CIRCUITOS DIGITALES Y MICROCONTROLADORES 2022

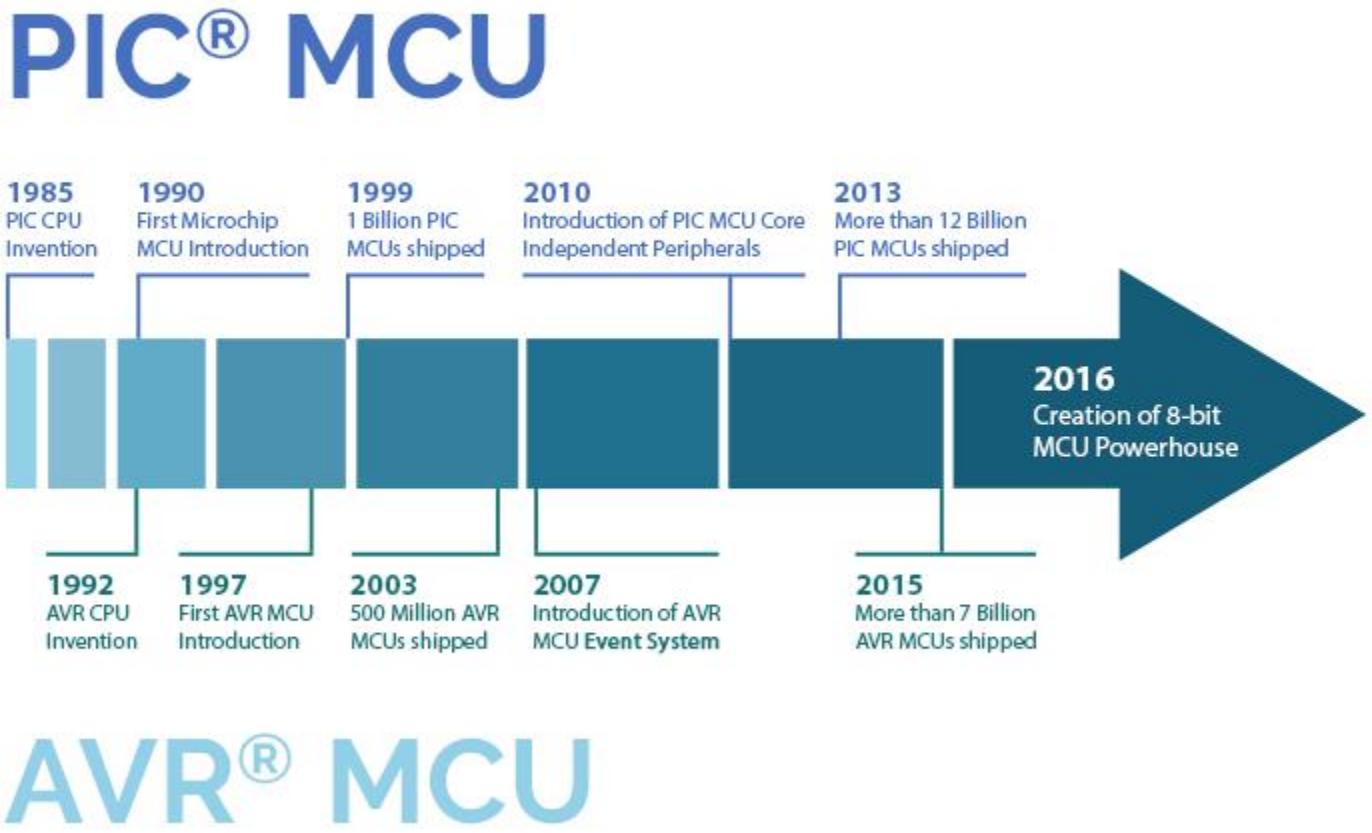
Facultad de Ingeniería UNLP

Introducción a la familia AVR

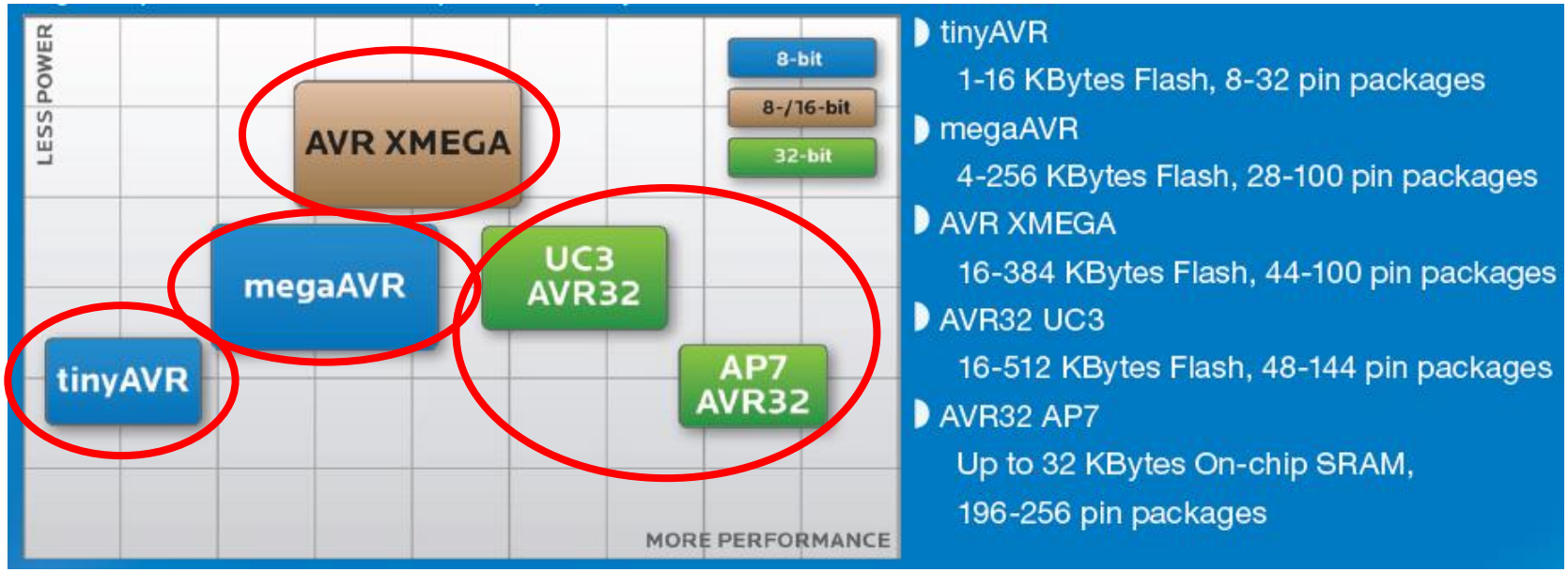
Ing. José Juárez

Historia de 8-bits según Microchip

- http://www.microchip.com/design-centers/8-bit



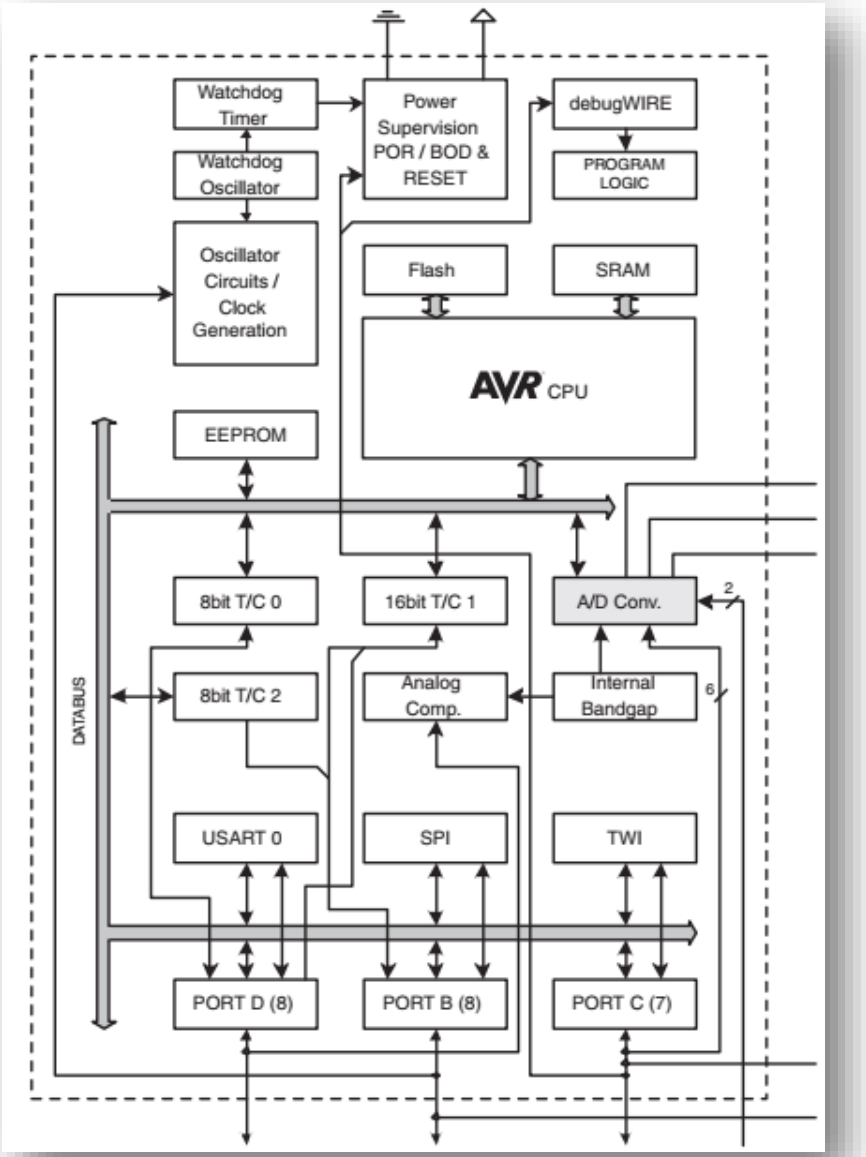
Familia Atmel AVR



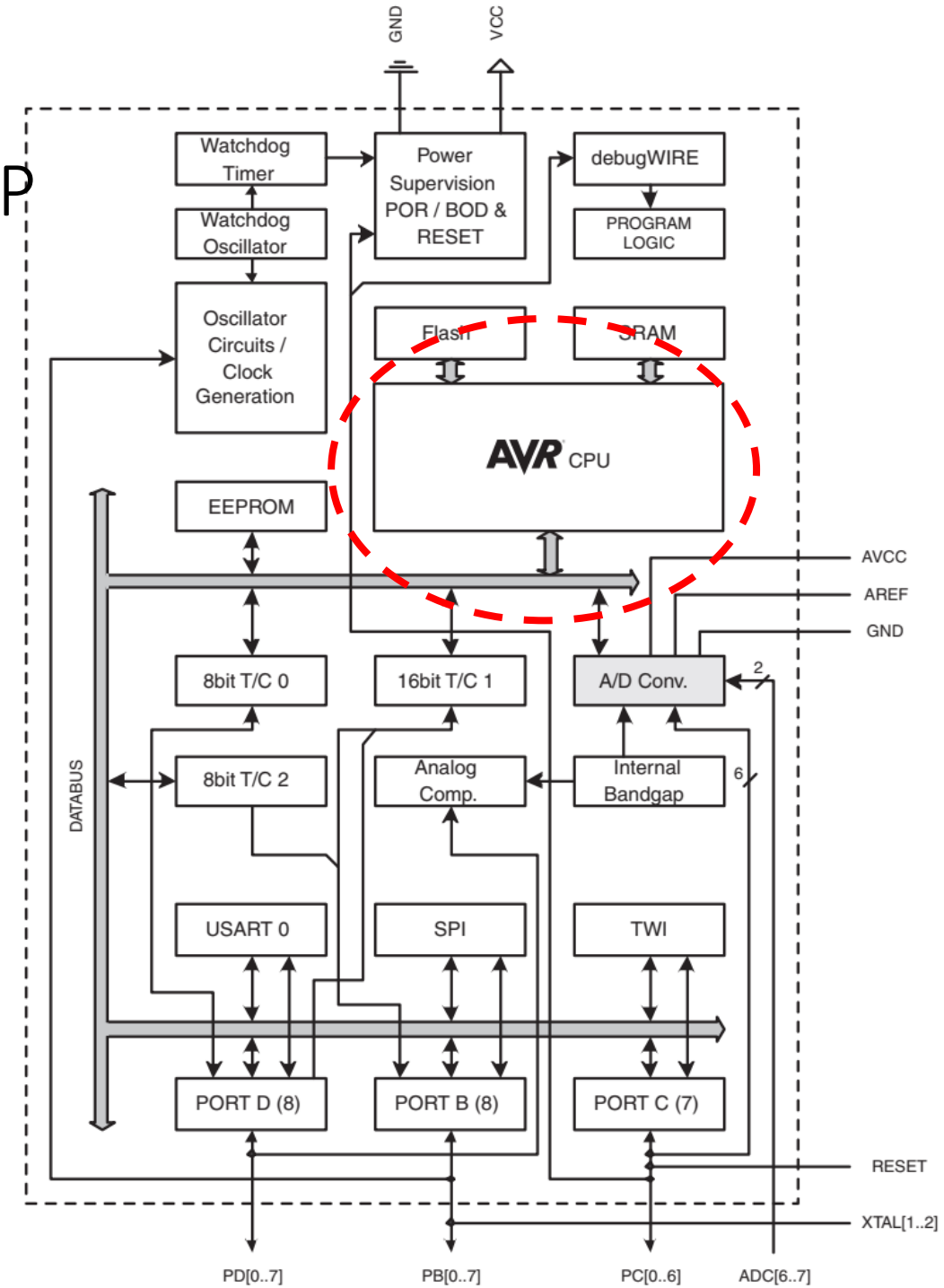
Folleto Atmel AVR: doc4064

http://ww1.microchip.com/downloads/en/DeviceDoc/doc4064.pdf

ARDUINO y el Atmega 328P [(Data sheet)]



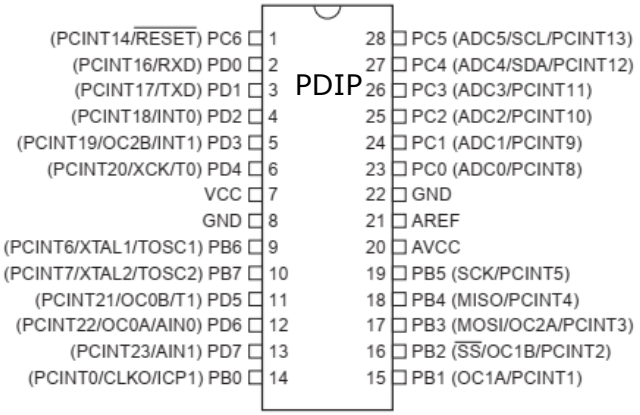
Atmega328P

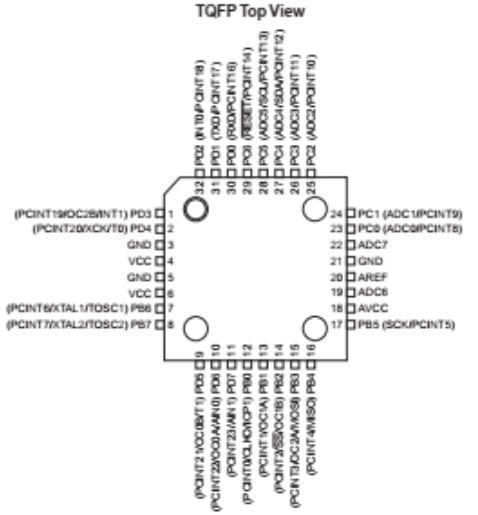


