7919 | Sistemas Embebidos

2º Cuatrimestre de 2017

CLASE 3: HARDWARE BÁSICO DE UN SISTEMA EMBEBIDO

Prof: José H. Moyano

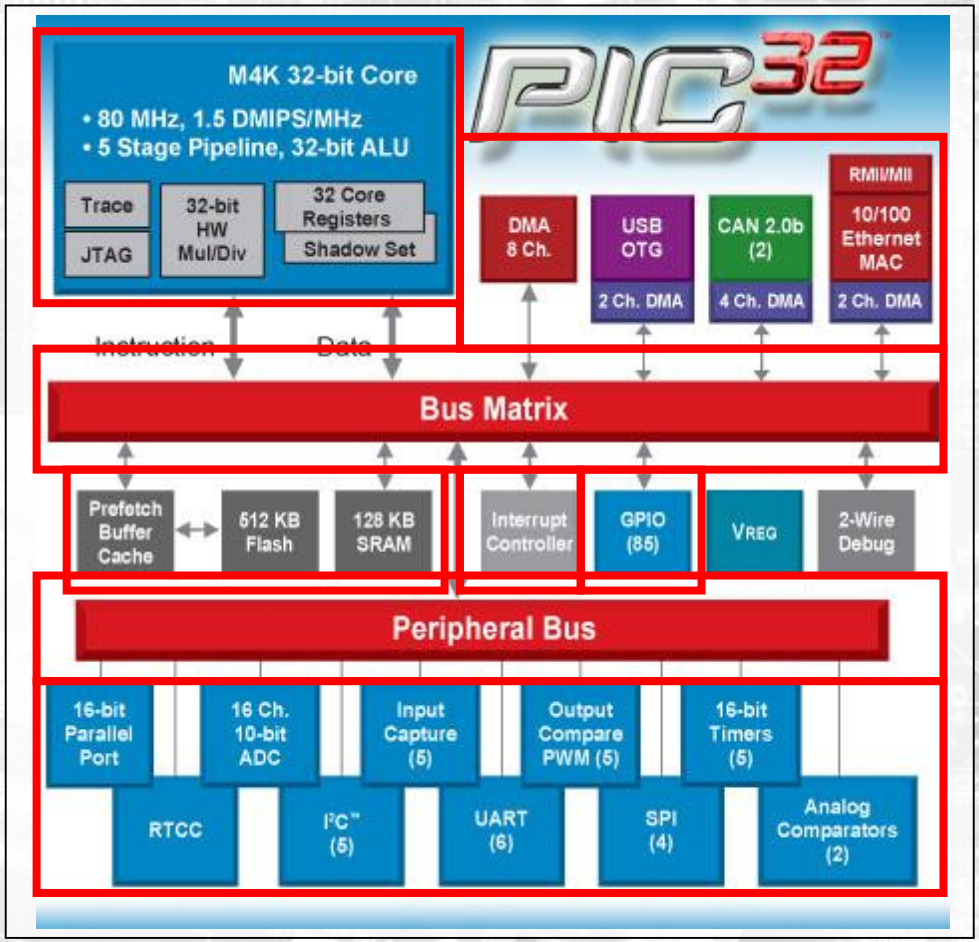
Autor original: Sebastián Escarza

Dpto. de Cs. e Ing. de la Computación
Universidad Nacional del Sur
Bahía Blanca, Buenos Aires, Argentina

GrIDSE

Caracterización de Sist. Embebidos

- Principales componentes:
  - Procesador central
  - Memoria
  - Entrada/Salida
    - Puertos
    - Interrupciones
    - Dispositivos e Interfaces
  - Buses
  - Clock



Definiciones

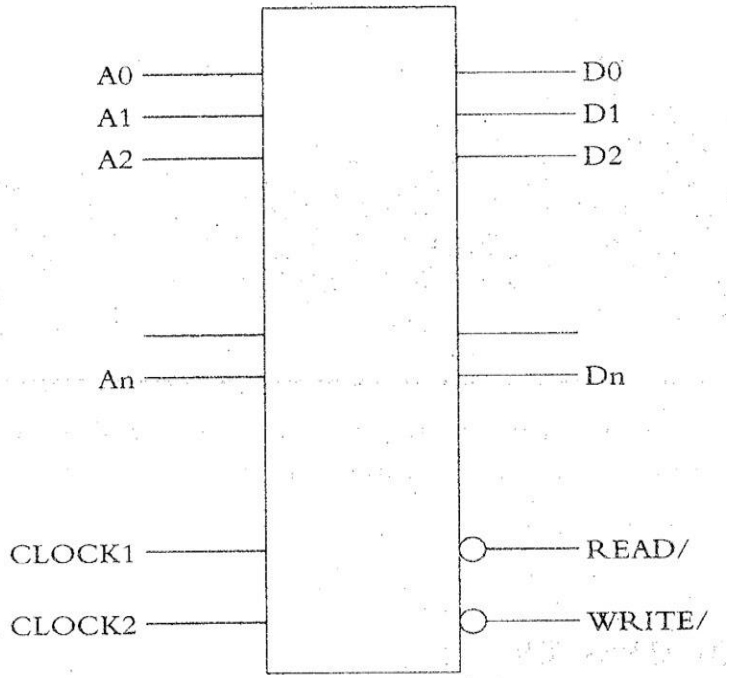
Definiciones

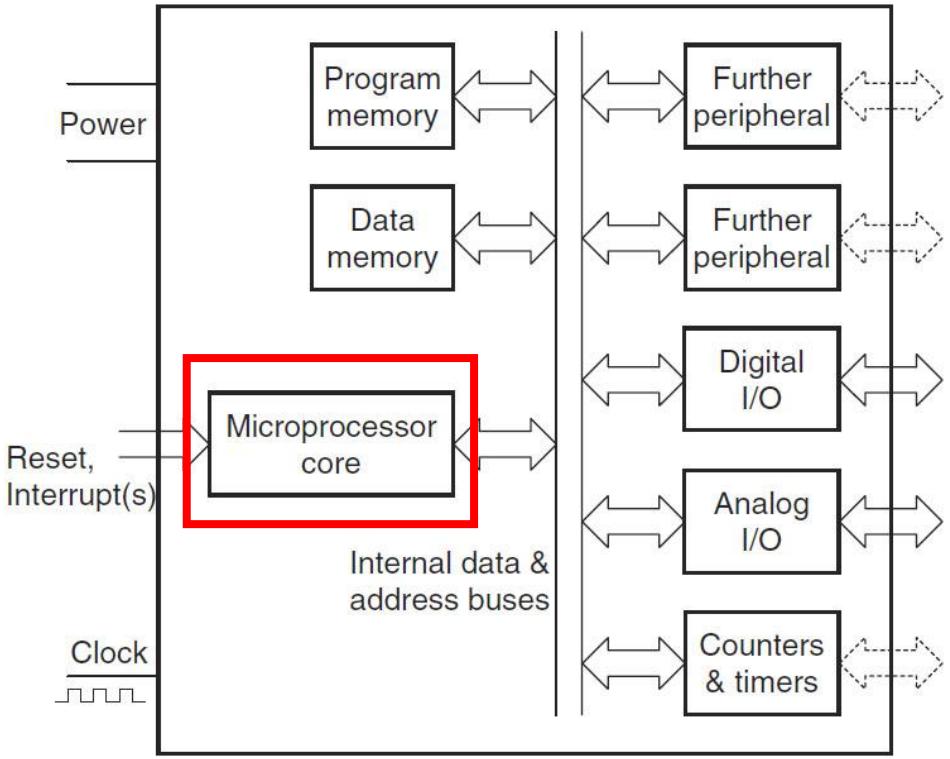
- **Microprocesador:** uno o más CPUs en un encapsulado (puede incluir controlador de memoria, etc).
- **Microcontrolador:** un conjunto de
  - uno o más CPUs
  - memoria
  - dispositivos, interfaces de E/S simples
- **Single Board Microcontroller:**
  - Un microcontrolador con otros dispositivos en una única tarjeta/placa.
- **System on a Chip (SoC):**
  - Similar a un microcontrolador, pero conteniendo dispositivos avanzados en el mismo integrado.

Definiciones

- Microprocesadores vs Microcontroladores

Figure 3.1 A Very Basic Microprocessor





Definiciones

- Microprocesadores vs Microcontroladores

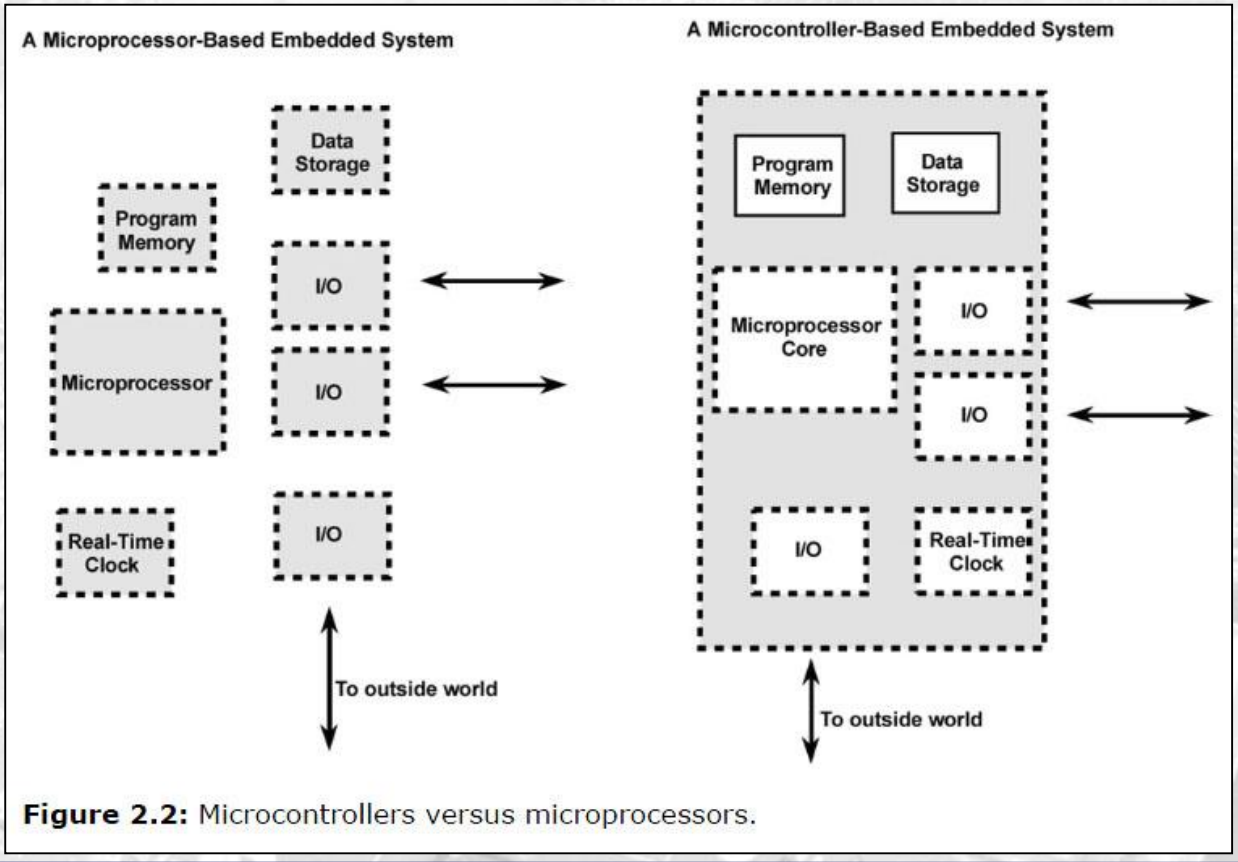
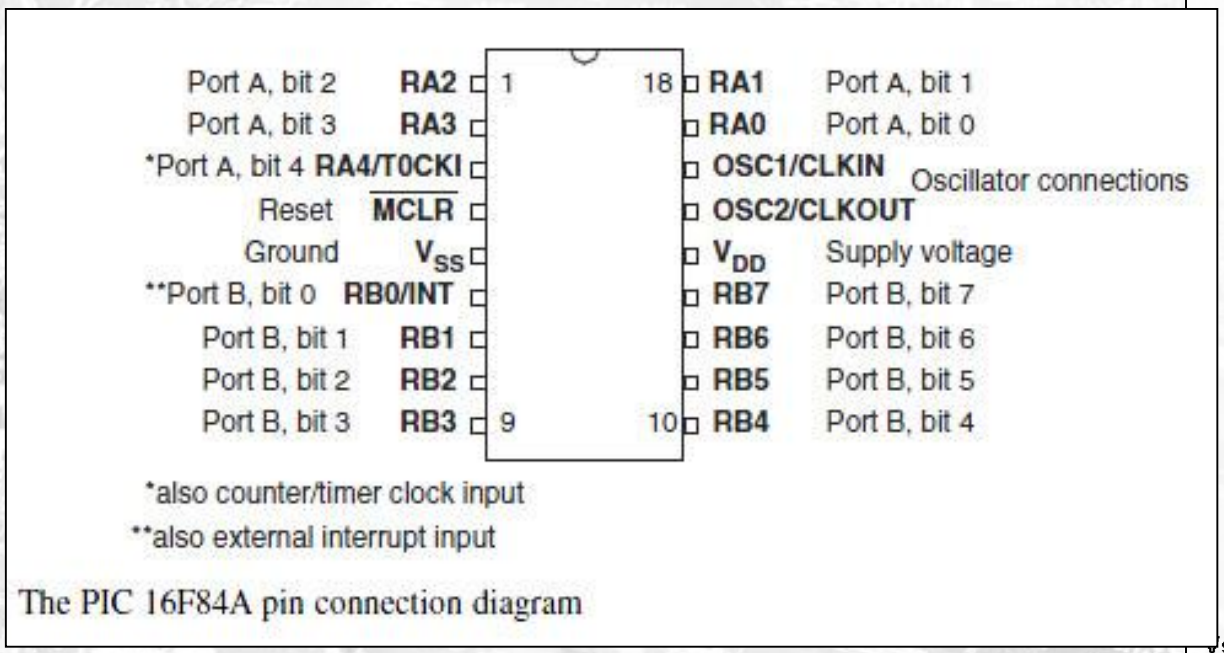


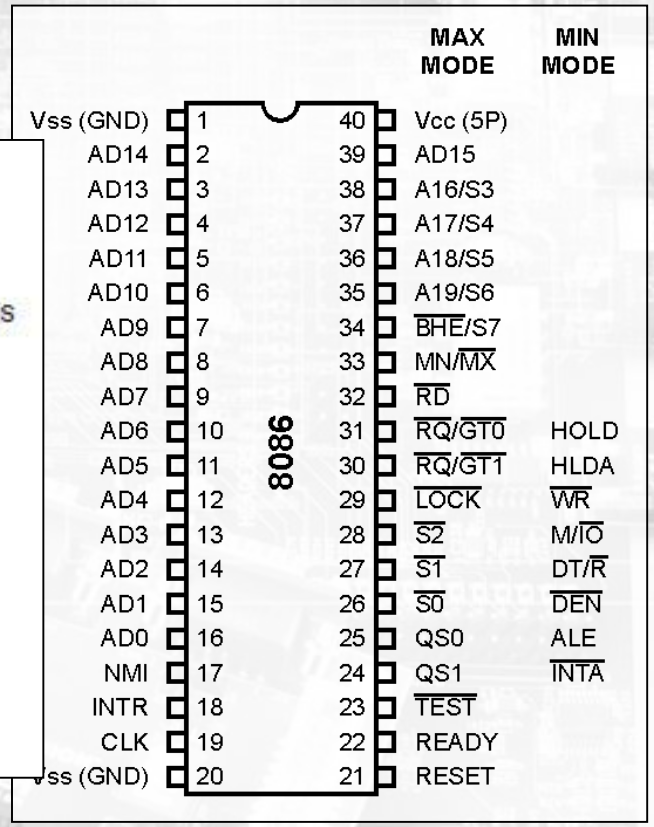
Figure 2.2: Microcontrollers versus microprocessors.

Definiciones

- Ej: µC PIC16 vs µP I8086



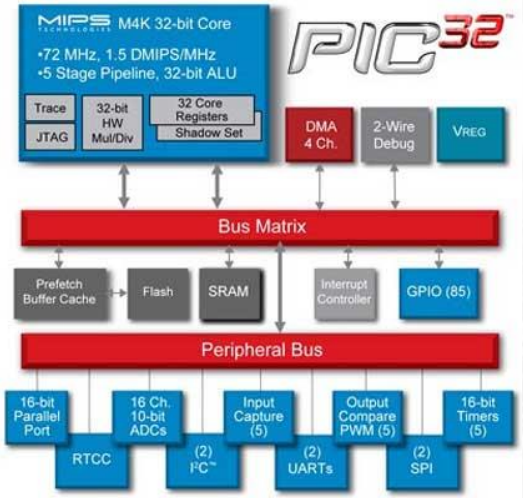
The PIC 16F84A pin connection diagram



Definiciones

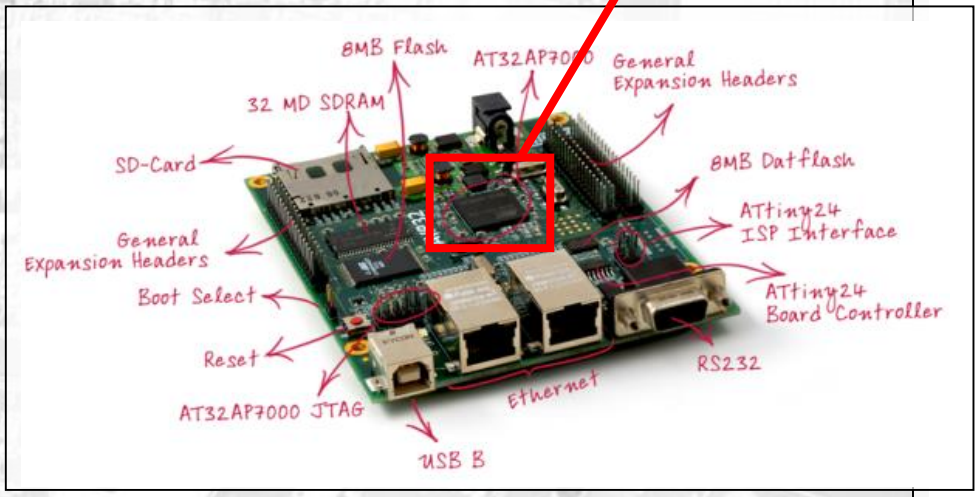
- *Single Board Microcontrollers vs Systems on Chip:*
  - *Varía el nivel de integración (comp. discretos vs ICs)*
  - *La distinción es a nivel de diseño de hardware*

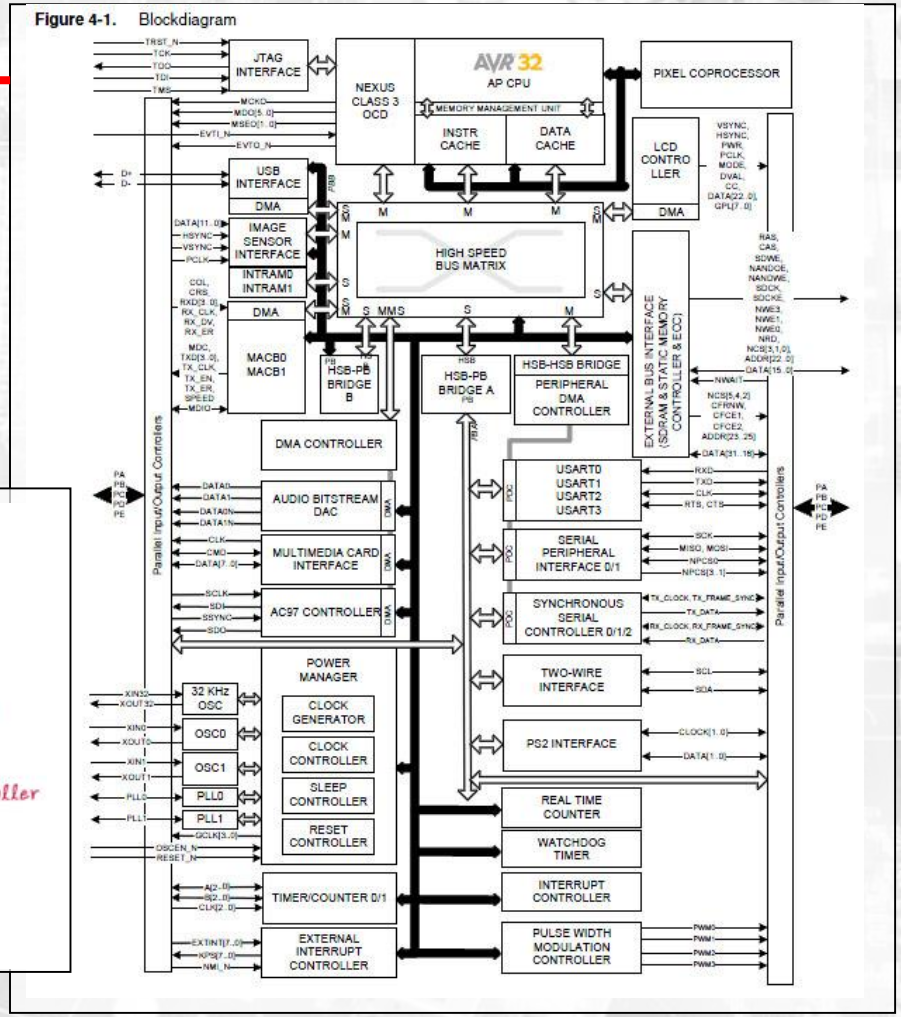
VS



Definiciones

- SB Microcontrollers vs Systems on a Chip
  - Muchos SoCs vienen en Demo Boards...





