Principios de Mecatrónica – SDI-11561
Ingeniería en Mecatrónica

Hugo Rodríguez Cortés

Departamento de Ingeniería Eléctrica y Electrónica

Instituto Tecnológico Autónomo de México

Agosto 2023

Microcontrolador AVR- de ATMEL

- La arquitectura básica fue diseñada por Alf-Egil Bogen y Vegard Wollan estudiantes del Norwegian Institute of Technology (NTH). Comprado y desarrollado por Atmel en 1996.

- El CPU de los AVR solamente puede realizar operaciones con 8 bits.[a] Para operar sobre datos de mayor longitud, estos deben separarse.

- Arquitectura Harvard RISC (Reduced instruction set computing) de 8 bits.

- En el chip se tienen memorias ROM, RAM y EEPROM, temporizadores y puertos de entrada y salida I/O

[a] Bit 0, Nibble 0000, Byte $0000\ 0000$, Word $0000\ 0000\ 0000\ 0000$

Microcontrolador AVR- de ATMEL

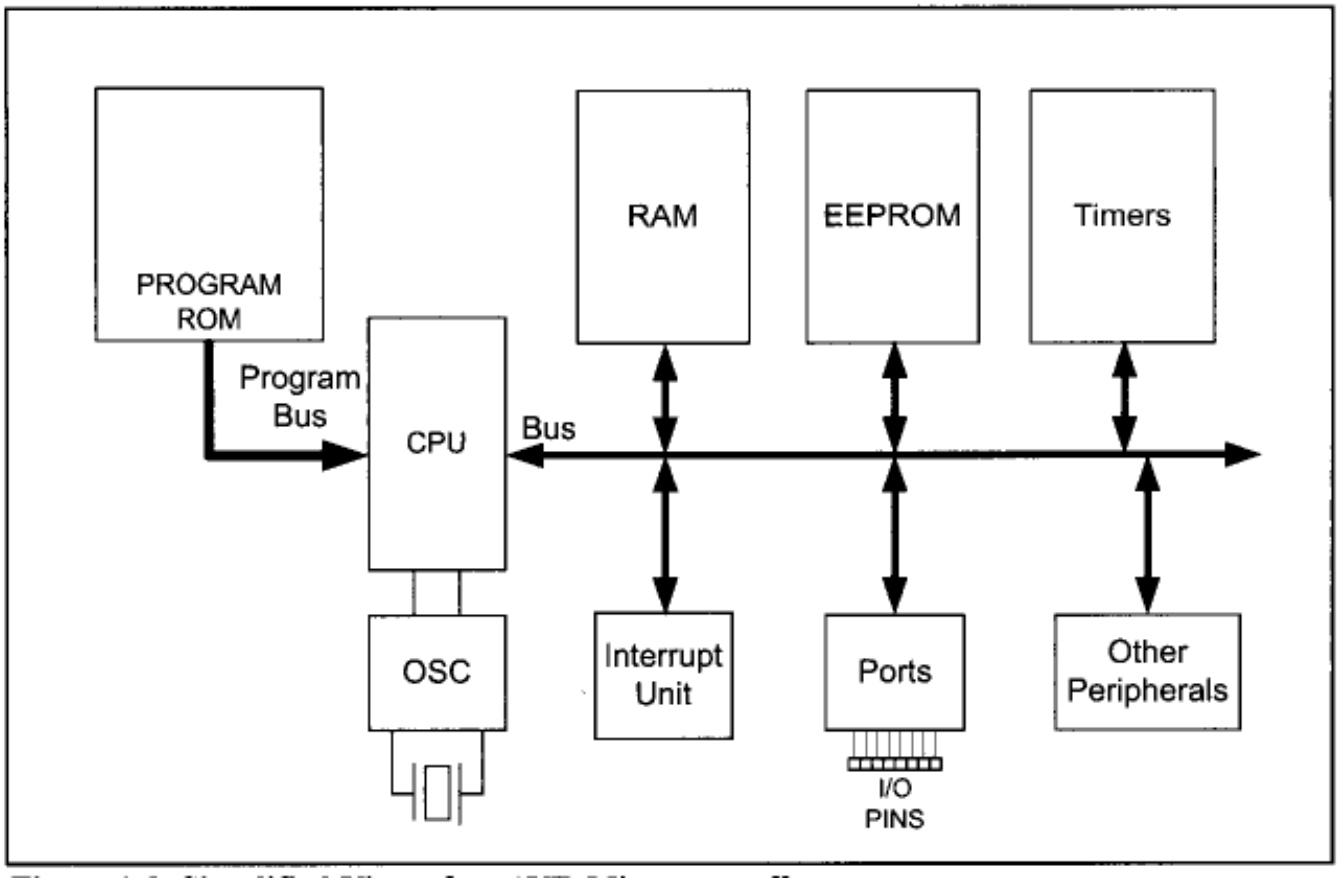


Figure 1: Componentes de un microcontralador

Características adicionales ADC, PWM. Interfaces seriales USART, SPI, I2C, CAN y USB.

Microcontrolador AVR- de ATMEL

1.- **Puertos I/O digitales** conocidos también como la Interfaces de usuario para interactuar con humanos. Teclados, interruptores, buzzer's, audio, luces, pantallas gráficas (digitales) son ejemplos de interfaces de usuario I/O.

2.- **Comunicación serial**: Puertos de comunicación para intercambios de información en serie y / o en paralelo con otros dispositivos o sistemas. Los puertos USB, puertos de impresora, wireless de radio frecuencia (RF) y puertos infrarrojos son algunos ejemplos representativos de dispositivos de comunicación serial.

3.- **Convertidores de datos (analógico a digital (ADC) y / o digital a analógico (DAC))**. Permiten la interacción con sensores y actuadores analógicos. Cuando la señal que sale de una interfaz de sensor es analógica, un ADC la convierte al formato digital comprendido por la CPU.

- De manera similar, cuando la CPU necesita controlar un actuador analógico, se requiere un DAC para cambiar el formato de la señal.
