Microcontroladores

Sistemas embebidos para tiempo real

Objetivos

- Describir los conceptos y bloques básicos de microcontroladores
  - CPU, ISA, arquitectura
- Comprender la importancia de conocer el uC
- Utilizar e interpretar manuales de usuario

Agenda

- Repaso:
  - uP vs. uC, CPU, tamaño de palabra.
- Arquitectura
  - RISC vs. CISC
  - Harvard vs. von Neumann
- Comparación AVR vs. MSP430

uP vs. uC

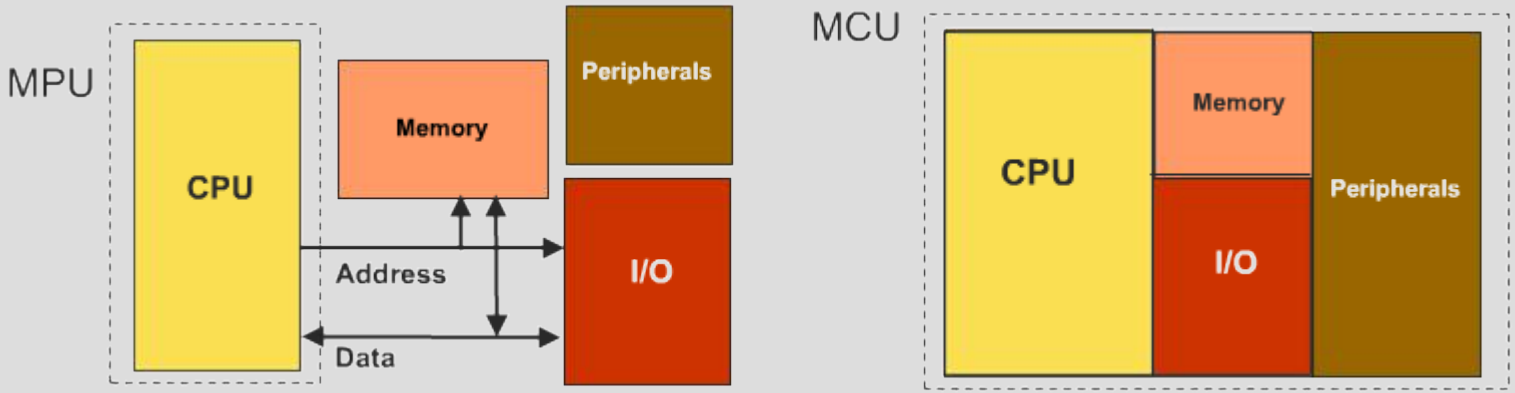


Figura modificada de “Fundamentals of Microcontrollers” de John Donovan (NXP)

CPU: unidad central de proc. (core)

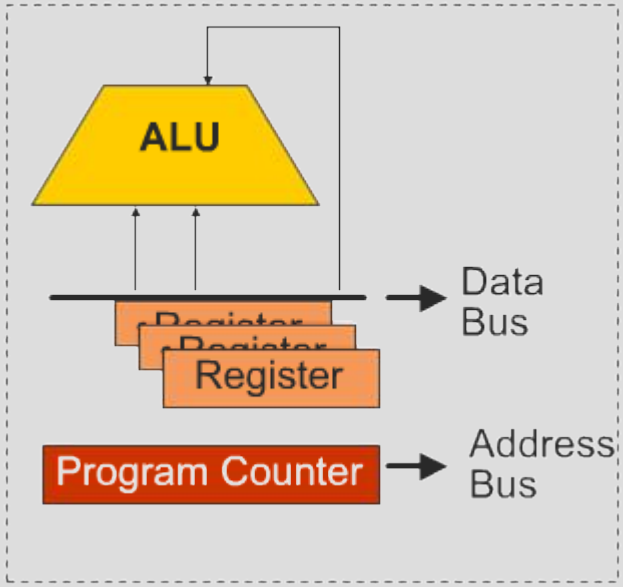
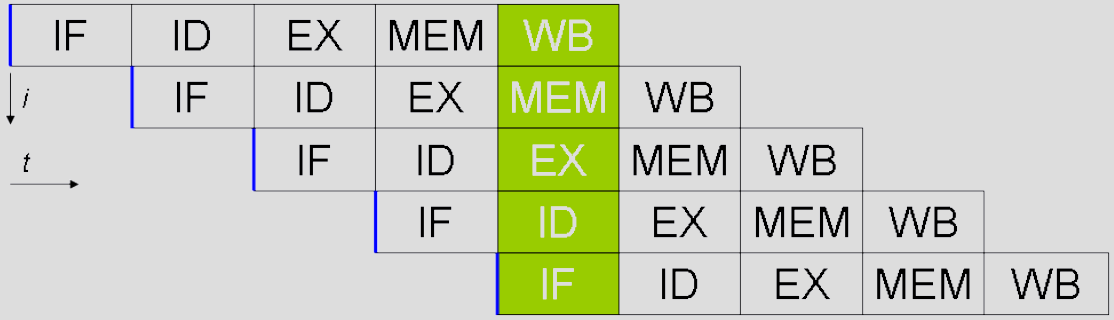
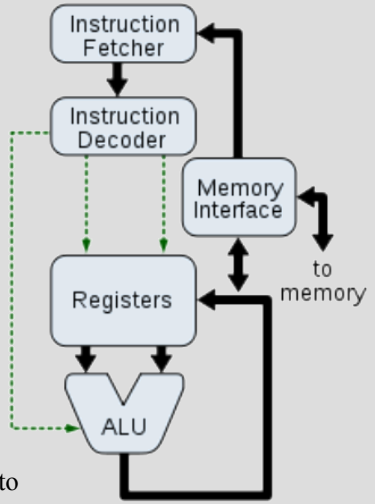