Definiciones

- ASIC (Application-Specific Integrated Circuit): es un circuito integrado de propósito particular.
  - Suelen acompañar al µP en Single Board µCs.
  - Se diseñan y manufacturan a medida.
  - Por ej:
    - UARTs, Conversores AD/DA
    - Controladores de Interrupciones
    - SoCs completos. etc.
  - Se contraponen a los ICs de propósito general (procesadores, dispositivos lógicos programables).
- ASSP (Application-specific standard product): es un ASIC de propósito general prediseñado por los fabricantes (COTS).

Componentes de HW: Clock

Clock y sincronismo

- Prácticamente todas las actividades en un microcontrolador están sincronizadas a partir de una señal de reloj (clock).

- Esta señal de clock es lo que hace evolucionar la operación de distintos componentes del sistema. El principal es el microprocesador.

Clock y sincronismo

- Mayor frecuencia implica:
  - Mayor velocidad de operación
  - Mayor consumo y posiblemente mayor interferencia electromagnética

- Además de la frecuencia, interesan cuestiones como la estabilidad de la señal de clock (variaciones respecto de la temperatura, humedad, alimentación suministrada, layout del circuito impreso (PCB), etc).

Clock y sincronismo

- La señal de clock puede generarse a partir de:
  - osciladores RC: Menos preciso, varía con temperatura, humedad, etc. Con los valores de R y C, se selecciona la frecuencia.
  - cristales: Son caros y frágiles mecánicamente. Señal más precisa y estable.
  - resonador cerámico: Mismo efecto piezoeléctrico de los cristales. Costo y estabilidad intermedia entre RC y cristales.

Clock y sincronismo

- La señal de clock puede generarse a partir de:
  - osciladores RC
  - cristales
  - resonador cerámico

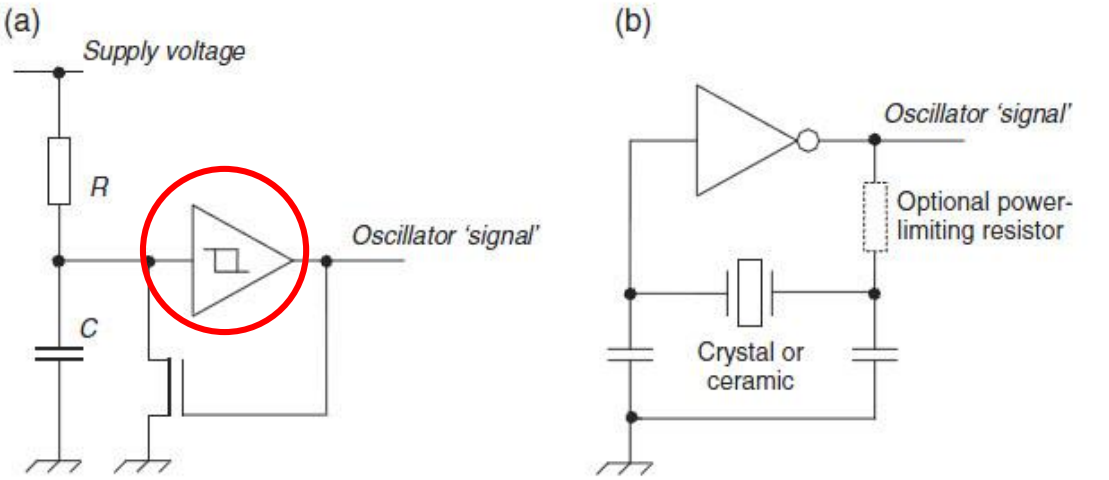


Figure 3.13 Microcontroller oscillator generator circuits. (a) Resistor–capacitor (RC). (b) Crystal or ceramic

Clock y sincronismo

- La señal de clock puede generarse a partir de:
  - osciladores RC
  - cristales
  - resonador cerámico

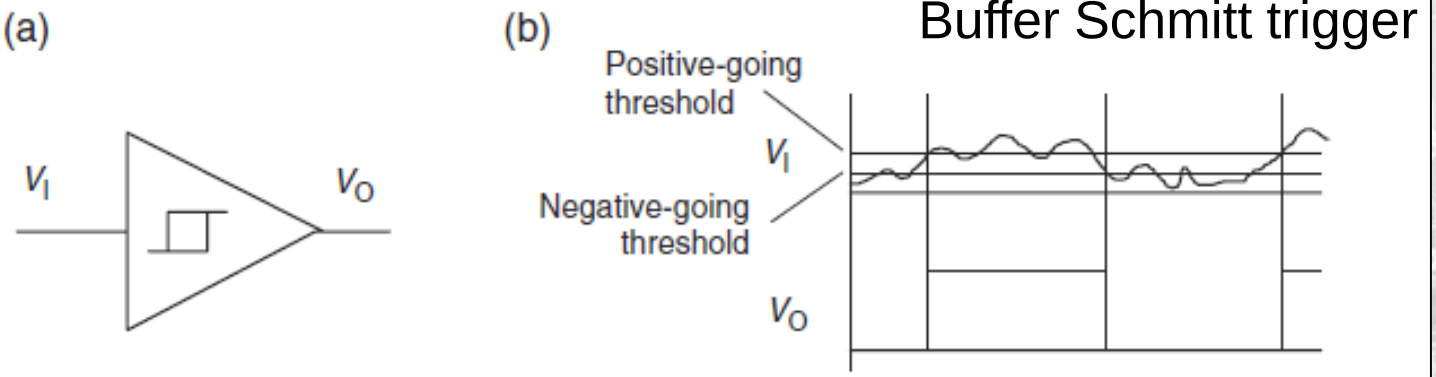


Figure 3.5 Schmitt trigger characteristics. (a) Buffer with Schmitt trigger input. (b) Input/output characteristic

Clock y sincronismo

- Los microcontroladores usualmente cuentan con osciladores internos (y registros de calibración), y con pines para interconectar osciladores externos.

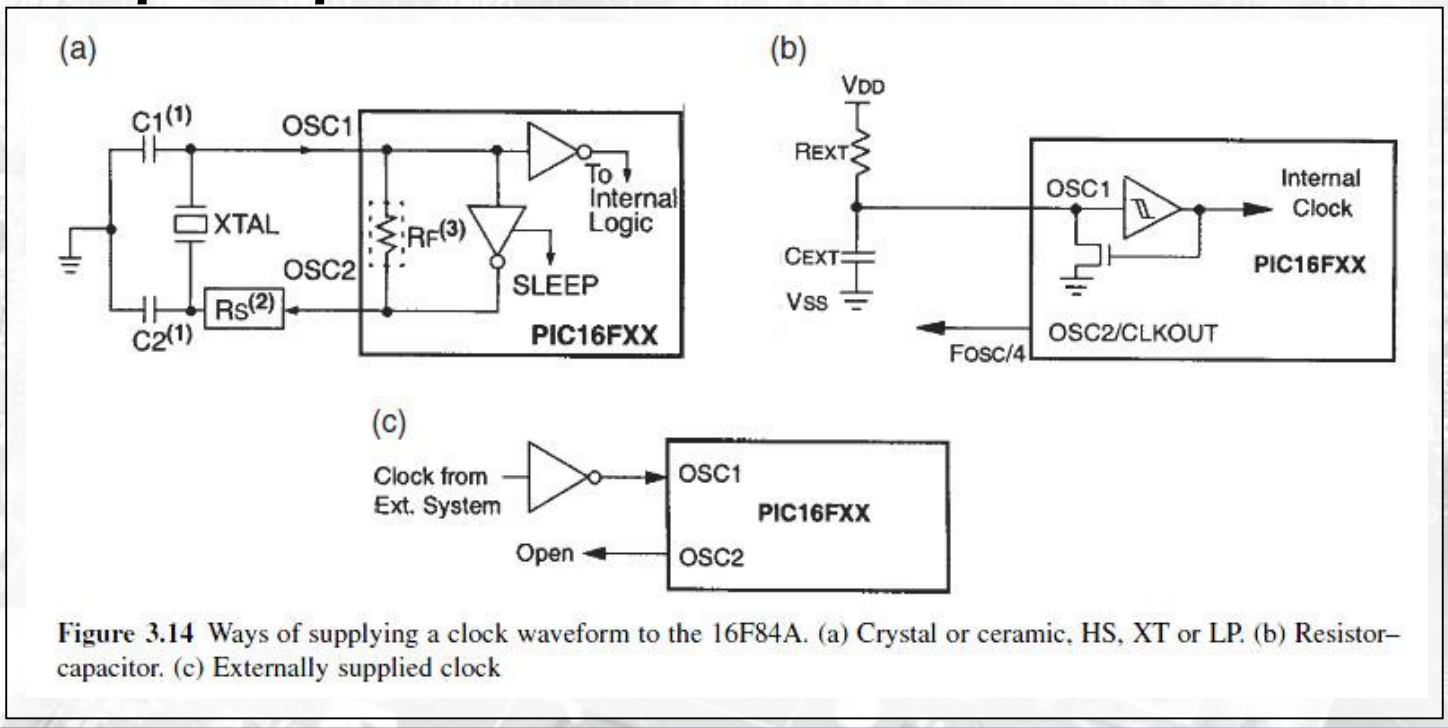


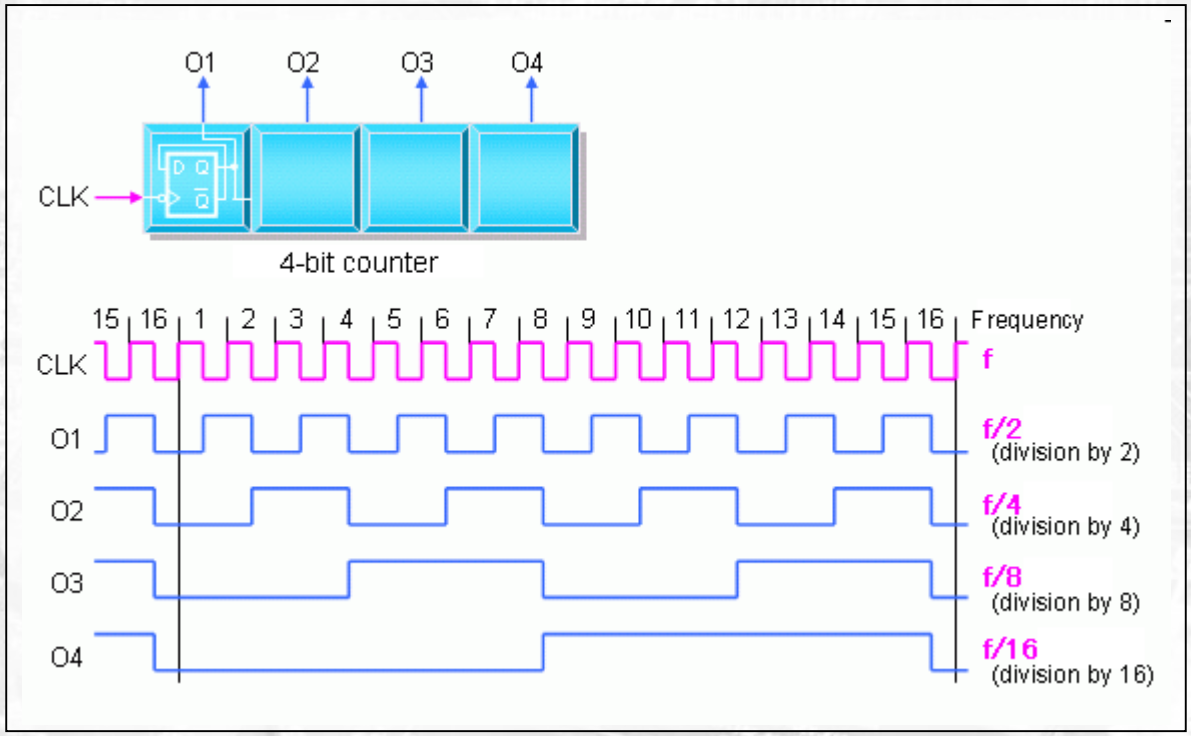
Figure 3.14 Ways of supplying a clock waveform to the 16F84A. (a) Crystal or ceramic, HS, XT or LP. (b) Resistor–capacitor. (c) Externally supplied clock

Clock y sincronismo

- La frecuencia debe seleccionarse de manera que:
  - sea suficientemente lenta para cubrir los requerimientos temporales de los diversos dispositivos (en general, los dispositivos no funcionan por encima de las frecuencias de reloj para las que fue diseñados).
  - sea suficientemente rápida para cubrir los requerimientos temporales de la aplicación.
  - contemple las restricciones de consumo del sistema (la performance de operación no lo es todo).
- Para lograr los primeros dos objetivos, a una señal de reloj base, se la suele dividir para lograr señales de reloj sincrónicas más lentas útiles para manejar diversos dispositivos.

Clock y sincronismo

- Divisor de reloj: mediante un contador...



Procesador Central

Procesador Central

- Es el centro de procesamiento de todo el sistema.
- La elección del CPU define la arquitectura del sistema:
  - ancho de palabra (data bus)
  - espacio direccionable (address bus) y su organización
  - set de instrucciones
  - registros internos
  - aritmética (floating/fixed point, si es saturada)
  - gestión de interrupciones
  - existencia de MMU (Memory Management Unit)/MPU (Memory Protection Unit)
- Condiciona consumo, desempeño, las posibles aplicaciones, etc.

Procesador Central

- Organización de la memoria (Von Neumann vs Harvard):

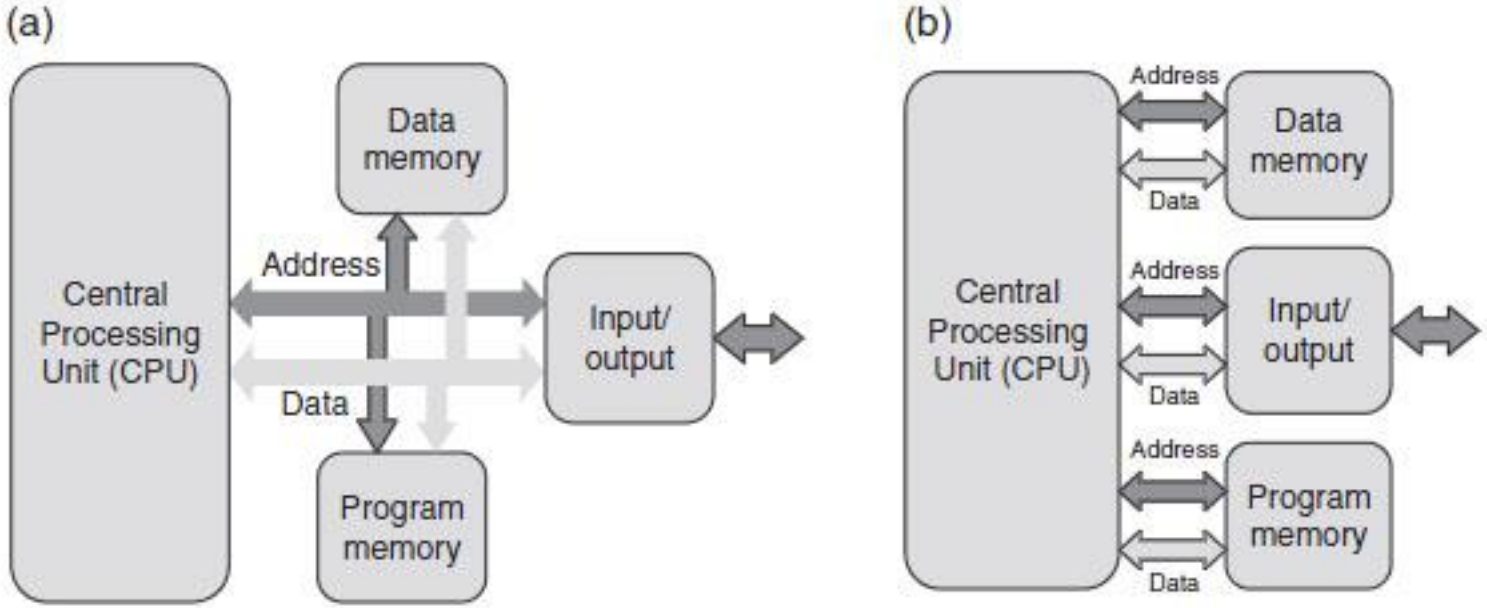
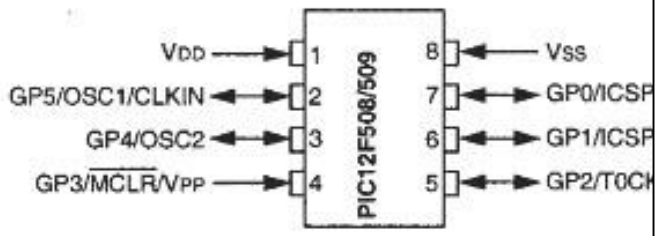


Figure 1.7 Organising memory access. (a) The Von Neumann way. (b) The Harvard way

Procesador Central

- Ej: PIC12F50X
  - Data (Addr) bus: 8bits
  - Instr. (Addr) bus: 12 bits
  - Arquitectura Harvard



Key

V_{DD}:	Power supply	V_{SS}:	
V_{PP}:	Programming voltage input	MCLR:	
OSC1, OSC2:	Oscillator pins	CLKIN:	
GP0 to GP5:	General-Purpose input/output pins (bidirectional		
CSPDAT:	In-Circuit Serial Programming™ data pin.		
CSPCLK:	In-Circuit Serial Programming™ clock pin.		