[Ver Familia AVR MCU]

Características Atmega328P

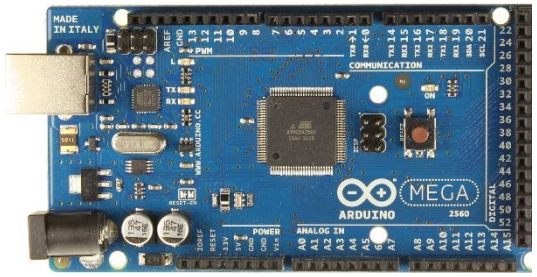
- CPU: AVR 8-bit
- 32 kB Memoria FLASH (ISP, 10.000 ciclos)
- 1 kB Memoria EEPROM (ISP, 100.000 ciclos)
- 2 kB Memoria SRAM
- 23 Entradas/salidas de propósito gral.
- 3 Puertos: PORTB, C y D
- Interfaz para depuración (debugWIRE)
- Reloj del sistema: hasta 20MHz
- Alimentación 1.8V a 5V
- 6 modos de bajo consumo (sleep modes)
- 2 Timers 8 bits y 1 Timer 16 bits
- 6 Generadores PWM
- 8 canales, 10bit ADC
- Interfaces Series: USART, SPI, I2C
- Watch Dog “perro guardián”
- Comparador analógico