Figura extraída de “Fundamentals of Microcontrollers” de John Donovan (NXP)

- Tipos:
  - Acumulador
  - Registros
  - Pila (stack)

CPU: pipeline





Figuras extraídas de http://es.wikipedia.org/wiki/Unidad_central_de_procesamiento

Tamaño de palabra

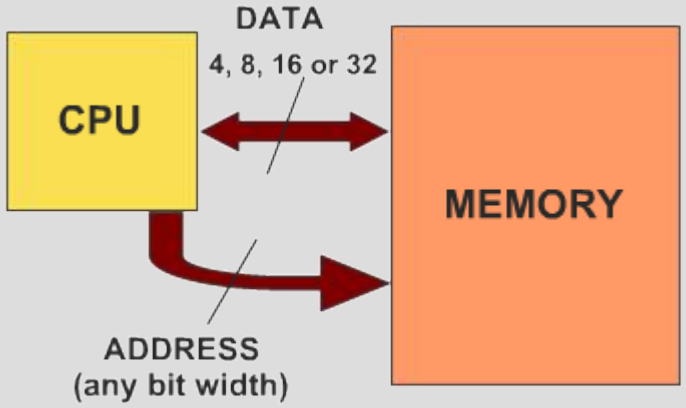


Figura extraída de “Fundamentals of Microcontrollers” de John Donovan (NXP)

- Bus de datos
- Bus de direcciones

Procesadores CISC vs. RISC

Complex Instruction Set Computing

Reduced Instruction Set Computing

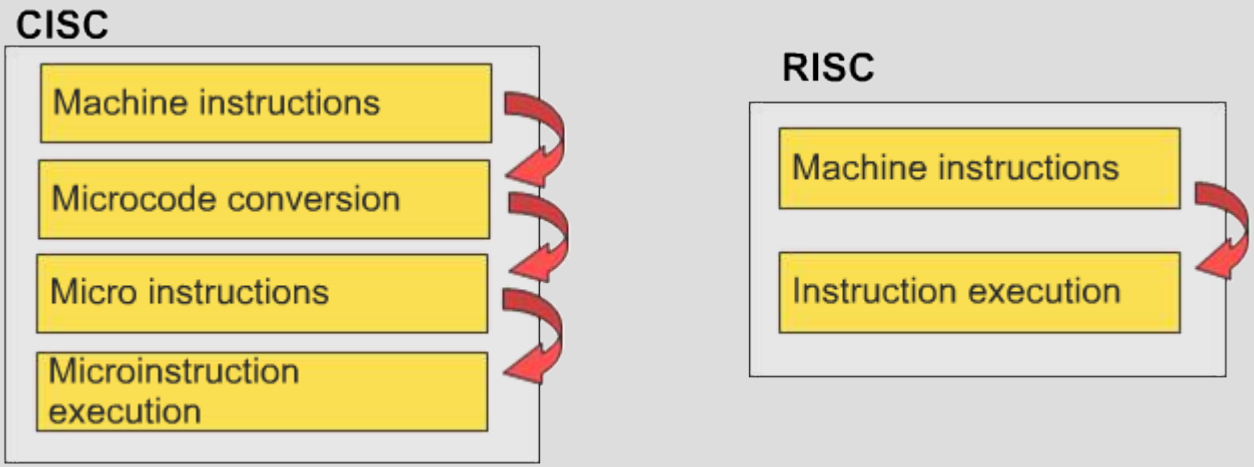


Figura modificada de “Fundamentals of Microcontrollers” de John Donovan (NXP)