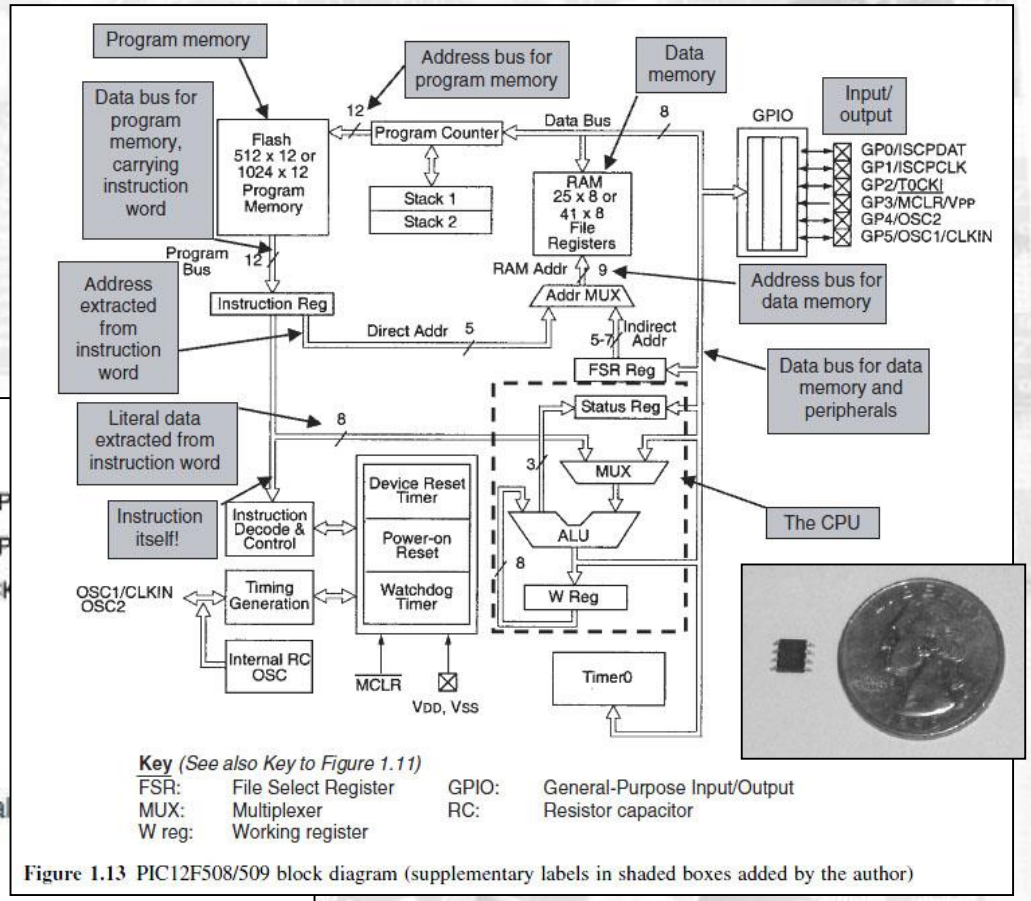


Figure 1.13 PIC12F508/509 block diagram (supplementary labels in shaded boxes added by the author)

Procesador Central

- Ej: Freescale MC68HC908
  - Bus: 8bits
  - Arquitectura Von Neumann

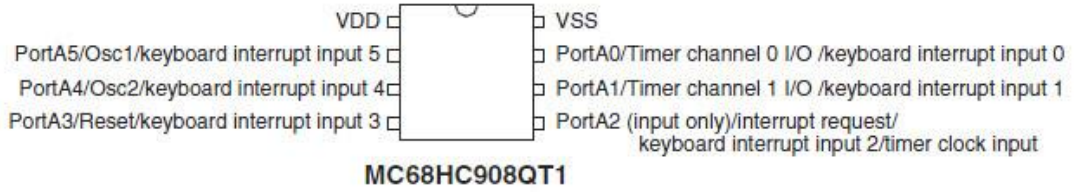


Figure 1.14 A Freescale 68HC908 microcontroller

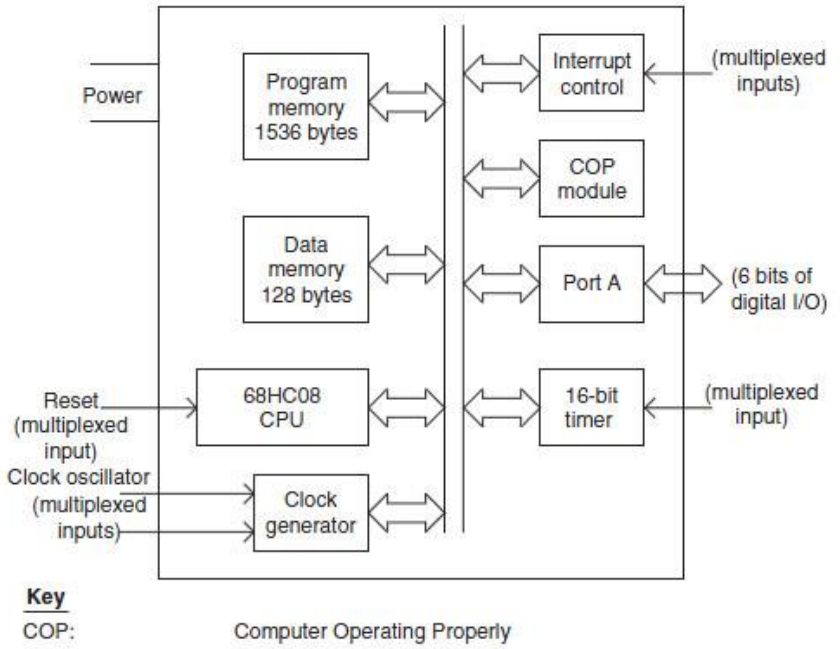


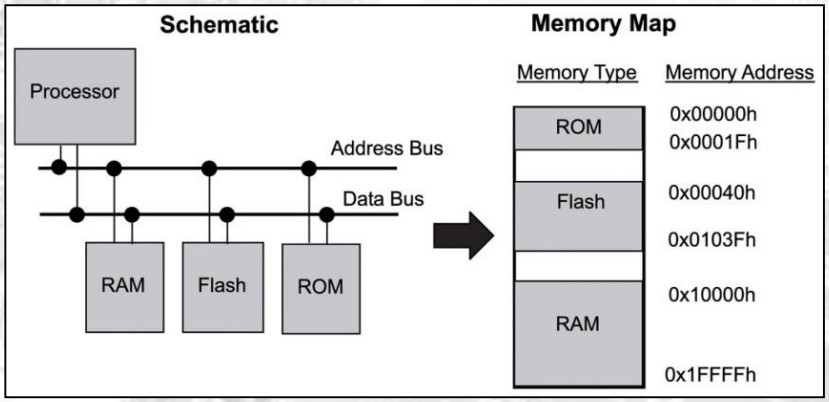
Figure 1.15 The Freescale MC68HC908QTI microcontroller – simplified block diagram

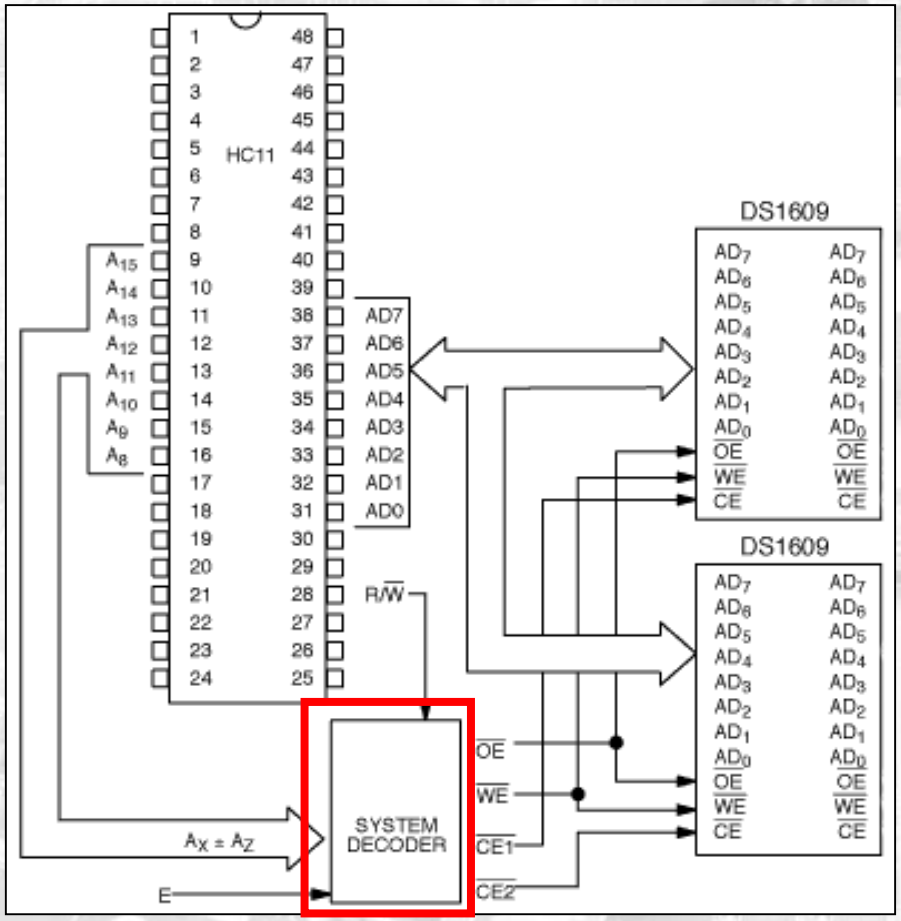
Procesador Central

- El procesador condiciona el mapa de memoria:
  - En qué direcciones están los registros y la memoria.
  - Memory mapped I/O vs Port mapped I/O.
  - Ubicación del vector de interrupciones (si existe).
  - Dirección/vector de reset (la primer instr. a ejecutar).
- En sistemas embebidos es esencial conocer la configuración del mapa de memoria (lo fija el diseñador de hardware).
  - En SoCs y Single Board µC está preestablecido.
  - Si implementamos el hardware, hay que decodificar correctamente las direcciones establecidas por el CPU.

Procesador Central

- El Mapa de Memoria se define por la lógica de decodificación de direcciones (lógica combinacional – glue logic).





Procesador Central

- Mapa de memoria: PIC16F84A

Table 2.2 16F84A memory features

Memory function	Technology	Size	Volatile/non-volatile	Special characteristics*
Program	Flash	1K × 14 bits	Non-volatile	10 000 erase/write cycles, typically
Data memory (file registers)	SRAM	68 bytes	Volatile	Retains data down to supply voltage of 1.5 V
Data memory (EEPROM)	EEPROM	64 bytes	Non-volatile	10 000 000 erase/write cycles, typically
Stack	SRAM	8 × 13 bits	Volatile	

*Information obtained from full 16F84A data sheet [Ref. 2.1].

Procesador Central

- Mapa de memoria: PIC16F84A

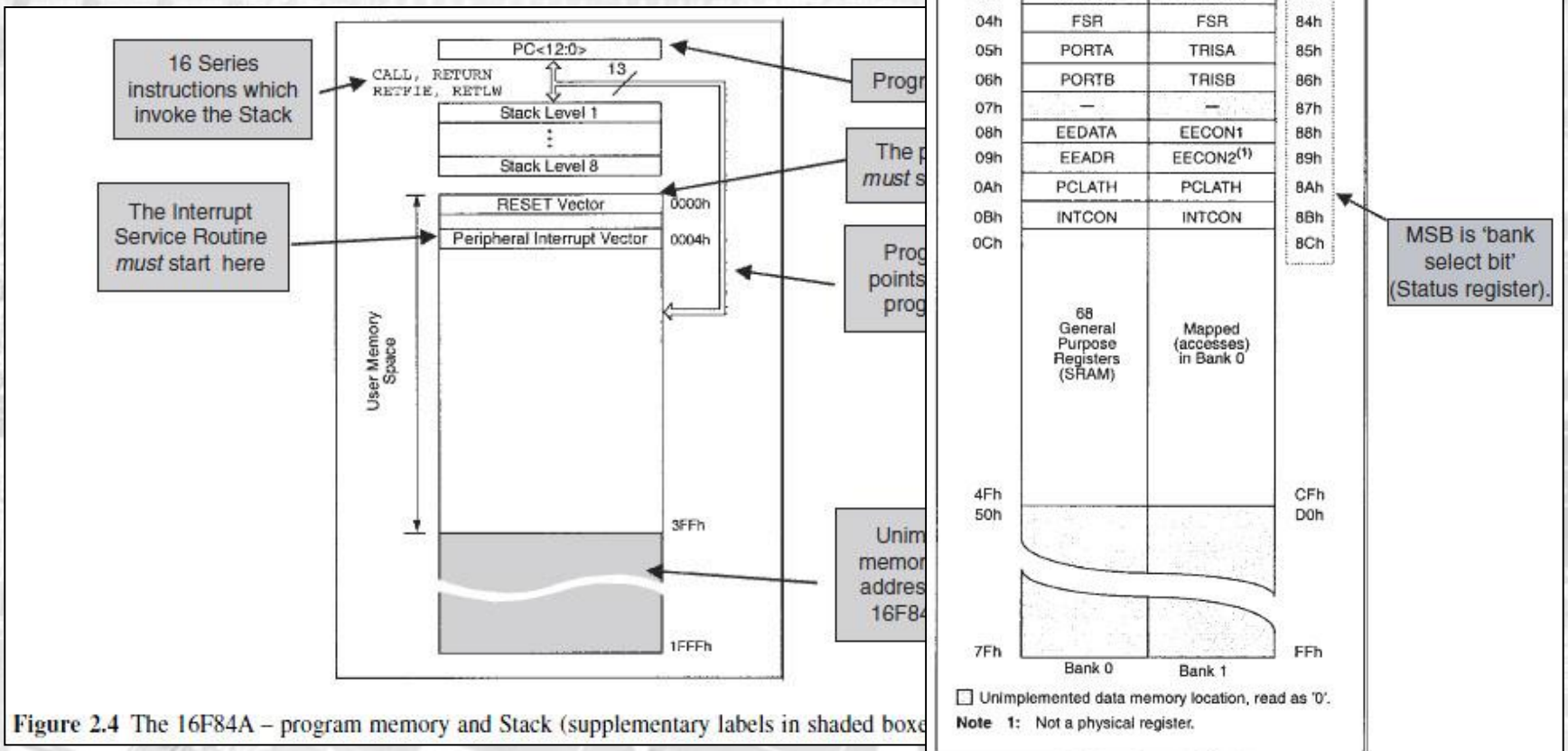
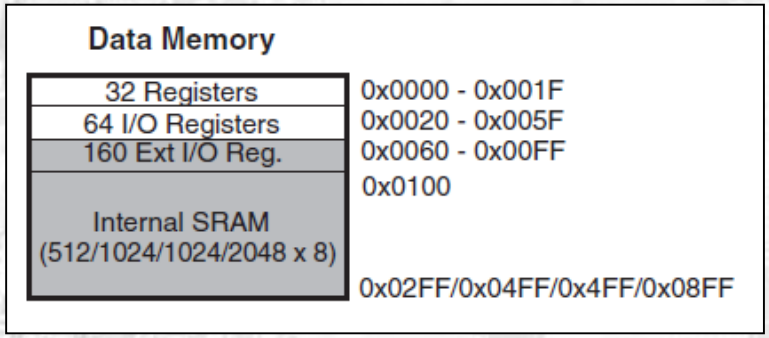
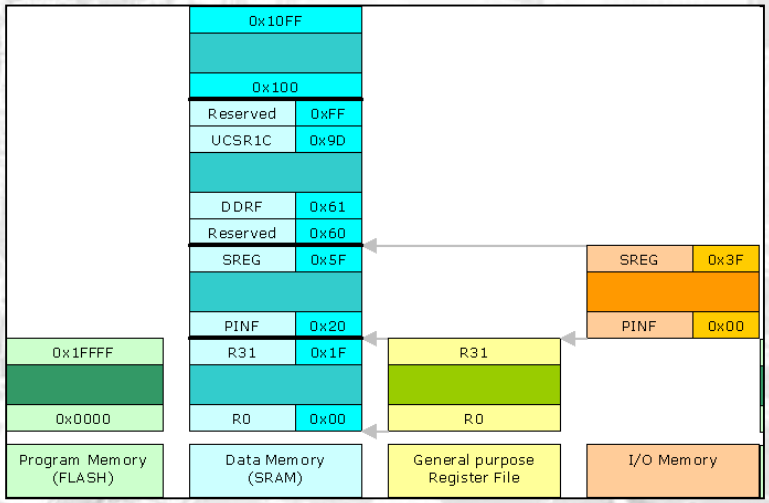


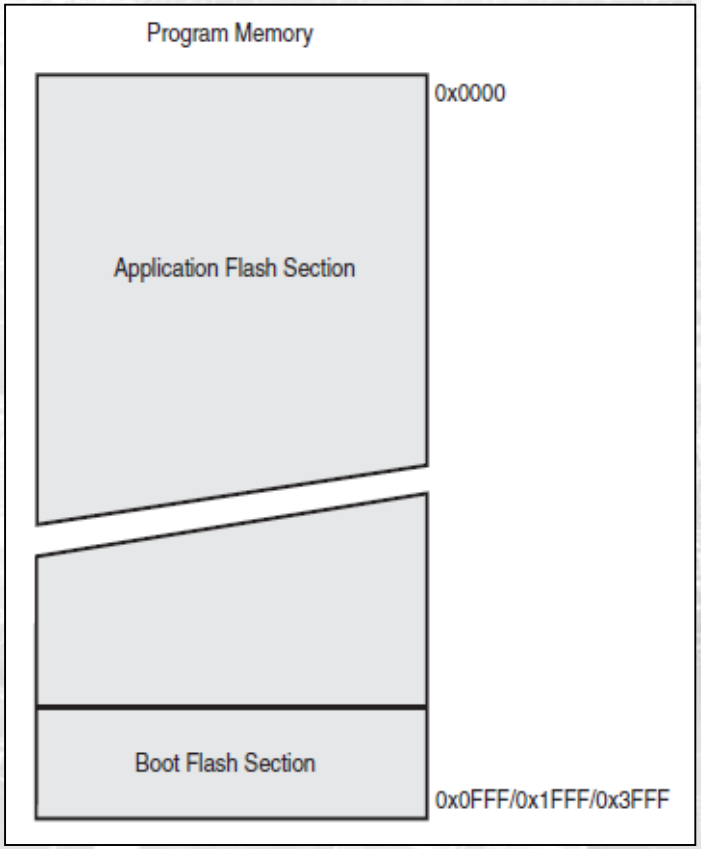
Figure 2.4 The 16F84A – program memory and Stack (supplementary labels in shaded boxe

