Microcontroladores de 8 bits

1. Motorola/Philips/NXP **CPU08** (**6805**)
2. Intel/Atmel **8051**
3. Atmel/Microchip **AVR**
4. STMicroelectronics **STM8**
5. Microchip **PIC16/18**

8 bits

1. Microchip **PIC24** (16 bits) y **dsPIC** (16 bits)
2. Texas Instruments **MSP430** (16 bits)

16 bits

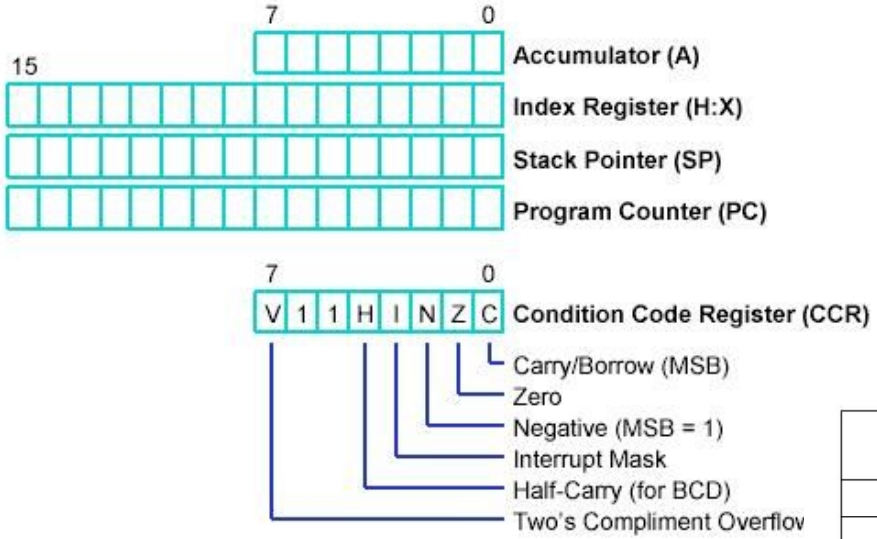
1. ARM **Cortex-M** (32 bits) - NXP, ST, Texas, Microchip, etc. (Materia optativa: SE)

1. Maxim **MAXQ10** (8bits) y **MAXQ20** (16 bits) (Sólo para valientes)

Motorola
68HC05

Modos de direccionamiento en la arquitectura CPU08

Motorola/Freescale/NXP 68HC08, tipo(1,1), acc 8 bits y registros especiales, inst. de largo variable, opcode=8 bits

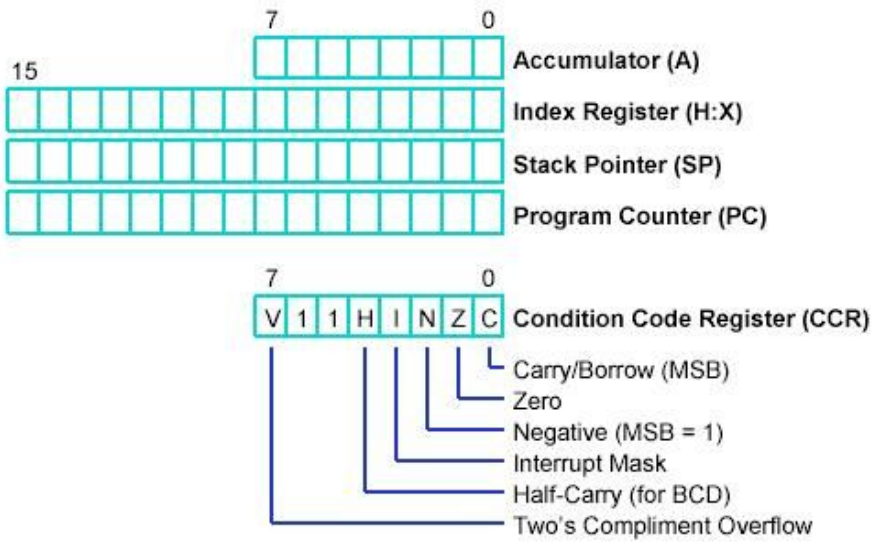


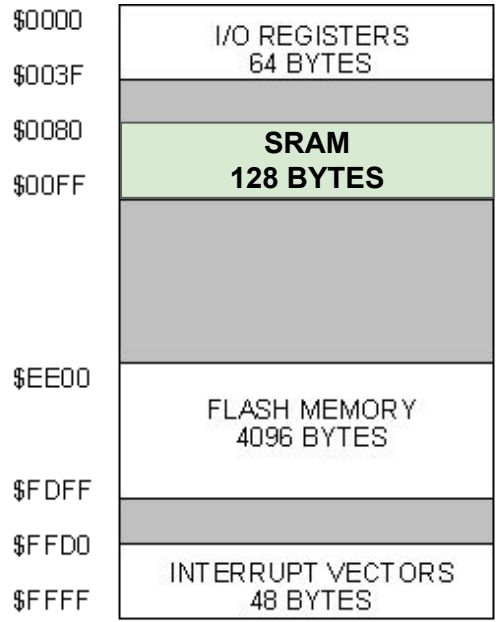
```
ADD #$01        ; inmediato #
ADD $0F         ; directo
ADD $010F       ; directo extendido
ADD ,X          ; indexado
ADD $02,X       ; indexado con offset
ADD $10FF,X
```

Source Forms		Addr Mode	Machine Code		HC08 Cycles
			Opcode	Operand(s)	
ADD	*#opr*	IMM	AB	ii	2
ADD	*opr*	DIR	BB	dd	3
ADD	*opr*	EXT	CB	hh ll	4
ADD	,X	IX	FB		2
ADD	*opr*,X	IX1	EB	ff	3
ADD	*opr*,X	IX2	DB	ee ff	4
ADD	*opr*,SP	SP1	9EEB	ff	4
ADD	*opr*,SP	SP2	9EDB	ee ff	5

CPU08 Stack Pointer

Init $00FF y crece hacia abajo (direcciones menores, en la figura hacia arriba)

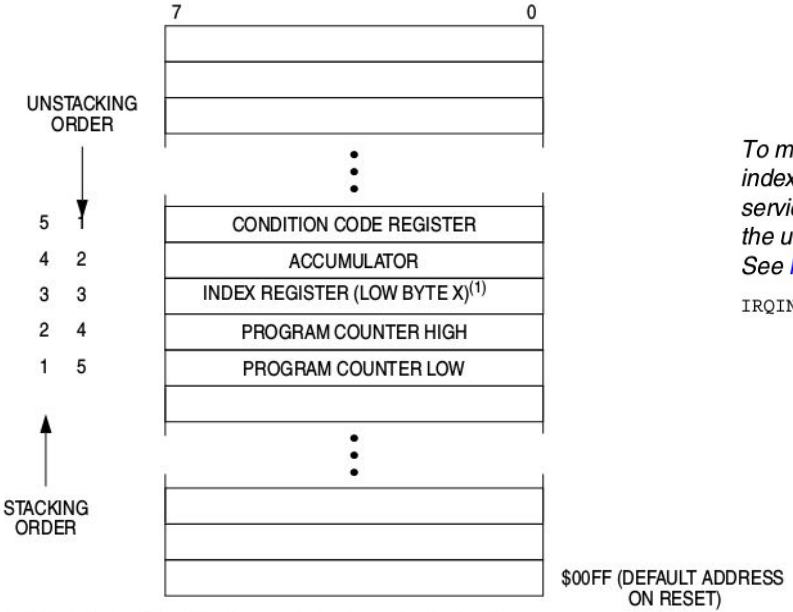




68HC908QT4 and 68HC908QY4 Memory Map

CPU08 interrupt stack

Orden de apilado: PC, X, A y CCR



1. High byte (H) of index register is not stacked.

Figure 3-1. Interrupt Stack Frame

NOTE

To maintain compatibility with the M6805 Family, H (the high byte of the index register) is not stacked during interrupt processing. If the interrupt service routine modifies H or uses the indexed addressing mode, it is the user's responsibility to save and restore it prior to returning.
See Figure 3-2.

```
IRQINT      PSHH
            |
            |Interrupt service routine
            |
            |
            PULH
            RTI
```

Figure 3-2. H Register Storage

MC9S08JM60 y MC9S08JM32

Microcontroladores HCS08

ISAs derivadas del 6800 (1974, 8 bits, acumulador+X+SP, arquitectura von Neumann, I/O mapeado en memoria, 192 opcodes: 72 inst y 7 modos de direccionamiento, 1MHz):
68HC05, **68HC08** (X de 16 bits), **68HCS08** (Freescale, nuevos modos de direccionamiento, Background debug mode (BDM)).
Evolución de los periféricos

MC9S08JM60 y MC9S08JM32

Microcontroladores HCS08

ISAs derivadas del 6800 (1974, 8 bits, acumulador+X+SP, arquitectura von Neumann, I/O mapeado en memoria, 192 opcodes: 72 inst y 7 modos de direccionamiento, 1MHz):
68HC05, **68HC08** (X de 16 bits), **68HCS08** (Freescale, nuevos modos de direccionamiento, Background debug mode (BDM)).
Evolución de los periféricos

El caso del Z80 (Zylog, 1976)
8-bit, 8080 mejorado

IP takes legacy 68HC08 architecture to 100million instructions/s

April 04, 2014 //By Graham Prophet

DCD's DF6808 IP Core is binary-compatible with the industry standard Motorola 68HC08 8-bit microcontroller, but with sophisticated on-chip peripheral capabilities, it performs 45-100 million instructions per second. The FAST architecture implemented in the DF6808 enables this MCU to run at least three times faster than the original.

MC9S08JM60 Series Features

8-Bit HCS08 Central Processor Unit (CPU)

- 48-MHz HCS08 CPU (central processor unit)
- 24-MHz internal bus frequency
- HC08 instruction set with added BGND instruction
- Background debugging system
- Breakpoint capability to allow single breakpoint setting during in-circuit debugging (plus two more breakpoints in on-chip debug module)
- In-circuit emulator (ICE) debug module containing two comparators and nine trigger modes. Eight deep FIFO for storing change-of-flow addresses and event-only data. Debug module supports both tag and force breakpoints.
- Support for up to 32 interrupt/reset sources

Memory Options

- Up to 60 KB of on-chip in-circuit programmable flash memory with block protection and security options
- Up to 4 KB of on-chip RAM
- 256 bytes of USB RAM

Clock Source Options

- Clock source options include crystal, resonator, external clock
- MCG (multi-purpose clock generator) — PLL and FLL; internal reference clock with trim adjustment

System Protection

- Optional computer operating properly (COP) reset with option to run from independent 1-kHz internal clock source or the bus clock
- Low-voltage detection with reset or interrupt
- Illegal opcode detection with reset
- Illegal address detection with reset

Power-Saving Modes

- Wait plus two stops

Peripherals

- **USB —** USB 2.0 full-speed (12 Mbps) device controller with dedicated on-chip USB transceiver, 3.3-V regulator and USBDP pull-up resister; supports control, interrupt, isochronous, and bulk transfers; supports endpoint 0 and up to 6 additional endpoints; endpoints 5 and 6 can be combined to provide double buffering capability
- **ADC** — 12-channel, 12-bit analog-to-digital converter with automatic compare function; internal temperature sensor
- **ACMP** — Analog comparator with option to compare to internal reference; operation in stop3 mode
- **SCI** — Two serial communications interface modules with optional 13-bit break LIN extensions

- **SPI** — Two 8- or 16-bit selectable serial peripheral interface modules with a receive data buffer hardware match function
- **IIC** — Inter-integrated circuit bus module to operate at up to 100 kbps with maximum bus loading; multi-master operation; programmable slave address; interrupt-driven byte-by-byte data transfer; 10-bit addressing and broadcast modes support
- **Timers** — One 2-channel and one 6-channel 16-bit timer/pulse-width modulator (TPM) modules: Selectable input capture, output compare, and edge-aligned PWM capability on each channel. Each timer module may be configured for buffered, centered PWM (CPWM) on all channels
- **KBI** — 8-pin keyboard interrupt module
- **RTC** — Real-time counter with binary- or decimal-based prescaler

Input/Output

- Up to 51 general-purpose input/output pins
- Software selectable pullups on ports when used as inputs
- Software selectable slew rate control on ports when used as outputs
- Software selectable drive strength on ports when used as outputs
- Master reset pin and power-on reset (POR)
- Internal pullup on $\overline{\text{RESET}}$, IRQ, and BKGD/MS pins to reduce customer system cost

Package Options

- 64-pin quad flat package (QFP)
- 64-pin low-profile quad flat package (LQFP)
- 48-pin quad flat no-lead (QFN)
- 44-pin low-profile quad flat package (LQFP)

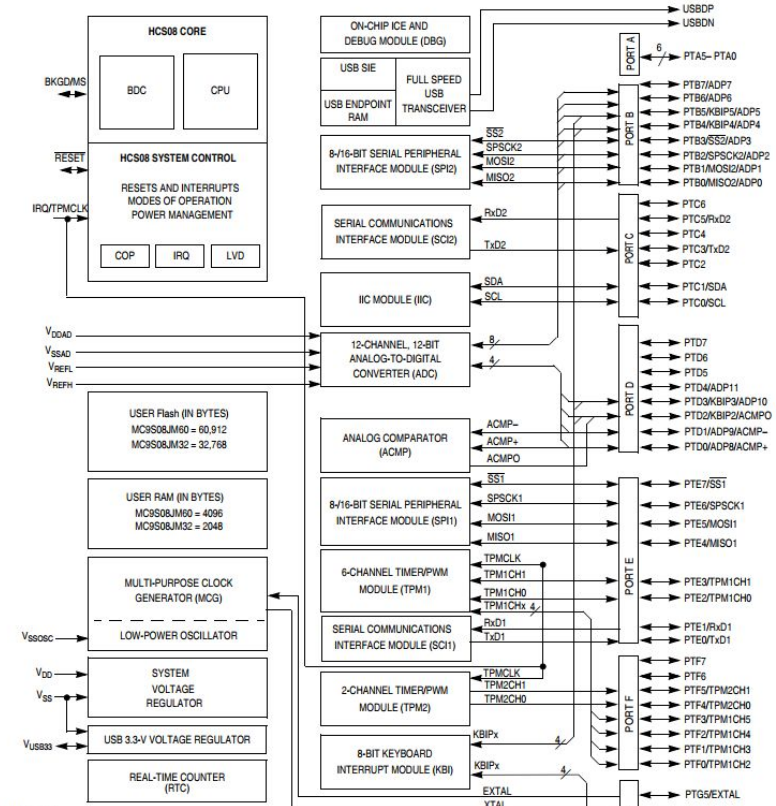
MC9S08JM60 Series Features

Table 1-1. Devices in the MC9S08JM60 Series

Feature	Device					
	MC9S08JM60			MC9S08JM32		
Package	64-pin	48-pin	44-pin	64-pin	48-pin	44-pin
Flash	60,912			32,768		
RAM	4096			2048		
USB RAM	256			256		
ACMP	yes			yes		
ADC	12-ch	8-ch	8-ch	12-ch	8-ch	8-ch
IIC	yes			yes		
IRQ	yes			yes		
KBI	8	7	7	8	7	7
SCI1	yes			yes		
SCI2	yes			yes		
SPI1	yes			yes		
SPI2	yes			yes		
TPM1	6-ch	4-ch	4-ch	6-ch	4-ch	4-ch
TPM2	2-ch			2-ch		
USB	yes			yes		
I/O pins	51	37	33	51	37	33
Package types	64 QFP 64 LQFP	48 QFN	44 LQFP	64 QFP 64 LQFP	48 QFN	44 LQFP

Table 1-2. Versions of On-Chip Modules

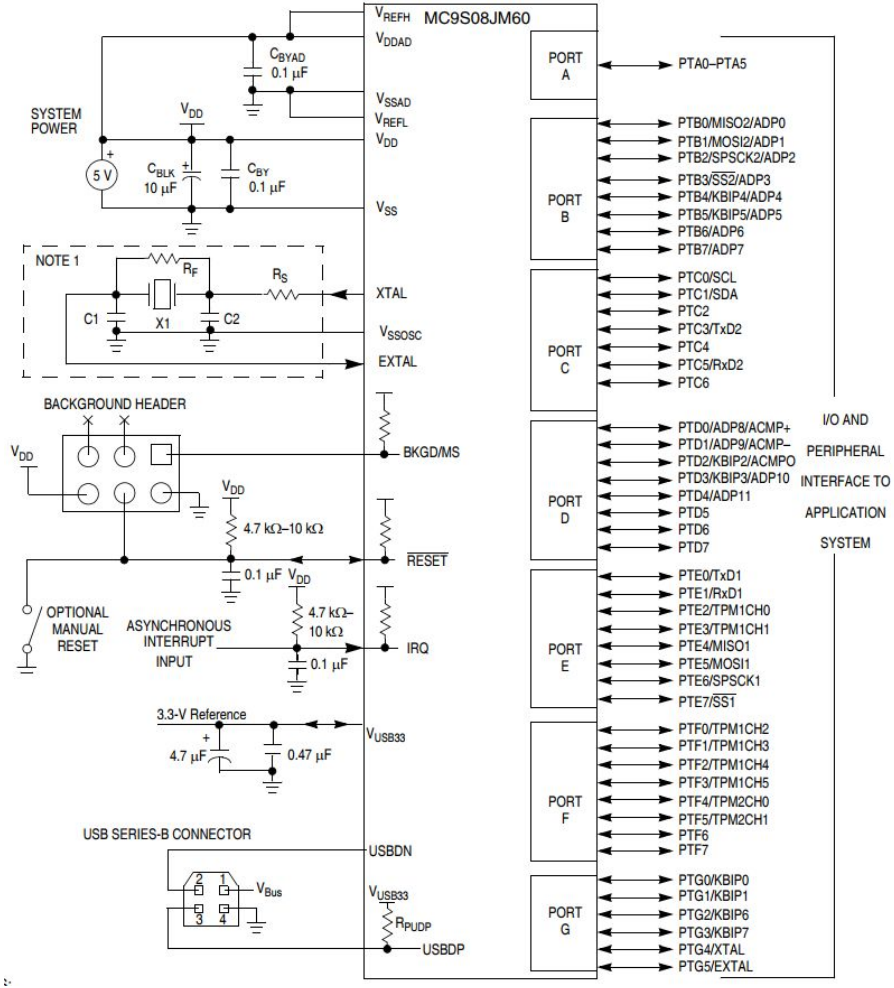
Module		Version
Analog Comparator	(ACMP)	2
Analog-to-Digital Converter	(ADC)	1
Central Processing Unit	(CPU)	2
IIC Module	(IIC)	2
Keyboard Interrupt	(KBI)	2
Multi-Purpose Clock Generator	(MCG)	1
Real-Time Counter	(RTC)	1
Serial Communications Interface	(SCI)	4
16-bit Serial Peripheral Interface	(SPI16)	1
Timer Pulse-Width Modulator	(TPM)	3
Universal Serial Bus	(USB)	1
Debug Module	(DBG)	2



NOTES:

1. Port pins are software configurable with pullup device if input port.
2. Pin contains software configurable pullup/pull-down device if IRQ is enabled (IRQPE = 1). Pull-down is enabled if rising edge detect is selected (IRQEDG = 1)
3. IRQ does not have a clamp diode to V_{DD}. IRQ must not be driven above V_{DD}.
4. Pin contains integrated pullup device.
5. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pull-down device.

Figure 1-1. MC9S08JM60 Series Block Diagram


MC9S08JM60
VREFH
VDDAD
VSSAD
VREFL
VDD
VSS
XTAL
VSSOSC
EXTAL
BKGD/MS
RESET
IRQ
VUSB33
USBDN
USBDP
CBYAD 0.1 µF
SYSTEM POWER
5 V
CBLK 10 µF
CBY 0.1 µF
NOTE 1
RF
RS
C1
X1
C2
BACKGROUND HEADER
4.7 kΩ–10 kΩ
0.1 µF
OPTIONAL MANUAL RESET
ASYNCHRONOUS INTERRUPT INPUT
3.3-V Reference
4.7 µF
0.47 µF
USB SERIES-B CONNECTOR
VBus
RPUDP
PORT A
PTA0–PTA5
PORT B
PTB0/MISO2/ADP0
PTB1/MOSI2/ADP1
PTB2/SPSCK2/ADP2
PTB3/SS2/ADP3
PTB4/KBIP4/ADP4
PTB5/KBIP5/ADP5
PTB6/ADP6
PTB7/ADP7
PORT C
PTC0/SCL
PTC1/SDA
PTC2
PTC3/TxD2
PTC4
PTC5/RxD2
PTC6
PORT D
PTD0/ADP8/ACMP+
PTD1/ADP9/ACMP–
PTD2/KBIP2/ACMPO
PTD3/KBIP3/ADP10
PTD4/ADP11
PTD5
PTD6
PTD7
PORT E
PTE0/TxD1
PTE1/RxD1
PTE2/TPM1CH0
PTE3/TPM1CH1
PTE4/MISO1
PTE5/MOSI1
PTE6/SPSCK1
PTE7/SS1
PORT F
PTF0/TPM1CH2
PTF1/TPM1CH3
PTF2/TPM1CH4
PTF3/TPM1CH5
PTF4/TPM2CH0
PTF5/TPM2CH1
PTF6
PTF7
PORT G
PTG0/KBIP0
PTG1/KBIP1
PTG2/KBIP6
PTG3/KBIP7
PTG4/XTAL
PTG5/EXTAL
I/O AND PERIPHERAL INTERFACE TO APPLICATION SYSTEM

Intel
8051

Intel 8051

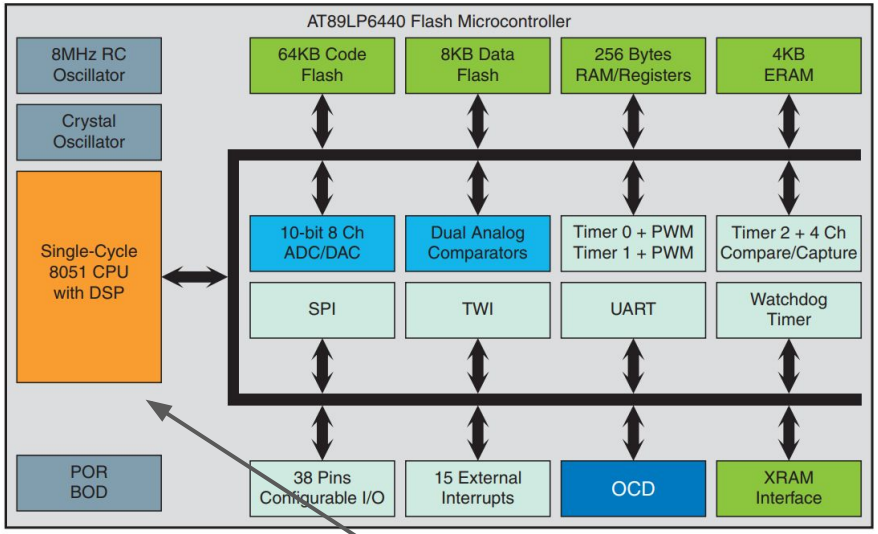
La arquitectura 8051 ha sido utilizada en la industria durante décadas, por lo que se mantiene popular hoy en día por su disponibilidad de bibliotecas y herramientas. El diseño original (1980) tomaba 12 ciclos de reloj (x1, x2, x3 o x4) por instrucción.