Procesadores CISC vs. RISC

- Procesador CISC (Complex Instruction Set Computing)
  - Instrucciones de largo variable
  - Decodificación de instrucciones complejo (microcoding)
  - Número de ciclos de reloj de ejecución variable
    - Ejemplo: shift and rotate (2 ciclos), integer multiply (~ 80)

- Procesador RISC (Reduced Instruction Set Computing)
  - Conjunto de instrucciones reducidas y ortogonal
  - Instrucciones de tamaño fijo (o varía muy poco) y similar formato
  - Decodificación de instrucciones más fácil
  - Ejecuta misma (aprox.) cantidad de ciclos

Arquitectura Harvard vs. von Neumann

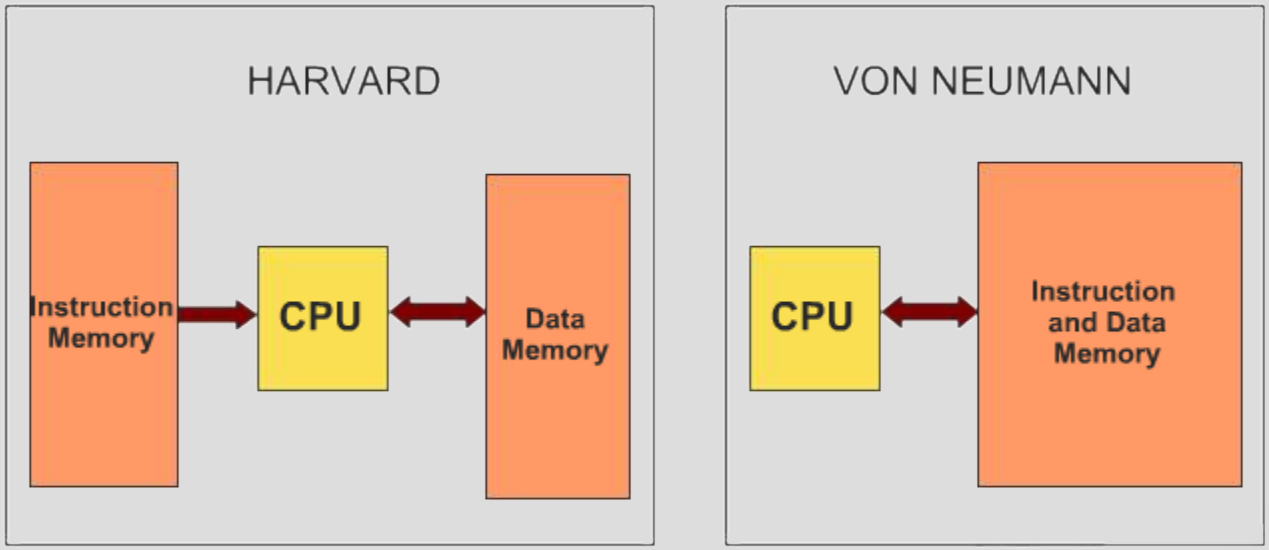
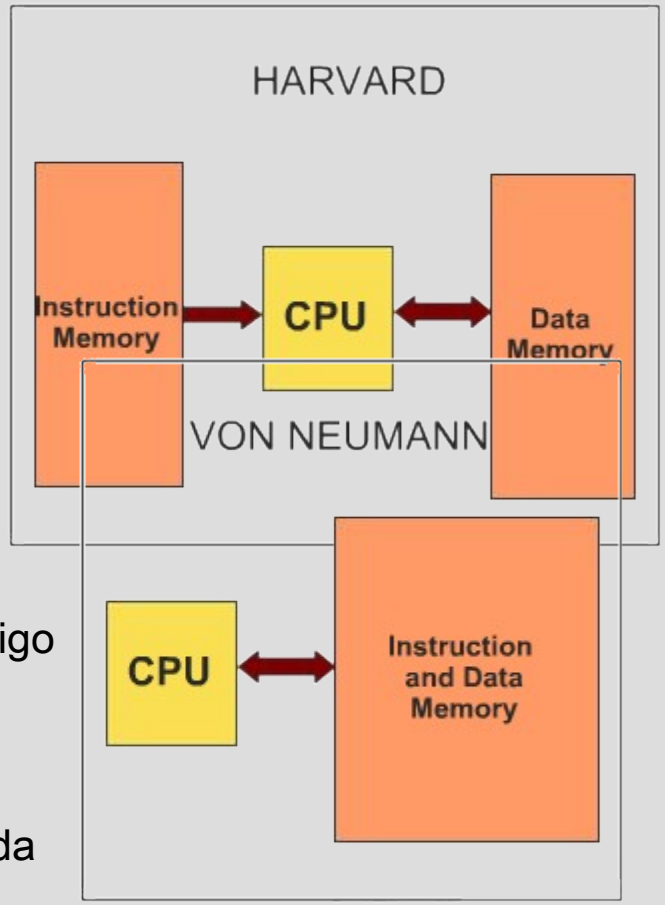