Procesador Central

- Mapa de memoria: AVR ATmega328P

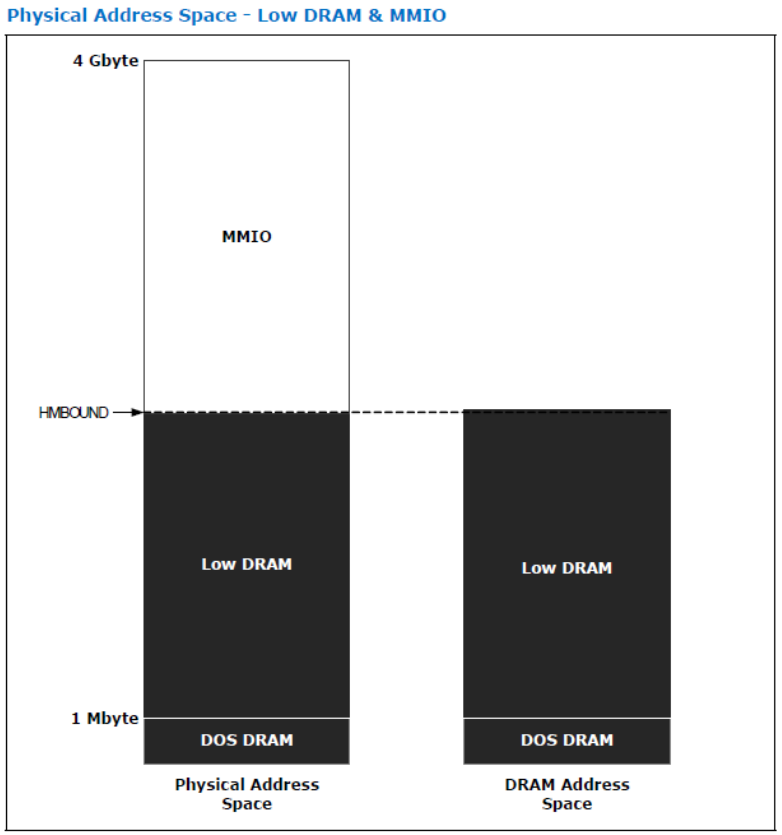


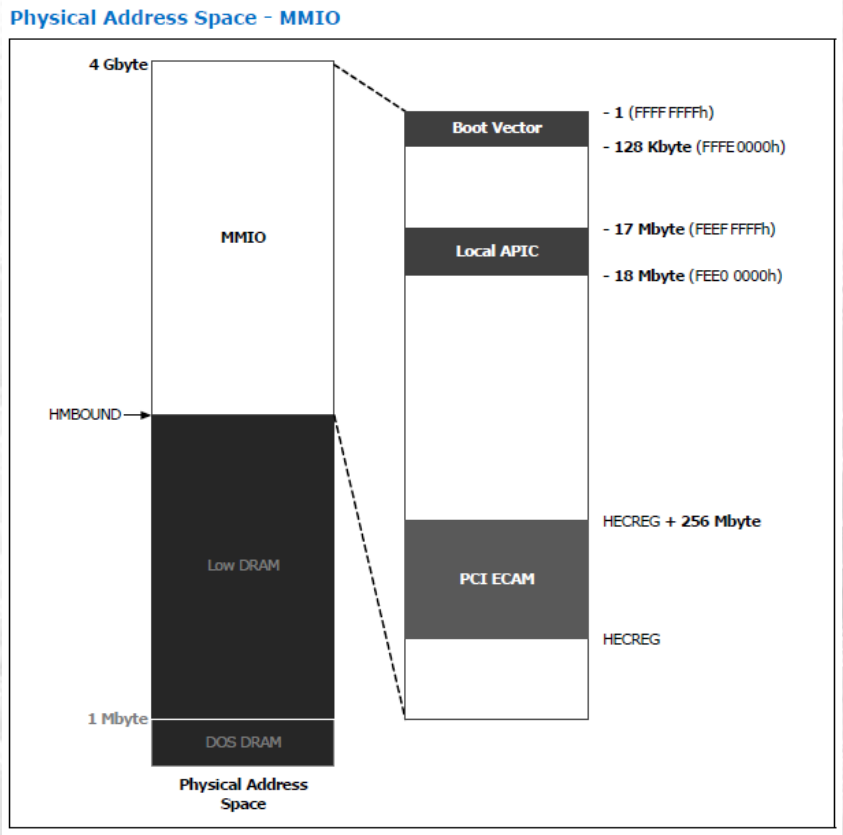




Procesador Central

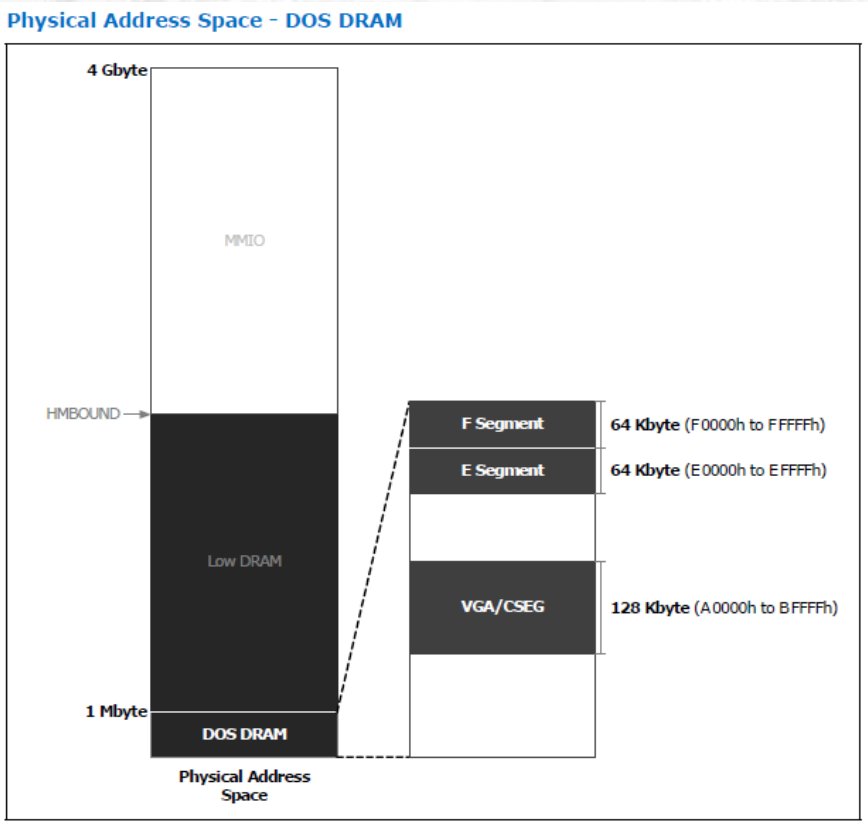
- Mapa de memoria: Intel Quark SoC X1000 (IA32)

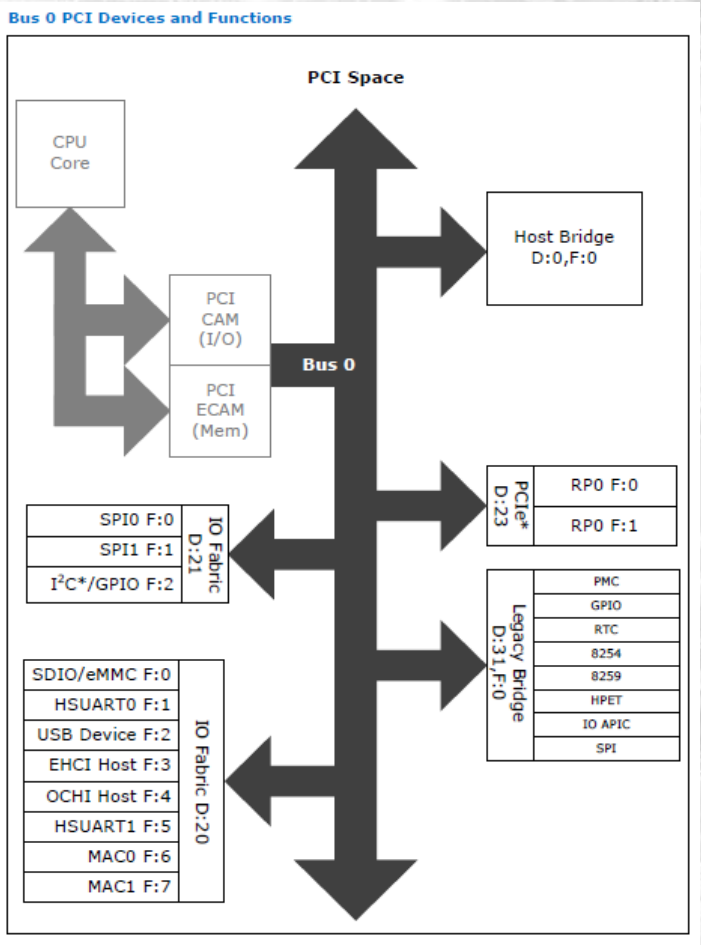




Procesador Central

- Quark SoC X1000: Mem/IO





Procesador Central

- Set de instrucciones:
  - RISC (Reduced Instruction Set Computing)
  - CISC (Complex Instruction Set Computing)

PIC 16 Series

```
movlw   22      ;1 cycle
movwf mem_loc   ;1 cycle
```

Atmel 8051

```
mov mem_loc,#22   ;2 cycles
```

(a)

PIC 16 Series

```
btfsc status,0  ;1 cycle (no skip)
goto new_place  ;2 cycles
```

Atmel 8051

```
jc new_place      ;2 cycles
```

(b)

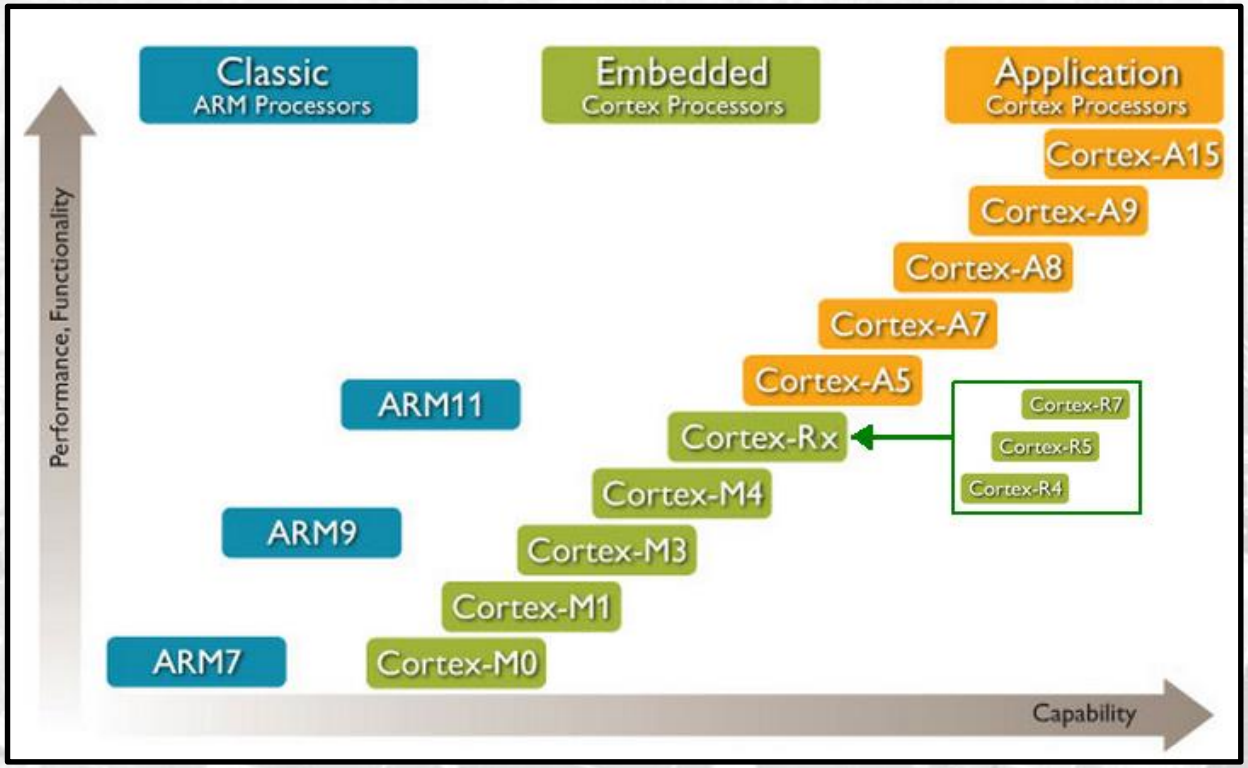
Program Example 4.4 Comparing RISC and CISC instruction capabilities. (a) Moving immediate/literal data to a memory location. (b) Branching if Carry bit set

Procesador Central

- El procesador y su set de instrucciones define:
  - la funcionalidad (instrucciones básicas, unidades funcionales, instrucciones SISD vs SIMD, instrucciones para DSP, etc).
  - el soporte de herramientas (compiladores, ensambladores, etc.) y librerías disponibles para la plataforma
  - el soporte de sistemas operativos (si se requiriese su uso) y middleware embebido en general.
- Existen arquitecturas/familias...
  - ... de procesadores: 80x86, PIC 16 Series, Cortex
  - ... de instruction sets: ARM, IA32, IA64, MIPS

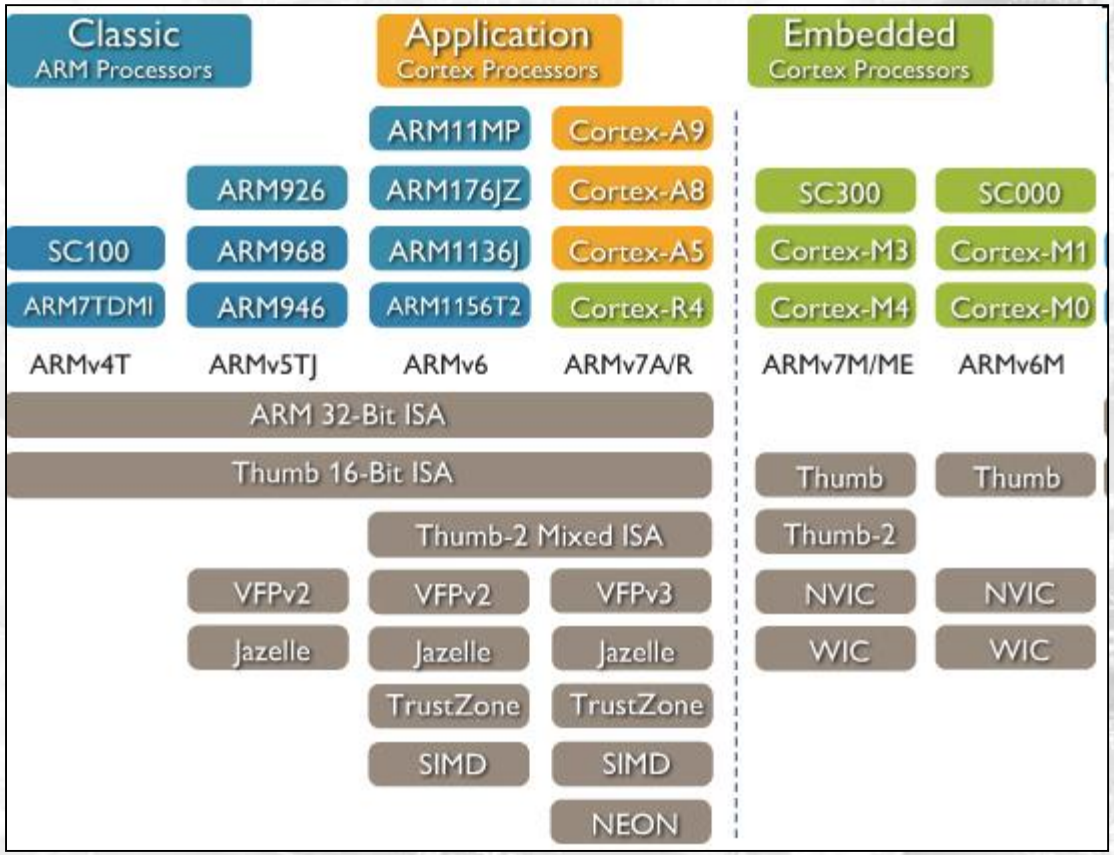
Procesador Central

- Procesadores ARM: Clásicos y Cortex
  - Cortex: A (Application)/R (Realtime)/M (Microcontroller)