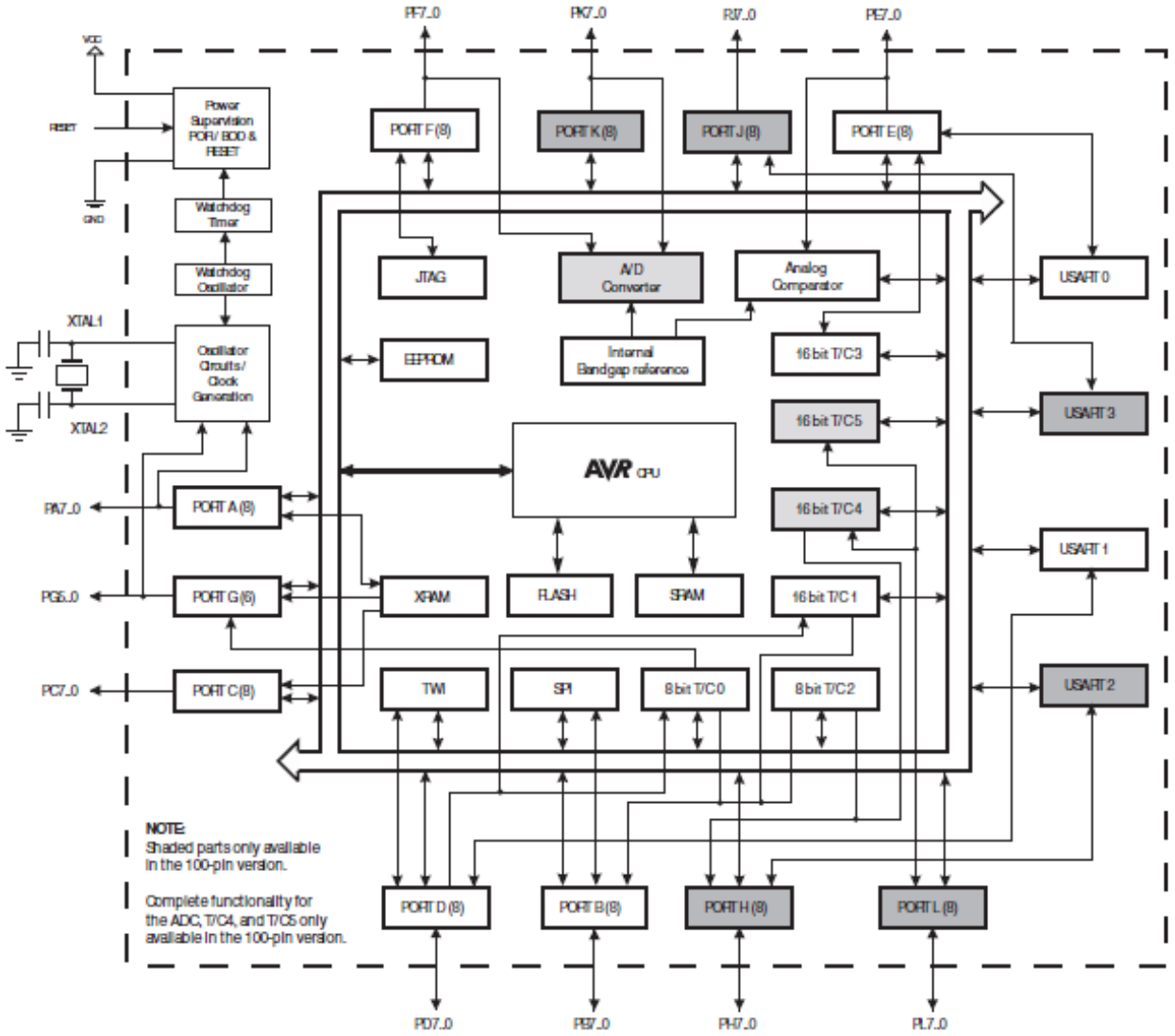




ARDUINO y el atmega 2560

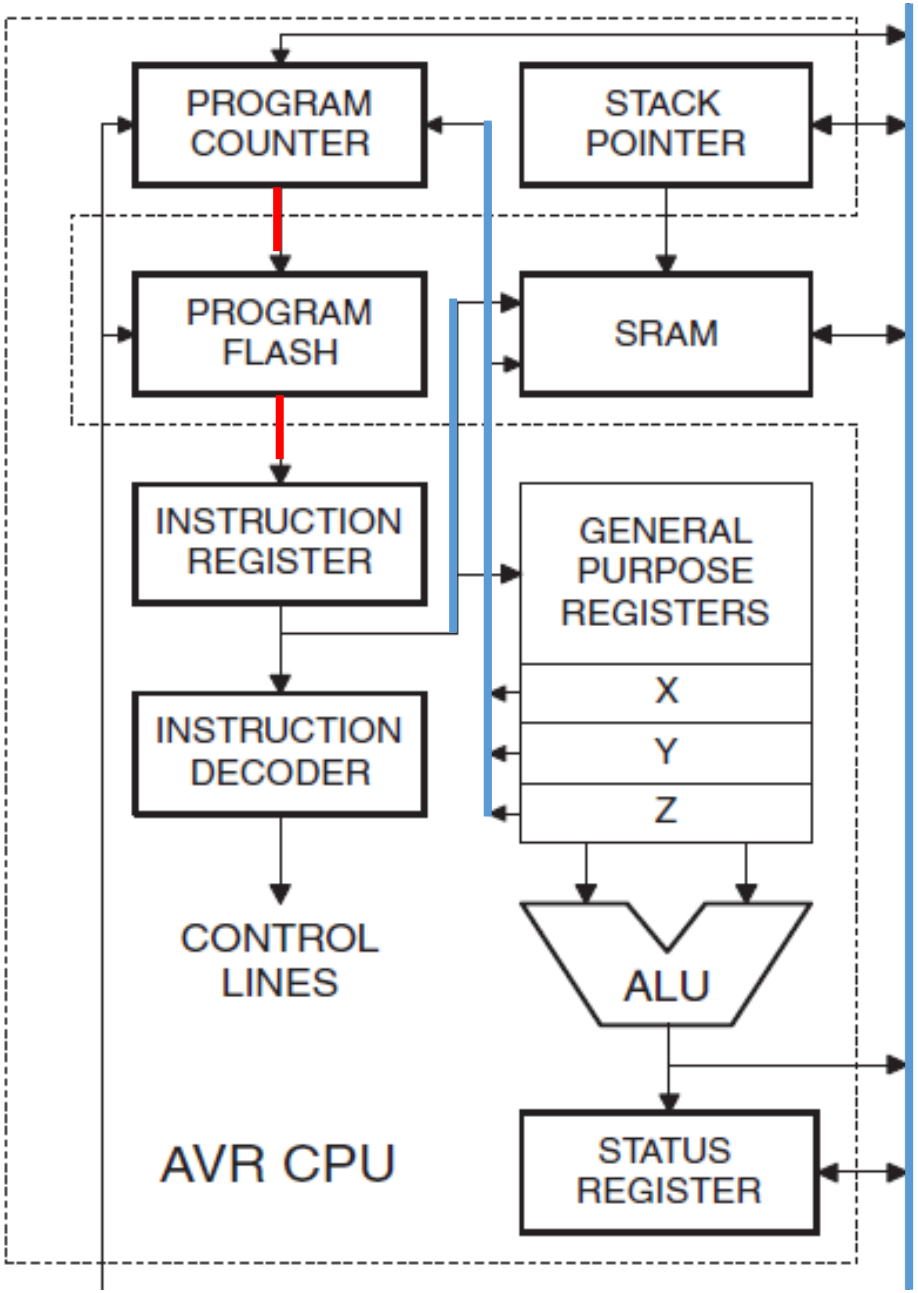
- AVR 8 bits
- 256kB Flash
- 4kB EEPROM
- 8kB RAM
- 16MHz
- 100 pines
- Puertos A,B,C,D
  E,F,G,H,J,K,L
- JTAG interface