  • Sensores Permiten interactuar con el entorno externo al sistema. Los sensores proporcionan entradas relacionadas con parámetros físicos como temperatura, presión, desplazamiento, aceleración, rotación, etc.
  • Controladores de velocidad del motor. Los controladores de motor de pasos o por PWM son ejemplos de muchos dispositivos que permiten la interacción con los procesos y el medio ambiente.

4. **Memoria RAM**. Memoria principal de la computadora, donde residen programas y datos, sobre la que se pueden efectuar operaciones de lectura y escritura. La memoria RAM es conocida como memoria volátil lo cual quiere decir que los datos no se guardan de manera permanente, es por ello, que cuando deja de existir una fuente de energía en el dispositivo la información se pierde.

- La memoria RAM puede ser reescrita y leída constantemente.

5. **Memoria ROM**. La memoria ROM es el medio de almacenamiento de programas o datos que permiten el buen funcionamiento de los ordenadores o dispositivos electrónicos a través de la lectura de la información sin que pueda ser destruida o reprogramable. La memoria ROM es conocida como memoria no volátil ya que la información contenida en ella no es borrable al apagar el dispositivo electrónico. En esta memoria se encuentran los programas que ponen en marcha el ordenador y realizan los diagnósticos.

6.- **Temporizadores (timers)**. Es un periférico que se utiliza para realizar la medición de frecuencia, implementación de relojes o para el trabajo de conjunto con otros periféricos que requieren una base estable de tiempo entre otras funcionalidades.

GPR en el AVR

- El CPU utiliza registros para almacenar información temporalmente.
- La información puede ser un byte de datos a procesar o una dirección que apunta a un dato que debe ser recolectado.
- Para entender el uso de los GPR se introducen las instrucciones LDI y ADD.
- Los ocho bits de un registro tienen la siguiente distribución

$$\mathrm{D7}\ \mathrm{D6}\ \mathrm{D5}\ \mathrm{D4}\ \mathrm{D3}\ \mathrm{D2}\ \mathrm{D1}\ \mathrm{D0}$$

con $\mathrm{D7}$ y $\mathrm{D0}$ los bits mas y menos significativos, respectivamente.