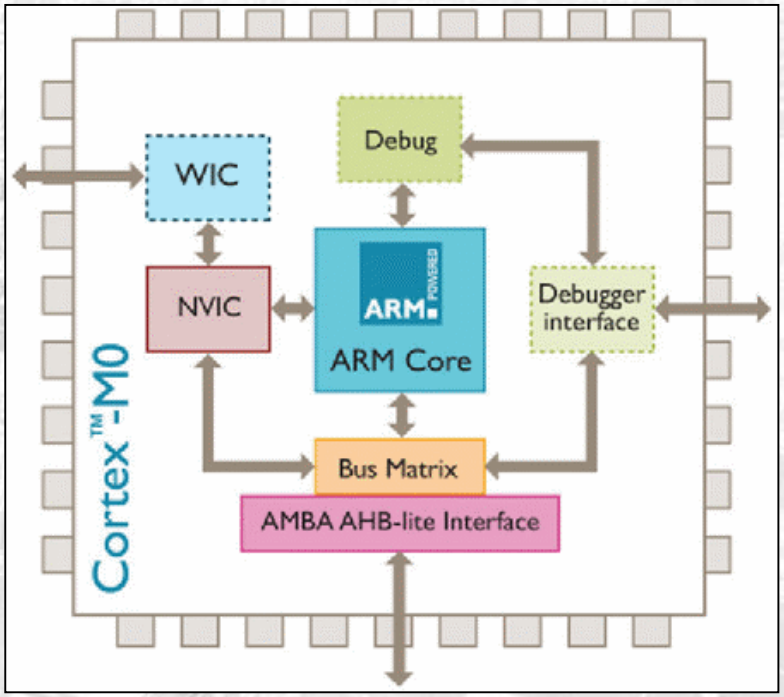
Procesador Central

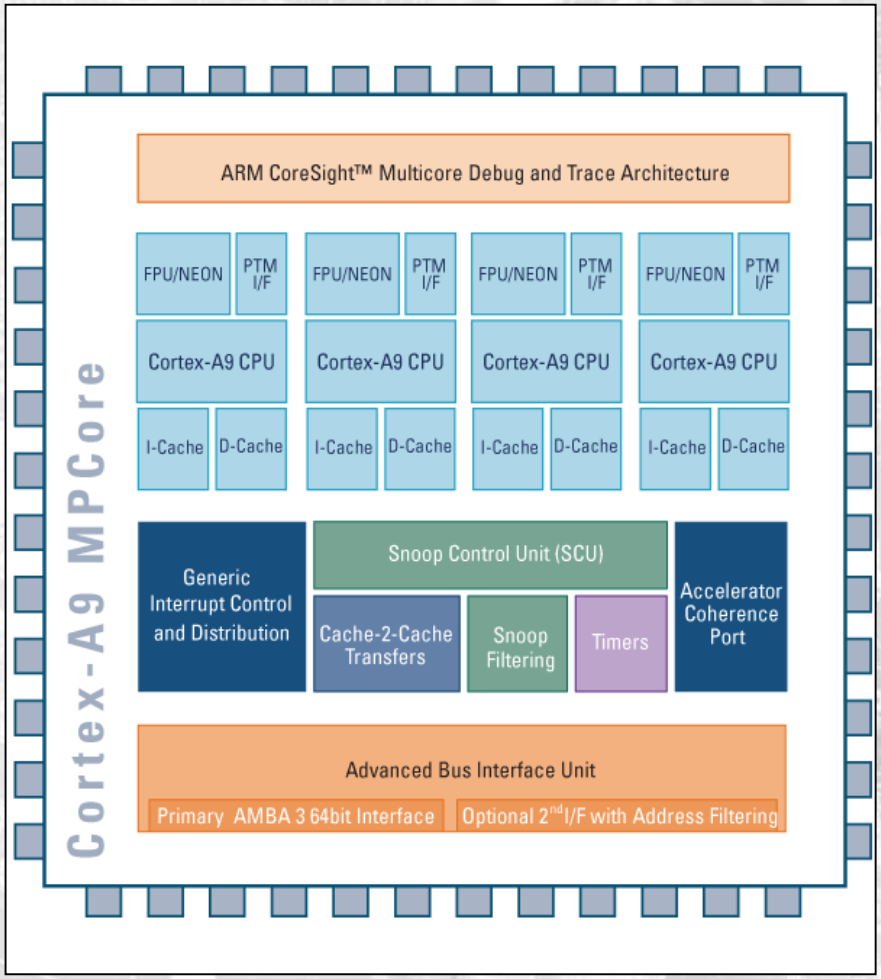
- Procesadores ARM en el contexto de la arquitectura ARM.



Procesador Central

- Procesadores ARM





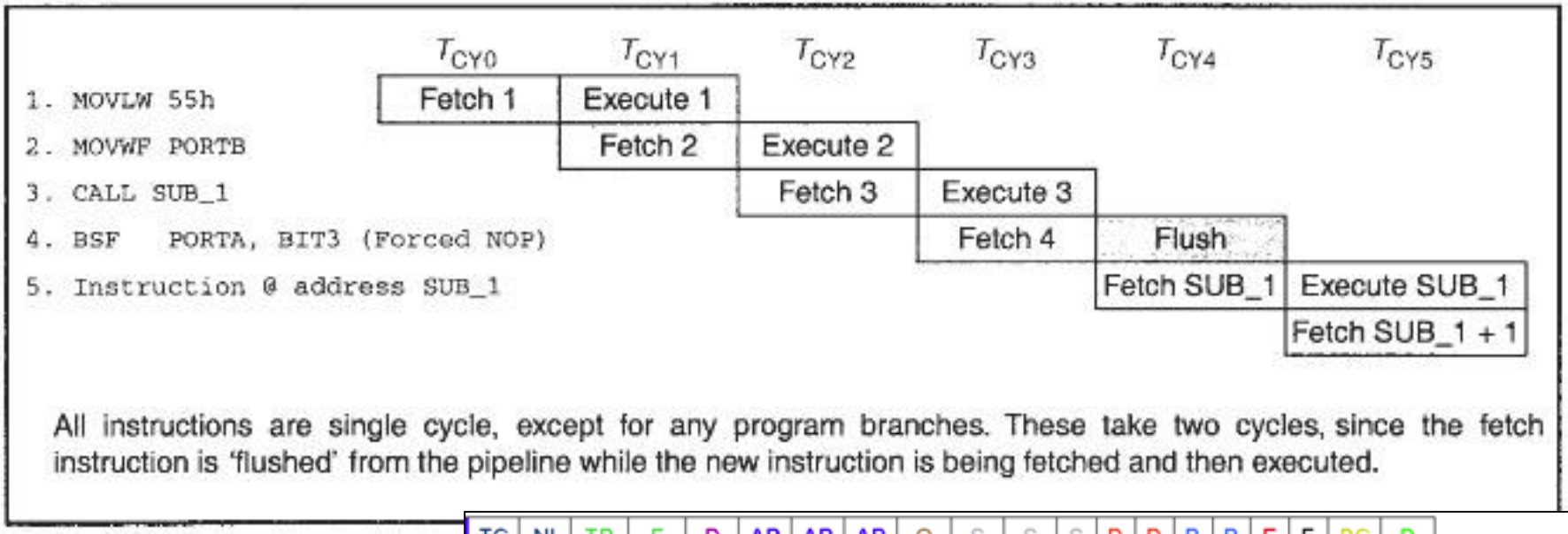
Procesador Central

- El pipelining (al igual que el uso de cachés) tiene implicancias en aplicaciones de tiempo real.

- Se busca temporizados estables (por ej, minimizar el tiempo de latencia de interrupciones), en lugar de incrementar el throughput de instrucciones.
  - Pipelines menos profundos
  - Arquitecturas más simples

Procesador Central

- Pipelining, hazards, etc. (PIC16F84A vs Pentium 4)



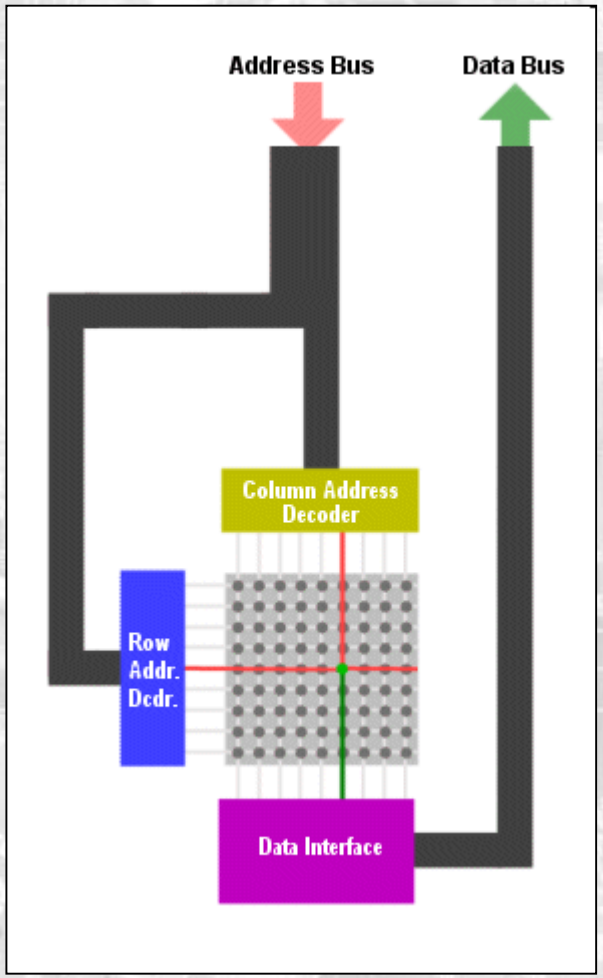
All instructions are single cycle, except for any program branches. These take two cycles, since the fetch instruction is 'flushed' from the pipeline while the new instruction is being fetched and then executed.

Figure 2.8 Instruction pipelining

Memorias

Memorias

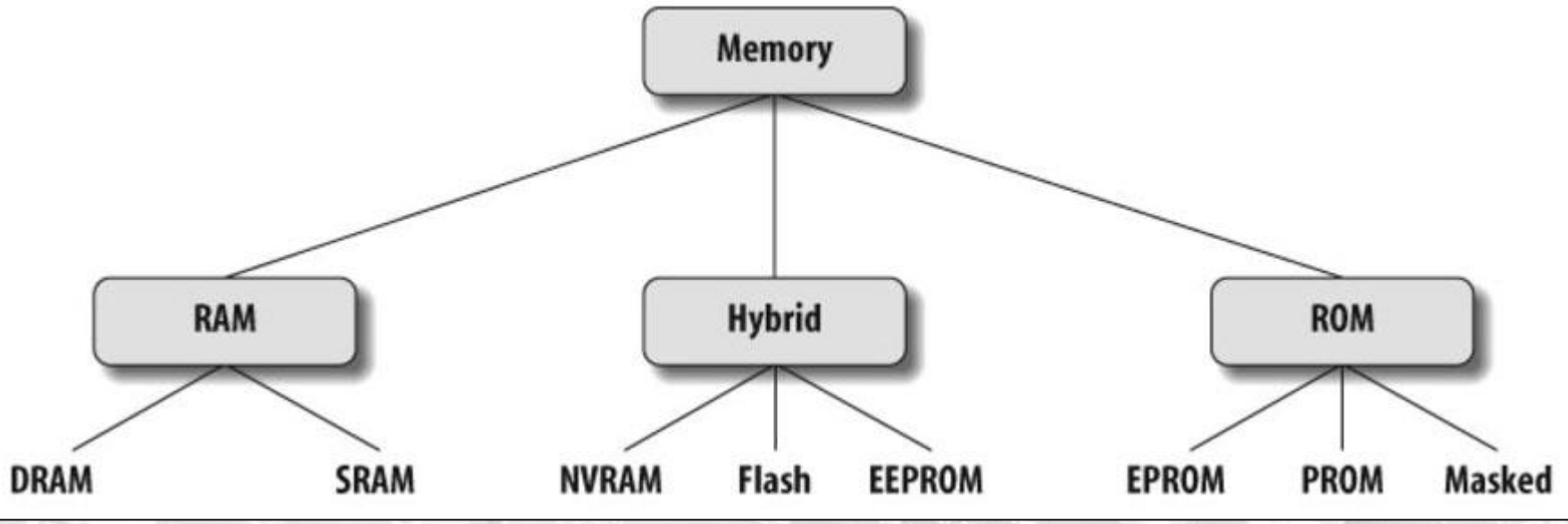
- Las memorias poseen diversas características:
  - Tipo (ROM, PROM, EPROM, EEPROM, Flash, RAM, etc.).
  - Ancho de palabra y tamaño
  - Tiempo de acceso (RAS time, CAS time)



Memorias

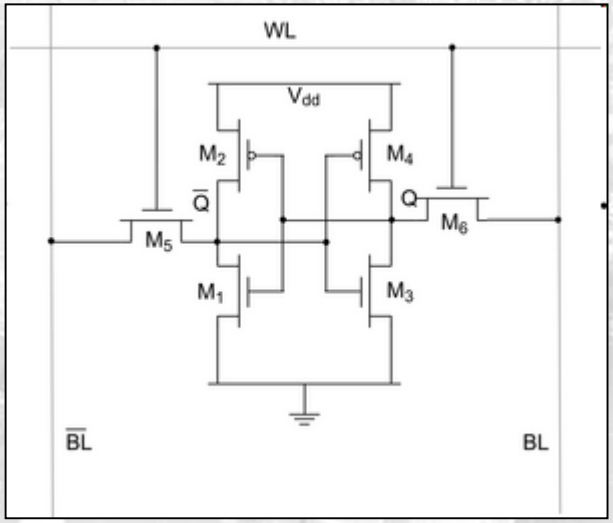
- Tipos de memorias:

Figure 6-1. Common memory types in embedded systems



Memorias

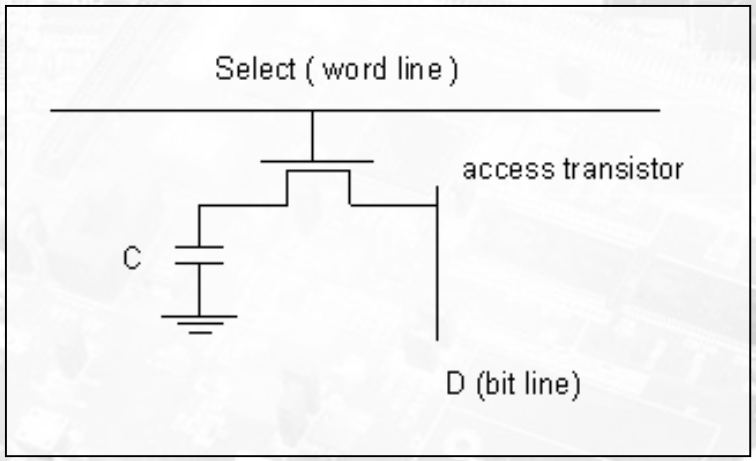
- Memorias SRAM (static RAM):
  - Volátil (mantiene datos mientras esté alimentada).
  - Cada celda de memoria es un flip-flop.
  - Poco densa (niveles de integración moderados).
  - Mantienen los datos con bajo consumo (CMOS).
  - Usualmente usada como memoria de datos en µC.
  - Interface sencilla.

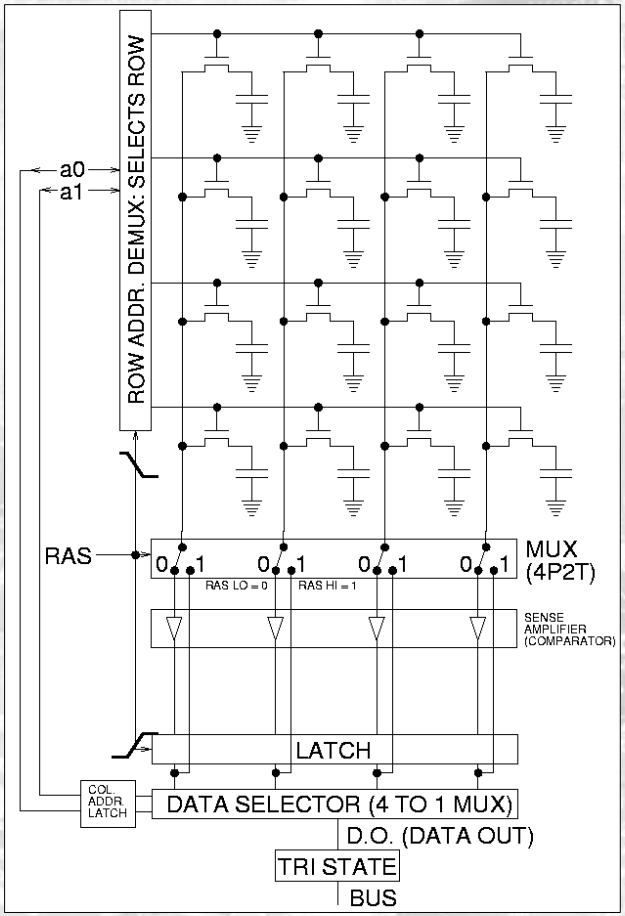


Celda SRAM

Memorias

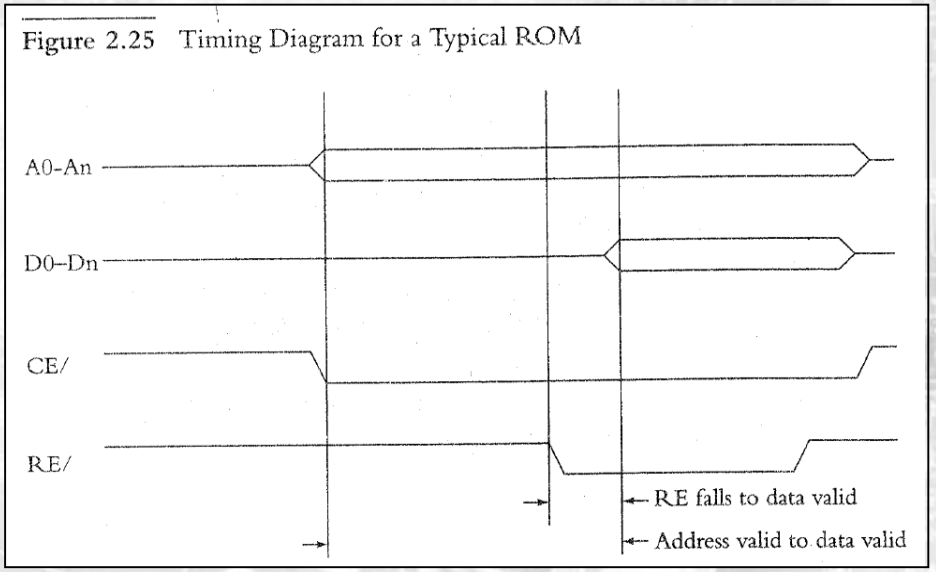
- Memorias DRAM (dynamic RAM):
  - Celdas basadas en capacitores.
  - Volátil (mantiene datos por cortos intervalos de tiempo).
  - Necesita ser refrescada periódicamente (leyéndola).