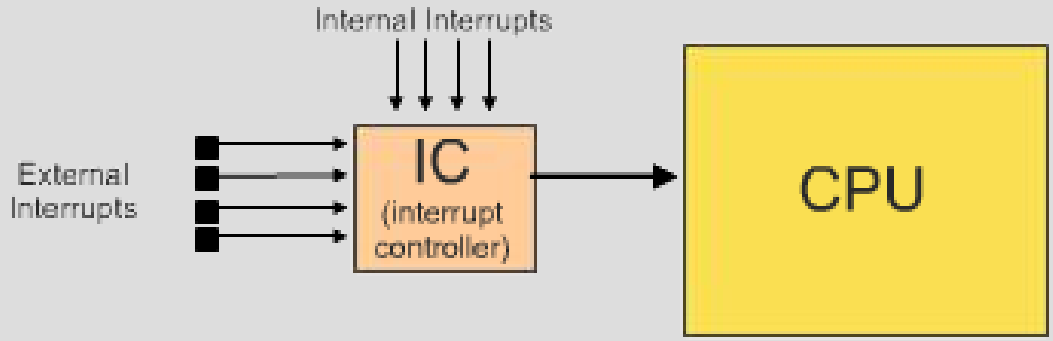
Figura modificada de “Fundamentals of Microcontrollers” de John Donovan (NXP)

Arquitectura Harvard vs. von Neumann

- Arquitectura Harvard
  - Memorias de código y datos
    - Espacio de memoria diferentes
    - Caminos separados hacia la CPU.
  - Más rápida
- Arquitectura von Neumann
  - Código y datos son mapeados en uno solo espacio.
  - Más flexible:
    - Fácilmente puede escribir en memoria de código
    - Pueden ejecutar código desde RAM
- Arquitectura Harvard modificada
  - Contenido de la memoria de código puede ser leída como datos.



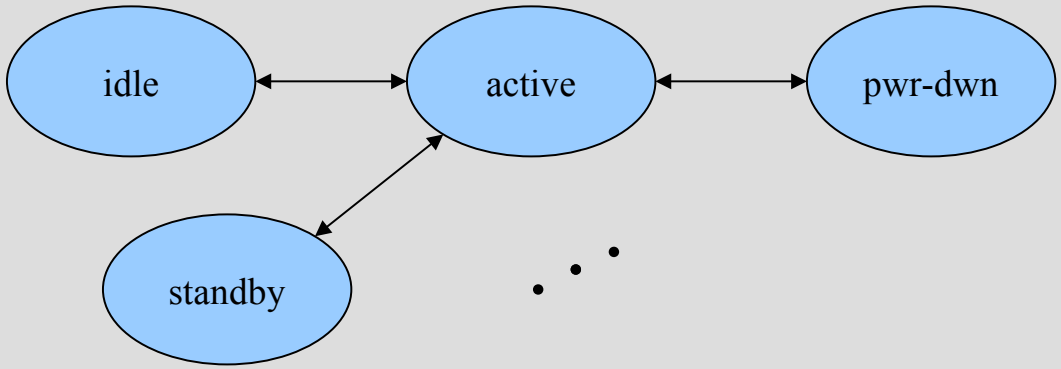
Controlador de interrupciones



Fuente: Figura modificada de “Fundamentals of Microcontrollers” de John Donovan (NXP)

- Interrupciones: fundamental importancia
  - Señal asíncrona
  - Flujo de ejecución del programa
  - Arquitectura de software

Modos de operación



- Idea básica:
  - Modos de bajo consumo: CPU y periféricos no usados.
  - Prever mecanismo para reactivarse.
  - Modo de operación compatible con arquitectura