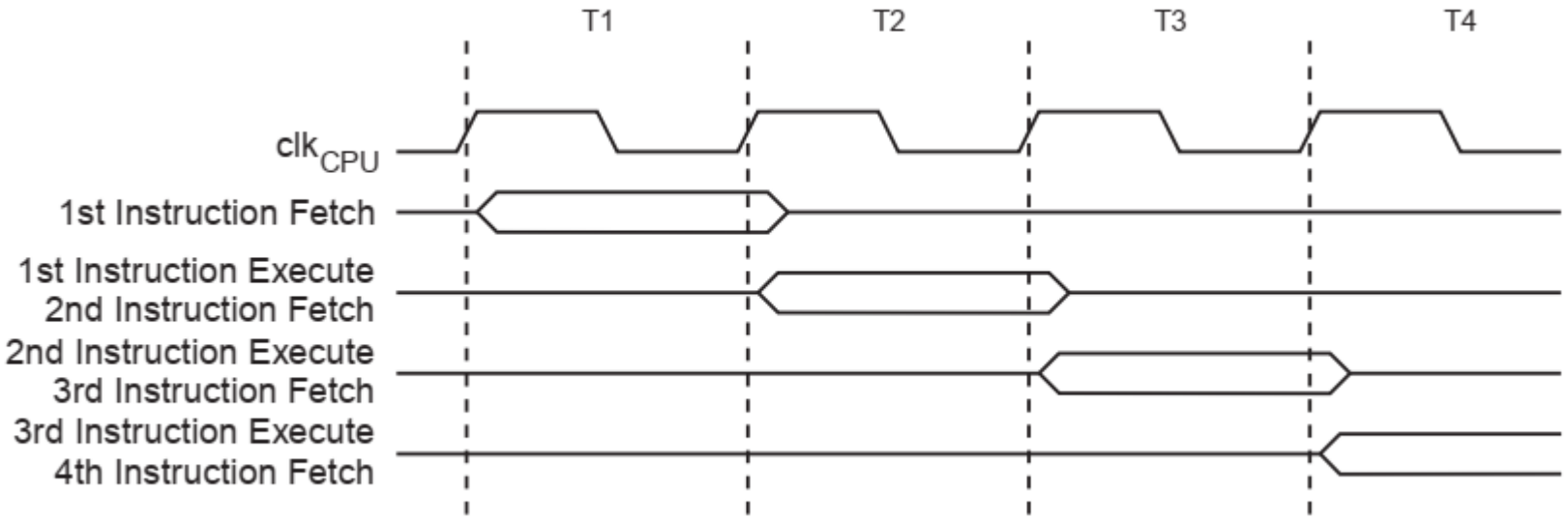
AVR- CPU

- RISC: 131 instrucciones ejecutadas en su mayoría en 1 ciclo de clock
- Hardvard: Memoria de programa y memoria de datos con buses independientes
- Reloj hasta 20MHz (20 MIPS ideal)
- Arquitecturas basadas en registros:
  - 32 registros CPU de 8 bits
  - Operaciones sobre registros CPU minimizando el acceso a memoria
- En “Instruction Register” permite seleccionar los operandos a usar en la ejecución.
- A su vez el OPcode es decodificado en el “decoder” para generar las señales de control para ejecutar la instrucción.



Ejecución de Instrucción

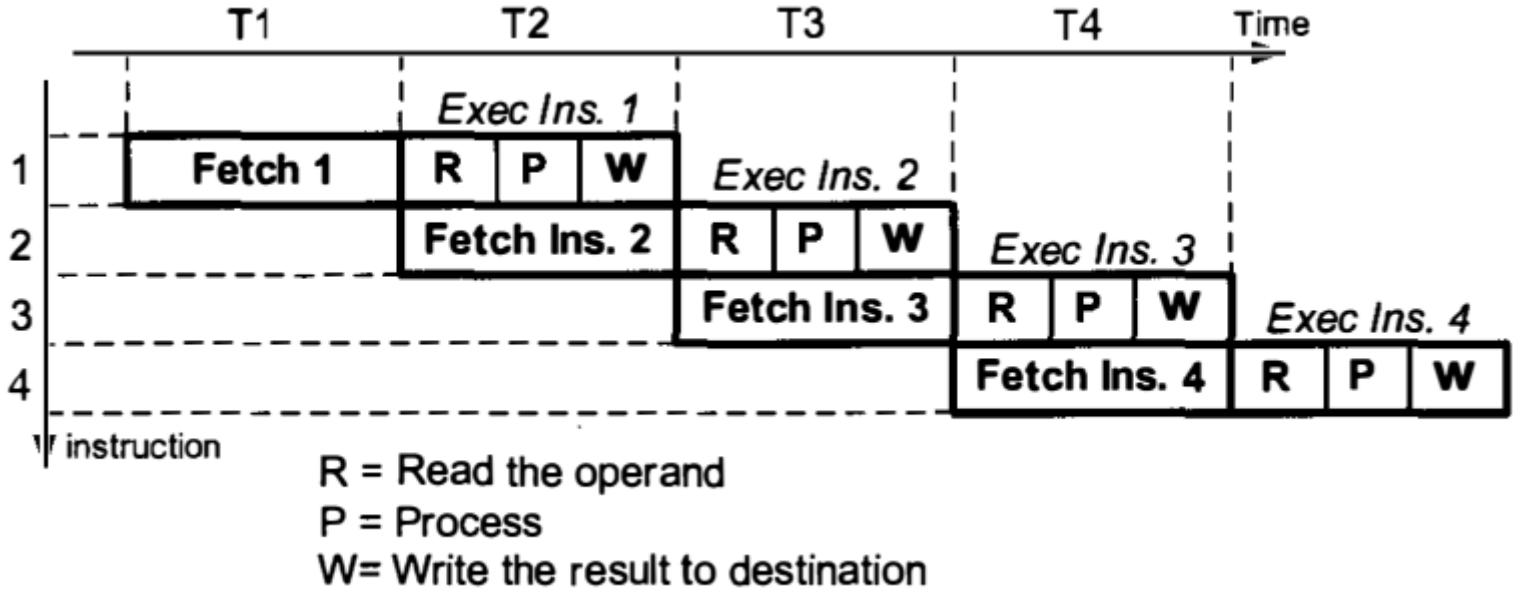
- AVR: Pipeline de 2 etapas
- Ejecución y búsqueda de instrucción en paralelo



CH6 data sheet
Libro de Espinosa CH 1 y 2
Libro de Mazidi CH 3

ALU

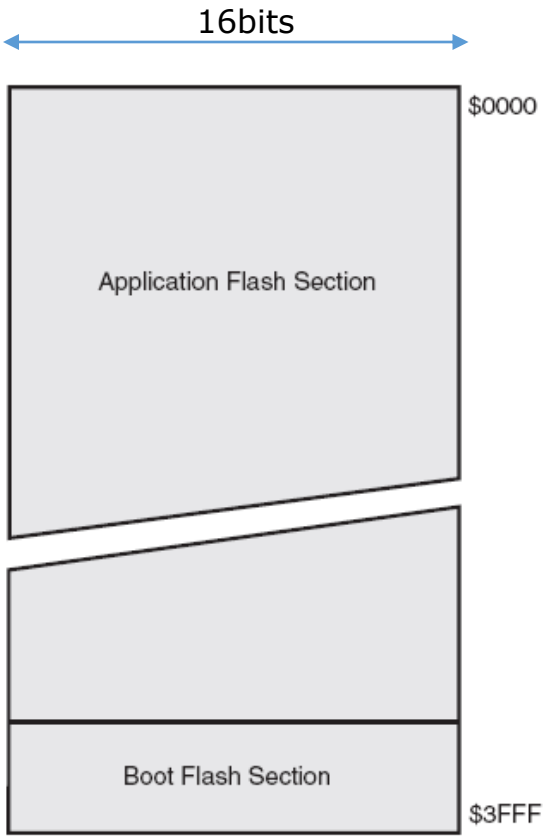
- Operaciones aritméticas y lógicas en 1 ciclo
- Multiplicación 8x8 en 2 ciclos
- Permite operaciones entre registros y entre registros y constantes
- El estado del resultado de una operación (Flags) se actualiza en el "status Register"