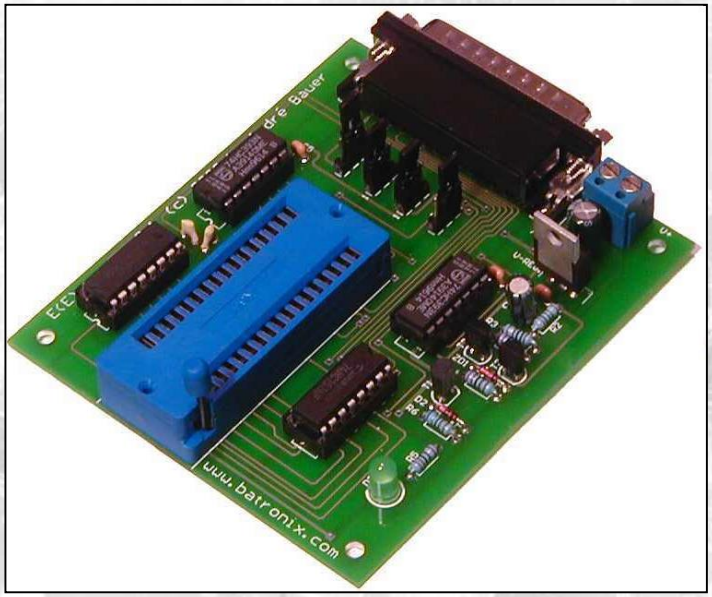
Memorias

- Memorias ROM (masked)
  - No volátil.
  - Lectura rápida.
  - La escritura se realiza durante el proceso de fabricación y es irreversible.
  - Interface sencilla.
  - CE: Chip enabled
    RE: Read enabled



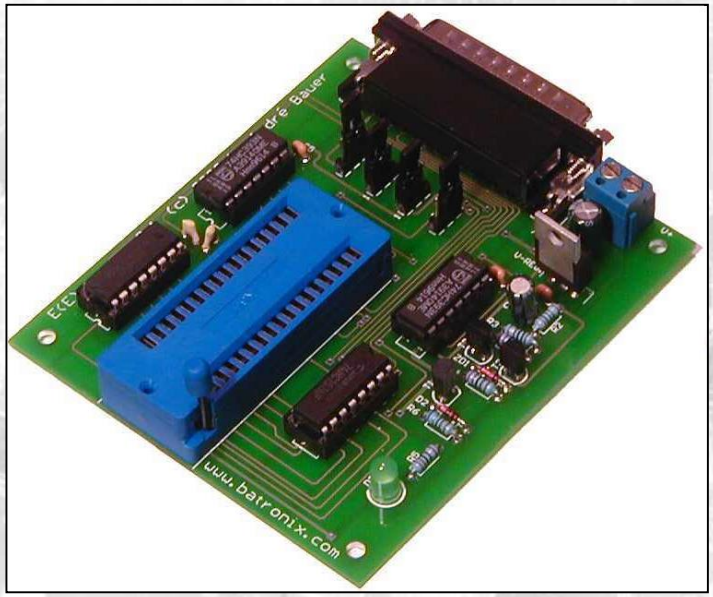
Memorias

- Memorias PROM (Programmable ROM)
  - Como una masked ROM pero...
  - La escritura se realiza mediante un programador y sólo puede realizarse una vez.



Memorias

- Memorias EPROM (Erasable Programmable ROM)
  - Como una PROM pero...
  - La escritura se realiza mediante un programador.
  - Se pueden borrar con luz ultravioleta.
  - Muy densas (1 trans. x celda).
  - Poco práctica (ventana)



Memorias

- Memorias EEPROM (Electrically Erasable Programmable ROM)
  - Como una EPROM pero...
  - La escritura/borrado se realiza eléctricamente sin HW adicional (un byte por vez).
  - Útil para guardar valores de configuración individuales.
  - Se degrada con las escrituras/lecturas.
  - Menos densas que las EPROMs.

Memorias

- Memorias Flash
  - Constituyen la evolución de las memorias EEPROMs.
  - Alta densidad (como las EPROMs).
  - Borrable en bloques (no byte a byte).
  - Lectura rápida, escritura lenta.
  - Se utilizan algoritmos de wear leveling.

Memorias

- Memorias NVRam
  - Memoria RAM no volátil
  - No tiene limitaciones en los ciclos de escritura/borrado (como las memorias Flash)
  - Escrituras palabra a palabra (no en bloques).
  - Similares en estructura a las DRAM. El tipo más común (RAM Ferroeléctrico – F-RAM/FeRAM) posee un material ferroeléctrico polarizable en lugar del dieléctrico existente en el capacitor de cada celda.
  - La estructura cristalina preserva polaridad sin energía (no requiere refresco ni energía p/mantener la información).
  - Mayor velocidad y menor consumo.

Memorias

- Comparación de tecnologías de memoria

Memory type	Volatile?	Writable?	Erase/rewrite size	Erase/rewrite cycles	Relative cost	Relative speed
SRAM	Yes	Yes	Byte	Unlimited	Expensive	Fast
DRAM	Yes	Yes	Byte	Unlimited	Moderate	Moderate
Masked ROM	No	No	N/A	N/A	Inexpensive (in quantity)	Slow
PROM	No	Once, with programmer	N/A	N/A	Moderate	Slow
EPROM	No	Yes, with programmer	Entire chip	Limited (see specs)	Moderate	Slow
EEPROM	No	Yes	Byte	Limited (see specs)	Expensive	Moderate to read, slow to write
Flash	No	Yes	Sector	Limited (see specs)	Moderate	Fast to read, slow to write
NVRAM	No	Yes	Byte	None	Expensive	Fast

Buses

Buses

- Buses: las líneas de comunicación entre el procesador y los diversos
  - Data bus
  - Instruction bus
  - Address (Data/Instr) bus

Figure 3.3 Another Look at the Address Space

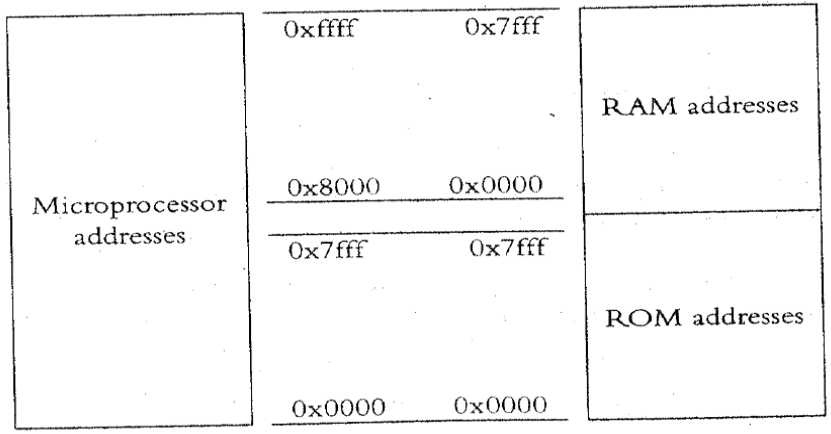
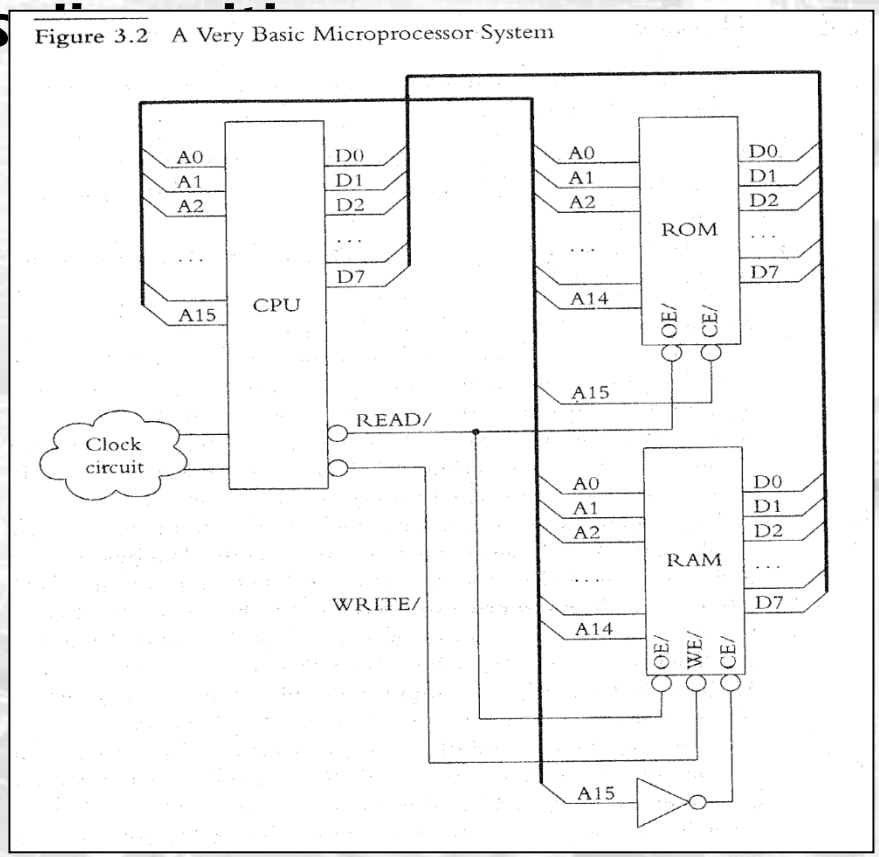


Figure 3.2 A Very Basic Microprocessor System

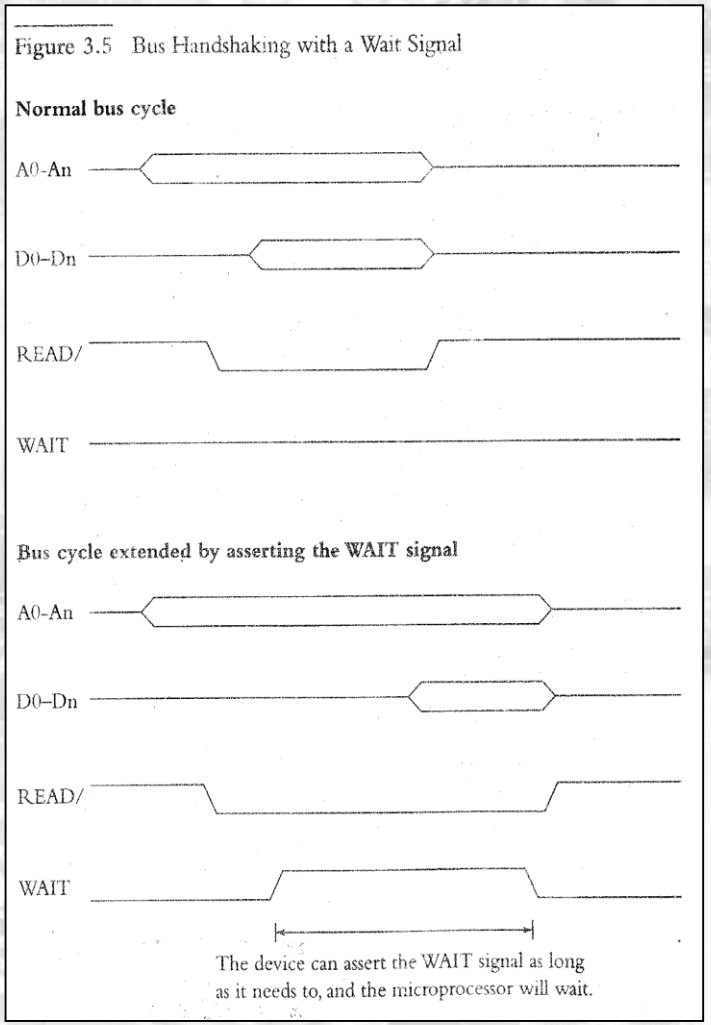


Buses

- Arbitraje de bus:
  - bus handshaking: CPU respeta protocolo y temporizados para obtener/proveer datos estables (bus cycle).
  - no handshaking: sólo si el CPU es más lento que todos los demás componentes (el CPU acciona las señales sin restricciones).
  - Uso de wait signals vs. wait states

Buses

- Wait signals:
  - los dispositivos fuerzan al CPU a que espere el tiempo necesario activando la línea de Wait
  - requiere soporte en dispositivos...



Buses

- Wait states:
  - CPU inserta ciclos de espera para obtener el dato del dispositivo
  - no requiere HW adicional en dispositivos...

Figure 3.6 The Microprocessor Clock Times the Bus

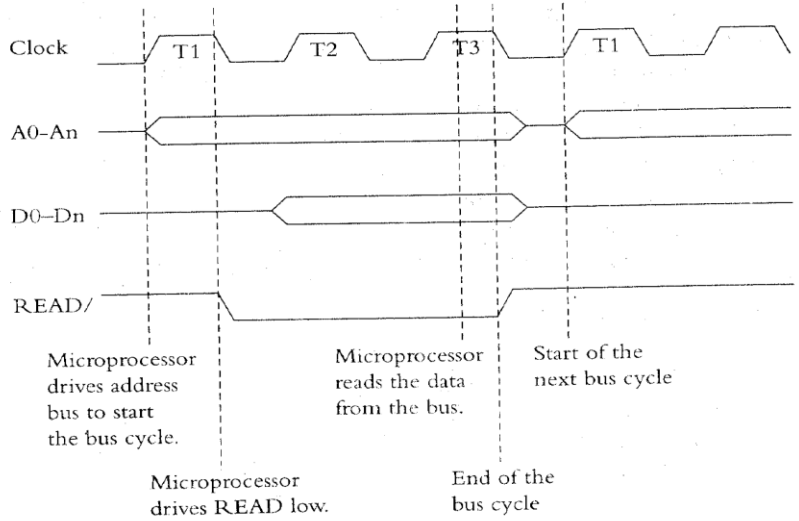
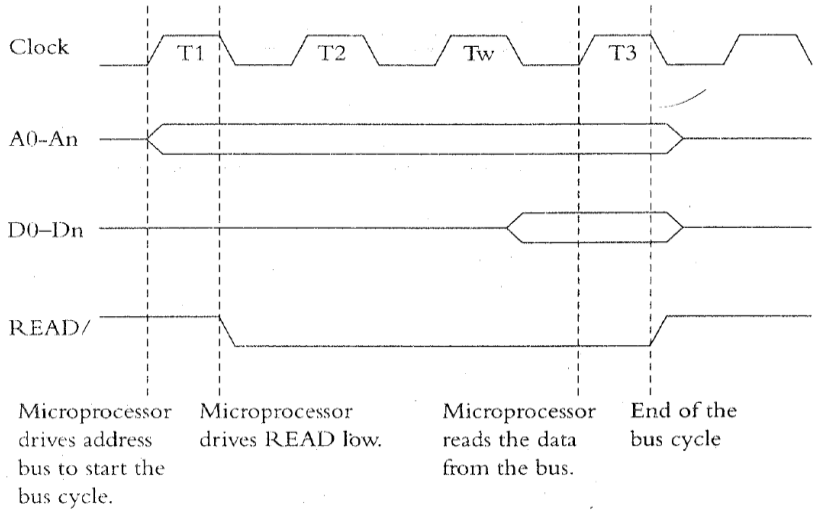
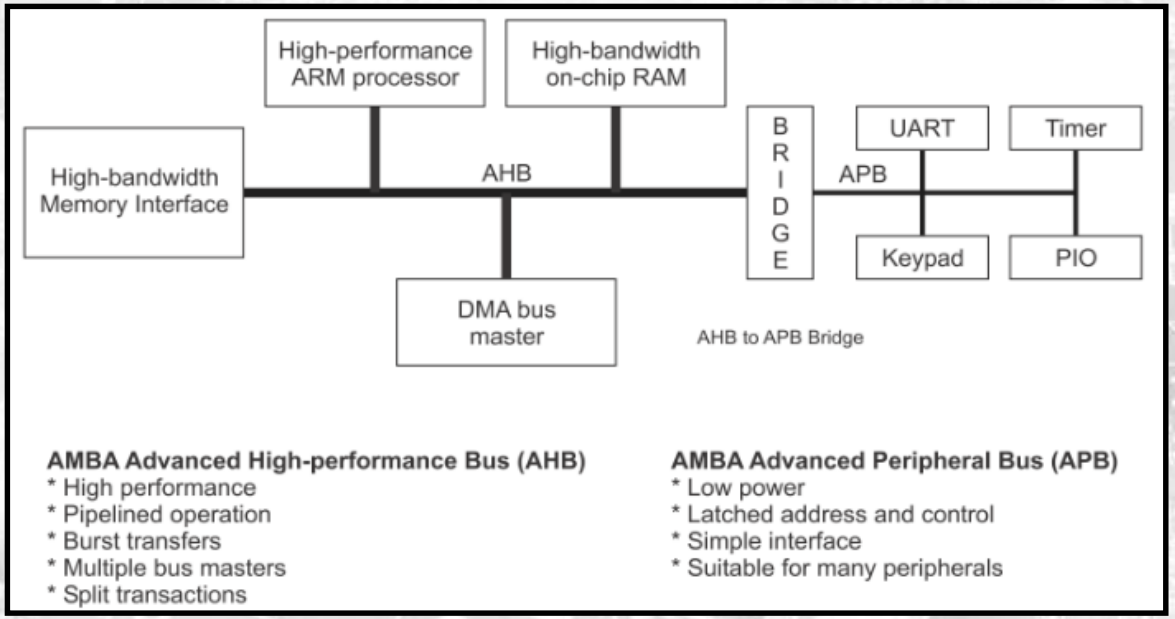


Figure 3.7 The Microprocessor Adds a Wait State



Buses - AMBA

- Ej: AMBA (Advanced Microcontroller Bus Arch.)
  - Desarrollado por ARM en 1996.
  - Conjunto de protocolos de interconexión de componentes "on-chip".
  - 4 generac.:
    - AMBA
    - AMBA 2
    - AMBA 3
    - AMBA 4



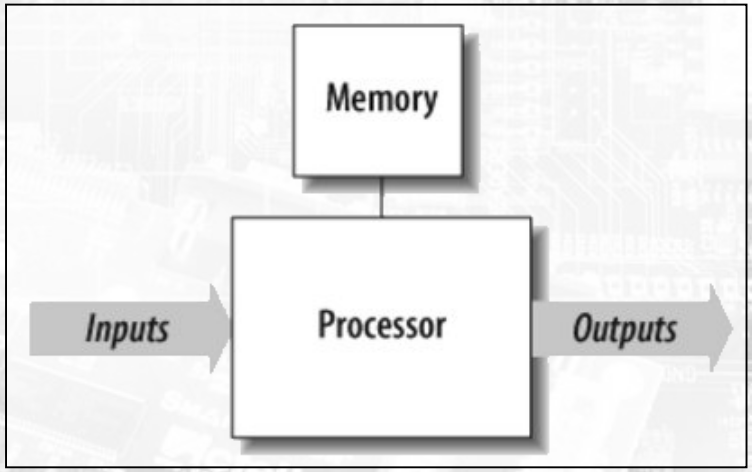
Buses - Ejemplos

- Arquitecturas de buses (on chip):
  - AMBA
  - IBM CoreConect (Power, Xilinx Microblaze)
  - Altera Avalon (Nios II)
- Alternativas (off chip)
  - Hyper Transport (AMD)
  - PCI Express
  - Rapid IO
- Veremos otros protocolos de interconexión:
  - I2C, SPI, CAN Bus, etc.