Periféricos

- Puertos digitales de E/S (I/O pin)
- Temporizadores (Timers)
  - Watchdog timer (WDT)
- Conversores A/D y D/A (ADC / DAC)
- Interfaz/protocolos de comunicación
  - E/S de datos digitales (UART, SPI, I2C)
- DMA
- Especializados (no siempre disponibles)
  - MPU / MMU (Memory Protection Unit / Memory Management Unit)
  - SVS

Watchdog Timer (WDT)

- Permite realizar un reset
- Protege de problemas de software
- Ejemplo:
  - Habilitado por defecto!
  - Si no quiero que resetee:
    - `WDTCTL = WDTHOLD | WDTPW; //deshabilita WDT`

Interfaces E/S

- Seriales
  - Asíncorono
    - UART (Universal Asynchronous Receiver / Trasmitter)
  - Síncronos
    - SPI (Serial Peripheral Interface)
    - I2C (Inter-Integrated Circuit Circuit)

Interfaces E/S

- Más complejos y mayores funcionalidades:
  - USB (Universal Serial Bus)
  - CAN (Controller Area NEtwork)
  - Ethernet
  - IEEE 802.15.4/Zigbee

Paseo guiado

- Documentos técnicos del MSP430G2553 (uC del laboratorio):
  - MSP430x2xx Family User's Guide (SLAU144J.pdf)
  - MSP430G2x53 MSP430G2x13 Mixed Signal Microcontroller (SLAS735J.pdf) Datasheet
  - MSP430G2553 Device Erratasheet (SLAZ440I.pdf)
  - MSP430G2553 LaunchPad Development Kit (SLAU772.pdf)
  - MSP430 Optimizing C/C++ Compiler v18.1.0.LTS User's Guide (SLAU132R.pdf)
  - Code Composer StudioTM IDE v8.x for MSP430TM MCUs (SLAU157AR.pdf)

Actividad en grupo

- Diferencias: ATmega32 versus MSP430 (familias)
  - Actividad: Comparar los μC en función de:
    - Tipo procesador, N-bits, frecuencia máxima
    - Arquitectura y mapa de memoria
    - Registros (cantidad, uso, etc.)
    - Características del PC, SR y SP
    - Modos de direccionamiento
    - Set de instrucciones
    - Modos de bajo consumo
  - Grupos: 4-5 estudiantes
  - Material: Manuales de los fabricantes
  - Tiempo: 10 minutos

Tabla comparativa

	ATmega32	MSP430
Tipo CPU / Tamaño palabra		
Arquitectura / Mapa de memoria		
Set de instrucciones		
Modos de direccionamiento		
Modos de bajo consumo		
Registros		