MSC®51 instruction set
8-bit data, 16-bit address, Harvard, **boolean**
Acc + 32 registros (4 bancos de 8)

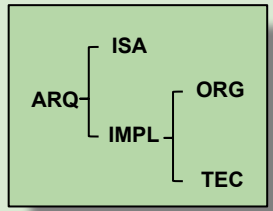
Hoy mismo ISA, segmentado, nueva tecnología
8051 original 12-cycle @12MHz, max 1 MIPS
8051 single-cycle @100MHz, max 100 MIPS

AT89LP51 @20MHz ($2.57)
Menor energía para la misma tarea
“less power consumption (up to 80% savings)”

FPGA
IP Core



100% de compatibilidad binaria (ISA)
Poco queda de la organización inicial (ORG)
Implementado con tecnología actual (TEC)



ADuC832 https://www.analog.com/en/products/aduc832.html

8051 - DISCUSIONES

https://semiengineering.com/the-case-against-8051/
Es chico, pero requiere mucha memoria
DMIPS malo

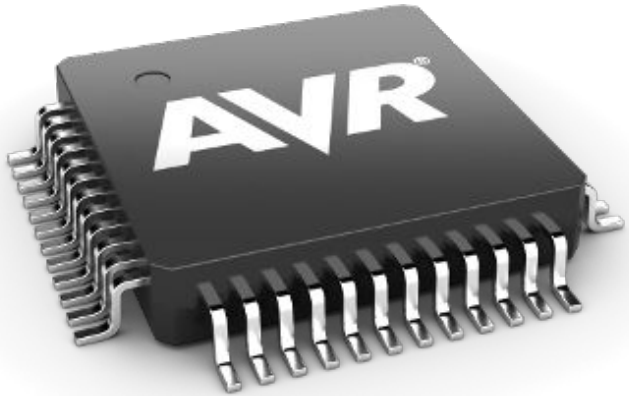
The classic 8051 architecture delivers less than 2 MIPS at 20 MHz
Single-cycle Atmel 1 MIPS per MHz of system frequency. At 20 MHz maximum clock frequency the AT89LP core is capable of 20 MIPS throughput

CAST's new super-fast 8051 MCU Core -- the S8051XC3 -- boasts a Dhrystone 2.1 performance rating of 0.252 DMIPS/MHz, which therefore offers a 26.85 times speed-up over the original 80C51 chip operating at the same frequency. Meanwhile, the S8051XC3's dynamic power consumption, which can be as low as 2.4µW/MHz, rivals that of low-power 32-bit processors and is lower than other 8051-compatible cores. “The Dhrystone score rates this new core at 26.85x the original at the same frequency, which just beats DCD's best 8-bit 8051 at 26.62x (as you can imagine, that's not an accident!) Our partners achieved this by starting from scratch and applying several modern processor architecture design techniques. Furthermore, our CPU size is about 7K gates, while DCD's is about 10K gates.

http://www.ecrostech.com/Other/Resources/Dhrystone.htm

		f	ROM	RAM	
MSP 430	14 DIP SMD	16MHz	512×8	128×8	1× $ 1,43 2000× $ 0,35
MAXQ	SMD	12MHz	16K×8	2K×8	1× $ 3,46
PIC16 ADC	8 DIP SMD	32MHz	2K×14	256×8	1× $ 0,83 Cottage $ 0,66
AT TINY ADC	8 DIP 6 SMD	20MHz 12 MHz	512×16 512×16	64×8 32×8	1× $ 0,82 9500× $ 0,27
CORTEX-M0 Cypress ADC	8 SMD	16MHz	8K×8	2K×8	1× $ 0,24 2500× $ 0,17
CORTEX-M3 ST ADC	20 SMD	48MHz	16K×8	4K×8	1× $ 1,15 2700× $ 0,55

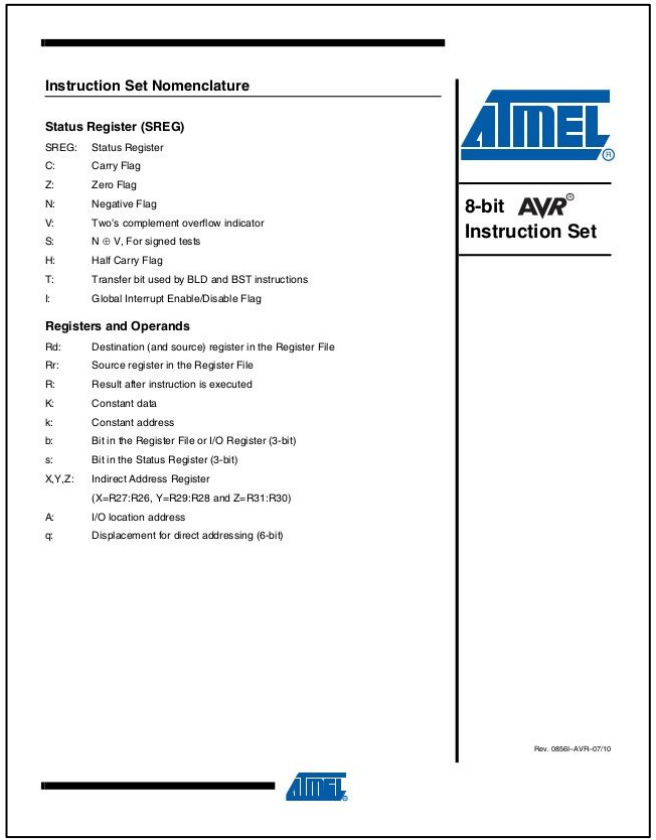
Atmel/Microchip AVR



Microcontroladores ATMEL ATtiny/ATmega

- AVR cpu
- Reloj
- Memoria Flash (programa)
- Memoria DRAM (datos)
- Memoria EEPROM (no volátil)
- Modulos de entrada/salida
  - I/O ports
  - Interfaces de comunicación
  - Temporizadores/contadores/pwm
  - ADC, comparadores, etc.
- Single supply, low cost, low power

https://microchipdeveloper.com/8avr:start



Instruction Set Nomenclature

Status Register (SREG)

SREG:	Status Register
C:	Carry Flag
Z:	Zero Flag
N:	Negative Flag
V:	Two's complement overflow indicator
S:	N ⊕ V, For signed tests
H:	Half Carry Flag
T:	Transfer bit used by BLD and BST instructions
I:	Global Interrupt Enable/Disable Flag

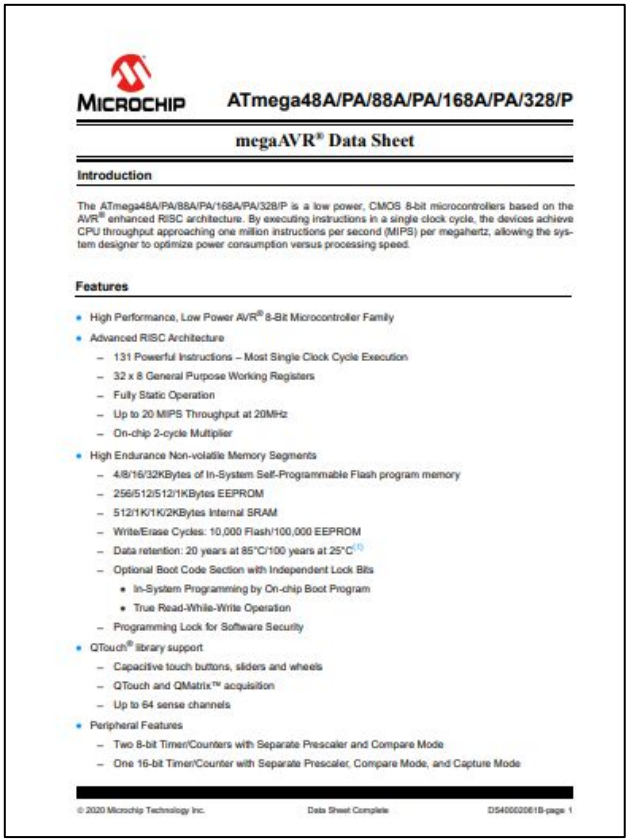
Registers and Operands

Rd:	Destination (and source) register in the Register File
Rr:	Source register in the Register File
R:	Result after instruction is executed
K:	Constant data
k:	Constant address
b:	Bit in the Register File or I/O Register (3-bit)
s:	Bit in the Status Register (3-bit)
X,Y,Z:	Indirect Address Register (X=R27:R26, Y=R29:R28 and Z=R31:R30)
A:	I/O location address
q:	Displacement for direct addressing (6-bit)

8-bit AVR® Instruction Set

Rev. 0856I–AVR–07/10

AVR Instruction Set Manual



MICROCHIP ATmega48A/PA/88A/PA/168A/PA/328/P

megaAVR® Data Sheet

Introduction

The ATmega48A/PA/88A/PA/168A/PA/328/P is a low power, CMOS 8-bit microcontrollers based on the AVR® enhanced RISC architecture. By executing instructions in a single clock cycle, the devices achieve CPU throughput approaching one million instructions per second (MIPS) per megahertz, allowing the system designer to optimize power consumption versus processing speed.

Features

- High Performance, Low Power AVR® 8-Bit Microcontroller Family
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 4/8/16/32KBytes of In-System Self-Programmable Flash program memory
  - 256/512/512/1KBytes EEPROM
  - 512/1K/1K/2KBytes Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
- QTouch® library support
  - Capacitive touch buttons, sliders and wheels
  - QTouch and QMatrix™ acquisition
  - Up to 64 sense channels
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode

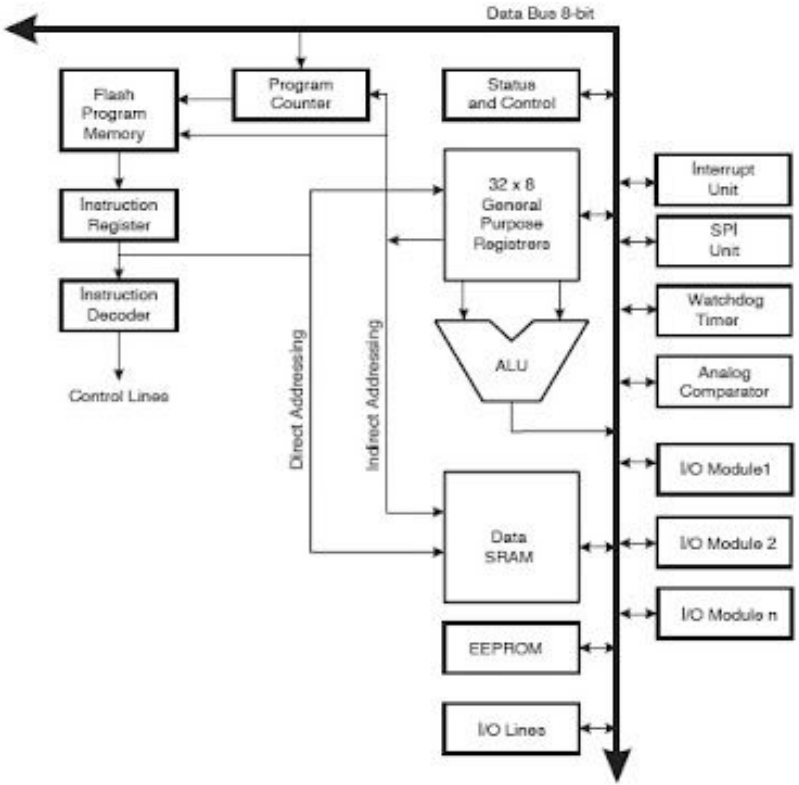
ATmega328p Datasheet

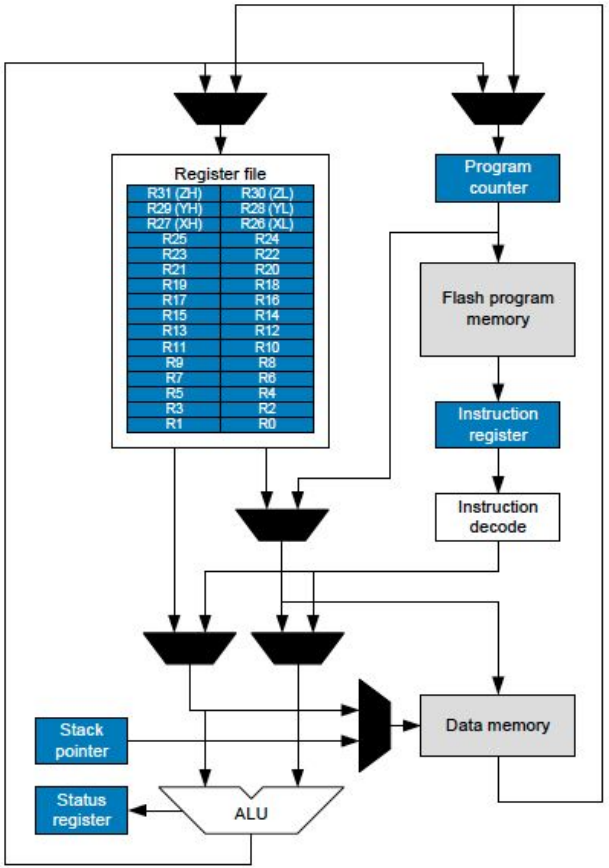
AVR Assembly Tutorials

http://www.rjhcoding.com/avr-asm-tutorials.php

AVRfreaks forum

https://www.avrfreaks.net/





AVR cpu: arquitectura Harvard, datos de 8 bits, instrucciones de 16 bits

Es un procesador de 8 bits (ALU y registros de 8 bits), pero puede direccionar 2^{16} posiciones de memoria (las direcciones son de 16 bits).

- 6 registros de 8 pueden ser utilizados como 3 punteros a memoria de datos de 16 bits: X, Y y Z. SP (pila) de 16 bits.

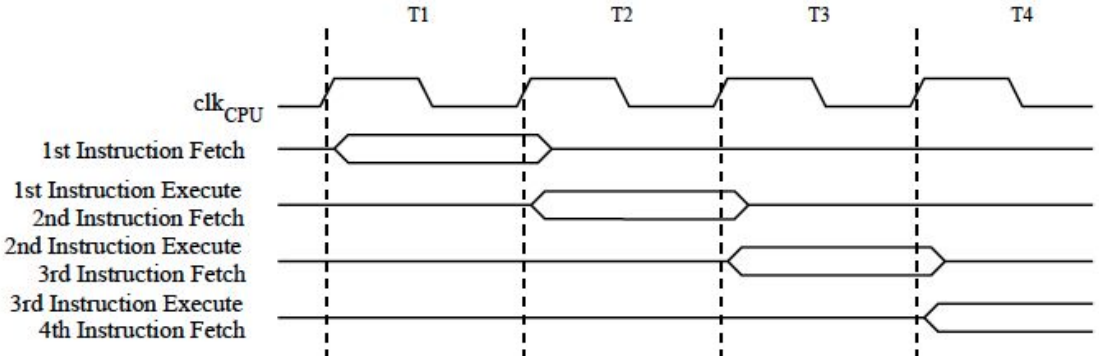
ATmegaXX puede direccionar XXK posiciones de memoria de programa (FLASH) de 16 bits. El PC de 16 bits puede extenderse a 22 bits.

Organización

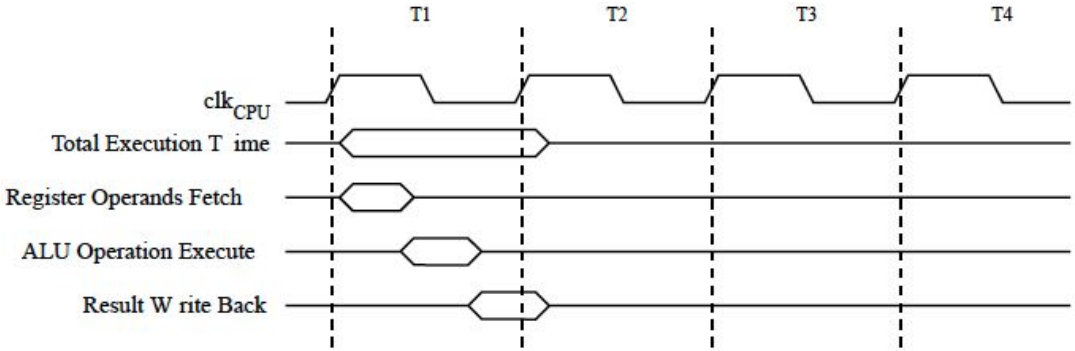
Arquitectura Harvard segmentada en dos etapas

Two-stage, single-level pipeline design (F-E)
la mayoría de las instrucciones ejecutan en 1 o 2 ciclos.

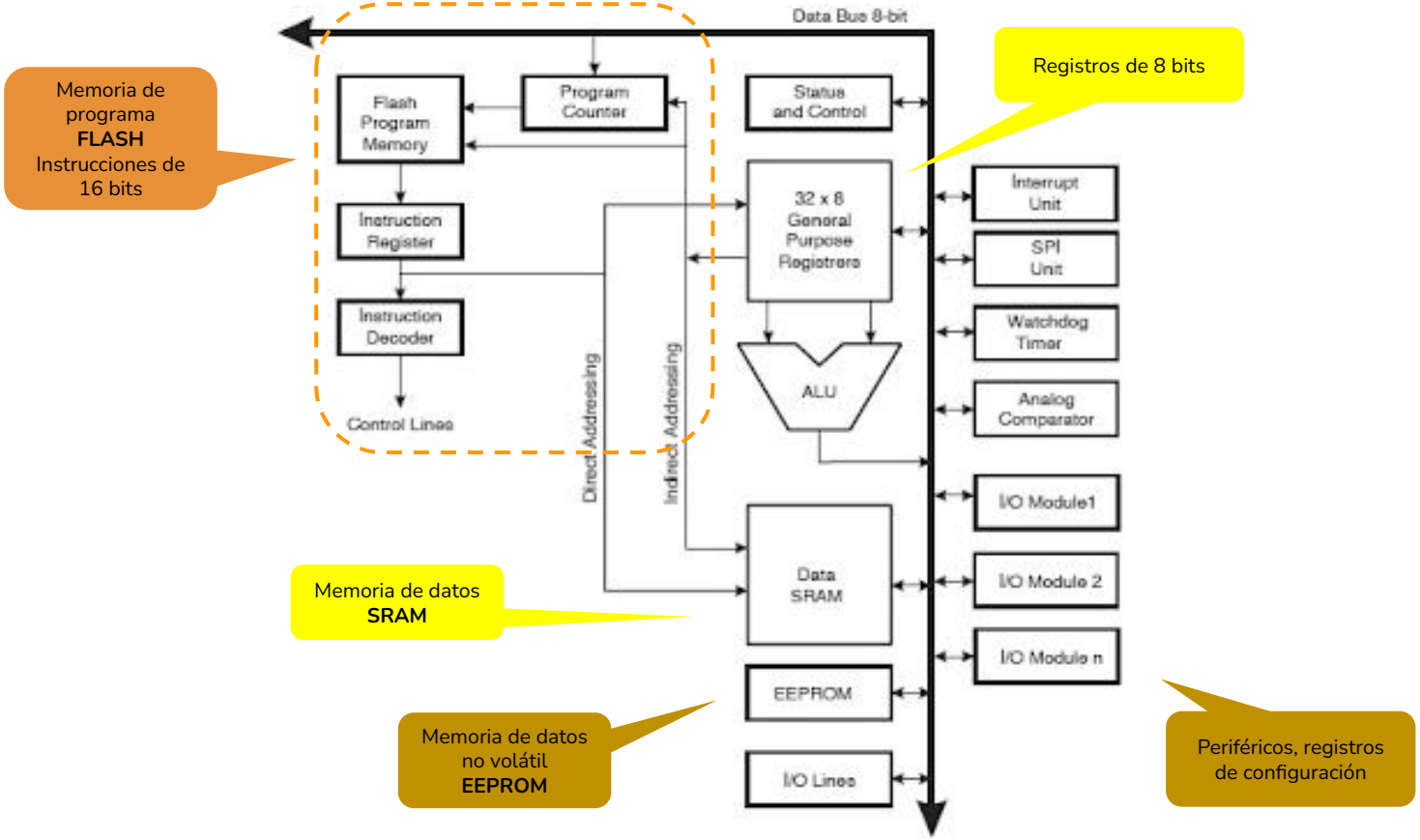
The Parallel Instruction Fetches and Instruction Executions