CH6 data sheet
Libro de Espinosa CH 1 y 2
Libro de Mazidi CH 3

Memoria de Programa –Atmega328P

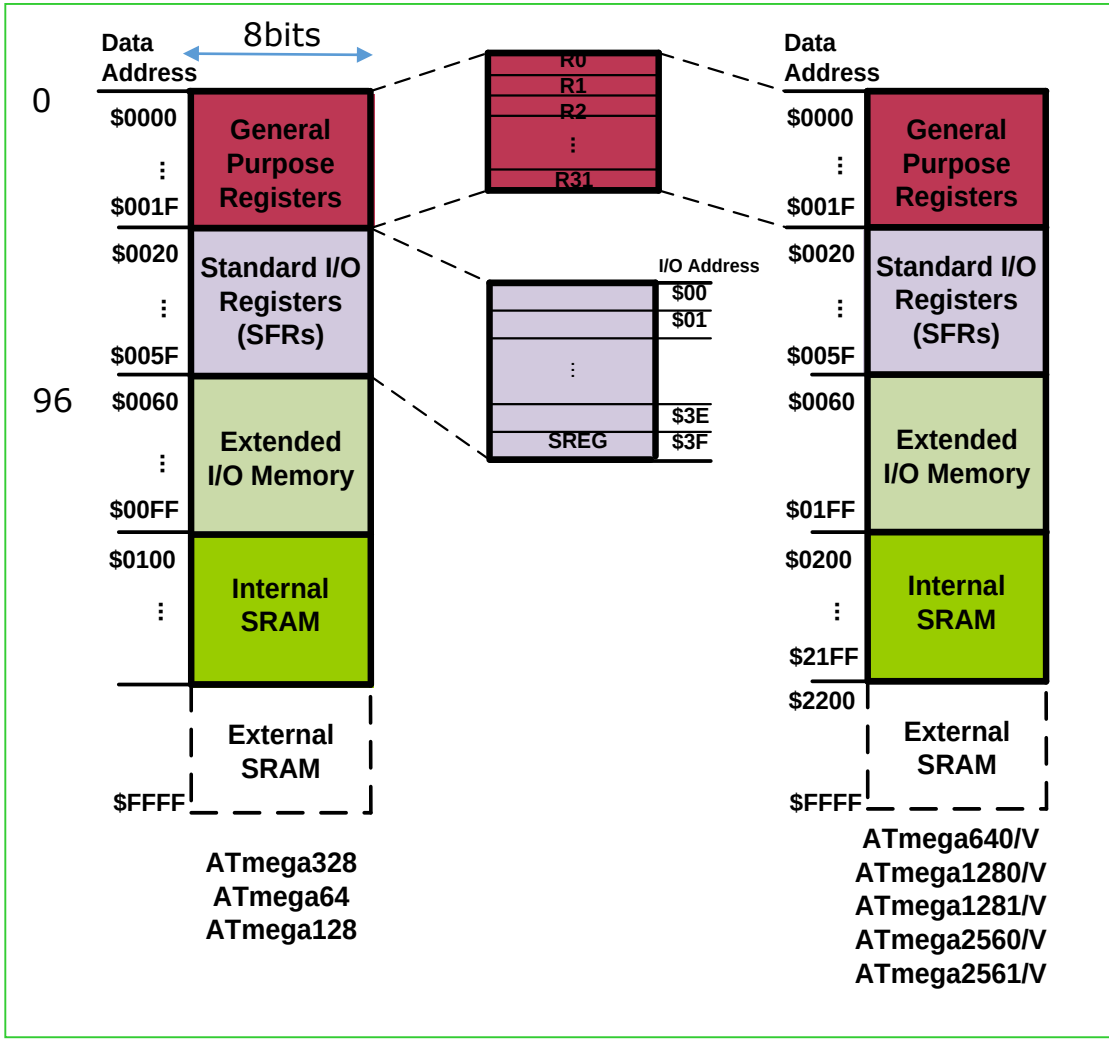
- 32kB FLASH
- Organización: 16K x 16bits (256 pag. de 64 words)
- Las instrucciones AVR son codificadas en 16 y 32 bits
- También se almacenan constantes o tablas
- Contador de programa (PC) de 14bits
- 2 instrucciones de acceso LPM (Load from Program Memory) SPM (Storage to Program Memory)
- Esta dividida en dos secciones: Boot (o arranque) y Aplicación (o programa de usuario)
- Programable externamente a través de Interfaz SPI o debugWire (requiere programador)
- Programable x Bootloader code (Read-While-Write Operation) (requiere interfaz serie)
- Trabas de seguridad contra escritura (Lock bits)



CH7 data sheet

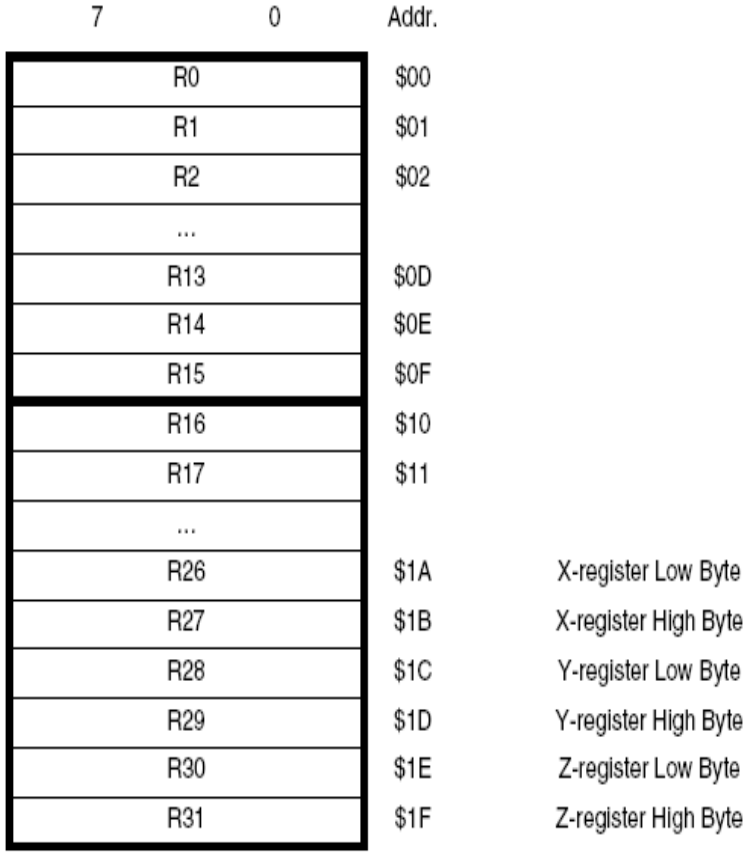
Memoria de Datos – Atmega328p

- Las primeras 96 direcciones corresponden a los registros CPU (32) y a los registros de los periféricos I/O (64)
- Este espacio de memoria puede accederse con las instrucciones IN,OUT en 1 ciclo de reloj
- A partir de las dirección 0x60 solo puede accederse a partir de las instrucciones LOAD/STORE.
- Se puede acceder de 5 formas o modos de direccionamiento
- Los registros I/O pueden ser accedidos como RAM, y también pueden accederse de a bits individuales.
- El acceso de lectura/escritura SRAM es de 2 ciclos de clock.



Registros CPU – 8 bits

- Optimizados para el conjunto de instrucciones RISC AVR permiten operaciones:
  - 1 operando de 8 bits -> 8
  - 2 operandos de 8 bits -> 8
  - 2 operandos de 8 bits -> 16
  - 1 operando de 16 bits -> 16
- La mayoría de las instrucciones que operan sobre los registros son de 1 ciclo de reloj
- "mapeados en memoria" en las 32 primeras direcciones para flexibilizar el acceso al espacio de datos
- Los pares de registros 26-27, 28-29 y 30-31 se denominan X, Y, Z para permitir el direccionamiento indexado
- X, Y, Z son punteros de 16 bits que permiten acceder a la memoria de datos de manera Indirecta, incluso pueden apuntar a los registros.



Registros I/O PORTS – 8bits

CH13 Data sheet

- Cada Puerto I/O tiene 3 Registros asociados
- Por ejemplo el puerto B:

PORTB – Port B Data Register

Bit	7	6	5	4	3	2	1	0	
	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
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	

DDRB – Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
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	

PINB – Port B Input Pins Address

Bit	7	6	5	4	3	2	1	0	
	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Registros I/O PORTS– 8bits

- Control de dirección de un puerto I/O

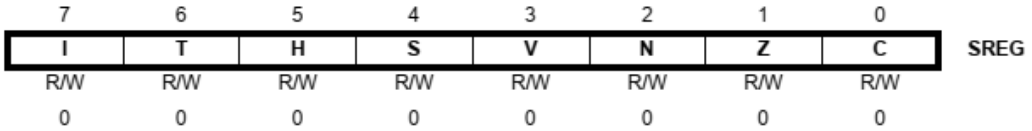
DDxn	PORTxn	PUD (in MCUCR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

Veremos en otra clase la estructura interna de los puertos I/O

Otros Registros I/O – 8bits

- Control de CPU en el espacio I/O:

  Stack pointer (SP)



- Ejemplos de Registos de Control de periféricos:

  Timer control (TCCRx)

  Timer counter (TCNTx)

  Timer output (OCRx)

  E2PROM control (EEARx)

  Serial Interface Control (BAUD, UCSR)

  ADC Control (ADLAR, ADCR…)

Listado completo de registros: pág. 423 ATMEGA328 Data Sheet

Conjunto de Instrucciones

- El conjunto de Instrucciones que un MCU puede ejecutar esta definido por el fabricante y es un único para cada familia o modelo de MCU
  - Documento: AVR Instrucción Set

- Programación en lenguaje assembler o ensamblador:
  - Desarrollado por el fabricante
  - Bajo nivel (instrucciones en lenguaje máquina)
  - No portable a otros fabricantes
  - Máx rendimiento y aprovechamiento de recursos (velocidad de ejecución y consumo de memoria)

Conjunto de Instrucciones

- Tipos de Instrucciones:
  - Aritméticas y lógicas
  - Bifurcaciones
  - Transferencia de datos
  - Operaciones de bits y bit test
  - Especiales

Table 2-2: ALU Instructions Using Two GPRs

Instruction		
ADD	Rd, Rr	ADD Rd and Rr
ADC	Rd, Rr	ADD Rd and Rr with Carry
AND	Rd, Rr	AND Rd with Rr
EOR	Rd, Rr	Exclusive OR Rd with Rr
OR	Rd, Rr	OR Rd with Rr
SBC	Rd, Rr	Subtract Rr from Rd with carry
SUB	Rd, Rr	Subtract Rr from Rd without carry