Ejemplo de uC: ATmega32

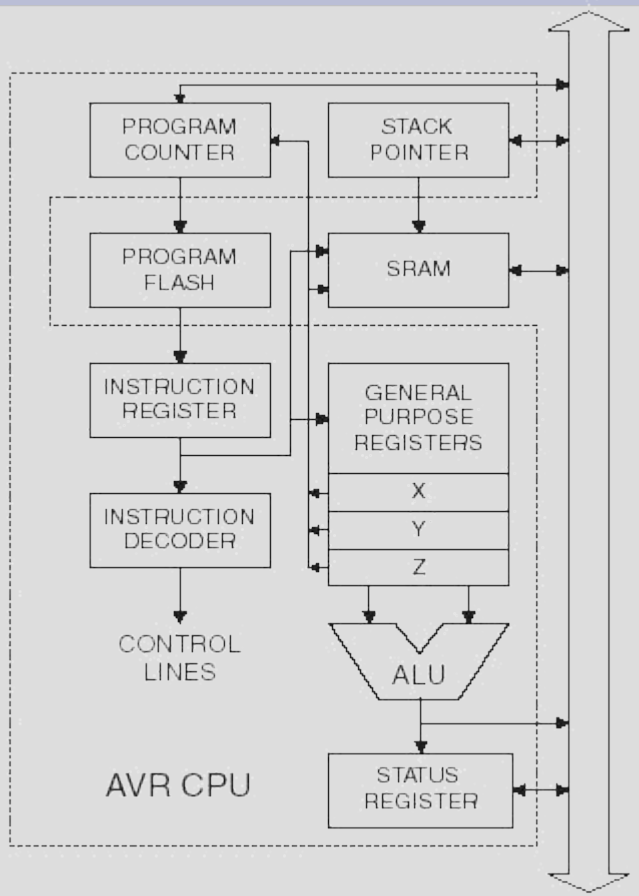
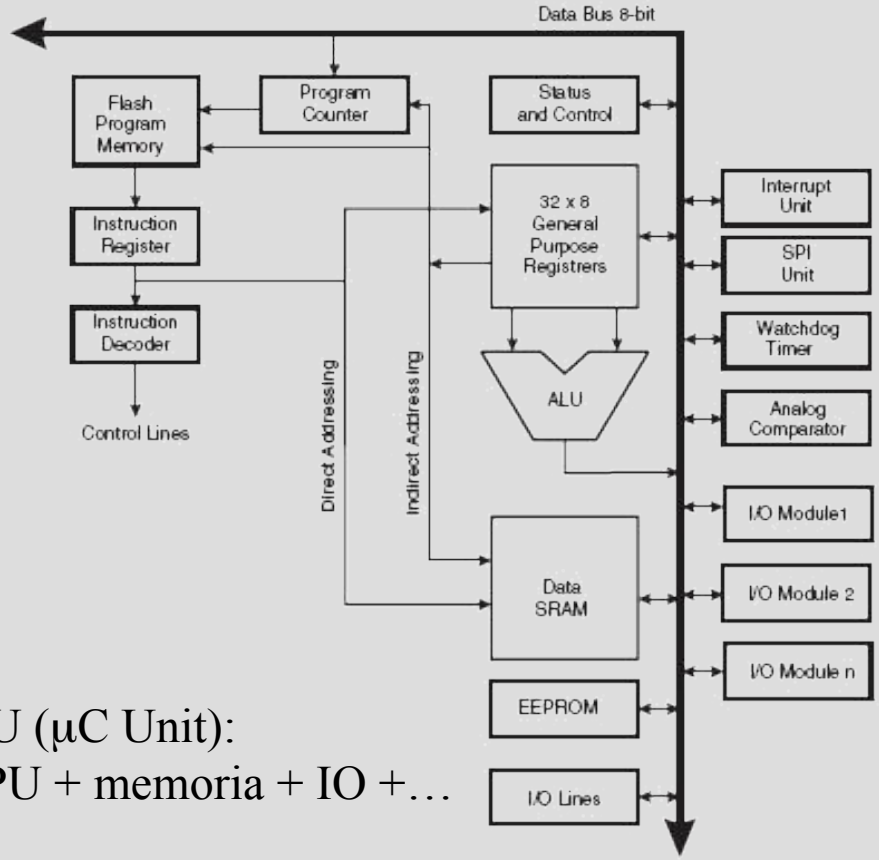