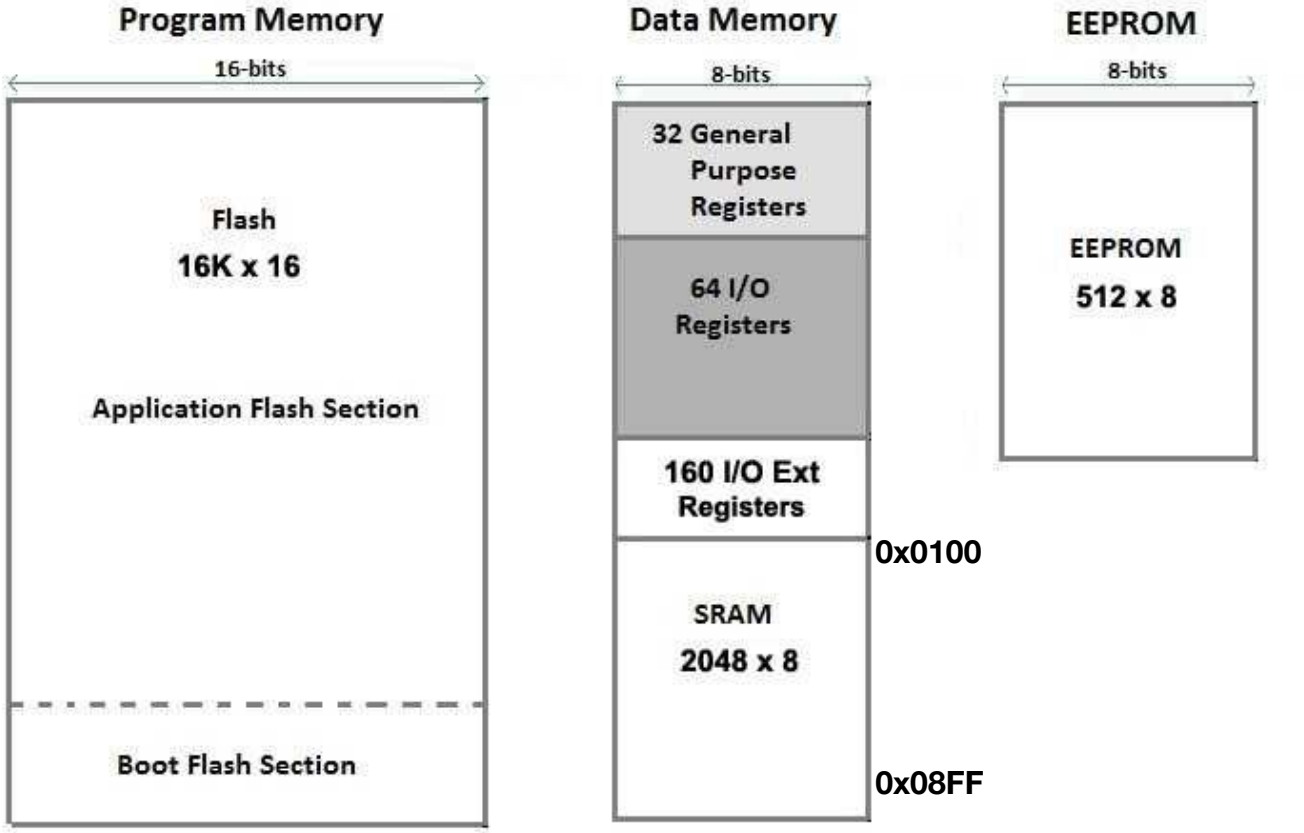
Single Cycle ALU Operation



Tipos de memoria



Mapa de memoria



Registros (GPR)

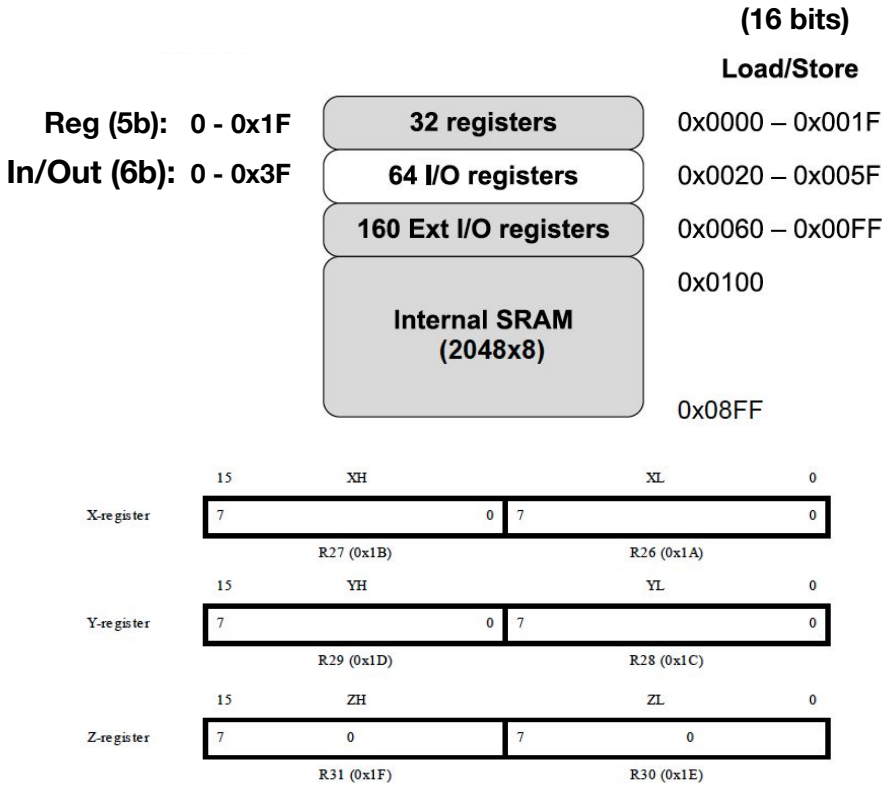
32 registros de 8 bits propósitos generales, operandos de la ALU

(0000–001F) 32 GPR de 8 bits r0-r31, algunas instrucciones rango limitado (ej. ldi, direccionamiento inmediato, r16-r31)

The following register pairs can work as address indexes (built-in 16-bit pointers useful for compiled C (X, Y y Z) direccionamiento indirecto. “Efficient C Language Code Density”)

- X, R26:R27
- Y, R28:R29
- Z, R30:R31

The following registers can be applied for specific use
R1:R0 store the result of multiplication instruction
R0 stores the data loaded from the program memory



Registros (I/O)

Address		Name
I/O	Mem.	
$00	$20	TWBR
$01	$21	TWSR
$02	$22	TWAR
$03	$23	TWDR
$04	$24	ADCL
$05	$25	ADCH
$06	$26	ADCSRA
$07	$27	ADMUX
$08	$28	ACSR
$09	$29	UBRRL
$0A	$2A	UCSRB
$0B	$2B	UCSRA
$0C	$2C	UDR
$0D	$2D	SPCR
$0E	$2E	SPSR
$0F	$2F	SPDR
$10	$30	PIND
$11	$31	DDRD
$12	$32	PORTD
$13	$33	PINC
$14	$34	DDRC
$15	$35	PORTC

Address		Name
I/O	Mem.	
$16	$36	PINB
$17	$37	DDRB
$18	$38	PORTB
$19	$39	PINA
$1A	$3A	DDRA
$1B	$3B	PORTA
$1C	$3C	EECR
$1D	$3D	EEDR
$1E	$3E	EEARL
$1F	$3F	EEARH
$20	$40	UBRRC
		UBRRH
$21	$41	WDTCR
$22	$42	ASSR
$23	$43	OCR2
$24	$44	TCNT2
$25	$45	TCCR2
$26	$46	ICR1L
$27	$47	ICR1H
$28	$48	OCR1BL
$29	$49	OCR1BH
$2A	$4A	OCR1AL

Address		Name
I/O	Mem.	
$2B	$4B	OCR1AH
$2C	$4C	TCNT1L
$2D	$4D	TCNT1H
$2E	$4E	TCCR1B
$2F	$4F	TCCR1A
$30	$50	SFIOR
$31	$51	OCDR
		OSCCAL
$32	$52	TCNT0
$33	$53	TCCR0
$34	$54	MCUCSR
$35	$55	MCUCR
$36	$56	TWCR
$37	$57	SPMCR
$38	$58	TIFR
$39	$59	TIMSK
$3A	$5A	GIFR
$3B	$5B	GICR
$3C	$5C	OCR0
$3D	$5D	SPL
$3E	$5E	SPH
$3E	$5E	SREG

Reg (5b): **0 - 0x1F**

In/Out (6b): **0 - 0x3F**

(16 bits)

Load/Store

32 registers	0x0000 – 0x001F
64 I/O registers	0x0020 – 0x005F
160 Ext I/O registers	0x0060 – 0x00FF
Internal SRAM (2048x8)	0x0100
	0x08FF

Stack Pointer

Status Register

ALU

Single cycle 8-bit ALU Instruction Set
(registro-registro o registro-inmediato)

Multiplicador por hardware 8x8, resultado de 16 bits en r0-r1.
2 ciclos.

Status Register mapeado en memoria

ARITHMETIC AND LOGIC INSTRUCTIONS					
Mnemonics	**Operands**	**Description**	**Operation**	**Flags**	**#Clocks**
ADD	Rd, Rr	Add two Registers without Carry	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add two Registers with Carry	Rd ← Rd + Rr + C	Z,C,N,V,H	1
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract two Registers with Carry	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract Constant from Reg with Carry.	Rd ← Rd - K - C	Z,C,N,V,H	1
AND	Rd, Rr	Logical AND Registers	Rd ← Rd · Rr	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd · K	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's Complement	Rd ← 0xFF - Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← 0x00 - Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd · (0xFF - K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd - 1	Z,N,V	1
TST	Rd	Test for Zero or Minus	Rd ← Rd · Rd	Z,N,V	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	Set Register	Rd ← 0xFF	None	1

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Saltos

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

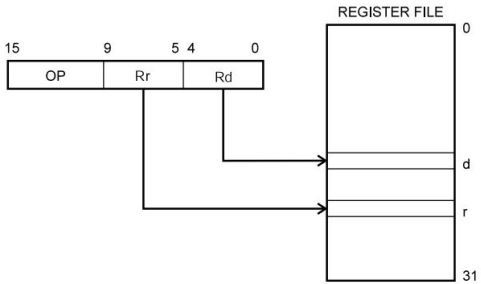
Conditional Branch Summary

Test	Boolean	Mnemonic	Complementary	Boolean	Mnemonic	Comment
Rd > Rr	Z•(N ⊕ V) = 0	BRLT(1)	Rd ≤ Rr	Z+(N ⊕ V) = 1	BRGE*	Signed
Rd ≥ Rr	(N ⊕ V) = 0	BRGE	Rd < Rr	(N ⊕ V) = 1	BRLT	Signed
Rd = Rr	Z = 1	BREQ	Rd ≠ Rr	Z = 0	BRNE	Signed
Rd ≤ Rr	Z+(N ⊕ V) = 1	BRGE(1)	Rd > Rr	Z•(N ⊕ V) = 0	BRLT*	Signed
Rd < Rr	(N ⊕ V) = 1	BRLT	Rd ≥ Rr	(N ⊕ V) = 0	BRGE	Signed
Rd > Rr	C + Z = 0	BRLO(1)	Rd ≤ Rr	C + Z = 1	BRSH*	Unsigned
Rd ≥ Rr	C = 0	BRSH/ BRCC	Rd < Rr	C = 1	BRLO/BRCS	Unsigned
Rd = Rr	Z = 1	BREQ	Rd ≠ Rr	Z = 0	BRNE	Unsigned
Rd ≤ Rr	C + Z = 1	BRSH(1)	Rd > Rr	C + Z = 0	BRLO*	Unsigned
Rd < Rr	C = 1	BRLO/BRCS	Rd ≥ Rr	C = 0	BRSH/BRCC	Unsigned
Carry	C = 1	BRCS	No carry	C = 0	BRCC	Simple
Negative	N = 1	BRMI	Positive	N = 0	BRPL	Simple
Overflow	V = 1	BRVS	No overflow	V = 0	BRVC	Simple
Zero	Z = 1	BREQ	Not zero	Z = 0	BRNE	Simple

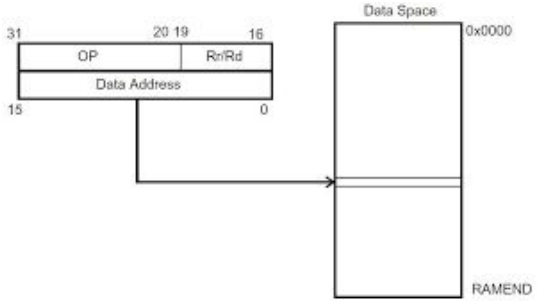
Note: Interchange Rd and Rr in the operation before the test, i.e., CP Rd,Rr → CP Rr,Rd.

Modos de direccionamiento

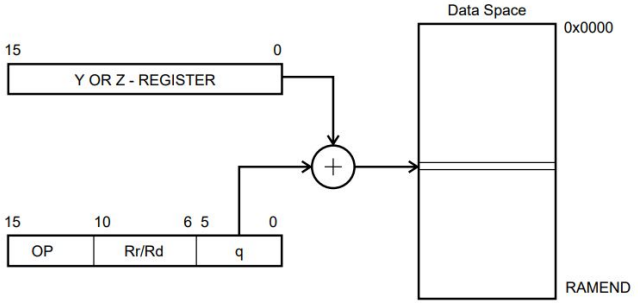
```
Immediate Addressing:       andi r16, $0F   (el operando está en la instrucción)

Register Direct Addressing: and r16, r0     (el operando está en un registro)
                            in r25, PINA    (el operando viene de i/o)
Data Direct Addressing:     lds r5, $F123   (la dirección está en la instrucción)

Indirect Addressing:        ld r11, X       (la dirección está guardada en un puntero)
      with displacement:    std Y+10, r14
      with Pre-decrement:   std -Y, r14
      with Post-increment   std Y+, r14
```



Direct Register Addressing

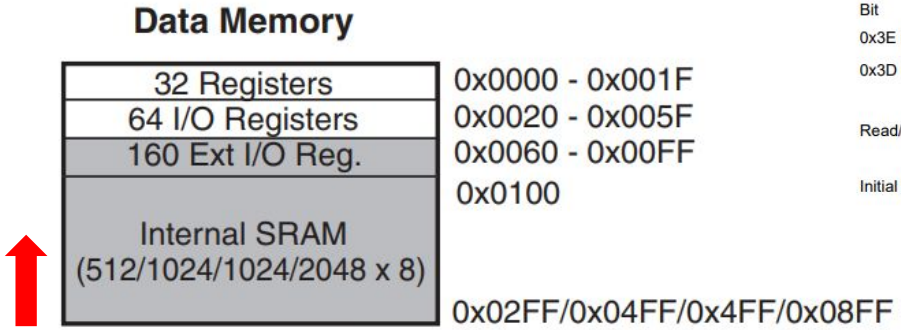


Direct Data Addressing



Data Indirect with Displacement

La pila



Bit	15	14	13	12	11	10	9	8	
0x3E (0x5E)	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SPH
0x3D (0x5D)	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	
	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	

Table 7-1. Stack Pointer instructions

Instruction	Stack pointer	Description
PUSH	Decremented by 1	Data is pushed onto the stack
CALL ICALL RCALL	Decremented by 2	Return address is pushed onto the stack with a subroutine call or interrupt
POP	Incremented by 1	Data is popped from the stack
RET RETI	Incremented by 2	Return address is popped from the stack with return from subroutine or return from interrupt

PUSH/POP 2 ciclos, RET/RETI 4 ciclos, ICALL/RCALL 3 ciclos, CALL 4 ciclos

```
ldi   r16,LOW(RAMEND)
out   SPL,r16
ldi   r16,HIGH(RAMEND)
out   SPH,r16
```

```
ATmega328p: M328DEF.INC
RAMEND = 0x08FF
SPL = 0x3D (0x5D)
SPH = 0x3E (0x5E)
```

Subrutinas

http://www.rjhcoding.com/avr-asm-functions.php (dibuja la memoria al revés)

```
      rcall doSomething          ; call subroutine

      ...                        ; rest of program



doSomething:
      push r16                   ; save r16 to the stack
      push r17                   ; save r17 to the stack

      ...                        ; código de la subrutina
                                 ; utiliza r16 y r17

      pop r17                    ; restore r17 from the stack
      pop r16                    ; restore r16 from the stack
      ret                        ; return from subroutine
```

Interrupciones

16.1. Interrupt Vectors in ATmega328/P

Table 16-1. Reset and Interrupt Vectors in ATmega328/P

Vector No	Program Address(2)	Source	Interrupts definition
1	0x0000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog System Reset
2	0x0002	INT0	External Interrupt Request 0
3	0x0004	INT1	External Interrupt Request 0
4	0x0006	PCINT0	Pin Change Interrupt Request 0
5	0x0008	PCINT1	Pin Change Interrupt Request 1
6	0x000A	PCINT2	Pin Change Interrupt Request 2
7	0x000C	WDT	Watchdog Time-out Interrupt
8	0x000E	TIMER2_COMPA	Timer/Counter2 Compare Match A
9	0x0010	TIMER2_COMPB	Timer/Coutner2 Compare Match B
10	0x0012	TIMER2_OVF	Timer/Counter2 Overflow
11	0x0014	TIMER1_CAPT	Timer/Counter1 Capture Event

prioridad

SREG – AVR Status Register

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

INTERRUPT ENABLE
STATUS: Status Register I : Global Interrupt Enable

SECUENCIA AVR

- When an interrupt occurs, the Global Interrupt Enable I-bit is cleared and all interrupts are disabled.
- The interrupt vector directs program control to the proper ISR or execution.
- That ISR can write logic one to the I-bit to enable nested interrupts.
- All enabled interrupts can then interrupt the current interrupt routine.
- When the ISR is completed and the return (RETI) command is executed from the ISR, the Global I-bit is automatically set to 'ON' and the program execution returns to the main program at the instruction that was interrupted.

Fuentes de interrupción

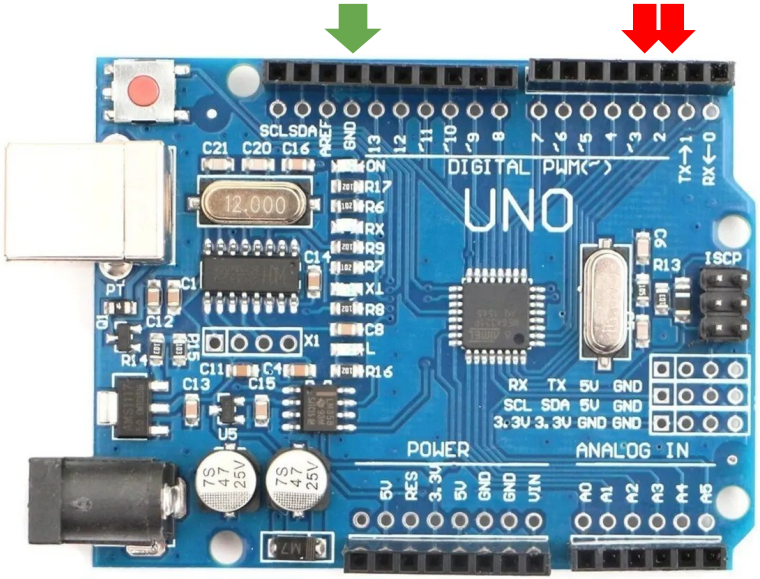
Port Pins: INT0, INT1 + Timers, UART, SPI, ADC, EEPROM, Analog Comparator, TWI or I2C

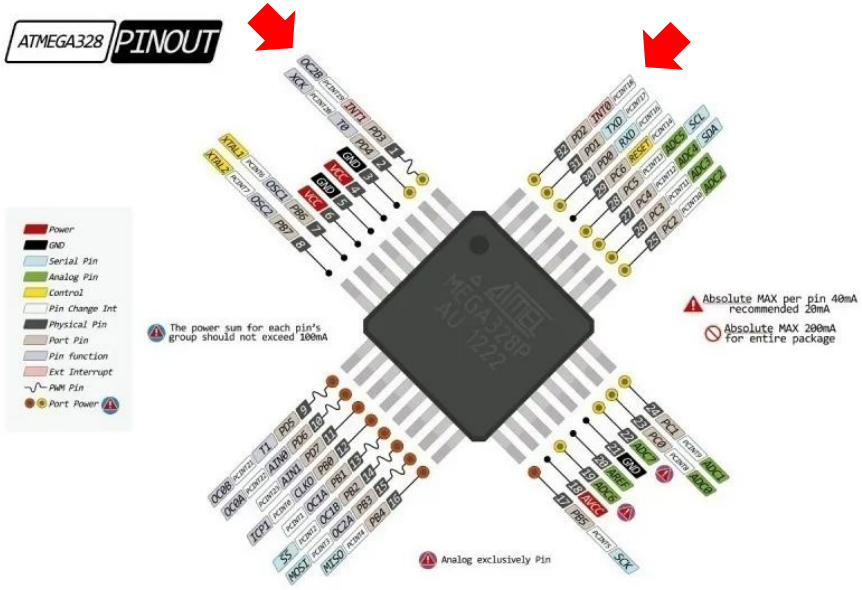
Dos posiciones de memoria, lo justo para poner un salto a la ISR (Interrupt Service Routine)

Interrupciones externas

- Configurar PD2 o PD3 como entrada con pull-up
- Configurar (registro EICRA)
- Habilitar la interrupción (registro EIMSK)
- Proveer una ISR
- Habilitar interrupciones en el status SREG

INT	Port	Pin	Kit
INT0	PD2	32	2
INT1	PD3	1	3





Interrupciones externas

```
; INT0 (activa baja) conmuta el led

.ORG 0x000              ; vector de reset
  jmp MAIN
.ORG 0x002              ; vector de INT0
  jmp EX0_ISR

MAIN:

  ldi r16,HIGH(RAMEND)          ; inicializa el
stack
  out SPH,r16
  ldi r16,LOW(RAMEND)
  out SPL,r16

  sbi DDRB,5            ; PORTB.5 = salida (LED)
  cbi DDRD,2            ; PORTD.2 = entrada (INT0)
  sbi PORTD,2           ; pull-up activado

  ldi r16,0x00                  ; INT0 activo bajo
(rojo)
  sts EICRA,r16
  ldi r16,0x01          ; INT0 habilitada (azul)
  out EIMSK,r16
  sei                   ; habilitar todas (verde)

LOOP: jmp LOOP          ; hacer nada

EX0_ISR:
  in r21,PORTB
  ldi r22,(1<<5)        ; conmuto el LED
  eor r21,r22
  out PORTB,r21
  reti
```

Bit	7	6	5	4	3	2	1	0	
(0x69)	–	–	–	–	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
0x1D (0x3D)	–	–	–	–	–	–	INT1	INT0	EIMSK
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
0x1C (0x3C)	–	–	–	–	–	–	INTF1	INTF0	EIFR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

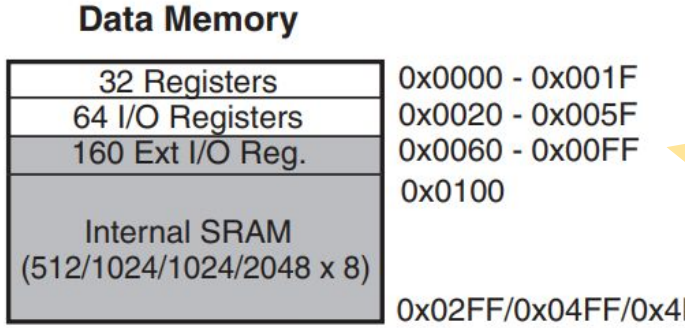
Interrupciones externas

```
; INT0 (activa baja) conmuta el led

.ORG 0x000              ; vector de reset
  jmp MAIN
.ORG 0x002              ; vector de INT0
  jmp EX0_ISR

MAIN:

  ldi r16,HIGH(RAMEND)          ; inicializa el
stack
  out SPH,r16
  ldi r16,LOW(RAMEND)
  out SPL,r16

  sbi DDRB,5            ; PORTB.5 = salida (LED)
  cbi DDRD,2            ; PORTD.2 = entrada (INT0)
  sbi PORTD,2           ; pull-up activado

  ldi r16,0x00                  ; INT0 activo bajo
(rojo)
  sts EICRA,r16
  ldi r16,0x01          ; INT0 habilitada (azul)
  out EIMSK,r16
  sei                   ; habilitar todas (verde)

LOOP: jmp LOOP          ; hacer nada

EX0_ISR:
  in r21,PORTB
  ldi r22,(1<<5)        ; conmuto el LED
  eor r21,r22
  out PORTB,r21
  reti
```

Bit	7	6	5	4	3	2	1	0	
(0x69)	–	–	–	–	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

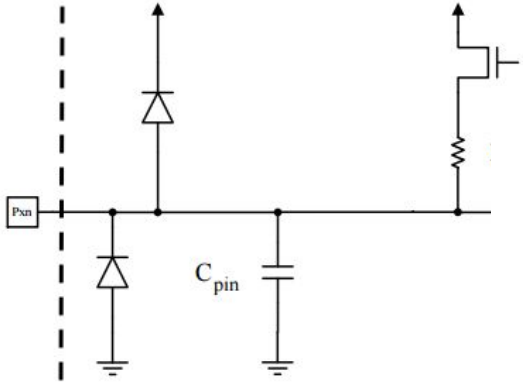
Table 12-2. Interrupt 0 Sense Control

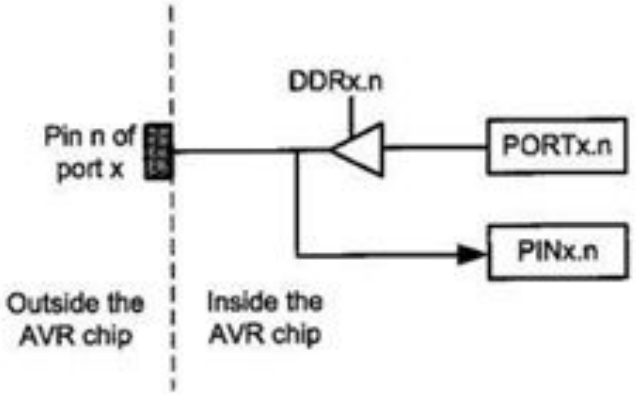
ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Any logical change on INT0 generates an interrupt request.
1	0	The falling edge of INT0 generates an interrupt request.
1	1	The rising edge of INT0 generates an interrupt request.