GPR en el AVR

- El AVR tiene 32 GPR

$$\mathrm{R0} \quad \mathrm{R1} \quad \mathrm{R2} \quad \mathrm{R3} \quad \mathrm{R4} \quad \cdots \quad \mathrm{R31}$$

- Los GPR en el AVR son los acumuladores en otros microcontroladores.

Conjunto y uso de instrucciones

- La sintaxis y operandos utilizados del conjunto de instrucciones que poseen los microcontroladores AVR se describen a continuación. El conjunto de instrucciones pueden ser de los siguientes tipos:
  - Instrucciones Aritméticas y Lógicas
  - Instrucciones de Desvío (salto)
  - Instrucciones de Transferencia de Datos
  - Instrucciones de Bit y prueba de Bit
- Una línea típica en lenguaje ensamblador está dada de la siguiente forma:

$$[\textbf{Mnemónico}] \quad \text{operando}1, \text{operando}2$$

- Los Mnemónicos son las instrucciones en lenguaje ensamblador que puede reconocer el microprocesador.

Conjunto y uso de instrucciones

- Dependiendo de la instrucción, puede haber uno o dos operandos, incluso pueden no existir; su contenido consiste en un número, una variable o una dirección; normalmente el resultado de una operación lógica, aritmética o de carga; y es almacenado en el operando1.

- El flujo del programa por naturaleza es secuencial. Puede ser modificado por instrucciones de saltos condicionales e incondicionales y llamadas a rutinas, que pueden abarcar completamente el espacio de direcciones.

Instrucción LDI

- Copia una dato de 8 bits en un GPR

```
LDI   Rd, K   ; carga al GPR Rd con el dato K inmediatamente.
              ; d debe tomar valores entre 16 y 31.
```

K es un dato de 8 bits puede ser $0 - 255$ en decimal o $00 - FF$ en hexadecimal.

- Ejemplos

```
LDI   R20, 0 × 25   ; carga al GPR R20 con el dato 0 × 25,
                    ; R20 = 0 × 25.
```

```
LDI   R31, 87   ; carga al GPR R31 con el dato 87,
                ; R31 = 87.
```

- Con esta instrucción NO se pueden cargar valores en los registros R0, $\cdots$ R15.

