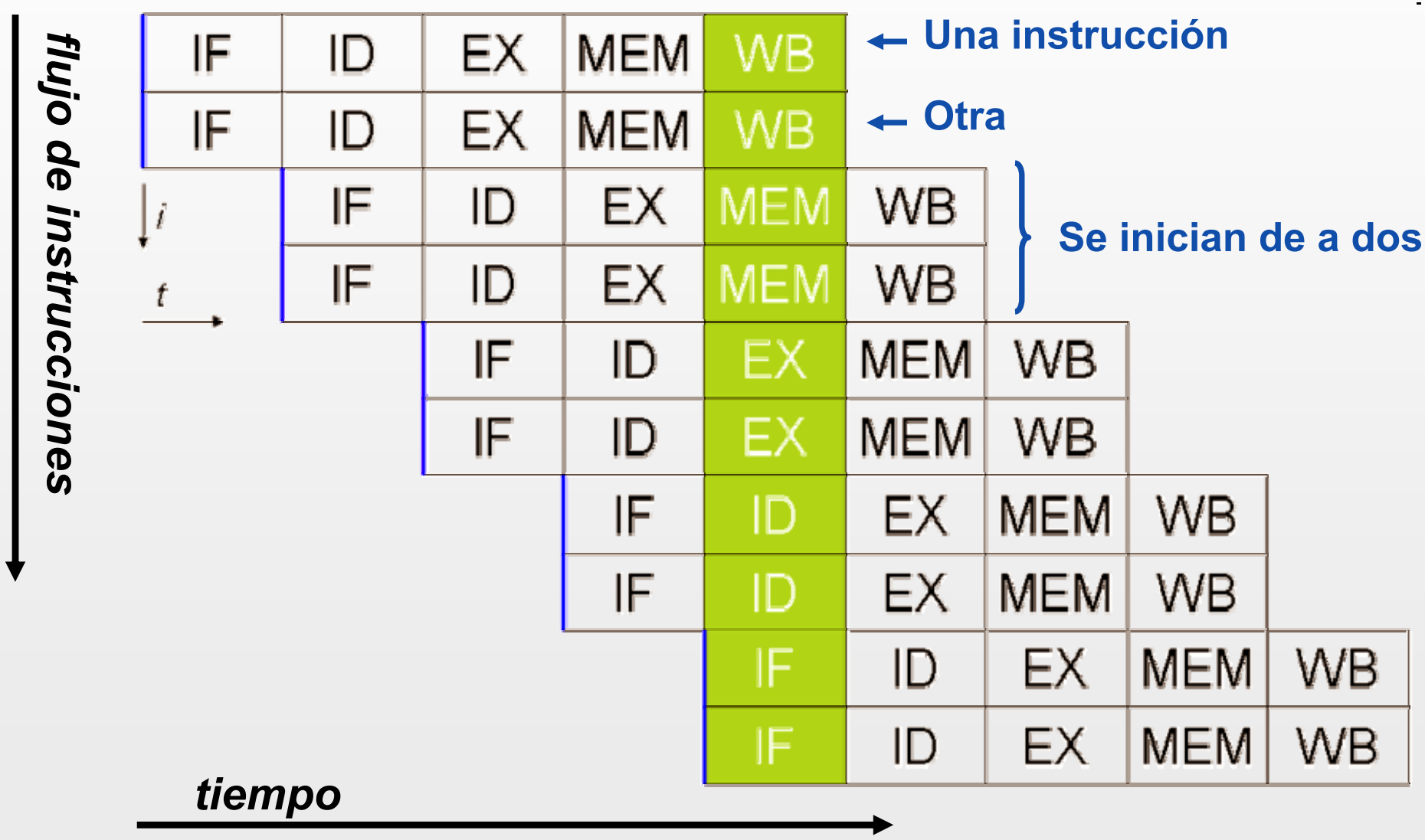
N = 715

Embedded Systems CONFERENCE SILICON VALLEY

Multiple Inicio de Instrucciones

- **Como vimos, un pipeline procesa varias instrucciones al mismo tiempo, pero en cada ciclo sólo se inicia una**
- **Pero un procesador puede tener "n" pipelines, para iniciar "n" instrucciones por ciclo y lograr así más paralelismo**
  - Los pipelines puede ser iguales o no
    - Ej., uno para instrucciones aritméticas, otro para *loads* y *stores*
    - Si el procesador puede iniciar "n" a la vez, se le dice ***n-issue***.
- **Para esto, hay dos tipos de arquitectura, que difieren en dónde se decide qué instrucciones se van a iniciar en paralelo:**
  - **Superescalar: Lo decide el procesador**, en tiempo de ejecución
    - Se usa, por ej., en los procesadores de PC desde el Pentium, y en procesadores para embebidos de alta gama
  - **VLIW** (Very-Long Instruction Word): **Lo decide el compilador o el programador**
    - Se usa, por ej., en los DSP de la serie C6x de Texas Instruments

Superescalares y VLIW



Memorias Cache

- **La tecnología de las memorias no experimenta tanto incremento de velocidad como la de los procesadores**
  - A frecuencias altas, la velocidad de procesamiento empieza a estar dominada por las esperas hasta que responda la memoria.