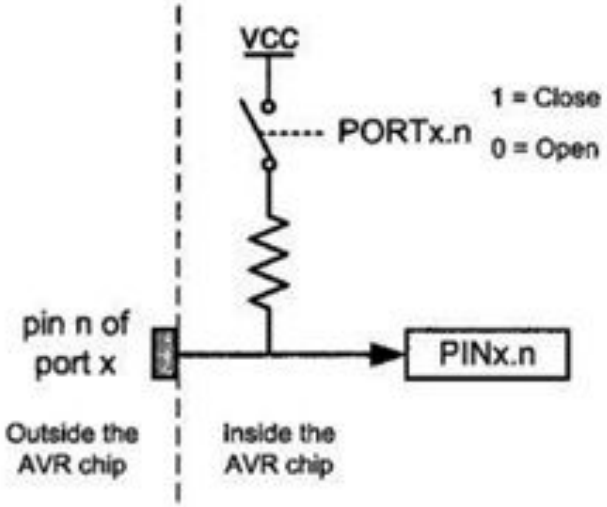
EICRA está en la zona I/O extendida, fuera del alcance de la instrucción OUT. Debe utilizarse STS de 2 words y 2 ciclos (ver la versión de 16 bits para el rango range 0x40...0xbf).

Ports de entrada/salida

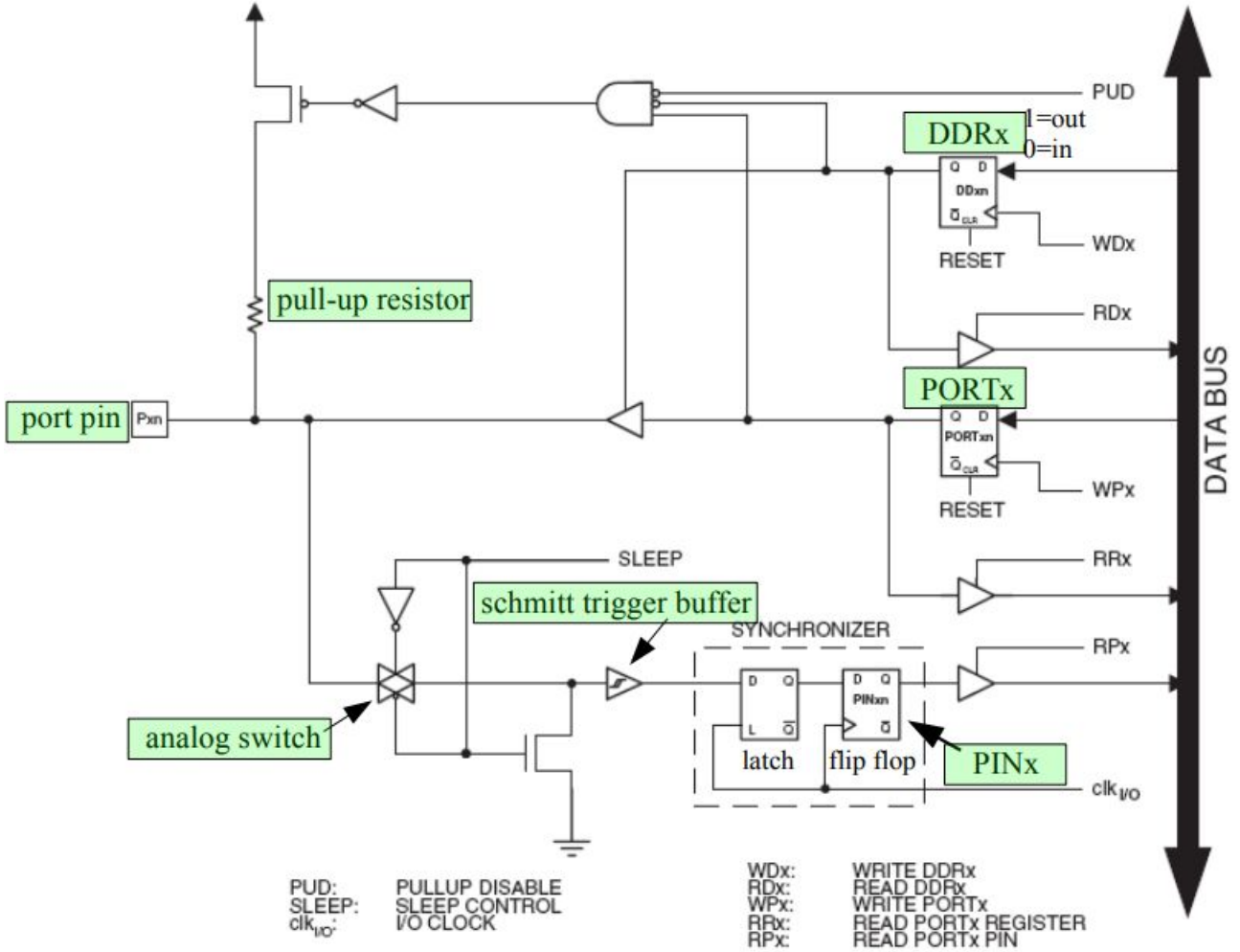




I/OPortin AVR microcontrollers



Pull-up Resistor


PUD
1=out
0=in
DDRx
DDxn
WDx
RESET
RDx
pull-up resistor
PORTx
port pin
Pxn
PORTxn
WPx
RESET
DATA BUS
SLEEP
RRx
schmitt trigger buffer
SYNCHRONIZER
RPx
PINxn
analog switch
latch
flip flop
PINx
clk I/O
PUD: PULLUP DISABLE
SLEEP: SLEEP CONTROL
clk I/O: I/O CLOCK
WDx: WRITE DDRx
RDx: READ DDRx
WPx: WRITE PORTx
RRx: READ PORTx REGISTER
RPx: READ PORTx PIN

Each port consists of three registers, where x = Port Name (A, B, C or D)

DDRx – Data Direction Register

The DDxn bits in the DDRx Register select the direction of this pin. If DDxn is written to '1', Pxn is configured as an output pin. If DDxn is written to '0', Pxn is configured as an input pin.

PORTx – Pin Output Register

As an Output: If a '1' is written to the bit when the pin is configured as an output pin, the port pin is driven high. If a '0' is written to the bit when the pin is configured as an output pin, the port pin is driven low.
As an Input: If a '1' is written to the bit when the pin is configured as an input pin, the pull-up resistor is activated. If a '0' is written to the bit when the pin is configured as an input pin, the port pin is tri-stated.

PINx – Pin Input Register

The PINxn bits in the PINx register are used to read data from port pin. When the pin is configured as a digital input (in the DDRx register), and the pull-up is enabled (in the PORTx register) the bit will indicate the state of the signal at the pin (high or low).
Note: If a port is made an output, then reading the PINx register will give you data that has been written to the port pins.
As a Tri-State Input: When the PORTx register disables the pull-up resistor the input will be tri-stated, leaving the pin left floating. When left in this state, even a small static charge present on surrounding objects can change the logic state of the pin. If you try to read the corresponding bit in the pin register, its state cannot be predicted.

Name: DDRB
Offset: 0x24
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x04

Bit	7	6	5	4	3	2	1	0
	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DDRBn: Port B Data Direction [n = 7:0]

Name: PORTB
Offset: 0x25
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x05

Bit	7	6	5	4	3	2	1	0
	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – PORTBn: Port B Data [n = 0:7]

Name: PINB
Offset: 0x23
Reset: N/A
Property: When addressing as I/O Register: address offset is 0x03

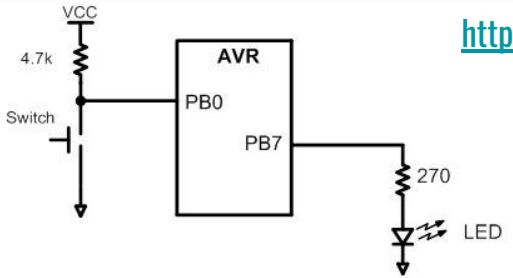
Bit	7	6	5	4	3	2	1	0
	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	x	x	x	x	x	x	x	x

Bits 7:0 – PINBn: Port B Input Pins Address [n = 7:0]

i/o bit-addressing

http://www.rjhcoding.com/avr-asm-io.php

Mnemonic	Description
in	in from I/O location
out	out from I/O location
cbi	clear bit in I/O register
sbi	set bit in I/O register
sbic	skip if bit in I/O register cleared
sbis	skip if bit in I/O register set



SBI: Set bit in IO register
CBI: Clear bit in IO register

SBIC: Skip if bit in IO register cleared

if

```
; Switch conectado al pin PB0 y LED al pin PB7
; Leer el estado del switch y enviarlo al LED.

       CBI DDRB,0    ; PB0 entrada
       SBI DDRB,7    ; PB7 salida
AGAIN: SBIC PINB,0   ; skip next if switch clear
       JMP ON
OFF:   CBI  PORTB,7  ; apagar LED
       JMP AGAIN
ON:    SBI PORTB,7   ; prender LED
       JMP AGAIN
```

while

```
wait:  sbic PINB,PINB0        ; skip if PINB0 is low
       rjmp wait              ; repeat loop
```

Codificación del repertorio de instrucciones

AVR

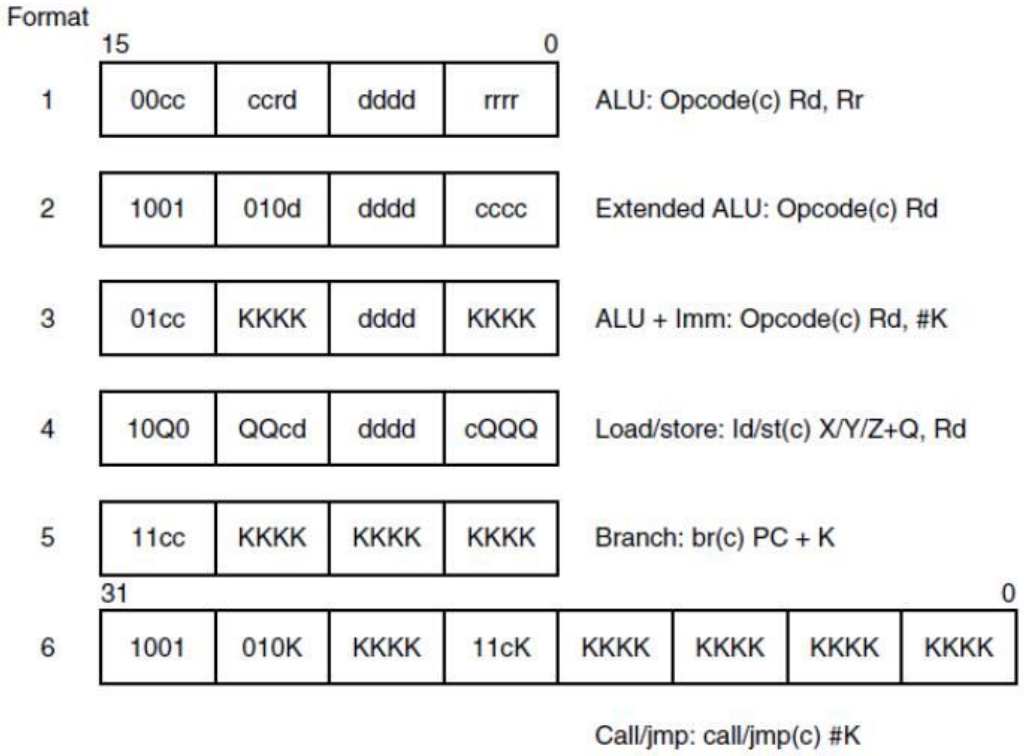
```
rrrrr = Source register
rrrr = Source register (R16–R31)
rrr = Source register (R16–R23)
RRRR = Source register pair (R1:R0–R31:R30)
ddddd = Destination register
dddd = Destination register (R16–R31)
ddd = Destination register (R16–R23)
DDDD = Destination register pair (R1:R0–R31:R30)
pp = Register pair, W, X, Y or Z
y = Y/Z register pair bit (0=Z, 1=Y)
u = FMUL(S(U)) signed with 0=signed or 1=unsigned
s = Store/load bit (0=load, 1=store)
c = Call/jump (0=jump, 1=call)
cy = With carry (0=without carry, 1=with carry)
e = Extend indirect jump/call address with EIND (0=0:Z,
1=EIND:Z)
q = Extend program memory address with RAMPZ (0=0:Z,
1=RAMPZ:Z)
aaaaaa = I/O space address
aaaaa = I/O space address (first 32 only)
bbb = Bit number (0–7)
B = Bit value (0 or 1)
kkkk = 4-bit unsigned constant (DES opcode)
kkkkkk = 6-bit unsigned constant
KKKKKKKK = 8-bit constant
```

El registro de destino **dddd** empieza **siempre** en el bit 4!

Atmel AVR instruction set overview

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Instruction
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NOP
0	0	0	0	0	0	0	1	d	d	d	d	r	r	r	r	MOVW Move register pair
0	0	0	0	0	0	1			d	d	d		r	r	r	Signed and fractional multiply (R16–R23 only)
0	0	0	0	0	1	d	d	d	d	d	d	r	r	r	r	2-operand instructions CPC, SBC, ADD, CPSE, CP, SUB. ADC, AND, EOR, OR, MOV
0	0	0	0	1												
0	0	0	1													
0	0	1	0													
0	0	1	1	K	K	K	K	h	h	h	h	K	K	K	K	Register-immediate instructions CPI, SBCI, SUBI, ORI, ANDI
0	1															
1	0	k	0	k	k	s	d	d	d	d	d	y	k	k	k	LDD/STD to Z+k or Y+k
1	0	0	1	0	0	s	d	d	d	d	d					LD/ST other
1	0	0	1	0	1	0	d	d	d	d	d	0				1-operand instructions (COM, NEG, SWAP, etc.)
1	0	0	1	0	1	0	0	B	b	b	b	1	0	0	0	SEx/CLx Status register clear/set bit
1	0	0	1	0	1	0	1					1	0	0	0	Misc instructions (RET, RETI, SLEEP, etc.)
1	0	0	1	0	1	0	c	0	0	0		1	0	0	1	Indirect jump/call to Z or EIND:Z
1	0	0	1	0	1	0	d	d	d	d	d	1	0	1	0	DEC Rd
1	0	0	1	0	1	0	0	k	k	k	k	1	0	1	1	DES round k
1	0	0	1	0	1	0	k	k	k	k	k	1	1	c	k	JMP/CALL abs22
1	0	0	1	0	1	1		k	k	p	p	k	k	k	k	ADIW/SBIW Rp,uimm6
1	0	0	1	1	0	B	a	a	a	a	a	b	b	b		I/O space bit operations
1	0	0	1	1	1	r	d	d	d	d	d	r	r	r	r	MUL, unsigned: R1:R0 = Rr×Rd
1	0	k	0	k	k	s	d	d	d	d	d	y	k	k	k	See 10k0 above
1	0	1	1	s	a	a	d	d	d	d	d	a	a	a	a	OUT/IN to I/O space
1	1	0	c	12 bit signed offset												Relative jump/call to PC ± 2×simm12
1	1	1	0	K	K	K	K	h	h	h	h	K	K	K	K	LDI Rh,K
1	1	1	1	0	B	7-bit signed offset						b	b	b		Conditional branch on status register bit
1	1	1	1	1	0	s	d	d	d	d	d	0	b	b	b	BLD/BST register bit to STATUS.T
1	1	1	1	1	1	B	d	d	d	d	d	0	b	b	b	SBRC/SBRS skip if register bit equals B

Codificación del repertorio de instrucciones



ATmega328p

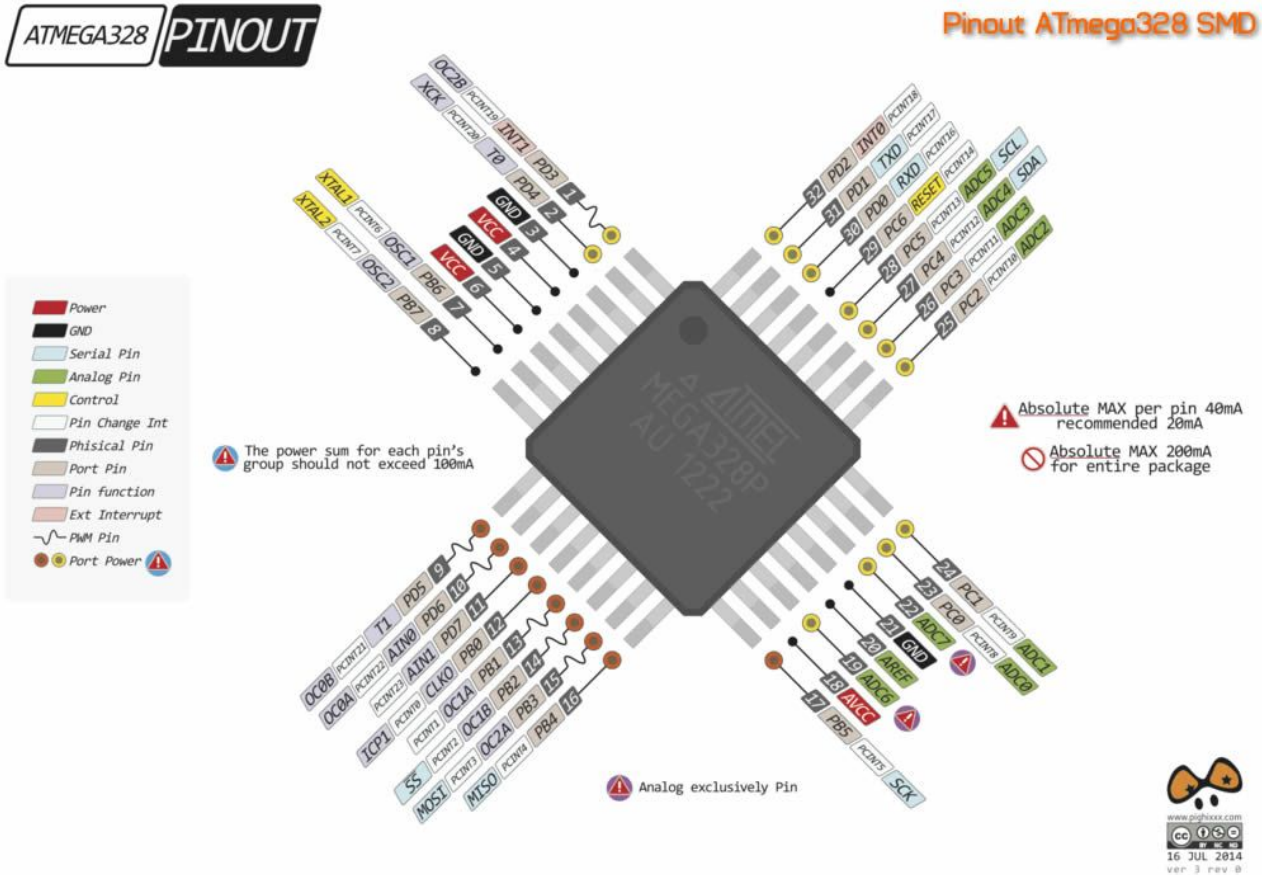
16 MHz, 32K Flash, 2K SRAM, 1K EEPROM

Encapsulado: 32-lead TQFP
Operating voltage:2.7V to 5.5V
Temperature range: Automotive temperature range: –40°C to +125°C
On-chip oscillator

Low power consumption
- Active mode: 1.5mA at 3V - 4MHz
- Power-down mode: 1µA at 3V
La tecnología con que está implementado le permite operar hasta 16 MHz.