Instrucción ADD

- Calcula la suma de los datos contenidos en dos GPR

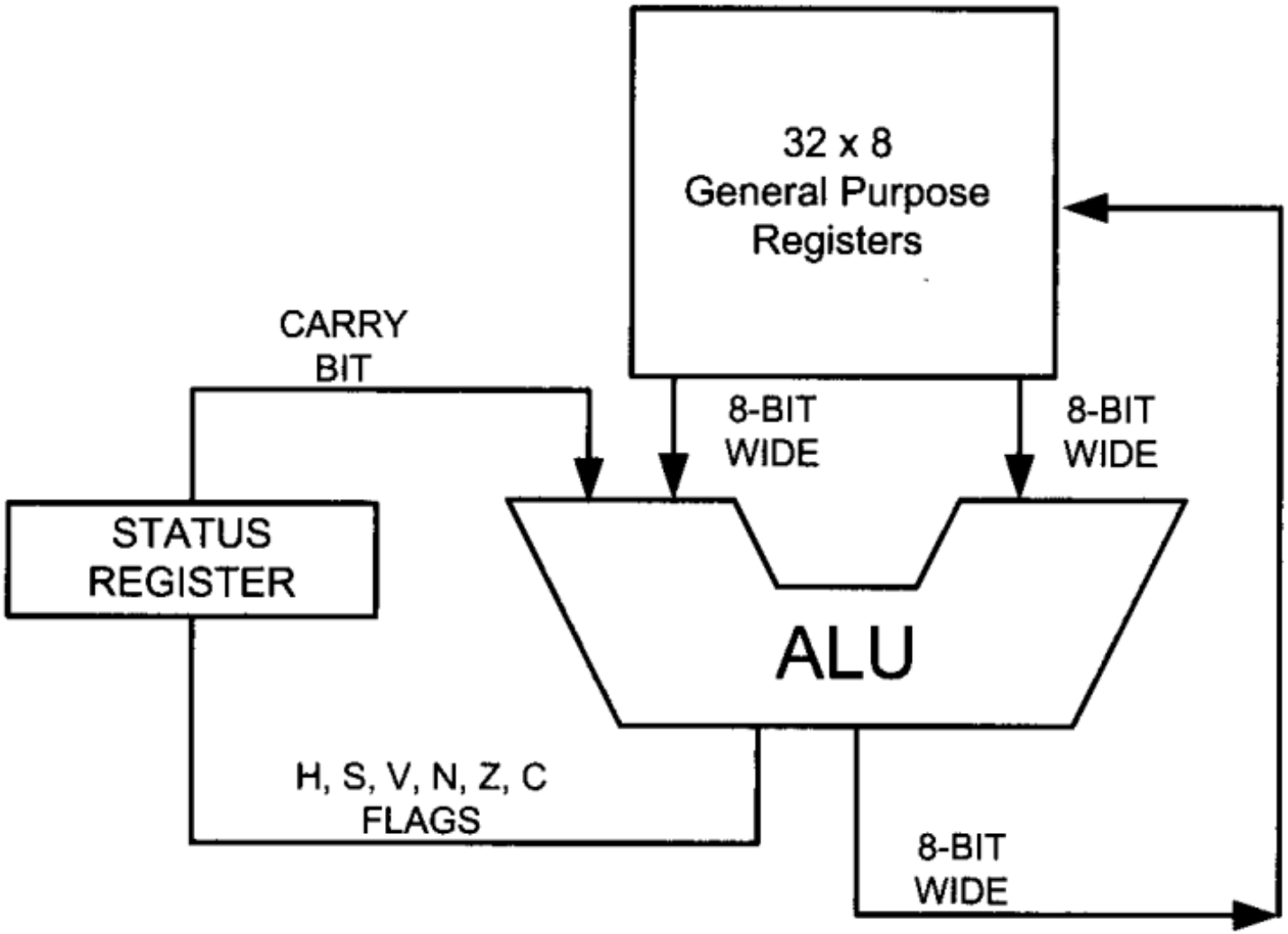
```
ADD   Rd, Rr   ; suma Rd con Rr y alamcena en Rd.
```

- Ejemplo

```
LDI   R20, 0 × 25   ; carga al GPR R20 con el dato 0 × 25,
LDI   R21, 0 × 37   ; carga al GPR R21 con el dato 0 × 37,
ADD   R20, R21      ; suma R21 con R20, R20= 0 × 59.
```