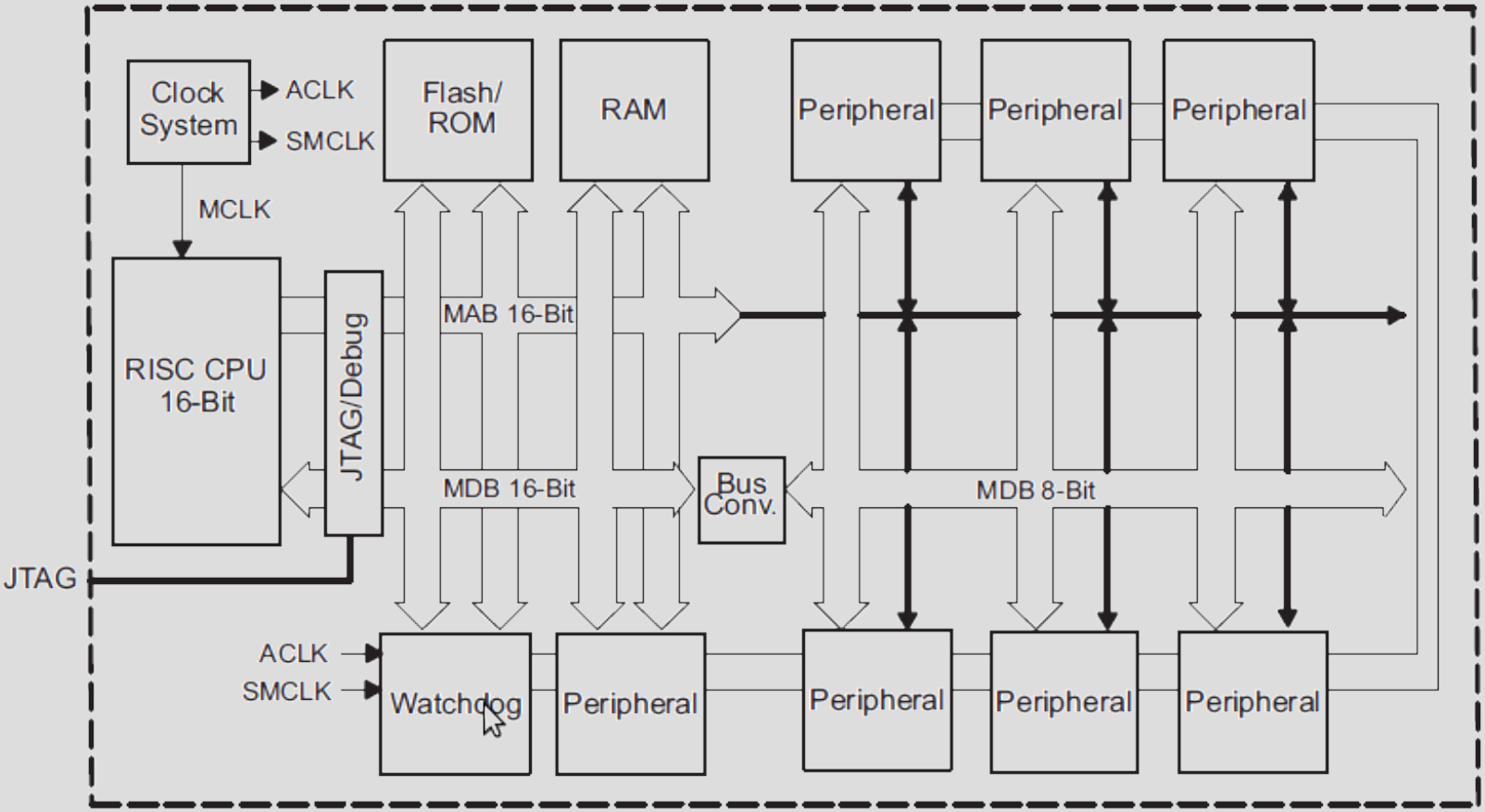
Figura modificada de: "ATmega32 (L)" datasheet (Figure 2. Block Diagram, page 3)



MCU (μC Unit):
CPU + memoria + IO +…

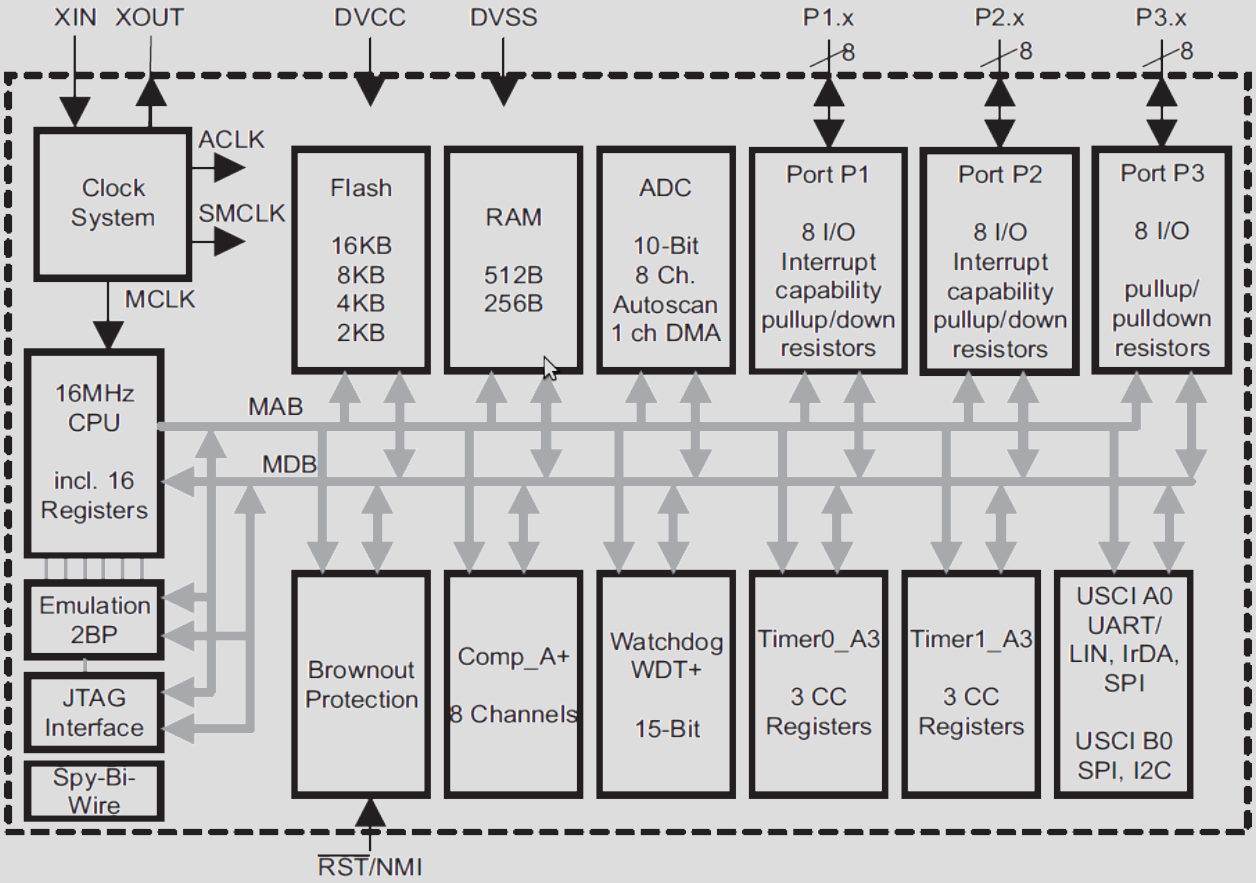
Fuente: "ATmega32 (L)" datasheet (Figure 2. Block Diagram, page 3)