Speed grade:
- 0 to 8MHz at 2.7 to 5.5V (automotive temperature range: –40°C to +125°C)
- 0 to 16MHz at 4.5 to 5.5V (automotive temperature range: –40°C to +125°C)

u$d 1.07 (x100)



La familia AVR de ATMEL

ATtiny/ATmega

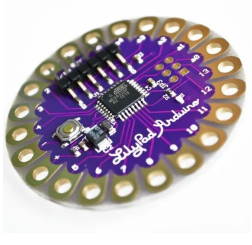
ATtiny menos de 1K RAM, menos de 16K Flash y menos de 20 patas.
ATmega "more muscle".
ATmegaXXY... XX kilobytes de memoria flash de programa, modelo Y

Kits

Arduino Uno/Nano: ATmega328P
16 MHz, 32K Flash, 2K SRAM, 1K EEPROM

Arduino Mega: ATmega2560
16 MHz, 256K Flash, 8K SRAM, 4K EEPROM

Robots Programación: ATmega32U4
16 MHz, 32K Flash, 2K5 SRAM, 1K EEPROM + USB



LilyPad: $400

8-bit AVR® Microcontrollers Peripheral Integration

Quick Reference Guide

Product Family	Pin Count	Program Flash Memory (KB)	SRAM (KB)	Peripheral Function Focus: Intelligent Analog: ADC (# of bits)	ADC (# of channels)	Comparators	ADC Gain Stage	DAC (# of bits)	Temperature Sensor	Internal Voltage Reference	Waveform Control: 8-bit PWM	16-bit PWM	Quadrature Decoder	Waveform Extension	Timing and Measurements: Real-Time Counter	8-bit Timer/Counters	12-bit Timer Counter	16-bit Timer/Counter	Logic, Crypto and Math: CCL	MULT	Crypto (AES/DES)	Safety and Monitoring: CRC	POR	BOD	WDT	Communications: UART	USART	USB	I²C	SPI	IRCOM	User Interface: QTouch® Technology	QTouch Technology with PTC[6]	LCD	System Flexibility: External Bus Interface	DMA Channels	Event System	SleepWalking	Sleep Modes	picoPower® Technology
ATtiny4/5/9/10	6	0.5–1	0.032	10[3]	4[3]	✓						2						1					✓		✓							✓							4	
ATtiny102/104	8/14	1	0.032	10	5/8	✓				✓		2						2					✓		✓		1											4	
ATtiny13A	8–20	1	0.064	10	4	✓						2											✓		✓							✓							3	✓
ATtiny20/40	12–20	2/4	0.128/0.256	10	8/12	✓			✓		2	2				1		1					✓	✓	✓				1	1		✓							4	
ATtiny24A/44A/84A	14–20	2–8	Up to 0.512	10	8	✓	✓		✓	✓	2	2				1		1					✓	✓	✓				1	1		✓							4	✓
ATtiny25(V)/45(V)/85(V)	8–20	2–8	Up to 0.512	10	4	✓	✓		✓	✓	4					2								✓	✓				1	1		✓							3	
ATtiny48/88	28–32	4/8	Up to 0.512	10	8	✓			✓	✓	1	1				1		1						✓	✓				1	1									3	✓
ATtiny87/167	20–32	8/16	0.512	10	11	✓			✓	✓	1	2				1		1						✓	✓	✓	1[1]			1	2								4	
ATtiny261A/461A/861A	20–32	2–8	Up to 0.512	10	11	✓	✓		✓	✓						1		1						✓	✓	✓				1	1		✓							4	✓
ATtiny20x/40x/80x/160x	8–24	2–16	Up to 1	10	12	✓			✓	✓		2			✓			1	✓	✓		✓	✓	✓	✓		1[5]		1	1						✓	✓		3	✓
ATtiny21x/41x/81x/161x/321x	8–24	2–32	Up to 2	10	12	✓		8	✓	✓		2			✓		1	1	✓	✓		✓	✓	✓	✓	✓	1[5]		1	1			✓[6]				✓	✓	3	✓
ATtiny441/841	14–20	4/8	Up to 0.512	10	12	✓	✓		✓		1	2				1		2						✓	✓	✓		2		1	1								4	✓
ATtiny1634	20	16	1	10	12	✓			✓	✓	2	2				1		1						✓	✓	✓		2		1			✓					4	✓
ATtiny2313A	20	2	0.128	–	–	✓				✓	2	2				1		1						✓	✓	✓		1		1	2							3	✓
ATmega8A/16A/32A	28–44	8–32	1–2	10	8	✓					2	1			✓	2		1		✓				✓	✓	✓		1		1	1		✓						5	
ATmega8U2/16U2/32U2	32	8–32	0.5–1	–	–	✓			✓	✓	4	6			✓	2		3		✓				✓	✓	✓		2	✓	2	2			✓					6	
ATmega16U4/32U4	32	16/32	1/2	10	12	✓			✓	✓	5					1		1		✓				✓	✓	✓		1	✓		1								6	
ATmega48PB/88PB/168PB/328PB	32	4–32	0.5–2	10	8	✓			✓	✓	4	2/6[8]			✓	2		1/3[8]		✓				✓	✓	✓		1/2[8]		1/2[8]	1/2[8]		✓	1[8]					6	
ATmega320x/480x	28–48	32–48	Up to 6	10	16	✓			✓	✓	4	3			1			5	✓	✓		✓	✓	✓	✓	4			1	1							✓		3	✓
ATmega64A/128A	64	64–128	4	10	8	✓	✓			✓	2	6				2		2		✓				✓	✓	✓		2		1	1		✓						6	
ATmega164PA/324PA/644PA/1284P	44	16–128	1–16	10	8	✓	✓			✓	4	2/2/4			✓	2		1/1/2		✓				✓	✓	✓		2		1	1		✓						6	✓
ATmega165PA/325PA/645P	44	16–64	1–4	10	8	✓				✓	4	6			✓	2		3		✓				✓	✓	✓		3		2	2			✓					6	✓
ATmega169PA/329PA/649P	64	16–64	1–4	10	8	✓				✓	2	2			✓	2		1		✓				✓	✓	✓		1		1	1		✓		✓				5	
ATmega324PB	44	32	2	10	8	✓				✓	2	2			✓	2		1		✓				✓	✓	✓		1		1	1		✓	✓					5	
ATmega640/1280/2560/1281/2561	64–100	64–256	8	10	8/16	✓	✓			✓	4	6/12			✓	2		4		✓				✓	✓	✓		2/4		1	1		✓			✓[9]			6	
ATmega3290PA/6490P	100	32–64	2–4	10	8	✓	✓			✓	2	2			✓	2		1		✓				✓	✓	✓		1		1	1		✓		✓				5	
ATmega3250PA/6450P	100	32–64	2–4	10	8	✓	✓			✓	2	2			✓	2		1		✓				✓	✓	✓		1		1	1		✓						5	
ATxmega A1U Family	100	64–128	4–8	12	16	✓	✓	12	✓	✓		8	✓	✓	✓			8		✓	✓	✓	✓	✓	✓	✓		8	✓	4	4	✓	✓			✓	4	✓	5	✓
ATxmega A3U Family	64	64–256	4–16	12	16	✓	✓	12	✓	✓		7	✓	✓	✓			7		✓	✓	✓	✓	✓	✓	✓		7	✓	2	3	✓	✓				4	✓	5	✓
ATxmega A4U Family	44–49	16–128	2–8	12	12	✓	✓	12	✓	✓		5	✓	✓	✓			5		✓	✓	✓	✓	✓	✓	✓		5	✓	2	2	✓	✓				4	✓	5	✓
ATxmega B1/B3 Family	64–100	64–128	4–8	12	8	✓	✓		✓	✓		2/3	✓	✓	✓			2/3		✓	✓	✓	✓	✓	✓	✓		1/2	✓	1	1	✓	✓			✓	2	✓	5	✓
ATxmega C3/D3 Family	64	32–384	4–32	12	16	✓	✓		✓	✓		5	✓	✓	✓			5		✓	✓	✓	✓	✓	✓	✓		3	✓[7]	2	2	✓	✓					✓	5	✓
ATxmega C4/D4 Family	44–49	16–128	2–8	12	12	✓	✓		✓	✓		4	✓	✓	✓			4		✓	✓	✓	✓	✓	✓	✓		2	✓[7]	2	2	✓	✓					✓	5	✓
ATxmega E5 Family	32	8–32	1–4	12	16	✓	✓	12	✓	✓		3	✓	✓	✓			3		✓	✓	✓	✓	✓	✓	✓		2		1	1	✓	✓				4	✓	5	✓

1: LIN port also 2: Peripheral Touch Controller 3: Only on the ATtiny5/10 4: Not on the ATtiny212/214/412/414/416 5: Only on the ATmega1281/2561 6: Only on the ATmega328PB 7: Only on the C3 and C4

www.microchip.com/8bit

MICROCHIP

```
; Ej 2-b

ldi xh,$01
ldi xl,$00
ldi r16,0
st X+, r16
ldi r17,1
st X+,r17
ldi r20,8

loop:
  mov r18, r17
  add r17, r16
  mov r16,r18
  st X+, r17
  dec r20
  brne loop

...
```

El código se puede mejorar nombrando las variable y constantes en forma descriptiva, agregando comentarios y un encabezado.

```
; Ej 2-b

ldi xh,$01
ldi xl,$00
ldi r16,0
st X+, r16
ldi r17,1
st X+,r17
ldi r20,8

loop:
  mov r18, r17
  add r17, r16
  mov r16,r18
  st X+, r17
  dec r20
  brne loop

...
```

```
; Primeros elementos de la secuencia de Fibonacci almacenados en memoria
; Created: 21/10/2020 10:09:51 a. m.
; Author : alejandro.veiga@ing.unlp.edu.ar

.EQU ELEMENTOS = 13     ; máximo 13 porque los registros son de 8 bits
.EQU TABLA = 0x0100     ; secuencia almacenada en el comienzo de la DRAM

.DEF anterior = r16
.DEF actual = r17
.DEF temp = r18
.DEF contador = r20

ldi XH,HIGH(TABLA)      ; indice de la tabla
ldi XL,LOW(TABLA)

ldi anterior,0          ; los dos primeros elementos son 0 y 1, por definición
st X+,anterior
ldi actual,1
st X+,actual

ldi contador,ELEMENTOS-2

loop:
  mov temp,actual
  add actual,anterior   ; cada elemento es la suma de los dos anteriores
  mov anterior,temp
  st X+,actual
  dec contador
  brne loop
...
```

```
; Ej 2-b

ldi xh,$01
ldi xl,$00
ldi r16,0
st X+, r16
ldi r17,1
st X+,r17
ldi r20,8

loop:
  mov r18, r17
  add r17, r16
  mov r16,r18
  st X+, r17
  dec r20
  brne loop

...
```

Ambos programas generan exactamente el mismo código

```
; Primeros elementos de la secuencia de Fibonacci almacenados en memoria
; Created: 21/10/2020 10:09:51 a. m.
; Author : alejandro.veiga@ing.unlp.edu.ar

.EQU ELEMENTOS = 13     ; máximo 13 porque los registros son de 8 bits
.EQU TABLA = 0x0100     ; secuencia almacenada en el comienzo de la DRAM

.DEF anterior = r16
.DEF actual = r17
.DEF temp = r18
.DEF contador = r20

ldi XH,HIGH(TABLA)      ; indice de la tabla
ldi XL,LOW(TABLA)

ldi anterior,0          ; los dos primeros elementos son 0 y 1, por definición
st X+,anterior
ldi actual,1
st X+,actual

ldi contador,ELEMENTOS-2

loop:
  mov temp,actual
  add actual,anterior   ; cada elemento es la suma de los dos anteriores
  mov anterior,temp
  st X+,actual
  dec contador
  brne loop
...
```

Example:

```
clr r27 ; Clear X high byte
ldi r26,$60 ; Set X low byte to $60
ld r0,X+ ; Load r0 with data space loc. $60(X post inc)
ld r1,X ; Load r1 with data space loc. $61
ldi r26,$63 ; Set X low byte to $63
ld r2,X ; Load r2 with data space loc. $63
ld r3,-X ; Load r3 with data space loc. $62(X pre dec)
```

Estilo de redacción

Mayúsculas o minúsculas? Espacios? Tabs?

```
clr r27      ; Clear X high byte
ldi r26,$60  ; Set X low byte to $60
ld r0,X+     ; Load r0 with data space loc. $60(X post inc)
ld r1,X      ; Load r1 with data space loc. $61
ldi r26,$63  ; Set X low byte to $63
ld r2,X      ; Load r2 with data space loc. $63
ld r3,-X     ; Load r3 with data space loc. $62(X pre dec)
```

In the Reduced Core tinyAVR the LD instruction can be used to achieve the same operation as LPM since the program memory is mapped to the data memory space.

The result of these combinations is undefined:

LD r26, X+

LD r27, X+

LD r26, -X

LD r27, -X

Using the X-pointer:

	Operation:		Comment:
(i)	Rd ← (X)		X: Unchang
(ii)	Rd ← (X) X ← X + 1		X: Post incremented
(iii)	X ← X - 1 Rd ← (X)		X: Pre decremented
	Syntax:	Operands:	Program Counter:
(i)	LD Rd, X	0 ≤ d ≤ 31	PC ← PC + 1
(ii)	LD Rd, X+	0 ≤ d ≤ 31	PC ← PC + 1

AVR Instruction Set Manual

ANALISIS DE PERFORMANCE

```
loop:
  mov temp, actual                  ; 1 ciclo
  add actual, anterior              ; 1
  mov anterior,temp                 ; 1
  st X+, actual                     ; 2
  dec contador                      ; 1  **
  brne loop                         ; 2  **
                                    ; Total: 8 ciclos de reloj por iteración


                                    ; Desenrollando: 5
  mov temp, actual                  ; 1 ciclo
  add actual, anterior              ; 1
  mov anterior,temp                 ; 1
  st X+, actual                     ; 2
```

Mejora = 8/5 = 60%

La familia XMEGA

AVR core :-) Mismo "Instruction Set manual"
Algunas pocas instrucciones nuevas (DES)
Algunas instrucciones toman menos ciclos

Nueva nomenclatura ATxmega_nnn_ F _m_ F: familia, por ejemplo A tienen todos un mismo manual.
"Xmega A Manual" has consistent naming of peripherals and their registers.
Pero para eso renombraron y reordenaron todo el sistema de entrada/salida.
Quedó muy bien, más intuitivo?, pero no es compatible.

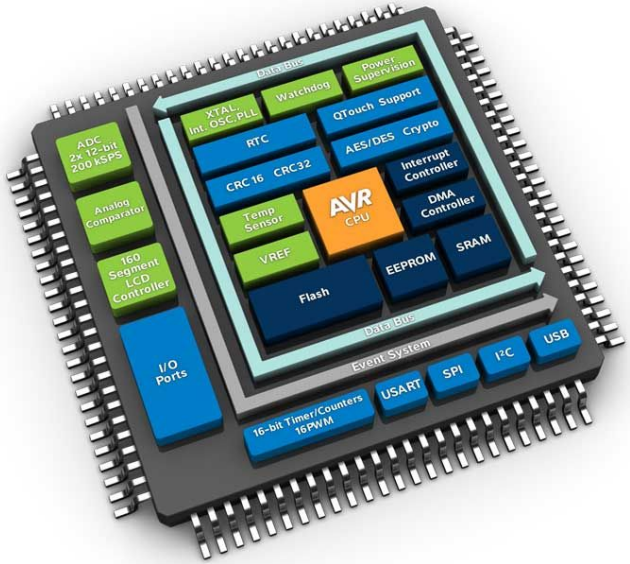
Hasta 32 MHz, directly from the internal oscillator (!)
Ultra-low-power (?) picoPower™

Multi-level interrupt system
Más periféricos y más sofisticados:

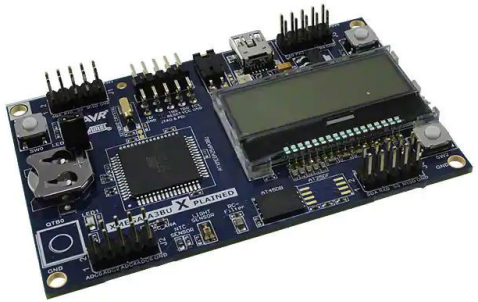
- Mejora en la operatoria de los ports: true Read-Modify-Write (RMW)
- Todos los timers son de 16 bits cascadeables a 32
- ADC de 12 bits, hasta 16 canales, hasta 2 Msps
- DAC de 12 bits, crypto engines, real-time clocks
- Hasta 8 UARTs

DMA transfer system. There's a very impressive new "event system" that allows an increased number of peripherals to communicate directly with each other, without taking up CPU cycles to do so. (VEER)

IDE: Atmel Studio soporta la familia (!)
No son pin compatible ni code compatible (debido al renombrado de los registros)
No-hobbista, más professional-oriented. No DIP ni 5V. Voltage 1.6-3.6V - NO LEGACY - FOR NEW DESIGNS
Nueva interfaz de programación (JTAG o PDI)
Kits: no hay en ML - Digikey: los ATXMEGA256 son mas baratos que los ATMEGA256? Kit: u$s 35, necesita un programador



Brochure
http://ww1.microchip.com/downloads/en/DeviceDoc/doc7925.pdf



La familia XMEGA - ISA

http://ww1.microchip.com/downloads/en/devicedoc/atmel-0856-avr-instruction-set-manual.pdf

Algunas instrucciones indican el número de ciclos diferente (menor) para XMEGA

Table 4-1. Versions of AVR 8-bit CPU

Name	Device Series	Description
AVR	AT90	Original instruction set from 1995.
AVRe	megaAVR®	Multiply (xMULxx), Move Word (MOVW), and enhanced Load Program Memory (LPM) added to the AVR instruction set. No timing differences.
AVRe	tinyAVR®	Multiply not included, but else equal to AVRe for megaAVR.
AVRxm	XMEGA®	Significantly different timing compared to AVR(e). The Read Modify Write (RMW) and DES encryption instructions are unique to this version.
AVRxt	(AVR)	AVR 2016 and onwards. This variant is based on AVRe and AVRxm. Closer related to AVRe, but with improved timing.
AVRrc	tinyAVR	The Reduced Core AVR CPU was developed for ultra-low pinout (6-pin) size constrained devices. The AVRrc therefore only has a 16 registers register-file (R31-R16) and a limited instruction set.

90. PUSH – Push Register on Stack

90.1. Description

This instruction stores the contents of register Rr on the STACK. The Stack Pointer is post-decremented by 1 after the PUSH.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) STACK ← Rr

	Syntax:	Operands:	Program Counter:	Stack:
(i)	PUSH Rr	0 ≤ r ≤ 31	PC ← PC + 1	SP ← SP - 1

16-bit Opcode:

1001	001d	dddd	1111

90.2. Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Example:

```
call routine ; Call subroutine
...
routine: push r14 ; Save r14 on the Stack
push r13 ; Save r13 on the Stack
...
pop r13 ; Restore r13
pop r14 ; Restore r14
ret ; Return from subroutine
```

Words	1 (2 bytes)
Cycles	2
Cycles XMEGA	1

La familia XMEGA - ISA

http://ww1.microchip.com/downloads/en/devicedoc/atmel-0856-avr-instruction-set-manual.pdf

Algunas instrucciones son exclusivas

Table 4-1. Versions of AVR 8-bit CPU

Name	Device Series	Description
AVR	AT90	Original instruction set from 1995.
AVRe	megaAVR®	Multiply (xMULxx), Move Word (MOVW), and enhanced Load Program Memory (LPM) added to the AVR instruction set. No timing differences.
AVRe	tinyAVR®	Multiply not included, but else equal to AVRe for megaAVR.
AVRxm	XMEGA®	Significantly different timing compared to AVR(e). The Read Modify Write (RMW) and DES encryption instructions are unique to this version.
AVRxt	(AVR)	AVR 2016 and onwards. This variant is based on AVRe and AVRxm. Closer related to AVRe, but with improved timing.
AVRrc	tinyAVR	The Reduced Core AVR CPU was developed for ultra-low pinout (6-pin) size constrained devices. The AVRrc therefore only has a 16 registers register-file (R31-R16) and a limited instruction set.

54. DES – Data Encryption Standard

54.1. Description

The module is an instruction set extension to the AVR CPU, performing DES iterations. The 64-bit data block (plaintext or ciphertext) is placed in the CPU register file, registers R0-R7, where LSB of data is placed in LSB of R0 and MSB of data is placed in MSB of R7. The full 64-bit key (including parity bits) is placed in registers R8-R15, organized in the register file with LSB of key in LSB of R8 and MSB of key in MSB of R15. Executing one DES instruction performs one round in the DES algorithm. Sixteen rounds must be executed in increasing order to form the correct DES ciphertext or plaintext. Intermediate results are stored in the register file (R0-R15) after each DES instruction. The instruction's operand (K) determines which round is executed, and the half carry flag (H) determines whether encryption or decryption is performed.

The DES algorithm is described in "Specifications for the Data Encryption Standard" (Federal Information Processing Standards Publication 46). Intermediate results in this implementation differ from the standard because the initial permutation and the inverse initial permutation are performed in each iteration. This does not affect the result in the final ciphertext or plaintext, but reduces the execution time.