- **Para alivianar el problema, a veces se usan memorias cache**
  - Son memorias rápidas, puestas entre la memoria principal y el procesador.
  - Conservan un subconjunto del contenido de la memoria principal
    - Gracias a la **localidad espacial y temporal del software**, con ese contenido se atiende la mayoría de los accesos que requiere el procesador.
- **Frecuentemente, se usan varios niveles de cache**
  - Un cache de nivel 1 (L1) pequeño y bien rápido
  - Un cache de nivel 2 (L2) más grande, aunque más lento
  - Quizás, un L3…
  - Las cache L1 y L2 suelen estar en el mismo chip del procesador

Memorias Cache

- **El cache L1 generalmente está dividido en cache de instrucciones y cache de datos**
  - Así, segmentos como IF y MEM del pipeline no compiten por el acceso a memoria.
- **Las memorias cache tienen sus contras:**
  - Complican la predicción del tiempo de ejecución
    - Porque no se sabe si la información va a estar o no en cache.
  - Ocupan superficie de silicio (=costo)
  - Consumen energía
- **Sin embargo, si la frecuencia de clock es alta, pueden ser imprescindibles para lograr buena performance.**

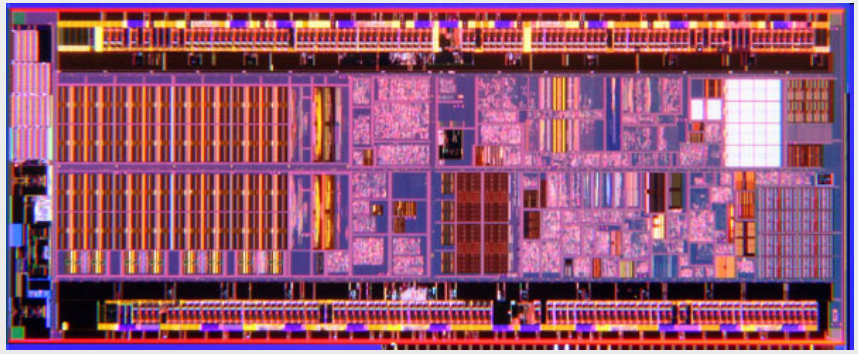
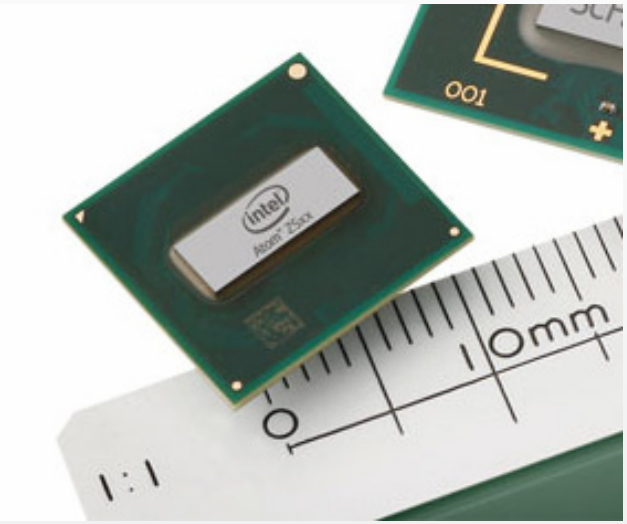