Entrada/Salida

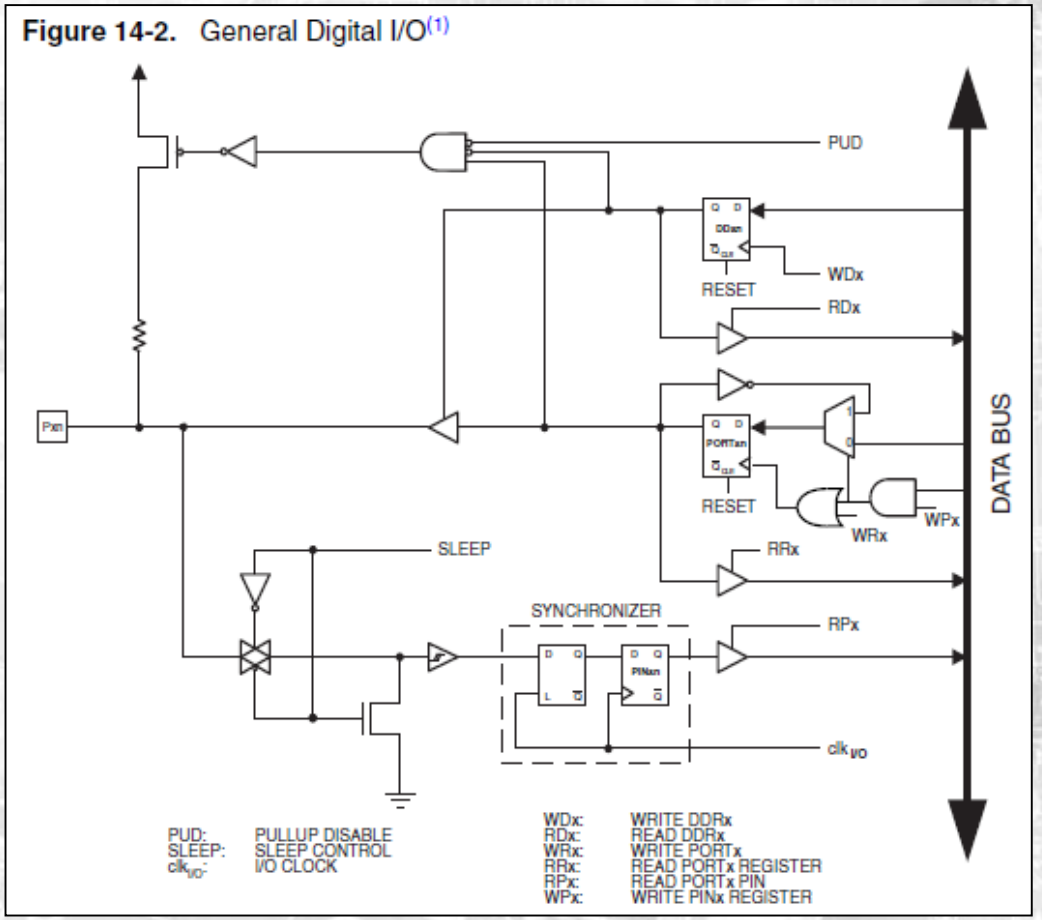
Entrada/Salida

- E/S: comunicación con el exterior
  - Memory Mapped I/O vs Port Mapped I/O (instrucciones generales vs particulares)
  - Transferir datos desde y hacia la memoria
  - Mediante:
    - Pines GPIO
    - Puertos
    - Dispositivos
  - Polling de dispositivos
  - Interrupciones



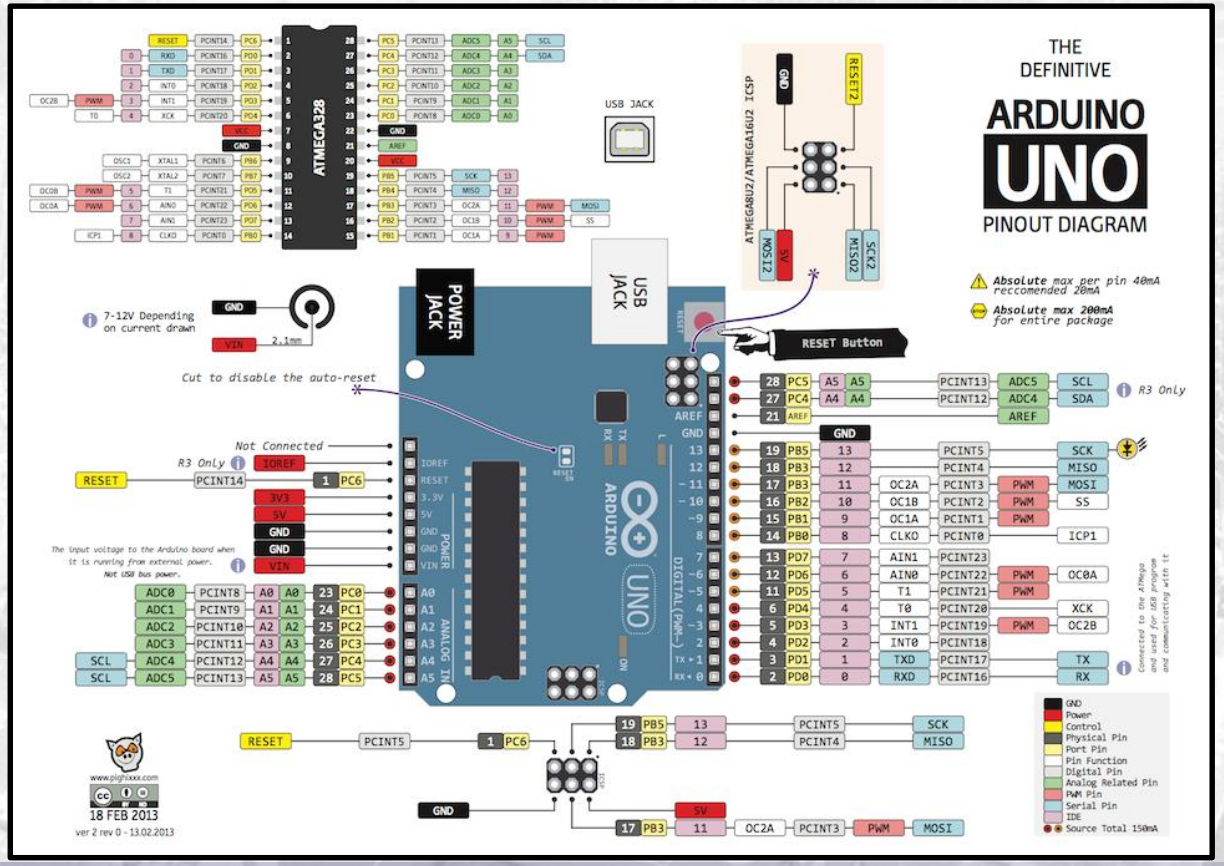
Entrada/Salida - Puertos

- Ej: puerto bidireccional programable (ATmega328P)



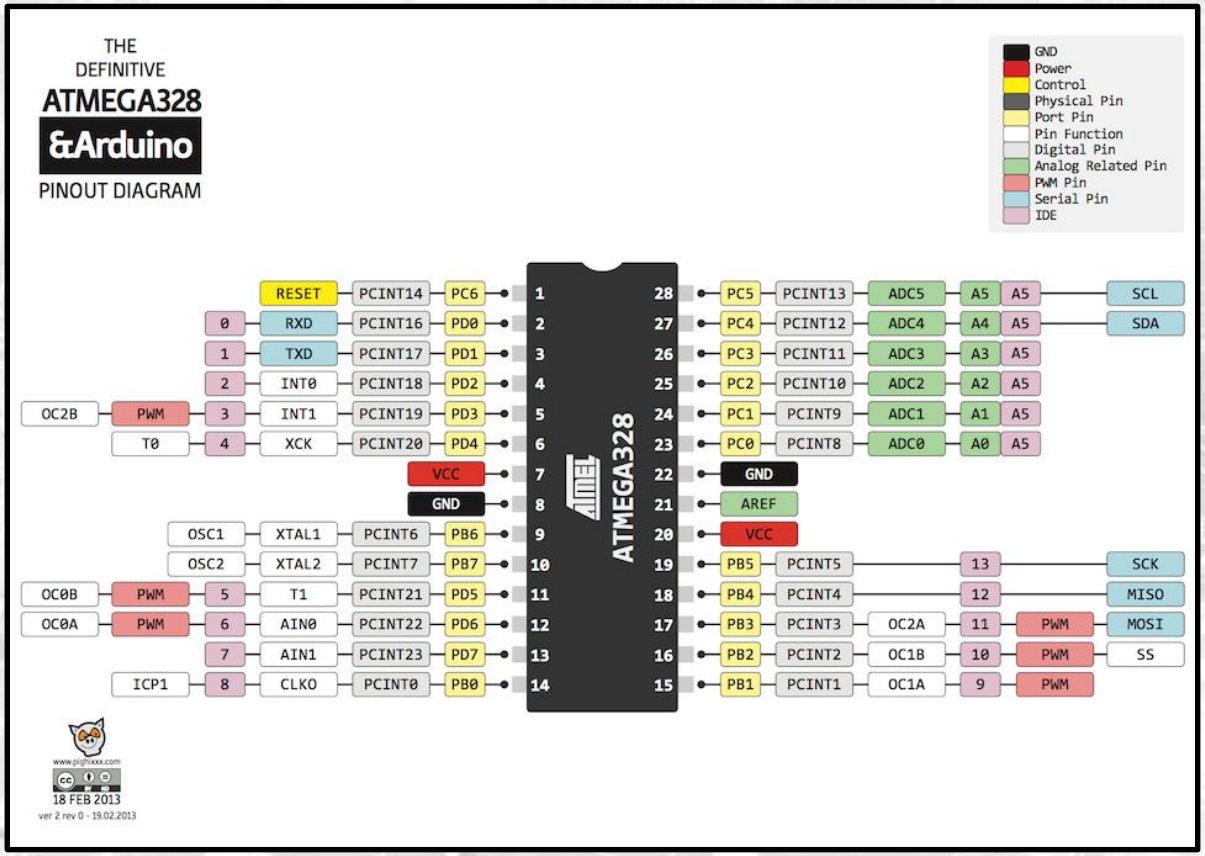
Entrada/Salida

- Pinout Diagram: Arduino Uno y ATMega328P



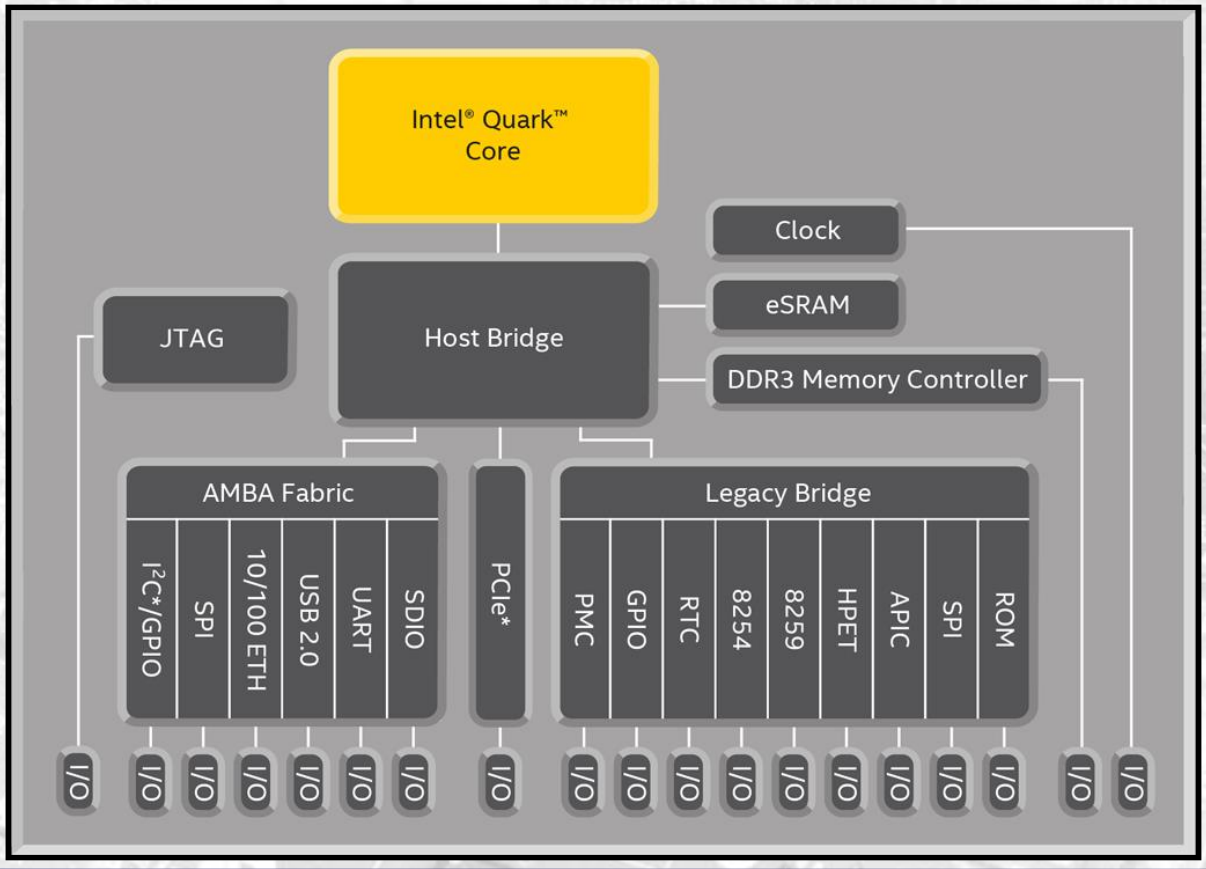
Entrada/Salida

- Pinout Diagram: Arduino Uno y ATMega328P



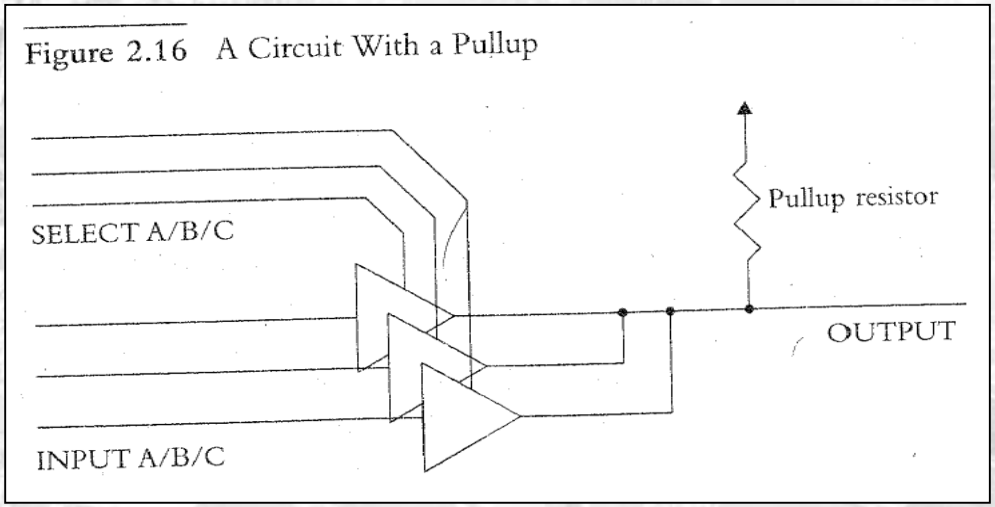
Entrada/Salida

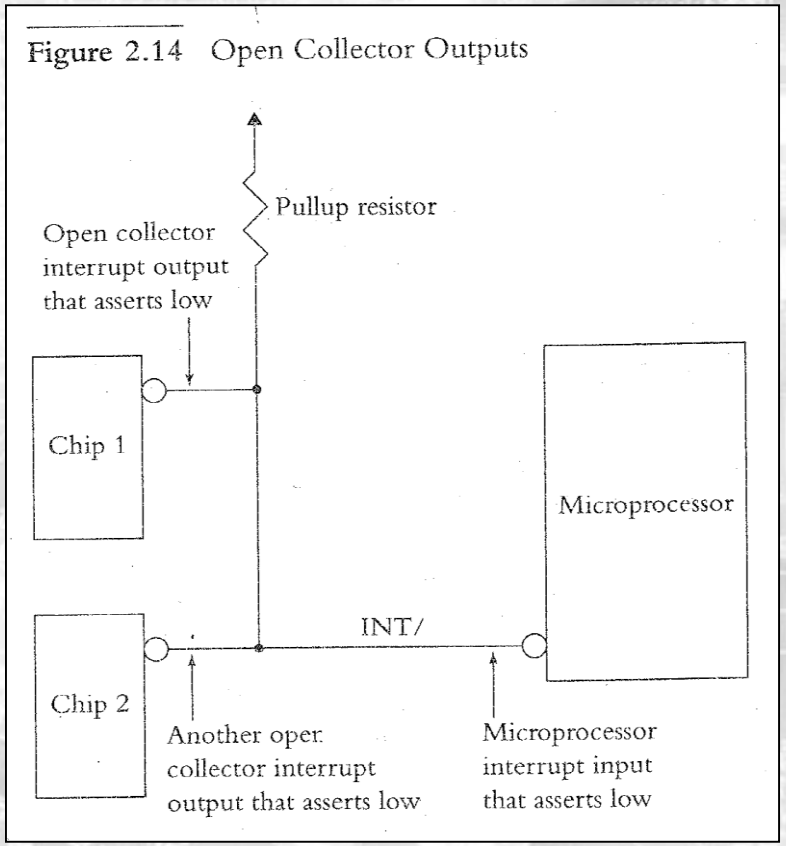
- Block Diagram: Intel SoC Quark X1000 (Galileo)



Entrada/Salida - Puertos

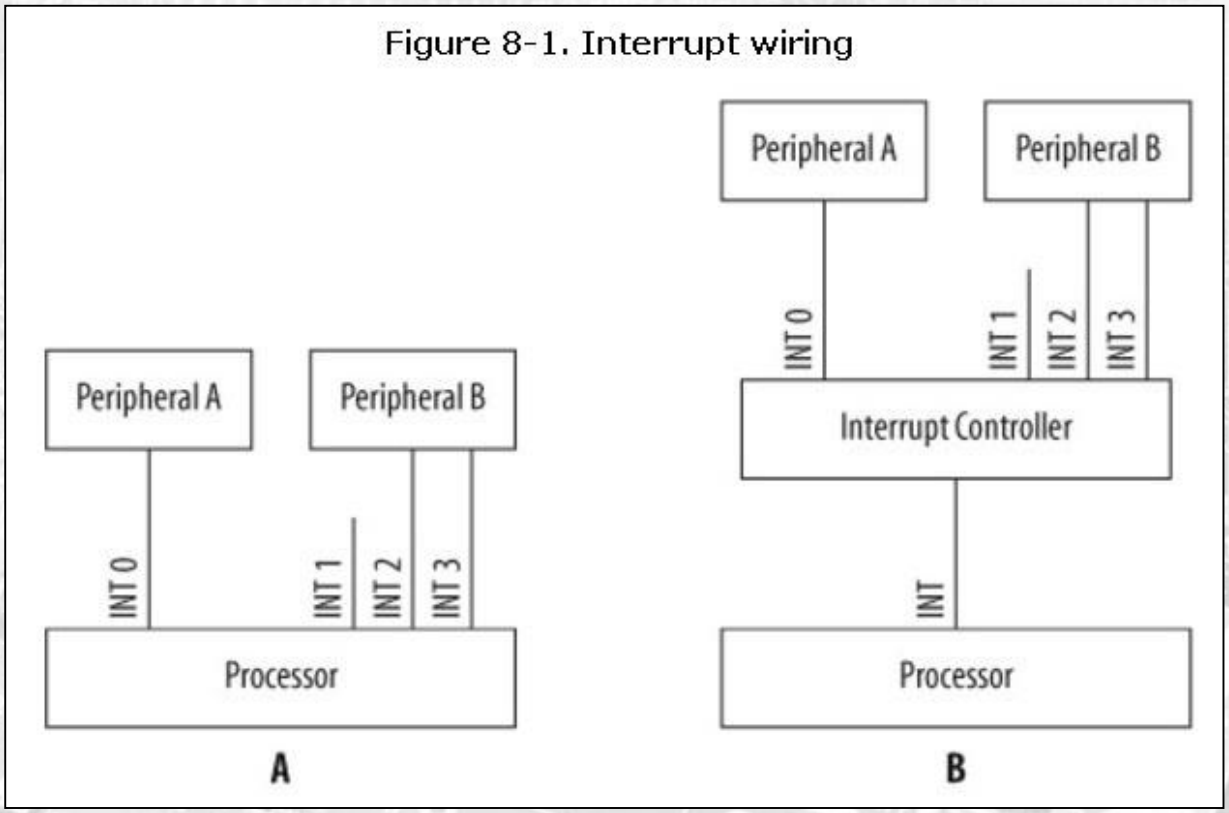
- Salidas Open Collector y Tri-state...
  - Para evitar conflictos en los buses y líneas compartidas.