Operation:

(i) If H = 0 then Encrypt round (R7-R0, R15-R8, K)
If H = 1 then Decrypt round (R7-R0, R15-R8, K)

	Syntax:	Operands:	Program Counter:
(i)	DES K	0x00≤K≤0x0F	PC ← PC + 1

16-bit Opcode:

1001	0100	KKKK	1011

Example:

```
DES 0x00
DES 0x01
...
DES 0x0E
DES 0x0F
```

Words 1 (2 bytes)

Cycles 1

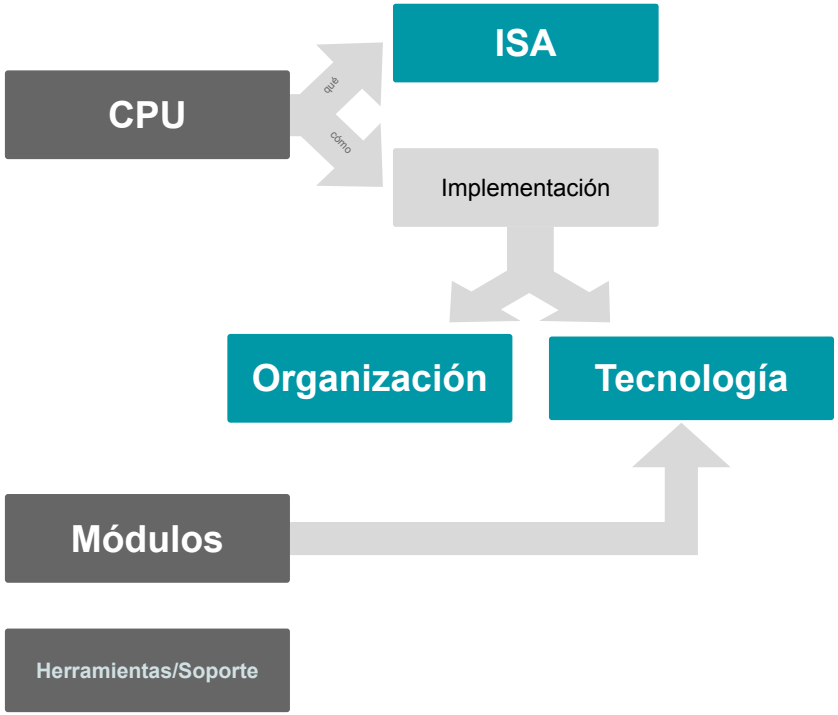
Note: If the DES instruction is succeeding a non-DES instruction, an extra cycle is inserted.

La familia XMEGA - Performance

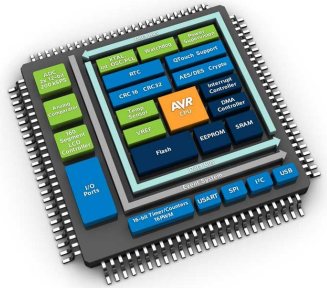
Clear Sel.	Processor	Cert.	Compiler	Execution Memory	MHz	Cores	CoreMark	CoreMark / MHz	Threads	Date↓
☐	Microchip ATXMEGA128A1U	✓	IAR C/C++ Compiler V7.10...	Stack	32	1	29.28	0.92	1	2018-09-20
☐	Atmel ATXMEGA128A1		GCC v4.5.1	internal SRAM	2	1	0.37	0.18	1	2012-01-15
☐	Atmel ATXMEGA128A1		GCC v4.5.1	internal SRAM	2	1	0.87	0.44	1	2012-01-15

Clear Sel.	Processor	Cert.	Compiler	Execution Memory	MHz	Cores	CoreMark	CoreMark / MHz	Threads	Date↓
☐	Microchip ATMEGA4809	✓	IAR C/C++ Compiler V7.10...	Stack	20	1	20.91	1.05	1	2018-09-20
☐	Atmel ATmega2560		avr-gcc 4.3.2	Internal flash & RAM...	8	1	4.25	0.53	1	2010-07-12
☐	Atmel ATmega644		avr-gcc-4.3.2	Flash & SRAM 20 M...	20	1	10.81	0.54	1	2009-11-17

	Compiler	Execution Memory	MHz	CoreMark	CoreMark / MHz↑
Atmel AT89C51RE2 in X2 mode (8051 - 6 clocks/machine cycle)	Keil C51 v8.18	Internal RAM & flash (static)	22	2.36	**0.106**
Microchip PIC18F46K2	Microchip MPLAB XC8 v1.32	Code in Flash, Data in RAM	64	7.23	**0.113**
STM8	Origen dudoso		24	5	**0.21**
Atmel ATmega644	avr-gcc-4.3.2	Flash & SRAM (static)	20	10.81	**0.54**
Microchip ATXMEGA128A1U	IAR C/C++ V7.10	Stack **???**	32	29.28	**0.92**
Microchip ATMEGA4809	IAR C/C++ V7.10	Stack **???**	20	20.91	**1.05**



XMEGA



CPU:

ISA: AVR con mínimas modificaciones
ORG: mejora x2 en la cantidad de ciclos sólo de algunas instrucciones
TEC: mejora x2 (32 MHz)

Módulos:

Modificación muy importante

Herramientas:

IDE: mismo
KITS: más avanzados

Soporte:

Recomendado para nuevos diseños (no hobbista)

Arquitectura AVR®

REPERTORIO DE INSTRUCCIONES
32 registros de 8 bits, ALU de 8 bits enteros, load/store tipo (0,2).
IO mapeada en memoria.
Dir. 16 bits (registros dobles) y pila.
126 instrucciones, 7 modos de direccionamiento, interrupciones.

Implementación

ATmega328

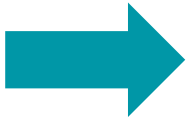
ORGANIZACIÓN
Harvard (datos de 8 bits, instrucciones de 16 bits).
Segmentación en dos etapas (F-E).

TECNOLOGÍA
130 nm, core 48,000 tr, SMD plástico, 3.3V, 16 MHz.
32K FLASH, 2K SRAM

Microchip PIC16

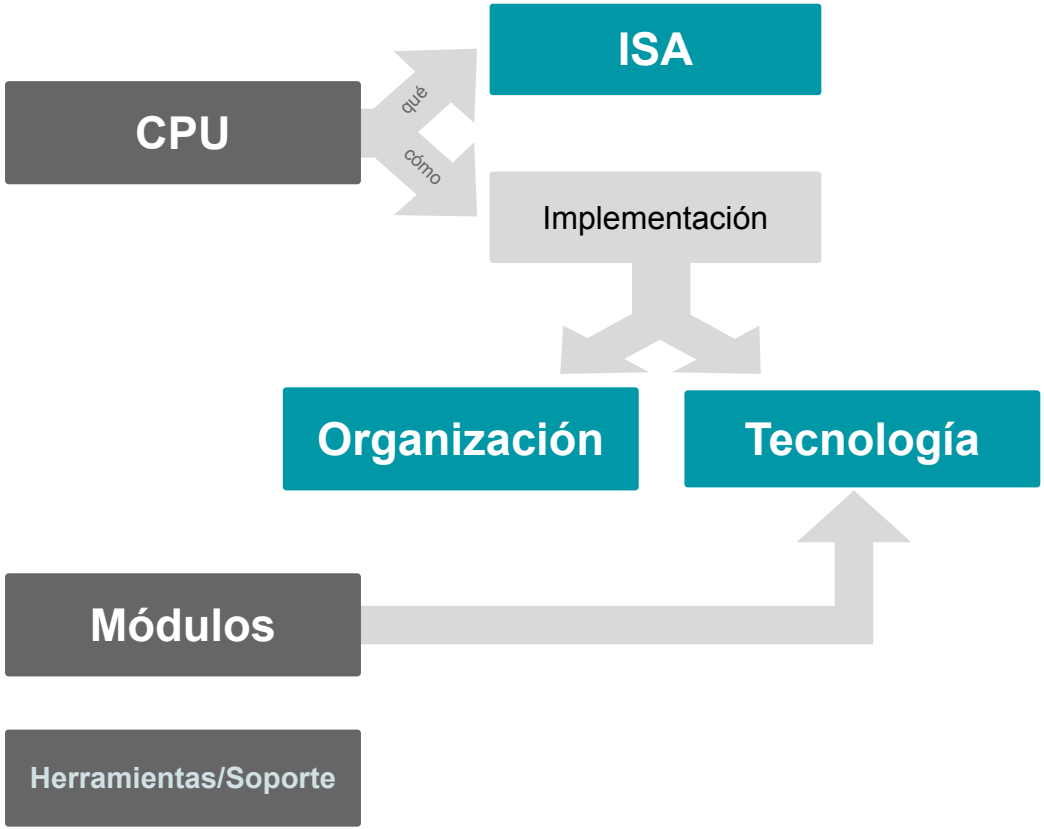
MICROCONTROLADORES ACTUALES (8 bits)

1. Motorola/Philips/NXP **CPU08**
2. Intel/Atmel **8051**
3. Atmel/Microchip **AVR**
4. STMicroelectronics **STM8**
5. Microchip **PIC16**



1. Microchip **PIC24** (16 bits) y **dsPIC** (16 bits)
2. Texas Instruments **MSP430** (16 bits) y **C2000** (DSP FPU 32 bits)
3. ARM **Cortex-M** (32 bits) - NXP, ST, Texas, Microchip, etc.

1. Maxim **MAXQ10** (8bits) y **MAXQ20** (16 bits) sólo para valientes



Medibles

Parámetros eléctricos (rango) —> V, I, T, f

Tamaño del código para un determinado problema y RAM necesaria —> Memoria de programa y de datos.

Potencia a fmax y en stand-by —> Watt y Watt/MHz

Performance a fmax —> Dhrystone/s y Coremark

Performance relativa —> DMIPS

Organización —> DMIPS/MHz y Coremark/MHz

Eficiencia —> DMIPS/Watt

Costo unitario —> $ MercadoLibre x1, uSs DigiKey x3000

Costo del sistema de desarrollo —> $ Kit, instrumentos y herramientas

No-tan-medibles

Tiempo de desarrollo y experiencia necesaria

Futuras migraciones y mejoras, soporte

Microchip PIC

Microchip Technology Inc., Chandler, Arizona, Estados Unidos. En 2016 compraron AVR, California.
La arquitectura PIC fue diseñada en los 80s como periférico de un procesador más poderoso.
Se utilizaron algunos criterios de diseño que más adelante se identificarían con los diseños RISC.
1993 primer lanzamiento con EEPROM y en 1998 primer lanzamiento con FLASH.
El largo de la instrucción varía según el modelo (12-16).

Fueron los primeros con FLASH (“F”) y low voltage (“L”)

ISA

8-bit RISC Harvard (8-bit data, 12/14/16-bit instrucciones con 3-6 opcode)
8-bit ALU, 1 **acumulador** de 8-bit (W: working register) IMPLICITO (no codificado)
Grupo reducido de instrucciones de largo fijo **RISC**
Toda la RAM funciona como **registros**, los primeros 12 son de de uso especial.
Para acceder a toda la RAM hay que cambiar de banco. Los 12 se preservan.
Todo mapeado en memoria: CPU, ports, peripheral registers, ALU status flags y PC.
Un registro para direccionamiento **indirecto** FSR (indice).
Pila en hardware, no direccionable. Complica código de alto nivel.
Repertorio poco ortogonal?

¿MARIE Harvard?

No hace diferencia entre RAM y Registros

La pila no está en RAM

PIC10F320
(SOT-23 2x3mm)

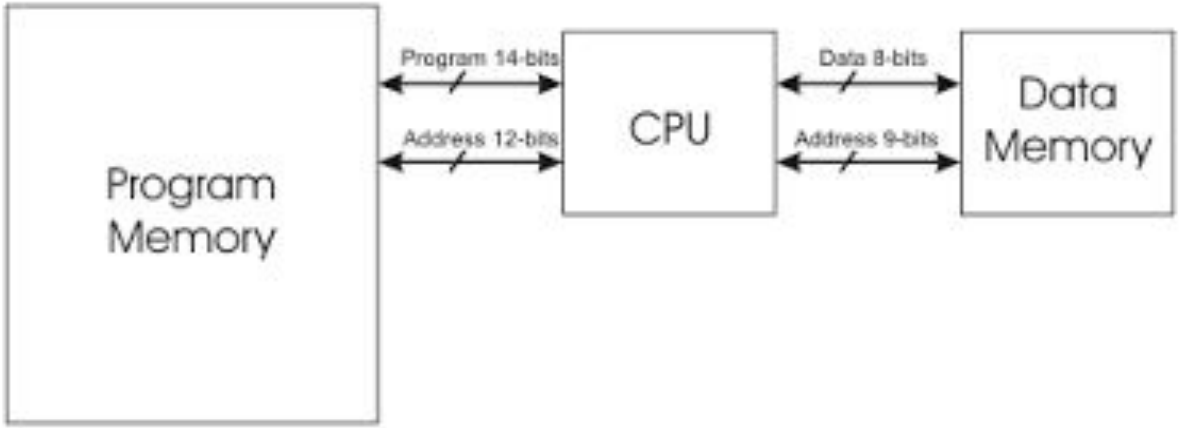
Implementación

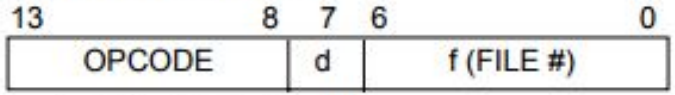
Harvard, ciclo de instrucción = 4 ciclos de reloj, pipeline de 2 etapas
Las instrucciones ejecutan en 1 o 2 ciclos de instrucción (saltos).
Rápido y bajo consumo. Gran variedad de módulos.

	Largo de la instrucción	No de instrucc	Max program words (ROM)	Registros 8-b (RAM)	Stack
PIC 10/12	12	35	512	32	2
PIC 16	14	49	8K (14KB)	368	8
PIC 18	16	83	2M (2^{21})	16x256	31

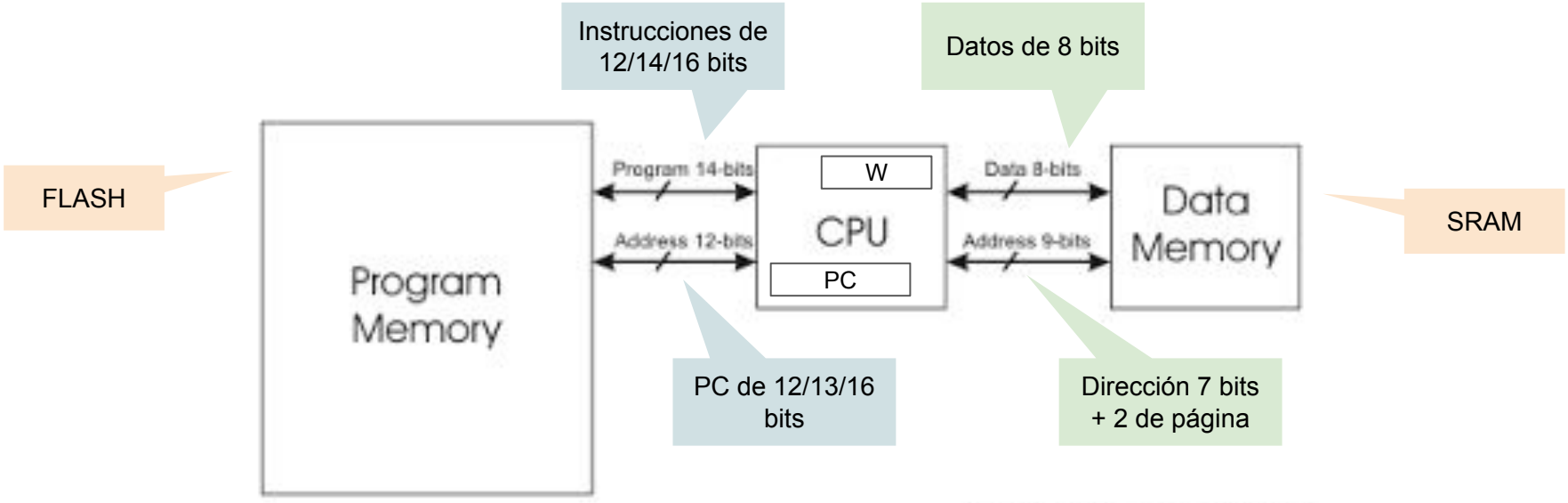
Herramientas de desarrollo

Compiladores C propios, grabadores baratos y bootloader. IDE MPLAB (oficial). Ver NUEVO “Microchip Studio”.

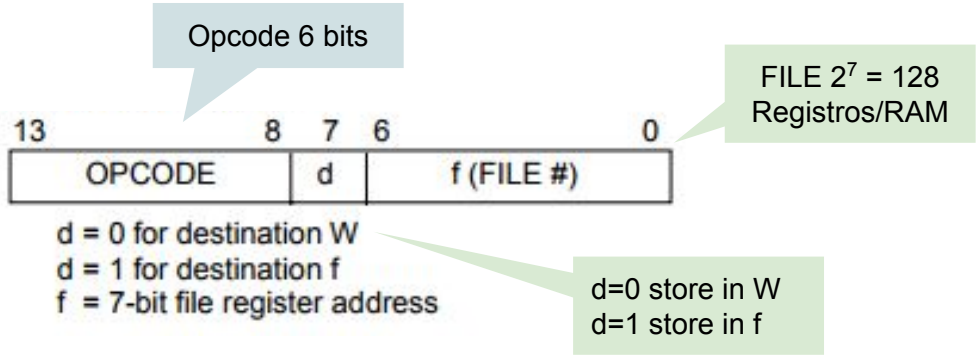




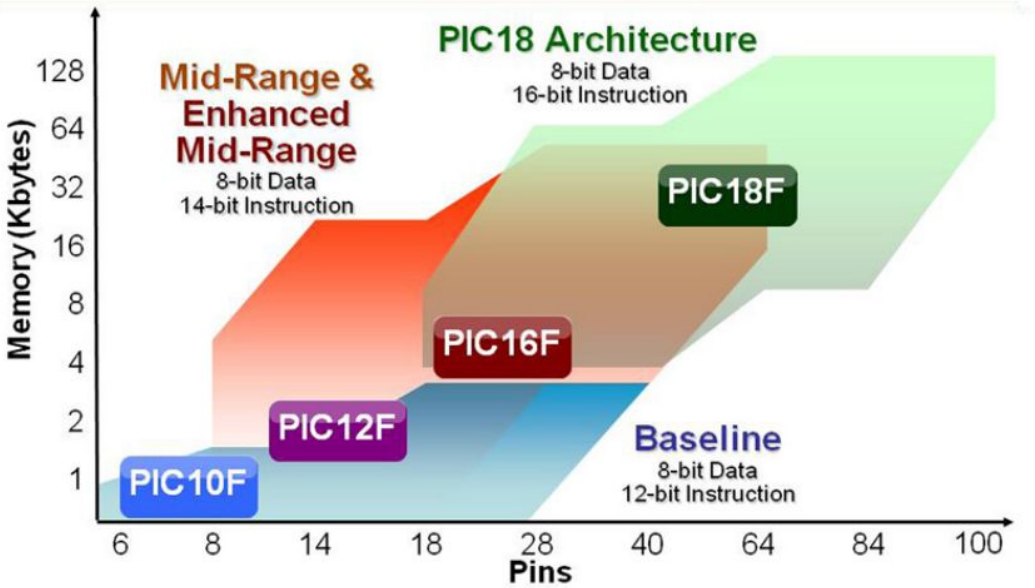
d = 0 for destination W
d = 1 for destination f
f = 7-bit file register address




```
; ADDWF suma un registro F con el acumulador W
; opcode 7 = 000111b

ADDWF 0x10,1      ; R16 <- W+R16
                  ; 000111 1 0010000b = 0x0790
ADDWF 0x05,0      ; W <- W+R5
                  ; 000111 0 0000101b = 0x0705
```

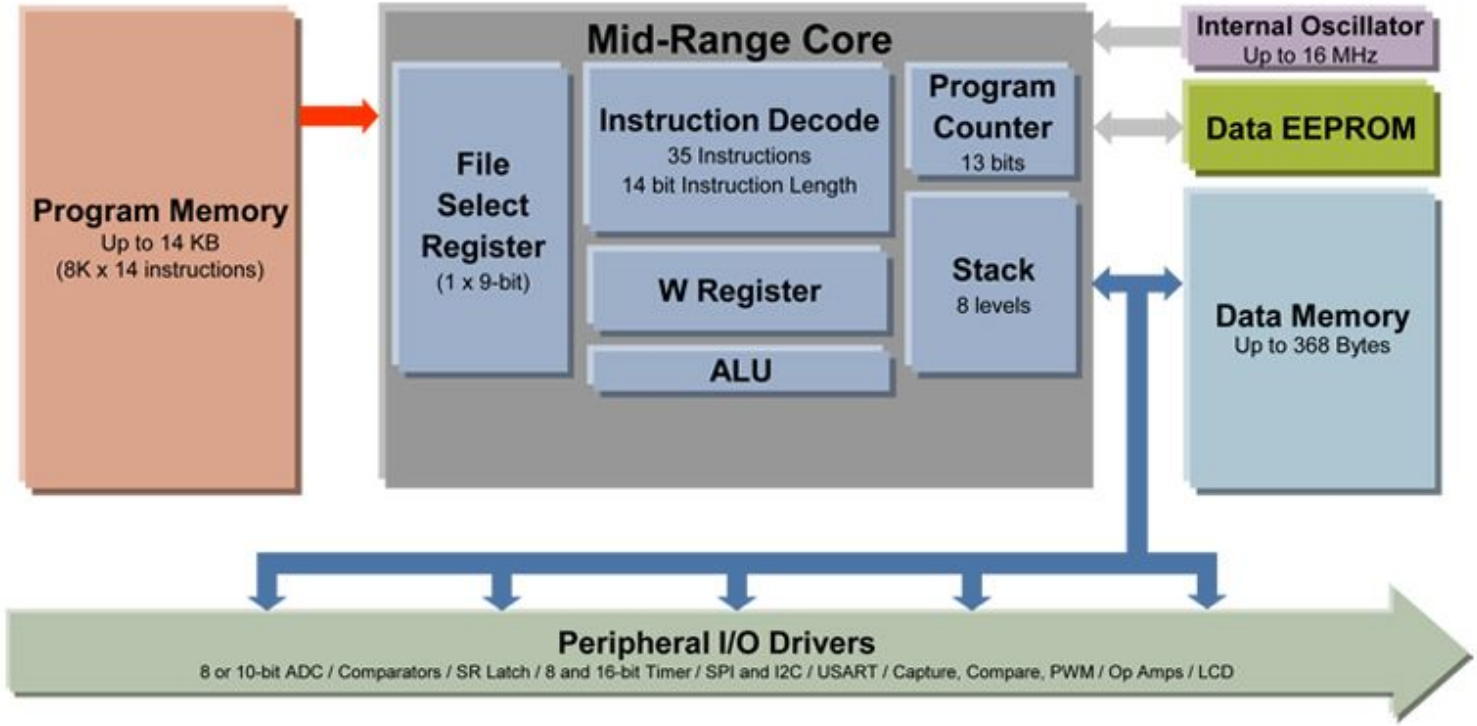


Microchip PIC16 - Mid range



	Largo de la instrucción	Program words (ROM)	Registros 8-b (RAM)	Stack
PIC 10/12	12	512	32	2
PIC 16	14	8K (14KB)	368	8
PIC 18	16	2M	16x256	31