Foto del *die* (pastilla) del Intel Atom

Intel Atom

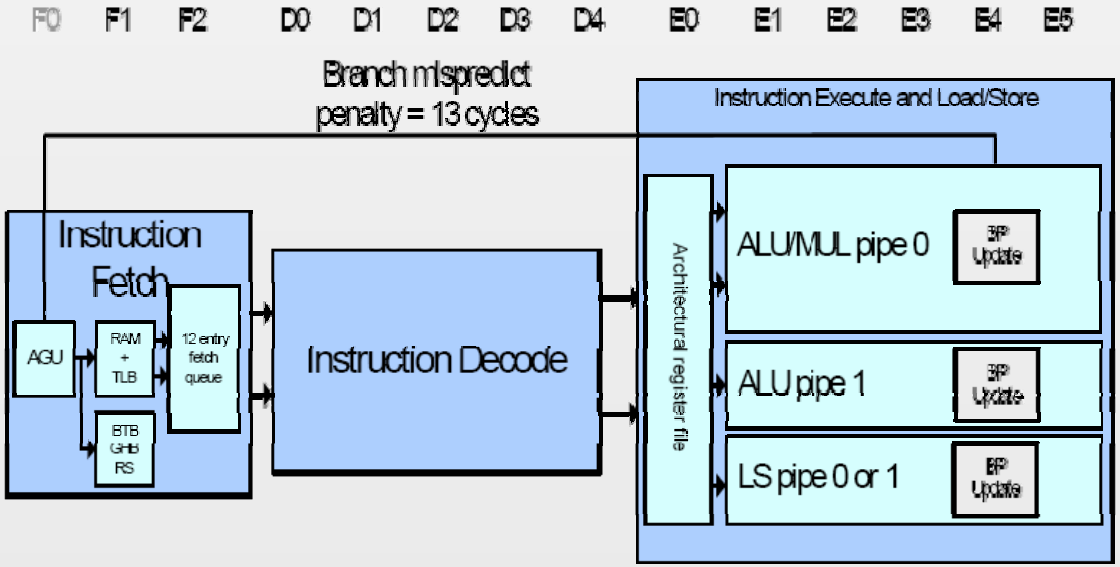
Z530



- 32 bits
- ISA x86 (PC compatible) pero con núcleo RISC
- 1,6 GHz
- Superescalar (*2 issue*)
- Lanzado en 2008
- Pipelines de 16 etapas
- Caches L1: split, 32 KB (I) + 24 KB (D)
- Cache L2: 512 KB
- Extensión para multimedia: SSE3
- Máx TDP (*thermal design power*): 2 W
- Vcore = 1,1 V
- 2 DMIPS/MHz
- Fabricado con un proceso de 45 nm
- Cuesta aprox. u$s 50

ARM Cortex A8

- 32 bits
- Arquitectura RISC
- ISA ARM (32 bits) y Thumb-2 (16/32 bits)
- desde 600 MHz hasta más de 1 GHz
- Superescalar (*2 issue*)
- Lanzado en 2009
- Pipelines de 13 etapas
- Caches L1: split, 16KB o 32 KB c/u
- Cache L2: 64 KB a 2 MB
- 2 DMIPS/MHz
- Extensión para multimedia: NEON
- Por unos u$s 50 se compra un TI OMAP 3515 (= Cortex A8 a 600 MHz + un DSP)



¿Preguntas?

Procesadores: Arquitecturas y Tecnologías

Andrés Djordjalian <andres@indicart.com.ar>
Indicart Carteles Electrónicos y Facultad de Ingeniería, UBA

Para el Simposio Argentino de Sistemas Embebidos (SASE 2010)