Instruction		
CLR	Rd	Clear Register Rd
INC	Rd	Increment Rd
DEC	Rd	Decrement Rd
COM	Rd	One's Complement Rd
NEG	Rd	Negative (two's complement) Rd
ROL	Rd	Rotate left Rd through carry
ROR	Rd	Rotate right Rd through carry
LSL	Rd	Logical Shift Left Rd
LSR	Rd	Logical Shift Right Rd
ASR	Rd	Arithmetic Shift Right Rd
SWAP	Rd	Swap nibbles in Rd

Set de Instrucciones

- LDI “carga inmediata”, solo con R16 a R31

```
LDI R20,0x25              ;load R20 with 0x25 (R20 = 0x25)
```

- ADD

```
LDI R16,0x25      ;load 0x25 into R16
LDI R17,0x34      ;load 0x34 into R17
ADD R16,R17       ;add value R17 to R16 (R16 = R16 + R17)
```

- LDS “carga desde RAM”

```
LDS   R0, 0x300   ;R0 = the contents of location 0x300
LDS   R1, 0x302   ;R1 = the contents of location 0x302
ADD   R1, R0      ;add R0 to R1
```

- STS “almacena en RAM”

```
LDS   R30, 0x220  ;load R30 with the contents of location 0x220
LDS   R31, 0x221  ;load R31 with the contents of location 0x221
ADD   R31, R30    ;add R30 to R31
STS   0x221, R31  ;store R31 to data space location 0x221
```

Set de Instrucciones

- IN

```
IN    R1,PIND     ;load R1 with PIND
IN    R2,PINB     ;load R2 with PINB
ADD   R1, R2      ;R1 = R1 + R2
STS   0x300, R1   ;store R1 to data space location $300
```

- OUT

```
IN    R0, PIND    ;load R20 with the contents of I/O reg PIND
OUT   PORTA, R0   ;out R20 to PORTA
```

- MOV

```
MOV   R10,R20     ;R10 = R20
```

- INC

```
LDS   R20, 0x430  ;R20 = contents of location 0x430
INC   R20         ;R20 = R20 + 1
STS   0x430, R20  ;store R20 to location 0x430
```

Set de Instrucciones

- Bifurcaciones y lazos
  - BRNE: Branch if not equal (Z≠1)

```
        LDI   R16, 10      ;R16 = 10 (decimal) for counter
        LDI   R20, 0       ;R20 = 0
        LDI   R21, 3       ;R21 = 3
AGAIN:ADD   R20, R21       ;add 03 to R20 (R20 = sum)
        DEC   R16          ;decrement R16 (counter)
        BRNE  AGAIN        ;repeat until COUNT = 0
        OUT   PORTB,R20    ;send sum to PORTB
```

  - Ejemplo de Lazos de retardo

```
        LDI   R16, 0x55    ;R16 = 0x55
        OUT   PORTB, R16   ;PORTB = 0x55
        LDI   R20, 10      ;load 10 into R20 (outer loop count)
LOP_1:LDI   R21, 70        ;load 70 into R21 (inner loop count)
LOP_2:COM   R16            ;complement R16
        OUT   PORTB, R16   ;load PORTB SFR with the complemented value
        DEC   R21          ;dec R21 (inner loop)
        BRNE  LOP_2        ;repeat it 70 times
        DEC   R20          ;dec R20 (outer loop)
        BRNE  LOP_1        ;repeat it 10 times
```

Set de Instrucciones

- BREQ: branch if equal (Z=1)

```
OVER: IN   R20, PINB    ;read PINB and put it in R20
      TST  R20          ;set the flags according to R20
      BREQ OVER         ;jump if R20 is zero
```

- Otros saltos condicionales

Table 3-1: AVR Conditional Branch (Jump) Instructions

Instruction	Action
BRLO	Branch if C = 1
BRSH	Branch if C = 0
BREQ	Branch if Z = 1
BRNE	Branch if Z = 0
BRMI	Branch if N = 1
BRPL	Branch if N = 0
BRVS	Branch if V = 1
BRVC	Branch if V = 0

- Saltos Incondicionales: JMP (hasta 64k), RJMP (hasta 2k), IJMP (usando Z)

Formato de Instrucción AVR

- Ejemplo: CLEAR REGISTER

Operation:

(i) Rd ← Rd ⊕ Rd

	Syntax:	Operands:	Program Counter:
(i)	CLR Rd	0 ≤ d ≤ 31	PC ← PC + 1

16-bit Opcode: (see EOR Rd,Rd)

0010	01dd	dddd	dddd

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	–	0	0	0	1	–

Example:

```
        clr   r18       ; clear r18
loop:   inc   r18       ; increase r18
        ...
        cpi   r18,$50   ; Compare r18 to $50
        brne  loop
```

Words: 1 (2 bytes)
Cycles: 1

- Código de operación de 6 bits
- Referencia al operando de 10 bits (con 5 alcanzaría)

Formato de Instrucción AVR

- Ejemplo: ADD with Carry

Operation:

(i) Rd ← Rd + Rr + C

	Syntax:	Operands:	Program Counter:
(i)	ADC Rd,Rr	0 ≤ d ≤ 31, 0 ≤ r ≤ 31	PC ← PC + 1

16-bit Opcode:

0001	11rd	dddd	rrrr

- Código de operación de 6 bits
- Referencias a los operandos 5 bits c/u

Example:

```
                    ; Add R1:R0 to R3:R2
add   r2,r0         ; Add low byte
adc   r3,r1         ; Add with carry high byte
```

Words: 1 (2 bytes)
Cycles: 1

Formato de Instrucción AVR

- Ejemplo: ADD with Carry

Status Register (SREG) Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	⇔	⇔	⇔	⇔	⇔	⇔

H: $Rd3 \bullet Rr3 + Rr3 \bullet \overline{R3} + \overline{R3} \bullet Rd3$
Set if there was a carry from bit 3; cleared otherwise

S: $N \oplus V$, For signed tests.

V: $Rd7 \bullet Rr7 \bullet \overline{R7} + \overline{Rd7} \bullet \overline{Rr7} \bullet R7$
Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R7
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is $00; cleared otherwise.

C: $Rd7 \bullet Rr7 + Rr7 \bullet \overline{R7} + \overline{R7} \bullet Rd7$
Set if there was carry from the MSB of the result; cleared otherwise.

R (Result) equals Rd after the operation.

Formato de Instrucción AVR

- Ejemplo: Instrucción IN (transferencia de registro I/O a registros CPU)

Operation:

(i) Rd ← I/O(A)

	Syntax:	Operands:	Program Counter:
(i)	IN Rd,A	0 ≤ d ≤ 31, 0 ≤ A ≤ 63	PC ← PC + 1

16-bit Opcode:

1011	0AAd	dddd	AAAA

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Example:

```
        in    r25,$16    ; Read Port B
        cpi   r25,4      ; Compare read value to constant
        breq  exit       ; Branch if r25=4
        ...
exit:   nop              ; Branch destination (do nothing)
```

Words: 1 (2 bytes)
Cycles: 1

Formato de Instrucción AVR

- Ejemplo: Instrucción LDS (carga directa desde RAM a registros)

Operation:

(i) Rd ← (k)

	Syntax:	Operands:	Program Counter:
(i)	LDS Rd,k	0 ≤ d ≤ 31, 0 ≤ k ≤ 65535	PC ← PC + 2

32-bit Opcode:

1001	000d	dddd	0000
kkkk	kkkk	kkkk	kkkk

Example:

```
lds   r2,$FF00     ; Load r2 with the contents of data space location $FF00
add   r2,r1        ; add r1 to r2
sts   $FF00,r2     ; Write back
```

Words: 2 (4 bytes)
Cycles: 2

Formato de Instrucción AVR

- Ejemplo: Instrucción JMP (jump)

	Syntax:	Operands:	Program Counter:	Stack:
(i)	JMP k	0 ≤ k < 4M	PC ← k	Unchanged

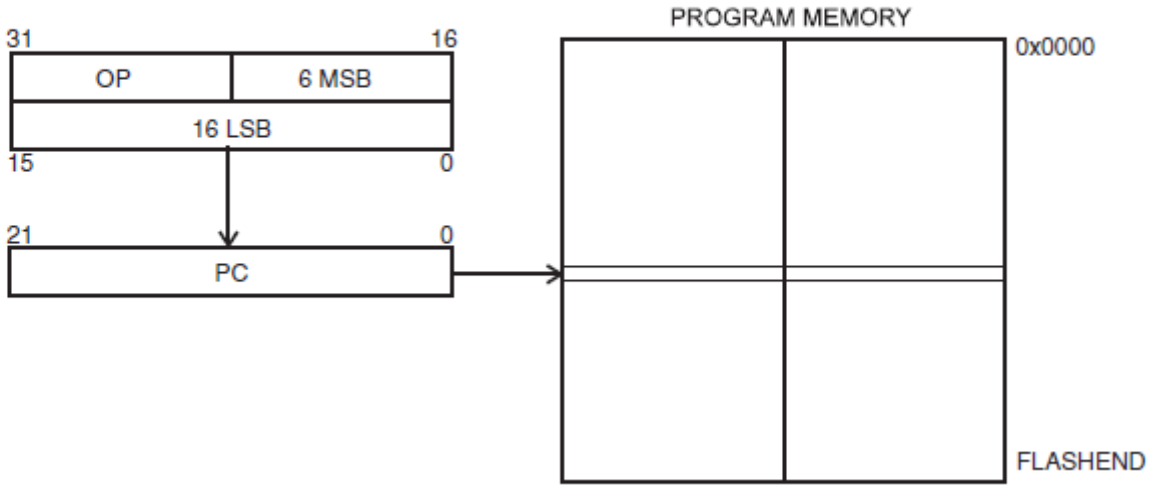
32-bit Opcode:

1001	010k	kkkk	110k
kkkk	kkkk	kkkk	kkkk

Words: 2 (4 bytes)
Cycles: 3

Direct Program Addressing, JMP and CALL

Figure 11. Direct Program Memory Addressing



Modos de direccionamiento

- Veremos ahora direccionamiento a memoria de datos (SRAM, Registros CPU y Registros I/O)

- Posibilidades:
  - El dato está contenido en la instrucción
  - El dato está en un registro Rx con 0 <= x <= 31
    - El número de registro (x) está en la instrucción.
  - El dato está en memoria SRAM o espacio I/O
    - La dirección del dato está en instrucción
    - La dirección está en un registro Rx con 0 <= x <= 31
      - El número de registro (x) está en la instrucción
    - La dirección se calcula con un registro Rx y un desplazamiento (offset)
      - El número de registro (x) está en la instrucción
      - El desplazamiento está en la instrucción

Modos de direccionamiento

- Direccionamiento Inmediato
  - El dato está en la instrucción
  - Ejemplo LDI o carga inmediata
    - LDI Rd,K ; k es un número de 8 bits
      - LDI r15,$FF; cargar el registro r15 con el valor $FF

Operation:

(i) Rd ← K

	Syntax:	Operands:	Program Counter:
(i)	LDI Rd,K	16 ≤ d ≤ 31, 0 ≤ K ≤ 255	PC ← PC + 1

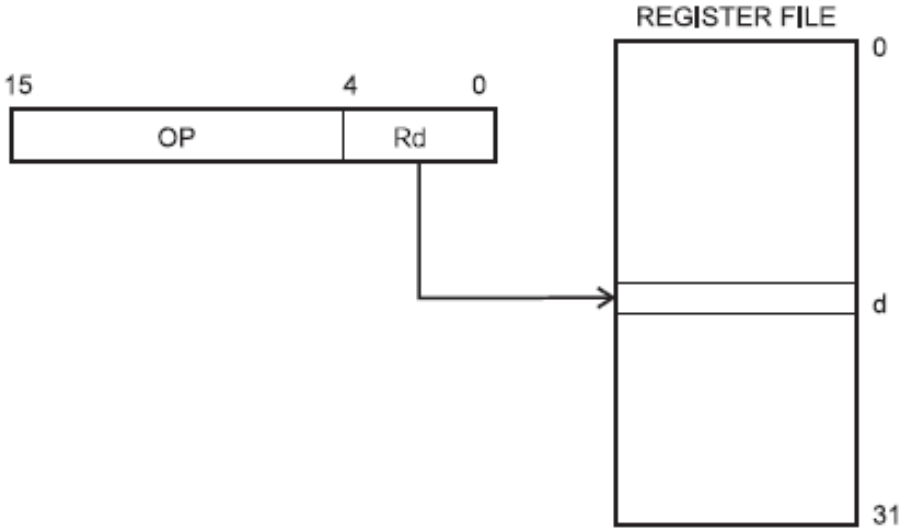
16-bit Opcode:

1110	KKKK	dddd	KKKK

Modos de direccionamiento

- Directo a Registros
  - Un único registro

Figure 1. Direct Single Register Addressing

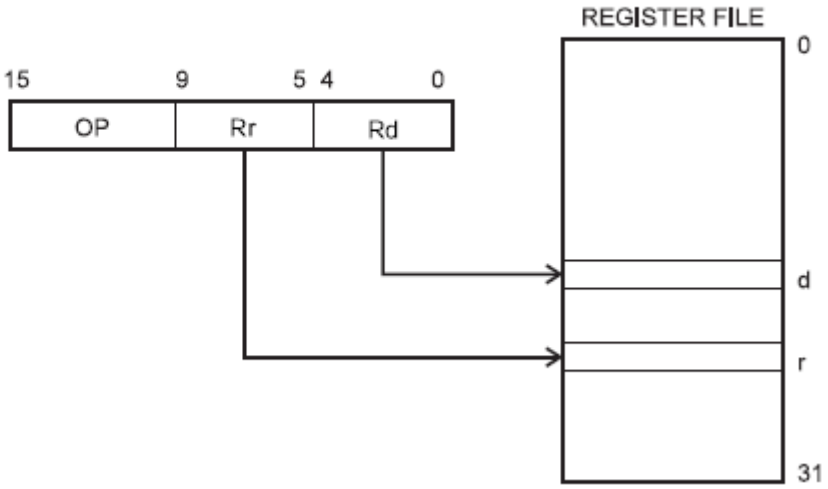


  - Ejemplo: CLR Rd

Modos de direccionamiento

- Directo a Registros
  - Dos Registros

Figure 2. Direct Register Addressing, Two Registers

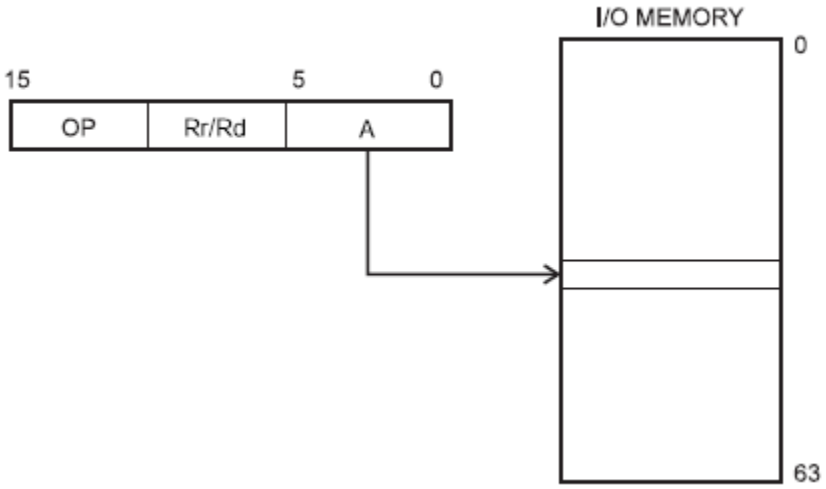


  - Operandos contenidos en registros Rr y Rd. El resultado se almacena en Rd
  - Ej: ADD Rd,Rr

Modos de direccionamiento

- Directo a Registros I/O

Figure 3. I/O Direct Addressing

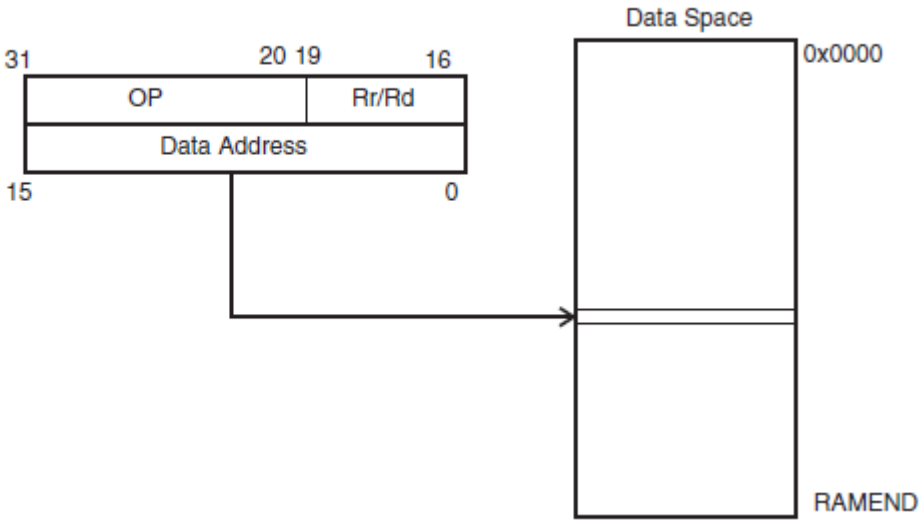


- La dirección del operando esta contenida en los últimos 6 bits
- Ejemplo: IN r25,PORTB