Entrada/Salida - Interrupciones

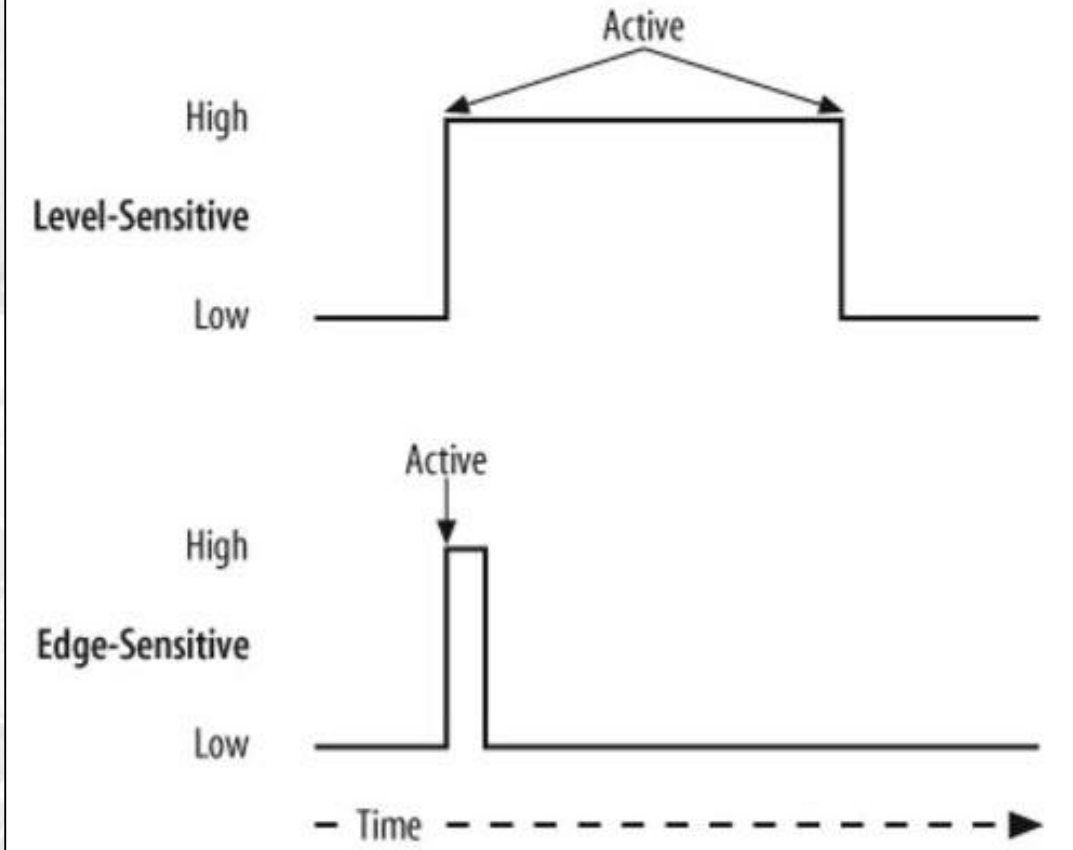
- *Alternativa al polling de dispositivos: Interrupciones*



Entrada/Salida - Interrupciones

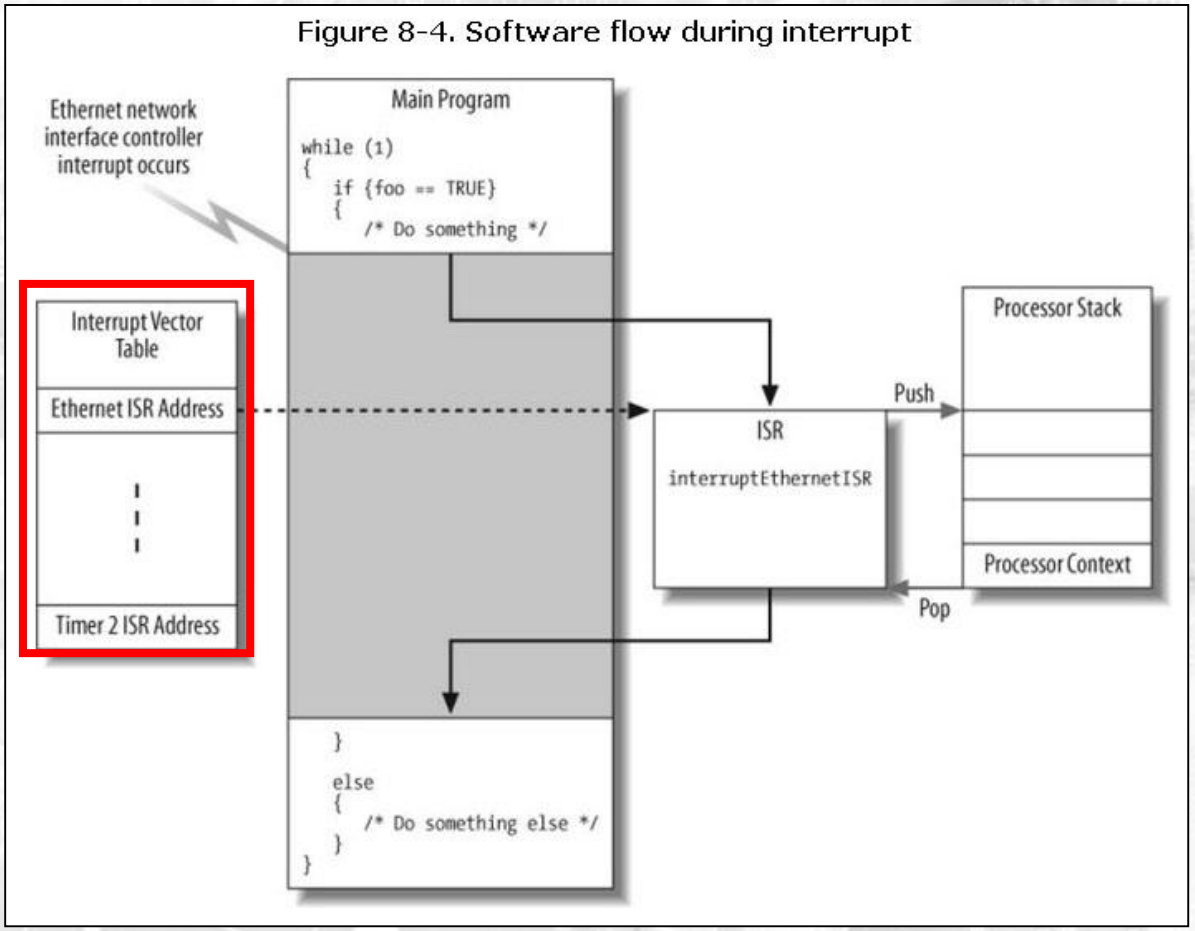
- Interrupciones
  - por niveles
  - por flancos

Figure 8-2. Level- and edge-sensitive interrupt signals



Entrada/Salida - Interrupciones

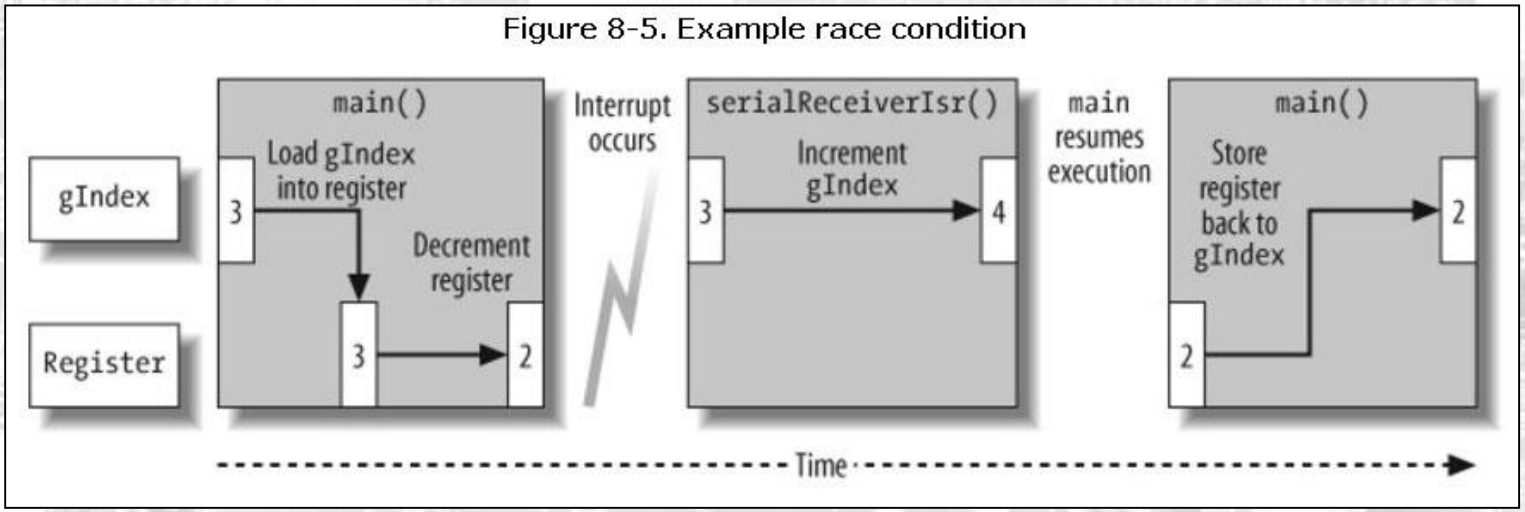
- IRQs
- ISRs
- Interrupciones vectoreadas
- Prioridades
- Anidamiento



Entrada/Salida - Interrupciones

- ISRs (Interrupt service routines)
  - Salvar/restaurar el contexto
  - Desactivar ints (no NMI) el menor tiempo posible.
  - Sección crítica / atomicidad
  - Reentrancia

Figure 8-5. Example race condition



Entrada/Salida - Interrupciones

- Ej: Condiciones de carrera (código de alto nivel y de bajo nivel).

Figure 4.5 Harder Shared-Data Problem

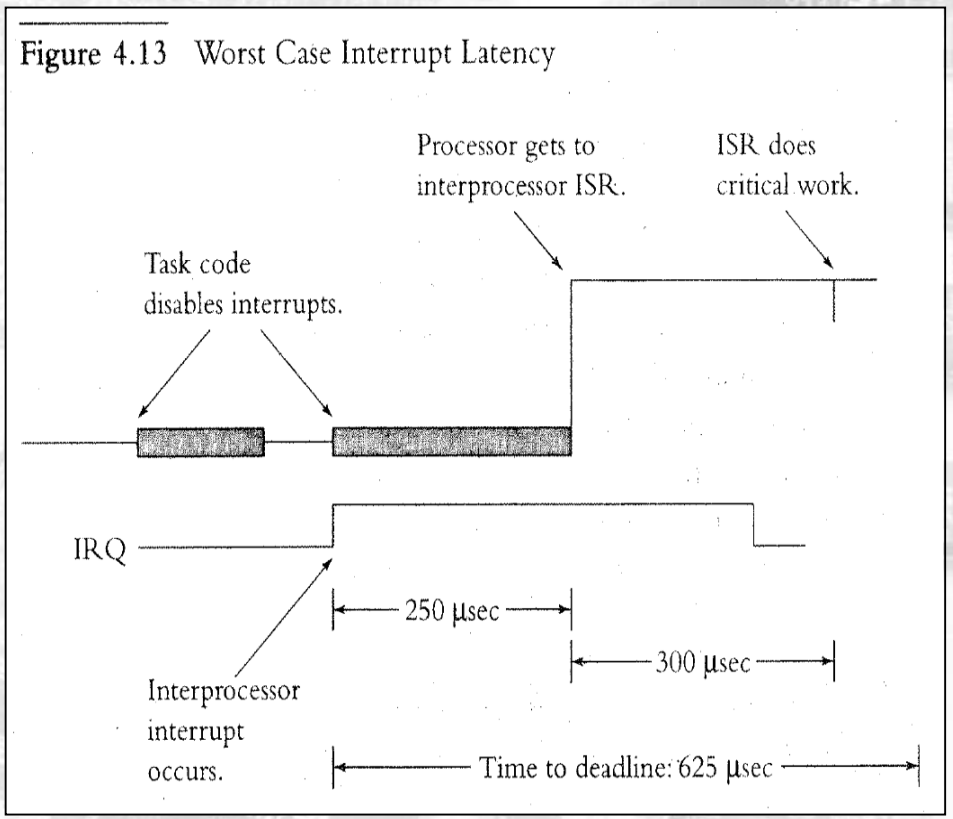
```
static int iTemperatures[2];

void interrupt vReadTemperatures (void)
{
   iTemperatures[0] = !! read in value from hardware
   iTemperatures[1] = !! read in value from hardware
}

void main (void)
{
   while (TRUE)
   {
      if (iTemperatures[0] != iTemperatures[1])
         !! Set off howling alarm;
   }
}
```

Figure 4.6 Assembly Language Equivalent of Figure 4.5

```
      .
      .
      .
        MOVE     R1, (iTemperatures[0])
        MOVE     R2, (iTemperatures[1])
        SUBTRACT R1, R2
        JCOND    ZERO, TEMPERATURES_OK
        .
        .
        .
        ; Code goes here to set off the alarm
        .
        .
        .
TEMPERATURES_OK:
        .
        .
        .
```

Entrada/Salida - Interrupciones

- La ISR debe:
  - atender la interrupción lo más rápido posible (bajar la latencia)
  - desactivar (sólo si es necesario), las interrupciones durante el menor tiempo posible
  - no hay que sacrificar la respuesta temporal del sistema



Entrada/Salida - Interrupciones

- Cuidado con las optimizaciones del copilador: volatile C keyword...

```
static volatile long int lSecondsToday;
```

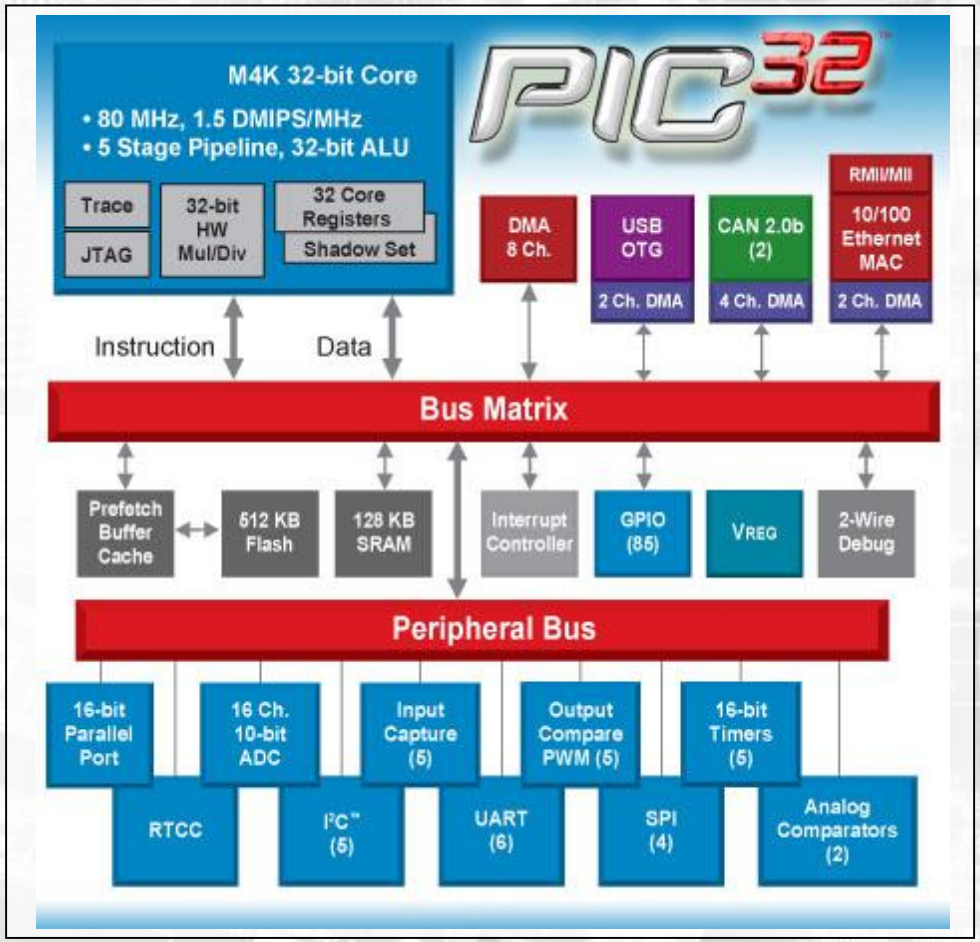
- Se evita el reordenamiento de código y otras optimizaciones del compilador.

Figure 4.12 A Program That Needs the volatile Keyword

```
static long int lSecondsToday;

void interrupt vUpdateTime (void)
{
   .
   .
   .
   ++lSecondsToday;
   if (lSecondsToday == 60L * 60L * 24L)
      lSecondsToday = 0L;
   .
   .
   .
}

long lSecondsSinceMidnight (void)
{
   long lReturn;

   /* When we read the same value twice, it must be good. */
   lReturn = lSecondsToday;
   while (lReturn != lSecondsToday)
      lReturn = lSecondsToday;

   return (lReturn);
}
```

E/S – Dispositivos e Interfaces

- Como parte de un sistema embebido se cuenta con dispositivos:
  - implementan funcionalidad particular
  - implementan interfaces estándares para la interacción entre sistemas



Referencias

- AMBA Specification.
- Atmel AVR ATmega328P Datasheet.
- Barr, M., Massa, A. Programming Embedded Systems: With C and GNU Development Tools, 2nd Edition. O'Reilly Media. 2006. ISBN: 978-0596009830. Capítulos 2, 6, 7 y 8.
- Intel® Quark™ SoC X1000 Datasheet
- Simon, D. An Embedded Software Primer. Addison-Wesley Professional. 1999. ISBN: 978-0201615692. Capítulos 2, 3 y 4.
- Wilmshurst, T. Designing Embedded Systems with PIC Microcontrollers: Principles and Applications. Newnes. 2006. ISBN: 978-0750667555. Capítulos 1, 2 y 3.