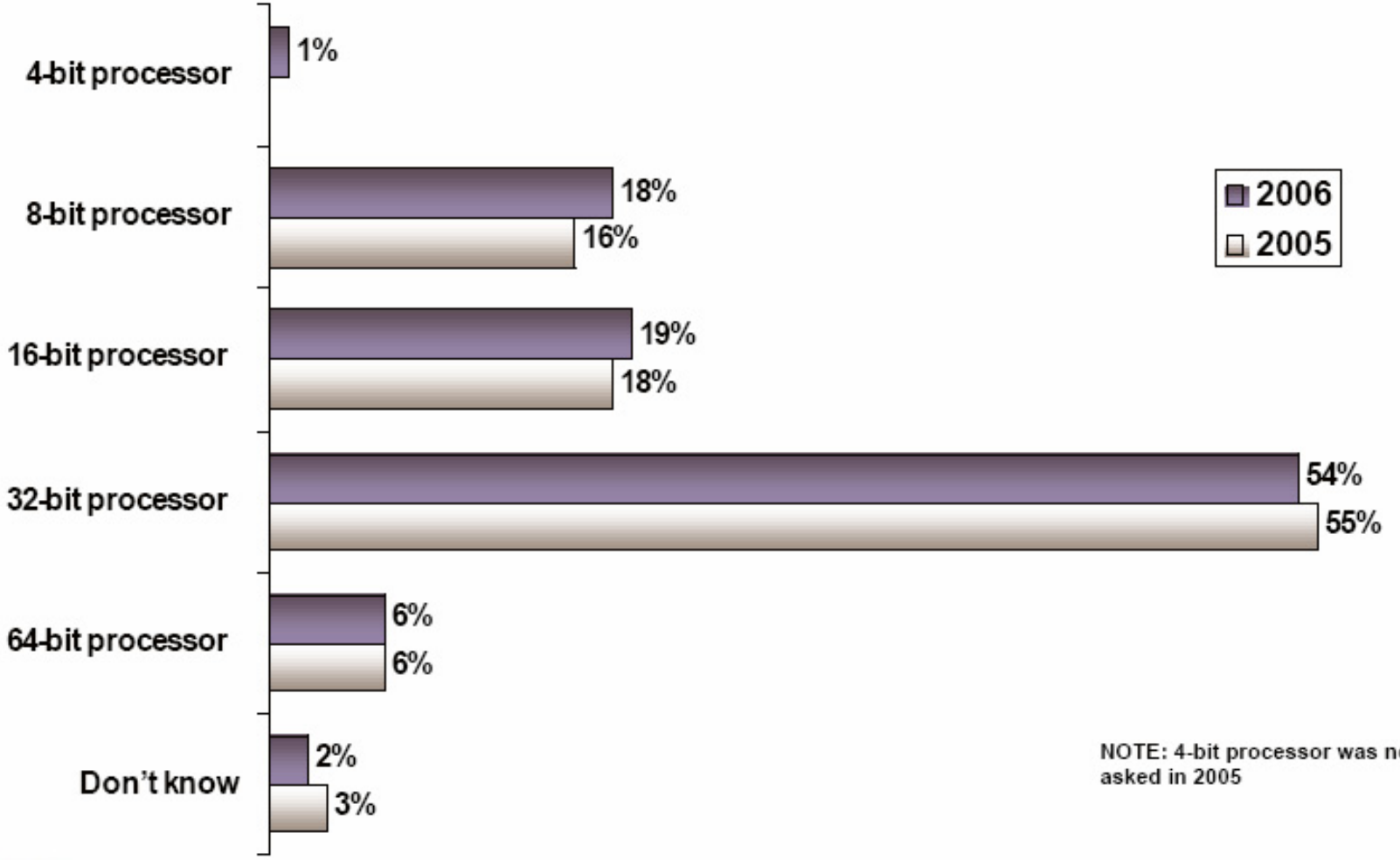
Marzo de 2010

Extra

CMP
United Business Media

2006 State of Embedded Market Survey

Current project's main processor



NOTE: 4-bit processor was not asked in 2005

N = 917

EETIMES
Embedded Systems Design

43. My current embedded project's *main* processor is a:

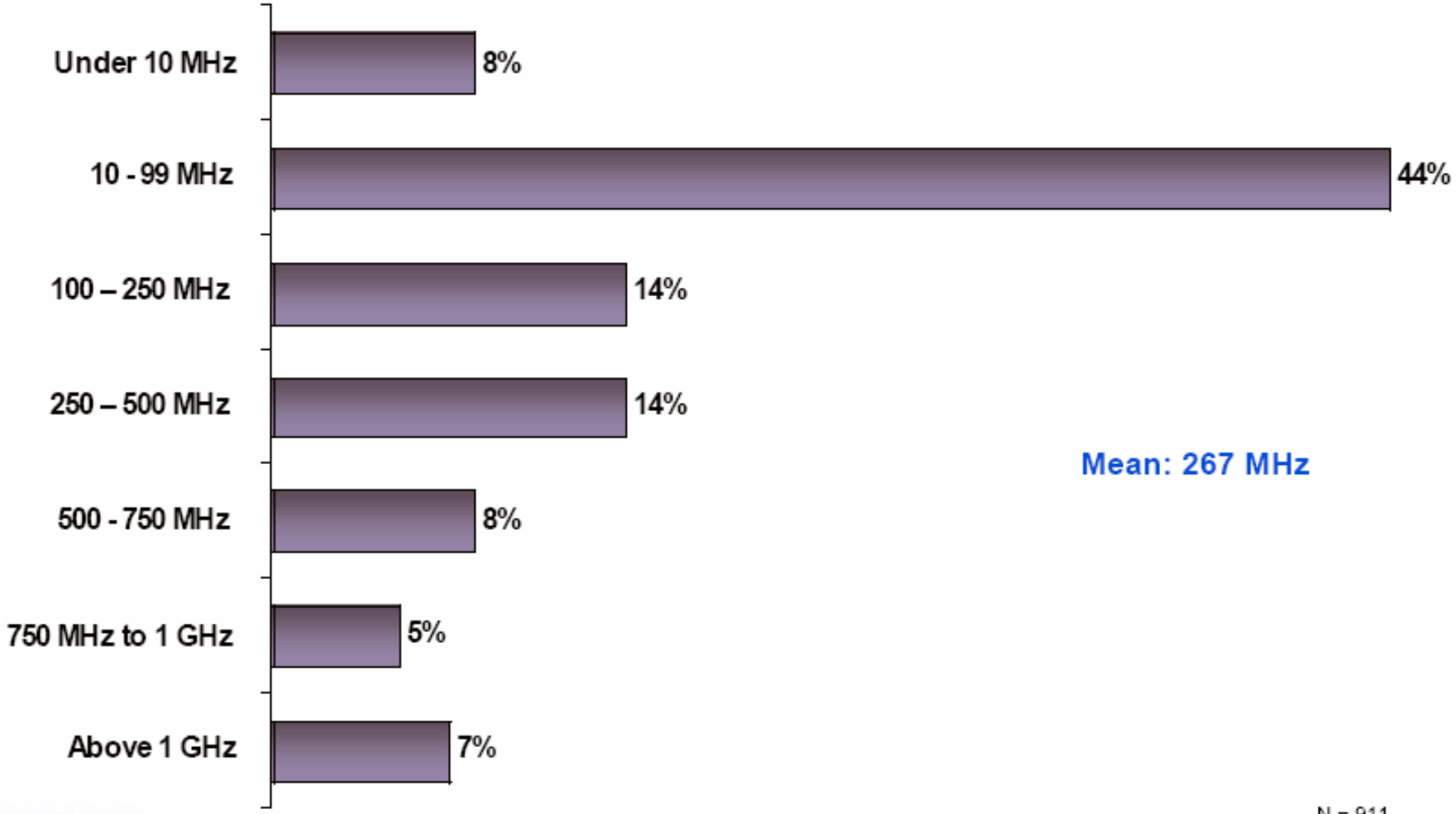
Embedded Systems CONFERENCE SILICON VALLEY

Extra

CMP
United Business Media

2006 State of Embedded Market Survey

Clock rate or main processor



Mean: 267 MHz

N = 911

EETIMES
Embedded Systems Design

Embedded Systems Conference Silicon Valley

44. My current embedded project's *main* processor clock rate is:

Extra

CMP
United Business Media

Current and projected embedded project programming language

Language	Current 2006 %	Next 2006 %
C	51	47
C++	30	32
Assembly language	8	6
Java	3	3
BASIC	1	1
UML, MatLab, or other modeling language	3	3
XML	*	1
LabView	2	2
Other	5	5

- Indicates not asked in 2005
* Indicates less than 0.5%

N = 1043

EETIMES
Embedded Systems Design

19. My current embedded project is programmed mostly in:
20. My next embedded project will likely be programmed mostly in

Embedded Systems Conference Silicon Valley

¡Gracias!

Procesadores: Arquitecturas y Tecnologías

Andrés Djordjalian <andres@indicart.com.ar>
Indicart Carteles Electrónicos y Facultad de Ingeniería, UBA

Para el Simposio Argentino de Sistemas Embebidos (SASE 2010)

Marzo de 2010