Modos de direccionamiento

- Directo a memoria de datos

Figure 4. Direct Data Addressing

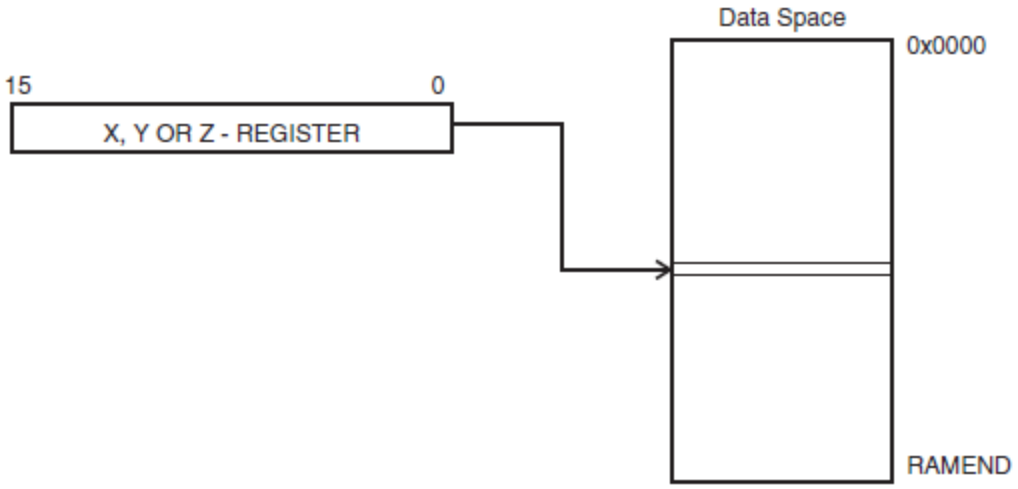


  - Data Address es la dirección (16 bits) donde se encuentra el dato. Ej:
    - LDS Rd,k ; Carga directa desde RAM
    - STS k,Rr ; Almacenamiento directo en RAM

Modos de direccionamiento

- Indirecto a memoria de datos

Figure 6. Data Indirect Addressing

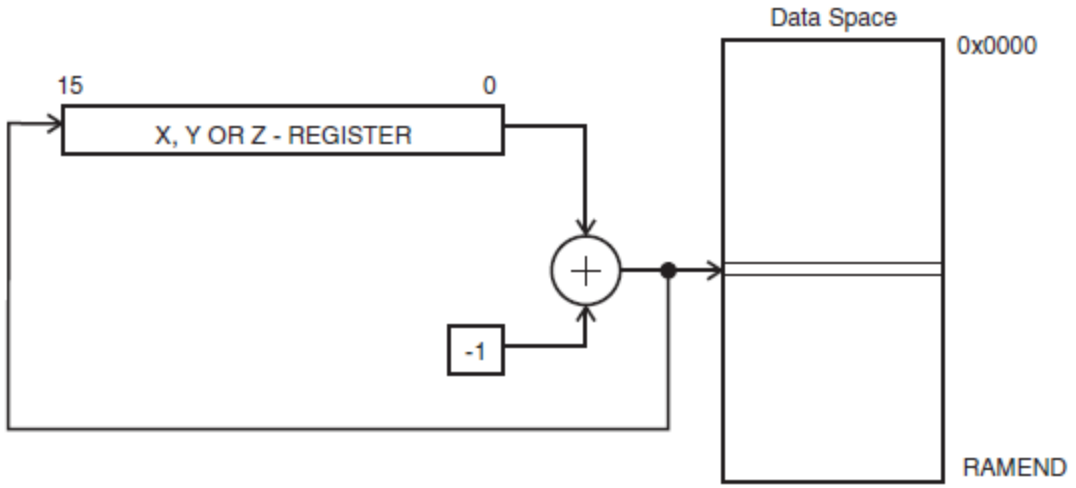


- La dirección del dato está contenida en alguno de los Registros X,Y o Z
  - LD Rd,X ; Rd ← (X)
  - ST Y,Rr ; (Y) ← Rr

Modos de direccionamiento

- Indirecto con Pre-decremento

Figure 7. Data Indirect Addressing with Pre-decrement

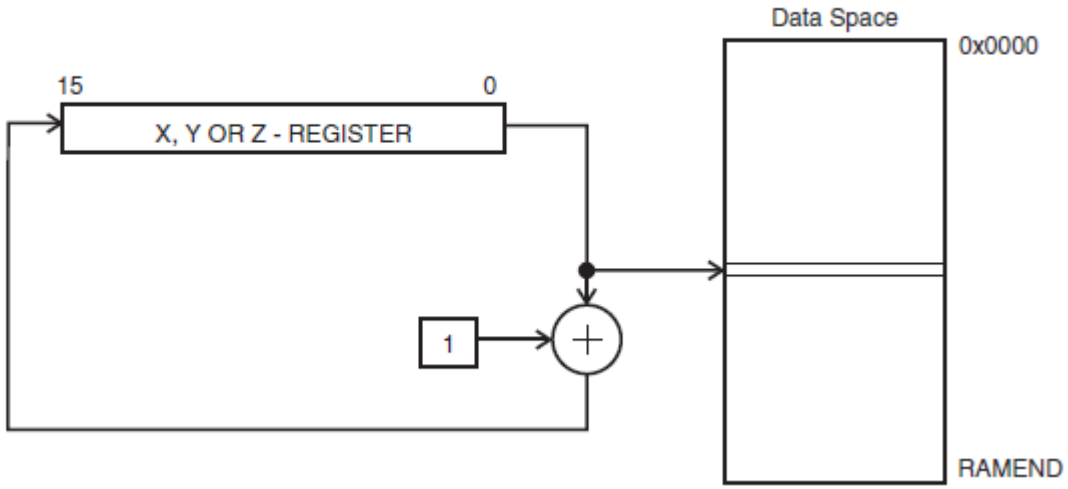


- Ejemplo:
  - LD r16, -X
  - ST –Z, R15

Modos de direccionamiento

- Indirecto con pos-incremento

Figure 8. Data Indirect Addressing with Post-increment

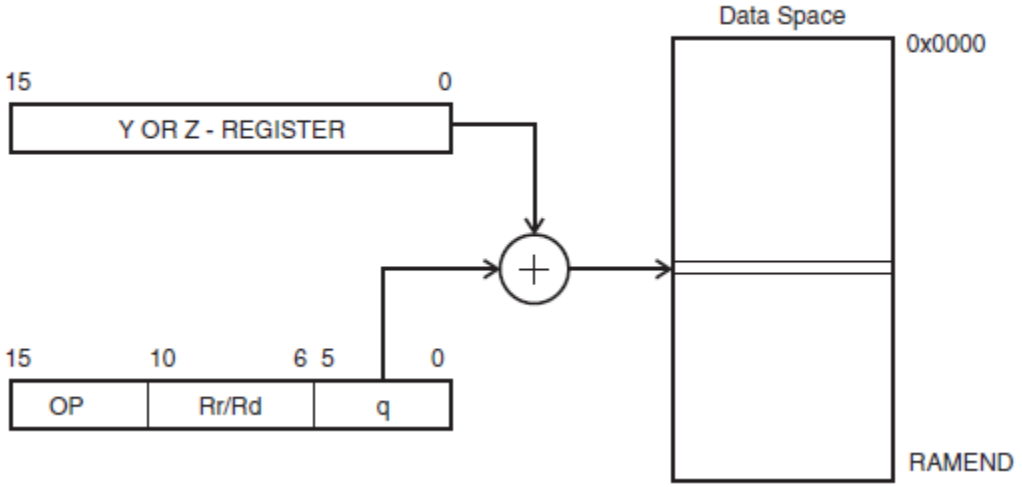


- Ejemplo:
  - LD r16,X+
  - ST Z+,r15

Modos de direccionamiento

- Indirecto con desplazamiento

Figure 5. Data Indirect with Displacement



- La dirección del dato se obtiene sumando un desplazamiento a los punteros Y o Z
- LDD Rd,Z+q; Rd ←(Z+q)
  - LDD r16,Y+$10
- STD Z+q,Rr ; (Z+q) ← Rr
  - STD Z+$20, r15

Resumen: Modos de Direccionamiento

- Direccionamiento a memoria de datos
  - Inmediato
  - Directo a 1 o 2 registros CPU
  - Directo a registros I/O
  - Directo a RAM
  - Indirecto a RAM
  - Indirecto a RAM con desplazamiento
  - Indirecto a RAM con pre decremento
  - Indirecto a RAM con pos incremento
- Direccionamiento a memoria de programa FLASH
  - Directo
  - Indirecto
  - Relativo
  - Direccionamiento a constante con y sin pos incremento

Ejemplo para probar Atmel Studio

```
;
; toggleProject.asm
;
        LDI R16,0xFF
        OUT DDRB,R16

L1:     OUT PORTB,R16
        LDI R20,0
        OUT PORTB,R20
        RJMP L1
```

Tarea:
Escribir cada línea en binario