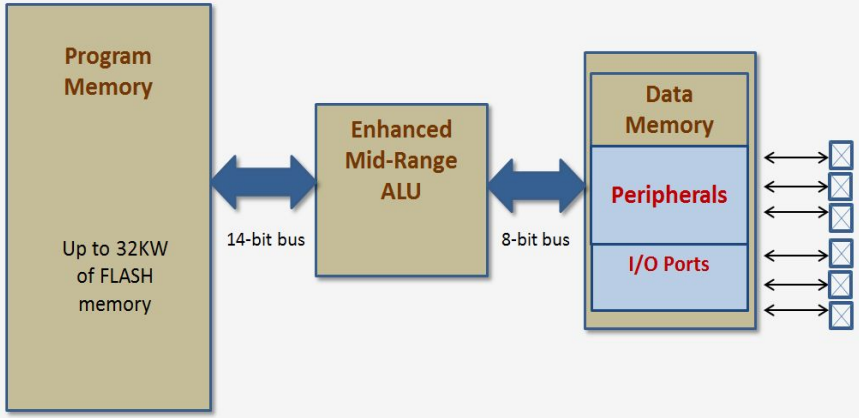
Reflexionar sobre las ventajas y desventajas de no poder direccionar la pila

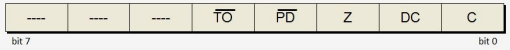


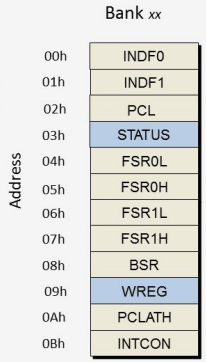
https://microchipdeveloper.com/8bit:emr-architecture

Microchip PIC16 - Memory map

Registros, ports y periféricos, todo mapeado en la memoria
Bancos de 128 bytes

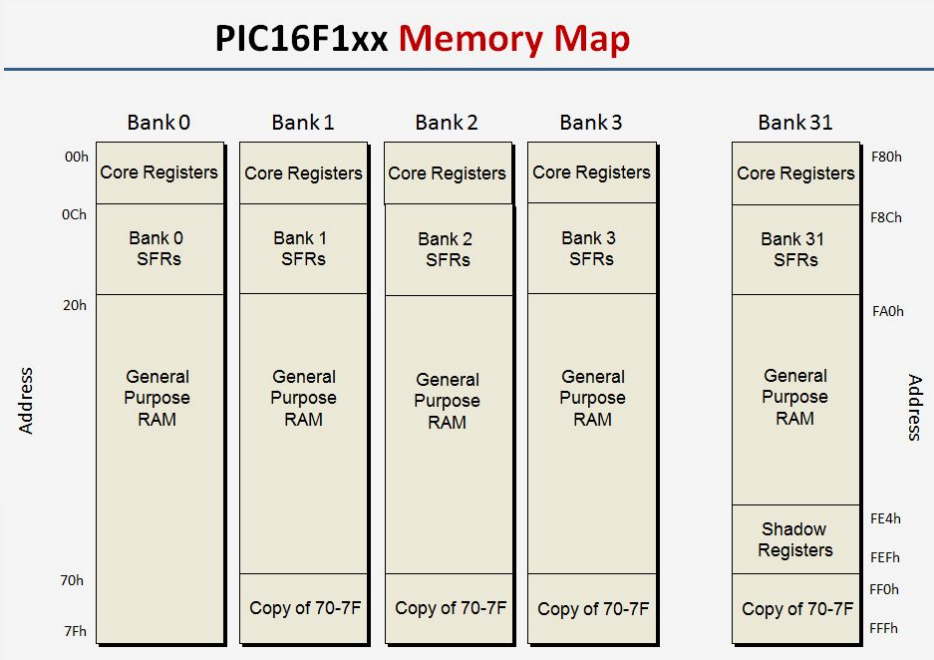






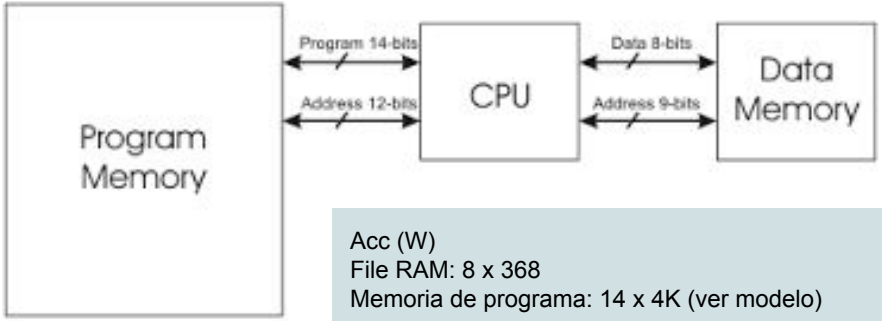
12 registros “core” son los mismos en todos los bancos, PC, W, STATUS, FSR, etc.

FSR selecciona el banco (2 bits) y sirve para **direccionamiento indirecto (7 bits)**



Microchip PIC16 - Organización

Harvard segmentado en 2 etapas FETCH-EXEC.
Cada etapa toma 4 ciclos de reloj.
Ciclo de instrucción = clock /4 (NO CONFUNDIRSE)
En la etapa de FETCH se captan los 14 bits en un ciclo de instrucción.
Las ejecuciones toman todas un ciclo de instrucción,
excepto las que alteran el PC (BRA, CALL, etc) que toman dos.
Etapa EXEC:



Acc (W)
File RAM: 8 x 368
Memoria de programa: 14 x 4K (ver modelo)

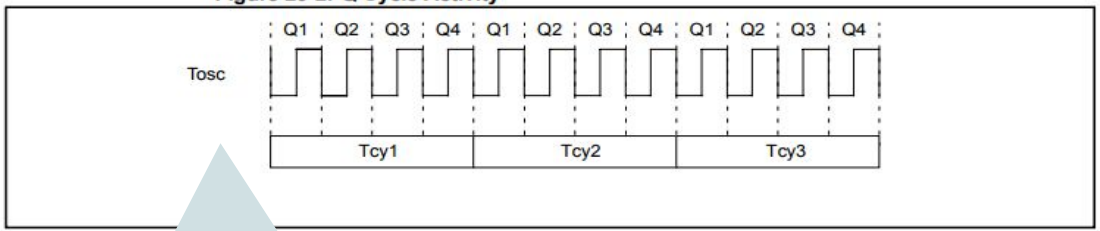
4 ciclos Q por ciclo de instrucción:
- Q1: Decodificación
- Q2: Leer dato
- Q3: Ejecutar la operación
- Q4: Escribir resultado

The four Q cycles that make up an instruction cycle (Tcy) can be generalized as:

- Q1: Instruction Decode Cycle or forced No Operation
- Q2: Instruction Read Cycle or No Operation
- Q3: Process the Data
- Q4: Instruction Write Cycle or No Operation

Each instruction will show the detailed Q cycle operation for the instruction.

Figure 29-2: Q Cycle Activity



FETCH-EXEC segmentado

OJO!

DATASHEET:
All instructions single cycle (400 ns @ 10 MHz) except for program branches which are two-cycle

< 2.5 MIPS @ 10 MHz

PIC vs. AVR: https://www.avrfreaks.net/forum/instruction-cycle-clock-tick

Microchip PIC16 - Organización

https://microchipdeveloper.com/8bit:emr-architecture

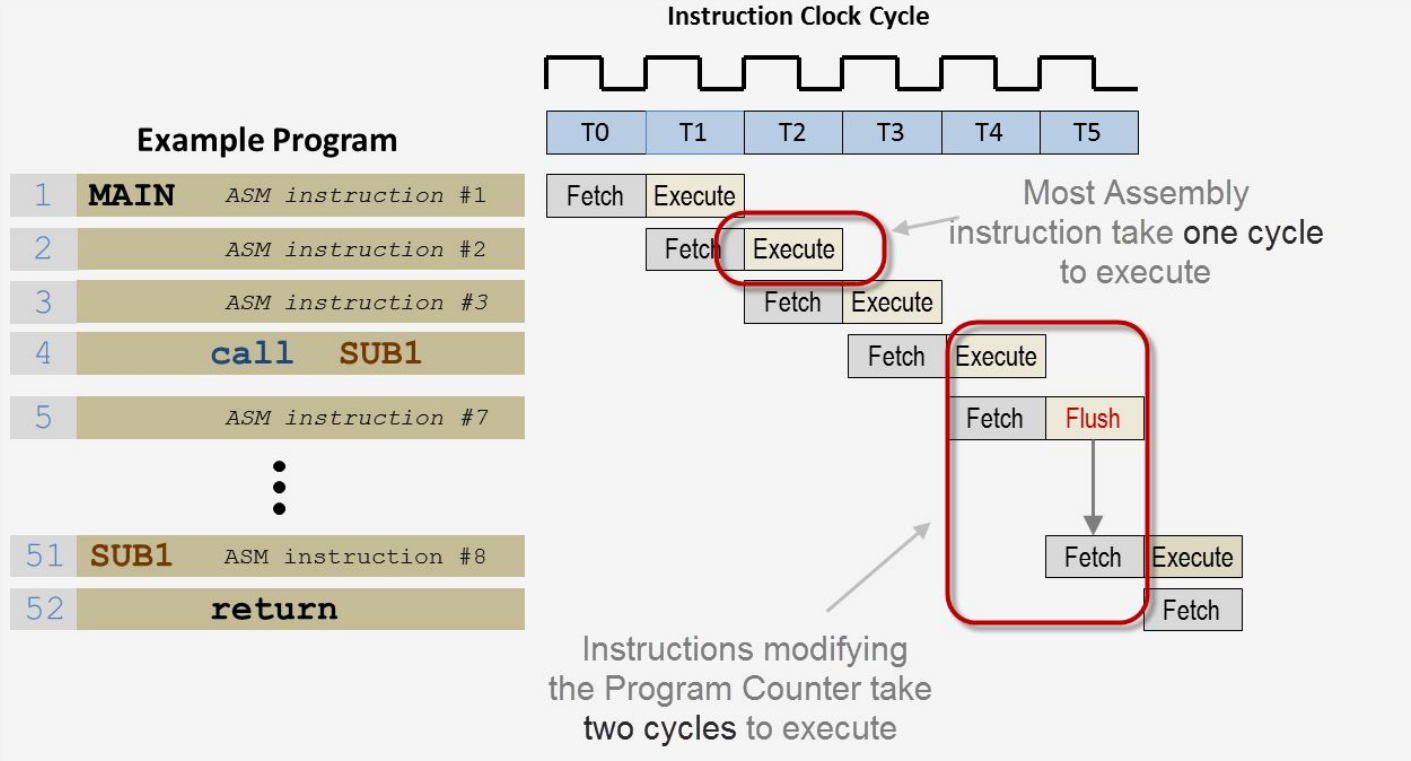


Table 29-1: Midrange Instruction Set

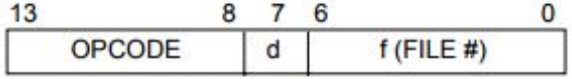
Mnemonic, Operands		Description	Cycles	14-Bit Instruction Word MSb			LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	1,2
CLRF	f	Clear f	1	00	0001	lfff	ffff	Z	2
CLRW	-	Clear W	1	00	0001	0xxx	xxxx	Z	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	1,2
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff		1,2,3
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	1,2
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	1,2
MOVWF	f	Move W to f	1	00	0000	lfff	ffff		
NOP	-	No Operation	1	00	0000	0xx0	0000		
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		1,2
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		3
LITERAL AND CONTROL OPERATIONS									
ADDLW	k	Add literal and W	1	11	111x	kkkk	kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11	1001	kkkk	kkkk	Z	
CALL	k	Call subroutine	2	10	0kkk	kkkk	kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00	0000	0110	0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10	1kkk	kkkk	kkkk		
IORLW	k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z	
MOVLW	k	Move literal to W	1	11	00xx	kkkk	kkkk		
RETFIE	-	Return from interrupt	2	00	0000	0000	1001		
RETLW	k	Return with literal in W	2	11	01xx	kkkk	kkkk		
RETURN	-	Return from Subroutine	2	00	0000	0000	1000		
SLEEP	-	Go into standby mode	1	00	0000	0110	0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11	110x	kkkk	kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z	

Table 29-2: Instruction Description Conventions

Field	Description
f	Register file address (0x00 to 0x7F)
W	Working register (accumulator)
b	Bit address within an 8-bit file register (0 to 7)
k	Literal field, constant data or label (may be either an 8-bit or an 11-bit value)
x	Don't care (0 or 1) The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select; d = 0: store result in W, d = 1: store result in file register f.
dest	Destination either the W register or the specified register file location
label	Label name
TOS	Top of Stack
PC	Program Counter
PCLATH	Program Counter High Latch
GIE	Global Interrupt Enable bit
WDT	Watchdog Timer
$\overline{TO}$	Time-out bit
$\overline{PD}$	Power-down bit
[ ]	Optional
( )	Contents
→	Assigned to
< >	Register bit field
∈	In the set of
italics	User defined term (font is courier)

Figure 29-1: General Format for Instructions

Byte-oriented file register operations



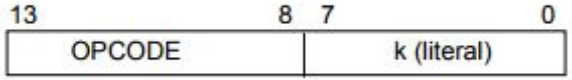
d = 0 for destination W
d = 1 for destination f
f = 7-bit file register address

Bit-oriented file register operations

b = 3-bit bit address
f = 7-bit file register address

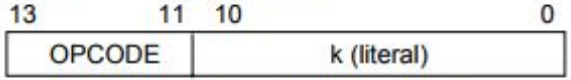
Literal and control operations

General



k = 8-bit literal (immediate) value

`CALL` and `GOTO` instructions only



k = 11-bit literal (immediate) value

Dos instrucciones de suma: direccionamiento directo (registro) e inmediato (literal) (idem SUB, AND, XOR)

ADDWF — Add W and f

Syntax: [*label*] ADDWF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (W) + (f) → destination

Status Affected: C, DC, Z

Encoding:

00	0111	dfff	ffff

Description: Add the contents of the W register with register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

Example 1

```
ADDWF   FSR, 0
```

Before Instruction
W = 0x17
FSR = 0xC2
After Instruction
W = 0xD9
FSR = 0xC2

Acumulador + “Registro” (f de 7 bits), con dos destinos posibles (d) [DIFERENCIA CON MARIE]

ADDLW — Add Literal and W

Syntax: [*label*] ADDLW k

Operands: $0 \le k \le 255$

Operation: (W) + k → W

Status Affected: C, DC, Z

Encoding:

11	111x	kkkk	kkkk

Description: The contents of the W register are added to the eight bit literal 'k' and the result is placed in the W register.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process data	Write to W register

Example1

```
ADDLW   0x15
```

Before Instruction
W = 0x10
After Instruction
W = 0x25

Direccionamiento inmediato sólo disponible para el acumulador W

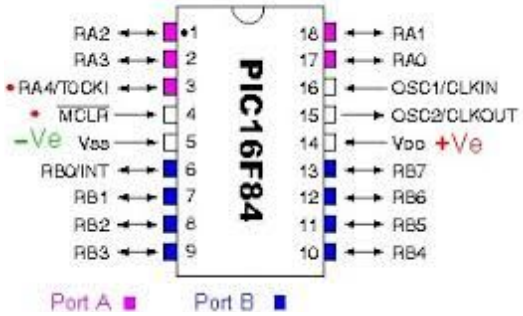
Distintos repertorios de instrucciones

8-bit PIC® Microcontrollers

Product Family	Pin Count	Program Flash Memory (KB)	RAM (KB)	Data EE (B)	Intelligent Analog							Waveform Control												Logic and Math			Safety and Monitoring			Communications					User Interface			Low Power and System Flexibility			
					ADC (# of bits)	Comp	HSComp	DAC (# of bits)	OPA	SlopeComp / PRG	ZCD	CCP/ECCP	10-bit PWM	16-bit PWM	COG	CWG	NCO	DSM	HLT (8-bit)	Universal Timer	NCO (20-bit)	SMT (24-bit)	RTCC	TEMP/TS	CLC	MULT	MathACC	CRC/SCAN	HLT	WWDT	USART	UART with Protocols	I2C/SPI	USB with ACT	LIN Capable	mTouch® Sensing	HCVD	LCD w/ charge pump	PPS	IDLE/DOZE/PMD	DMA/VI	DIA/MAP
PIC10(L)F3XX	6	384–896 B	0.064	HEF	8								✓			✓	✓				✓			✓	✓										✓						
PIC16F152XX	8–40	3.5–28	0.5–2	–	10							✓	✓			✓	✓	✓	✓					✓						1		✓		✓				✓	✓		✓
PIC12/16 LF155X/6X	14–20	7–14	1.024	HEF	10[2]								✓											✓						✓		✓		✓	✓	✓					
PIC16(L)F145X	14–20	14	1.024	HEF	10	✓							✓			✓								✓						✓		✓	✓	✓	✓						
PIC1X(L)F157X	8–20	1.75–14	1.024	HEF	10	✓		5						✓		✓				✓				✓						✓				✓	✓			✓			
PIC16(L)F153XX	8–48	3.5–28	2.048	HEF	10	✓		5			✓	✓	✓			✓	✓		✓		✓			✓	✓				✓	✓	2		✓		✓	✓			✓	✓		✓
PIC1X(HV)F752/53	8–14	1.75–3.5	0.128	–	10		✓	5/9	✓	SC		✓			✓				✓										✓						✓						
PIC1X(L)F161X	8–14	3.5	0.256	HEF	10	✓		8			✓	✓				✓			✓			✓		✓			✓	✓	✓						✓						
PIC16(L)F161X[8]	14–20	7–14	1.024	HEF	10	✓		8			✓	✓	✓			✓			✓			✓		✓	✓		✓	✓	✓	✓		✓		✓	✓			✓			
PIC18-Q40/1	14–20	16–32	1–4	512	12[4]	✓		8	✓[5]		✓	✓		✓		✓	✓	✓	✓			✓		✓	✓	✓		✓	✓	✓	2	1	✓		✓	✓	✓		✓	✓	✓	✓
PIC16(L)F170X/71X	14–40	3.5–28	2.048	HEF	10		✓	5/8	✓		✓	✓	✓		✓		✓				✓			✓	✓						✓		✓		✓	✓	✓		✓			
PIC16(L)F176X/7X	14–40	7–28	2.048	HEF	10		✓	5/10	✓	✓	✓	✓	✓	✓	✓			✓	✓	✓				✓	✓				✓		✓		✓		✓	✓			✓			
PIC16(L)F183XX	8–20	3.5–14	2.048	256	10	✓		5				✓	✓			✓	✓	✓			✓			✓	✓						✓		✓		✓	✓			✓	✓		
PIC16(L)F184XX	14–28	7–28	2.048	256	12[4]	✓		5			✓	✓	✓			✓	✓		✓		✓	✓		✓	✓					✓	✓		✓			✓	✓		✓	✓		✓
PIC16(L)F188XX	28–40	7–56	4.096	256	10[4]	✓		5			✓	✓	✓			✓	✓	✓	✓		✓	✓		✓	✓			✓	✓	✓	✓		✓		✓	✓	✓		✓	✓		
PIC18-Q10	28–40	16–128	1–3.6	256-1K	10[4]	✓		5			✓	✓	✓			✓		✓	✓					✓	✓		✓	✓	✓	✓	2		✓		✓	✓	✓		✓	✓		
PIC18-Q43	28–48	32–128	2–8	1024	12[4]	✓		8			✓	✓		✓		✓	✓	✓	✓			✓		✓	✓			✓	✓	✓	4	1	✓		✓	✓	✓		✓	✓	✓	✓
PIC18-Q84[8]	28–48	64–128	8–13	1024	12[7]	✓		8			✓	✓		✓		✓	✓	✓	✓	✓	✓	✓		✓	✓			✓	✓	✓	3	2	✓		✓		✓		✓	✓	✓	✓
PIC16(L)F191XX	28–64	14–56	4.096	256	12[4]	✓		5			✓	✓	✓			✓			✓			✓	✓	✓	✓				✓	✓	✓		✓		✓	✓	✓	✓	✓	✓		✓
PIC18-K40	28–64	16–128	3.728	256-1K	10[4]	✓		5			✓	✓	✓			✓		✓	✓					✓		✓		✓	✓	✓	5		✓		✓	✓	✓		✓	✓		
PIC18-K42	28–48	16–128	8.192	256-1K	12[4]	✓		5			✓	✓	✓			✓	✓	✓	✓		✓	✓		✓	✓	✓		✓	✓	✓	1	1	✓		✓	✓	✓		✓	✓	✓	✓
PIC18-J94	64–100	32–128	4.096	–	12	✓						✓											✓		✓					4		✓	✓	✓	✓		✓	✓	✓		

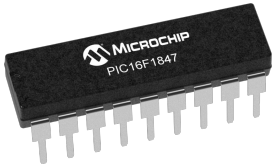
PIC16C84 (1993)

Fue el primero con EEPROM, grabación serie (boom hobbista 90s, como hoy es Arduino).
El modelo “F” posterior tiene FLASH.
68 bytes de RAM
Un 8-bit timer y 13 I/O pins
64 Bytes of EEPROM
10 MHz internal



PIC16F1847

Evolución pin compatible en producción:
https://www.microchip.com/wwwproducts/en/PIC16F1847
Nanowatt XLP Technology variant, 8K program memory, 1024 bytes data memory, 256 bytes EEPROM, 5× timers, hardware PWM, onchip 32 MHz/31 kHz precision oscillator, 12-input 10-bit ADC, 4× PLL. 5-bit DAC.
32 MHz internal