Ejemplo de uC: MSP430 (familia)



Fuente: "MSP430x2xx Family User's Guide“ (file: SLAU144H.pdf) Figure 1-1. MSP430 Architecture, page 26.

Ejemplo de uC: MSP430 (device)



Fuente: "MSP430G2x53 MSP430G2x13 MIXED SIGNAL MICROC." (file: SLAS735J.pdf) Functional Block Diagram, MSP430G2x53, page 5.

Deberes

- Compilación “manual”
  - Actividad
    - Bosquejar el código assembler del siguiente código C
    - Contabilizar:
      - memoria
      - ciclos
  - Grupos:
    - MSP430
    - AVR (ATmega32)
  - Materiales
    - Manuales correspondientes
  - Puesta en común
    - Comparación programas: complejidad, ciclos reloj, uso memoria

```
int a, b, c;

int main( void )
{
  a = 1;
  b = 2;
  c = a + b;
}
```

Comparación

- AVR (ATmega32)

```
main:
    000006   E001            LDI     R16,0x01
    000008   E010            LDI     R17,0x00
    00000A   E6E0            LDI     R30,0x60
    00000C   8300            ST      Z,R16
    00000E   8311            STD     Z+1,R17
  b = 2;
    000010   E002            LDI     R16,0x02
    000012   E010            LDI     R17,0x00
    000014   E6E2            LDI     R30,0x62
    000016   8300            ST      Z,R16
    000018   8311            STD     Z+1,R17
  c = a + b;
    00001A   E6E0            LDI     R30,0x60
    00001C   8100            LD      R16,Z
    00001E   8111            LDD     R17,Z+1
    000020   E6E2            LDI     R30,0x62
    000022   8120            LD      R18,Z
    000024   8131            LDD     R19,Z+1
    000026   0F02            ADD     R16,R18
    000028   1F13            ADC     R17,R19
    00002A   E6E4            LDI     R30,0x64
    00002C   8300            ST      Z,R16
    00002E   8311            STD     Z+1,R17
}
    000030   9508            RET
```

- MSP430

```
main:
 001118     4392 0200          mov.w   #0x1,&a
  b = 2;
 00111C     43A2 0202          mov.w   #0x2,&b
  c = a + b;
 001120     421F 0200          mov.w   &a,R15
 001124     521F 0202          add.w   &b,R15
 001128     4F82 0204          mov.w   R15,&c
  return 0;
 00112C     430C               clr.w   R12
 00112E     4130               ret
```

Comparación

- AVR (ATmega32)
  - ciclos: 139-104 = 35
  - memoria código: 44 (0006-0031, 0x2c bytes)
- MSP430
  - ciclos: 95-74 = 21
  - memoria código: 22 (021E – 0233, 0x16 bytes)

Bibliografía

- “An Embedded Software Primer”
  - David E. Simon
- “MSP430x2xx Familiy User's Guide”
- “ATmega32 (L)”