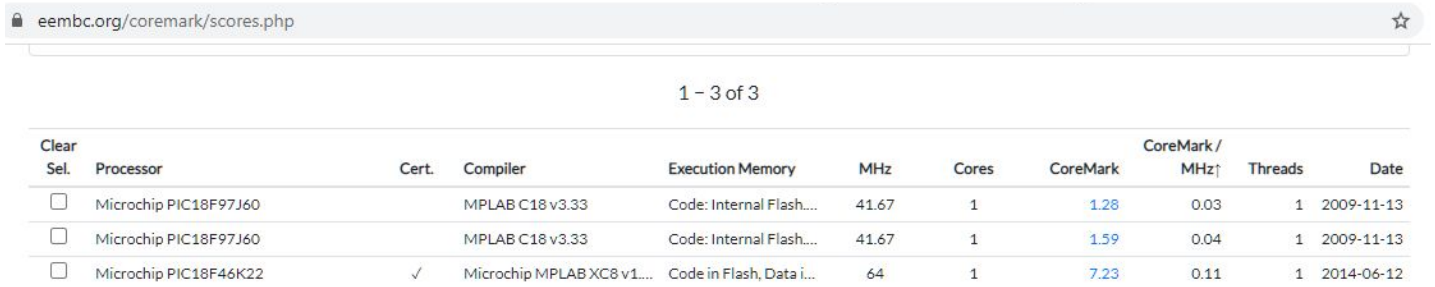
PIC18F46K20

CoreMark®
An EEMBC Benchmark

https://www.microchip.com/wwwproducts/en/PIC18F46K40
64K program memory, 3936 bytes data memory, 1024 bytes EEPROM
64 MHz internal

PERFORMANCE

CoreMark®
An EEMBC Benchmark


eembc.org/coremark/scores.php

1 – 3 of 3

Clear Sel.	Processor	Cert.	Compiler	Execution Memory	MHz	Cores	CoreMark	CoreMark / MHz↑	Threads	Date
☐	Microchip PIC18F97J60		MPLAB C18 v3.33	Code: Internal Flash....	41.67	1	1.28	0.03	1	2009-11-13
☐	Microchip PIC18F97J60		MPLAB C18 v3.33	Code: Internal Flash....	41.67	1	1.59	0.04	1	2009-11-13
☐	Microchip PIC18F46K22	✓	Microchip MPLAB XC8 v1....	Code in Flash, Data i...	64	1	7.23	0.11	1	2014-06-12

Los tres tienen instrucciones de 16 bits y datos de 8 bits

	Compilador	MHz	CoreMark	CoreMark / MHz↑
Microchip PIC18F46K20	Microchip MPLAB XC8 v1.32	64	7.23	**0.113**
STM8	Origen dudoso	24	5	**0.21**
Atmel ATmega644	avr-gcc-4.3.2	20	10.81	**0.54**

Si al PIC lo pienso como un micro de 16 MHz -> **0.452**

Si al AVR lo lo hago funcionar a 16 MHz -> **8.64**

A igual reloj, AVR/PIC=0.54/0.452=1.2
AVR es 20% más rápido

A fondo, AVR/PIC=10.81/7.23
AVR es 50% más rápido

REFERENCIAS

Arquitectura “Enhanced Mid-Range”, repertorio, ejemplos
https://microchipdeveloper.com/8bit:emr

PIC16C84 Datasheet
http://ww1.microchip.com/downloads/en/DeviceDoc/30445d.pdf

PIC16F1847 Datasheet
http://ww1.microchip.com/downloads/en/DeviceDoc/40001453G.pdf

Folleto PIC-AVR
https://ww1.microchip.com/downloads/en/DeviceDoc/30009630M.pdf

IDEs
https://www.microchip.com/mplab/mplab-x-ide
https://www.microchip.com/mplab/microchip-studio

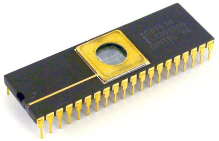
60-day trial

STMicroelectronics
STM8

Legacy 8-bit

(DIP8)

- **NXP CPU08:** Motorola 68HC05 y derivados (1980s hasta la actualidad), von Neumann, acumulador de 8 bits.
- **Intel 8051** (1981, open): 8-bit data, 16-bit addr, Harvard, 8 registros de 8 bits.



(DIP40, cerámica, oro y vidrio)

8-bit

(DIP8)

(SOP-8)

- **Microchip PIC16/18:** 8-bit Harvard modificado (8-bit data, 14/16-bit instr), 8-bit ALU y acc (W), RAM = bancos de 8-bit register file + SFR. Ciclo de máquina = 4 ciclos de reloj. Saltos dos ciclos.
- **Atmel AVR** (ATmega, ATtiny): 8-bit Harvard modificado (puede escribir mem prog), 32 GPR de 8 bits y 64 I/O mapeados en memoria, 131 instr. de 16 bits, 5 modos de direccionamiento, PC y SP de 16 bits, multiplicador. Ciclo de máquina = 1 ciclo de reloj. Instrucciones de 1 o más ciclos.
- **STMicroelectronics STM8**: 8-bit Harvard acc 3-stage pipeline, 6 reg mapeados en memoria: ACC 8-b, X-Y 16-b index reg, PC 24-b (16MB), SP 16-b. CC; 80 inst de 16-32 bits, 20 modos dir.

16-bit

(DIP28)

(SOP-20)

- **Texas Instruments MSP430**: 16-bit von Neumann, instrucciones de largo fijo, repertorio ortogonal, 16 registros de 16 bits, ultra-low-power modes, constant generator, 27 instrucciones (+24 simuladas) y 7 modos de direccionamiento, byte operations.
- **Microchip PIC24:** 16-bit Harvard (instrucciones de 24-bit y 16-bit registers y ALU). Similar a PIC16.
- **Maxim MAXQ**: 8/16-bit RISC (dos modelos MAXQ10/20), Harvard modificado (MMU), transport-triggered (una única instrucción: MOVE, 16 módulos de 32 registros). Ultra-quiet ("intelligent clock management"), single-cycle instructions (más proporción que la competencia) pero sin segmentación. 8 a 16 acumuladores A[0] a A[15] con un reg puntero AP: A[AP], destino implícito para operaciones de ALU. Stack aparte. Loop counters, DJNZ single-cycle.

32-bit

(QFP-48) (DIP-28)

ARM Cortex-M: 32-bit RISC segmentado
Instrucciones 32-bit, todas condicionales, barrel shifter.

NXP LPC800:
Cortex-M0 (DIP-8)

Se dejó de fabricar?

STM8

“ST's 8-bit microcontroller platform is implemented around a high-performance 8-bit core and a state-of-the-art set of peripherals. This platform is manufactured using an ST-proprietary 130 nm embedded non-volatile memory technology. The STM8 allows fast and safe development through enhanced stack pointer operations, advanced addressing modes and new instructions.“

ST es una empresa franco-italiana con sede en Suiza y domicilio fiscal en Holanda. Desde 1987 fabricaron 6800, Z80, etc.
Fueron exitosas sus implementaciones de Arm® Cortex®-M: STM32.
Fueron menos conocidos sus microcontroladores de 8 bits de diseño propio ST6 y ST7, previos a éste.
Este es un nuevo intento de 8 bits.

https://www.st.com/en/microcontrollers-microprocessors/stm8-8-bit-mcus.html

STM8 8-bit MCUs
Core up to 24 MHz

Mainstream	**Industrial, consumer and mass market**	Robust and reliable Up to 125 °C	**STM8S** Data EEPROM, 3 and 5 V families, precise RC
Ultra-low-power	**Ideal combination of low-power performance and features**	High-end analog IPs Active Halt < 1 µA	**STM8L** Data EEPROM, 1.65 and 3 V families, strong analog, LCD drivers, low-leakage technology
Automotive	**Long-term guarantee**	AEC-Q100 Up to 150 °C	**STM8AF** Data EEPROM, 3 and 5 V families, precise RC, LIN, CAN, grade 0 **ASIL Ready**
	Long-term guarantee	AEC-Q100 Up to 125 °C	**STM8AL** Data EEPROM, 1.65 and 3 V families, strong analog, LCD drivers, low-leakage technology **ASIL Ready**

Argumentos de venta para la serie S (mainstream)
https://www.st.com/en/microcontrollers-microprocessors/stm8s-series.html

- ST's STM8S series of mainstream 8-bit microcontrollers covers a large variety of applications in the industrial, consumer and computer markets, particularly where large volumes are concerned.
- Based on the STM8 **proprietary** core, the STM8S series benefits from ST's 130 nm technology and advanced core architecture performing up to **20 MIPS at 24 MHz**.
- In addition to embedded EEPROM and RC oscillators, the rich number of I/Os offered and the **performance of its embedded peripherals** are recognized as differentiating key points versus competition.
- The STM8S series is part of ST's **10-year product longevity** commitment program for STM32 and STM8 microcontrollers, ensuring a robust and reliable solution for designers.
- The associated development **toolchain** from affordable discovery kits to more complex evaluation kits and third-party tools makes it easy to develop with STM8S microcontrollers.

Argumentos de venta para la serie L (low power)
https://www.st.com/en/microcontrollers-microprocessors/stm8l-series.html

- ST's ultra-low-power product lines support a wide number of applications where consumption is critical, such as in portable devices. The STM8L, based on the 8-bit STM8 core, benefits from our proprietary **ultra-low-leakage process**, shared with the STM32L family, and features an ultra-low power consumption of 0.30 µA with the lowest power mode.
- **PRESENTA NUMEROS (ver)**

Argumentos de venta para las series AF y AL (automotive)
https://www.st.com/en/microcontrollers-microprocessors/stm8af-series.html

- ST's STM8AF series is intended for automotive applications where no compromise on parameters is possible, from reliability to system cost effectiveness. The STM8AF series is modular, provides high performance and offers the flexibility required for short development cycles. Its true data EEPROM, combined with the capability to withstand **up to 150 °C ambient temperature**, make the series a sustainable choice for automotive applications.
- ST's STM8AL **ultra-low-power** series for automotive applications puts green energy, application safety and power efficiency at the forefront. It is particularly suited to battery-operated functions such as remote keyless entry and tire pressure monitoring, as well as for applications where power consumption is critical over time: companion microcontroller, immobilizers and sensors.

STM8

- 8-bit Harvard architecture (dos buses, pero “unified 24-bit address space”, von Neumann, ISP)
- Tipo (1,1) - Acumulador
- 3-stage pipeline, 32-bit wide program memory bus - single cycle fetching for most instructions,
- 6 reg mapeados en memoria: (A, X, Y, SP, PC, CC) - Todo mapeado en un único mapa de memoria
  - A: acumulador de 8 bits
  - X-Y: 16-bit index registers (*) indexed indirect addressing
  - SP: 16-bit stack pointer (16K stack)
  - PC: 24-bit program counter (16MB) (**)
  - CC: Condition Code register, second interrupt enable bit, allowing four interrupt priority levels.
- 80 instrucciones de 16-32 bits (las de 16 bits sólo pueden acceder a los primeros 64K de memoria, casi todos tiene menos)
- 20 modos de direccionamiento (?!?!)
- Multiplicaciòn 8x8, divisiòn 16/8 y 16/16, bit manipulation
- Nested interrupts with three software priority levels, 32 interrupt vectors
- Periféricos según familias de aplicación
  - STM8S, mainstream MCUs
  - STM8L, ultra-low-power MCUs (<1uA)
  - STM8AF and STM8AL, automotive MCUs (150°C)
- Muy bajo costo
- ST-proprietary 130 nm embedded non-volatile memory technology
- Operación hasta 24 MHz (16 MHz interno) @24MHz & 3V -> 11mA
- Encapsulado hasta SO8

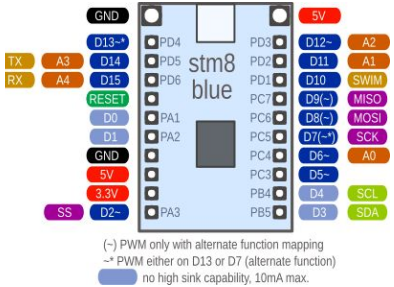
Can store your constants in the program memory and even run code from RAM

Tipo (1,1)

```
RRWA  X,A
RET
ADD   A,(0x03,X)
LD    (0x03,X),A
JRNC  0xf2b4
INC   (0x02,X)
JRNE  0xf2b4
INC   (0x01,X)
JRNE  0xf2b4
INC   (X)
RET
```

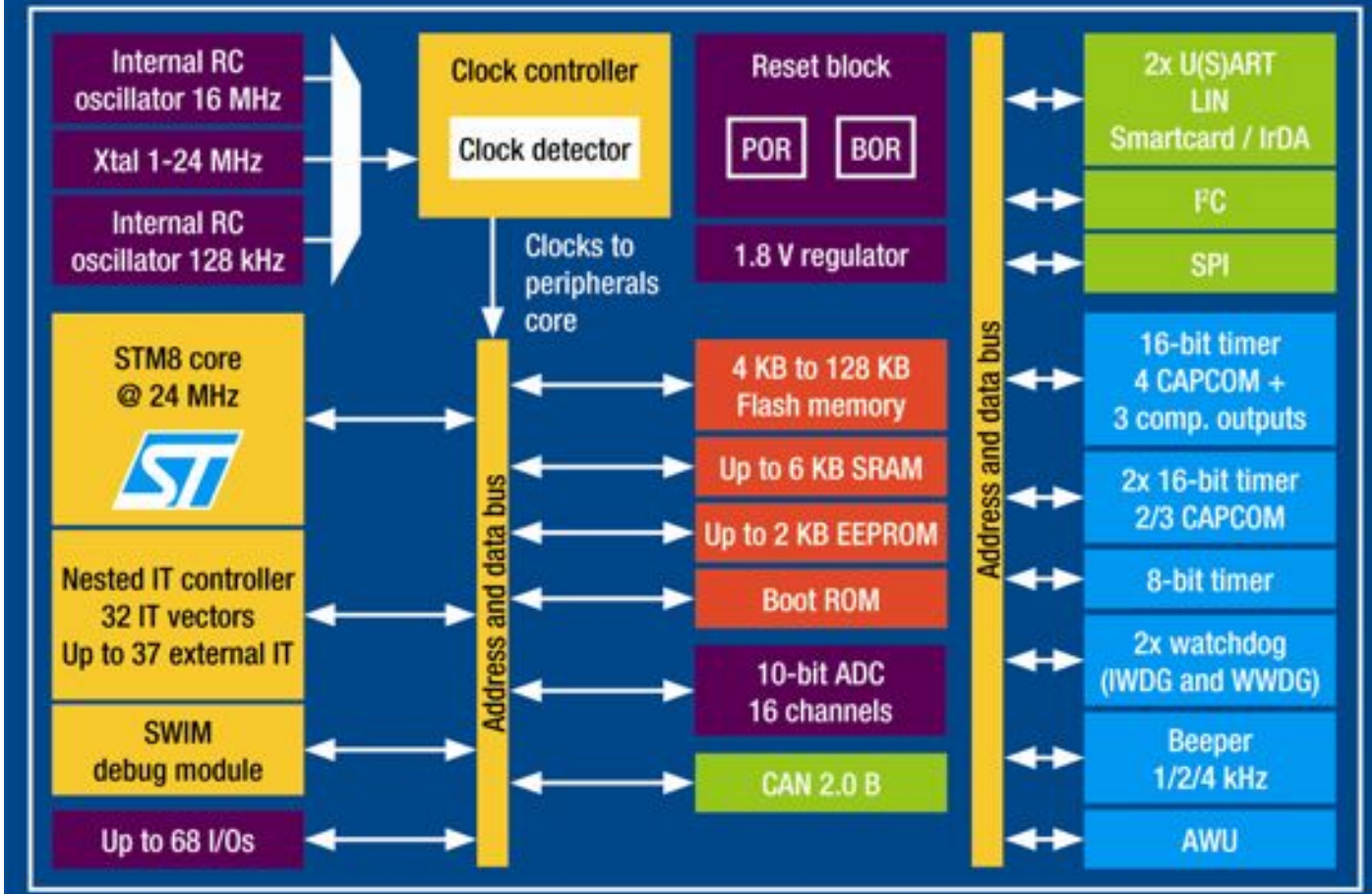
ST-Link compatible

STM8S103F3P6 equivalente a ATmega8
Flash 8kB
RAM 1kB
EEPROM 640 byte ML: S300

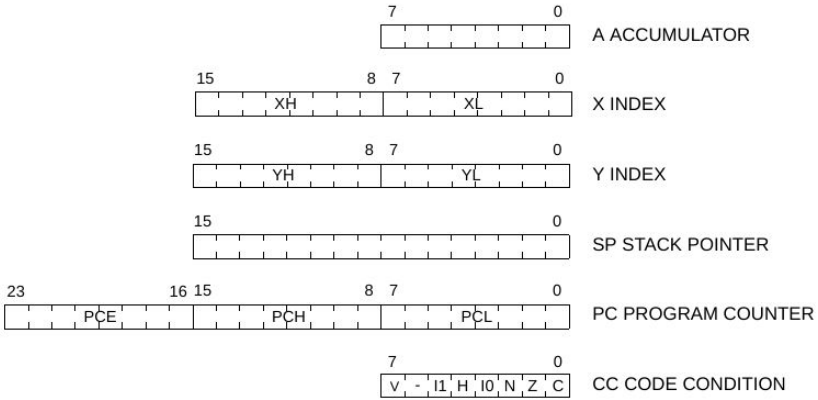


Mejoras al ST7 previo: (*) extendidos a 16 bits , (**) extendido a 24 bits (6x-10x performance)
Ver (muy bien explicado): https://en.wikipedia.org/wiki/ST6_and_ST7

STM8S block diagram



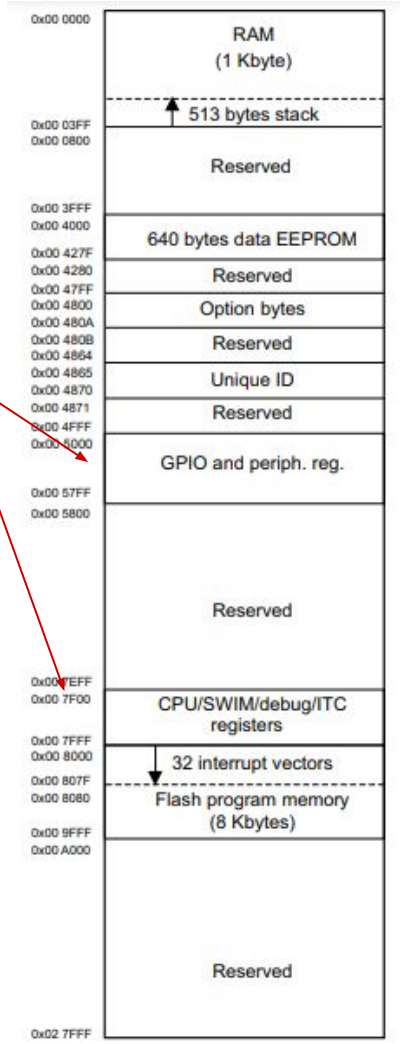
STM8: Programming model & memory map

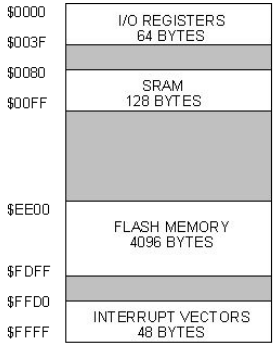


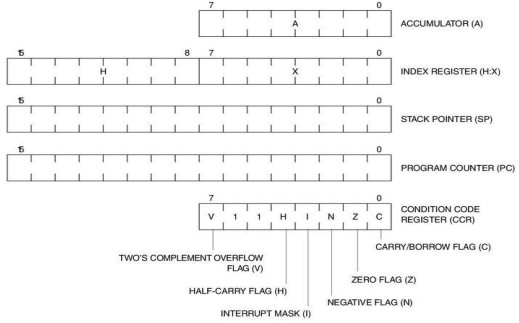
STM8

6 registros mapeados en memoria
IO mapeado en memoria

Organización:
Pipeline **3 etapas**, 24 MHz
Instrucciones de 1 ciclo si memoria < 64K (single-cycle), 24 MIPS?
Por qué dicen 20 MIPS a 24 MHz?

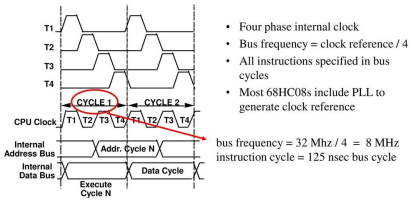






CPU08

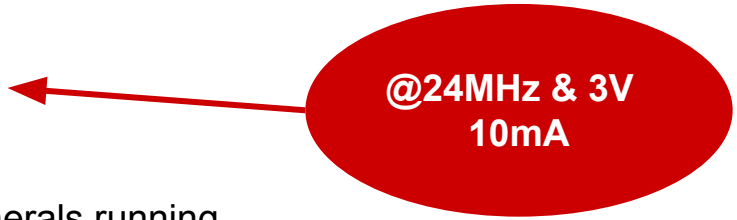
5 registros no pertenecen a la memoria. IO si.
Organización multiciclo:
Bus freq = clock/4 (original 8 MHz, nuevos 48 MHz).
Instrucciones de 1, 2, 3 o más bus cycles (multi-cycle), muy menor a 12 MIPS



ALGUNOS MEDIBLES QUE ENCONTRÉ POR AHÍ

Wide operating voltage range from 1.65V to 5V

5V: Run current 1mA + (0.6mA x MHz) (typical)
3V: Run current 1mA + (0.4mA x MHz) (typical)



Wait Current 1.6mA - CPU not clocked, all peripherals running
Halt Mode current 10µA typical, 25µA max

Average 1.8 cycles/instruction
No era 20 MIPS a 24 MHz???

Performance: @24MHz – 6 DMIPS – 5 CoreMark = 0.21 Coremark/MHz

ST marketing materials put STM8 Dhrystone performance at 0.29 DMIPS / Mhz.
Optimizaciones:
https://community.st.com/s/question/0D50X00009Xki9LSAR/stm8-dhrystone-performance-new-record-at-0355-dmips-mhz-using-sdcc

INFORMACION OFICIAL

Tienen que poder leer esto, no hay necesidad de ver videos en YouTube o leer tutoriales de dudoso origen.

STM8 Reference manual

STM8S Series and STM8AF Series 8-bit microcontrollers

https://www.st.com/resource/en/reference_manual/cd00190271-stm8s-series-and-stm8af-series-8bit-microcontrollers-stmicroelectronics.pdf

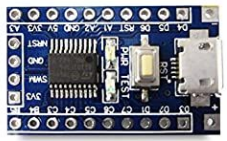
STM8S103F3 Datasheet

8-bit MCU with 8K Flash, 1K RAM, 16 MHz, 640 bytes EEPROM, 10-bit ADC, 3 timers, UART, SPI, I²C

https://www.st.com/resource/en/datasheet/stm8s103f3.pdf

Digikey x1000 $0.76 - x1 $1.45

Mouser 1: $1.30 10: $1.12 100: $0.859 500: $0.768 1,000: Ver



Más barato que Atmega328 pero menos memoria