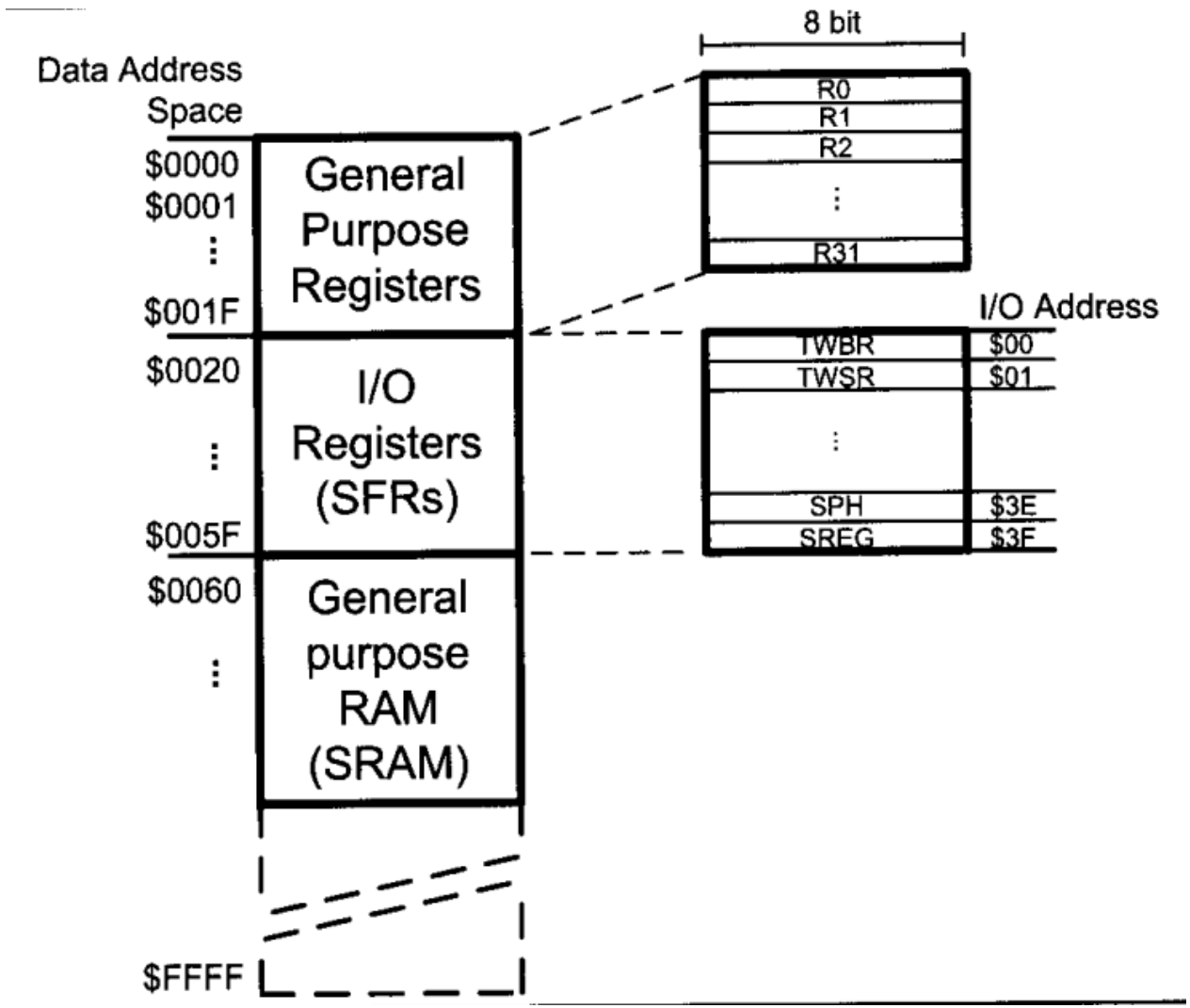
- Escriba las instrucciones para sumar 0×16 y $0 \times CD$. Colocando el resultado en R19.

GPR y ALU (arithmetic logic unit)



Memoria de datos del AVR

- Espacio de memoria de programa y espacio de memoria de datos. El espacio de memoria de datos tiene la siguiente estructura



Memoria de datos del AVR

- Memoria I/O
  - Dedicada a funciones específicas: temporizadores, registro de estatus, comunicación serial, puertos I/O, ADC. La designación alterna corresponde del número de pin y las funciones periféricas asignadas.
  - Todos los AVR tienen al menos 64 bytes de ubicaciones de memoria I/0.
  - En otros microcontroladores los registros I/O se denominan como SFR.
- Datos internos SRAM
  - Se utiliza para guardar datos y parámetros. Puede accesarse directamente desde su dirección.
  - El tamaño de la SRAM varía entre chips.

Instrucciones para Memoria de datos

- Instrucción LDS

```
LDS   Rd, K   ; carga en Rd con el contenido de la dirección K.
              ; K es una dirección entre $0000 y $FFFF.
```

LDS ordena al CPU a cargar un byte desde la dirección de memoria K al GPR.

- Al ejecutarse el GPR tendrá el valor que almacenado en la dirección de memoria.

- Ejemplo

```
LDS   R5, 0 × 200   ; carga en R5 el contenido de la dirección $200.
```

Instrucciones para Memoria de datos

- Ejemplo

```
LDS   R0, 0 × 300   ; R0 contiene el dato de la ubicación $300.
LDS   R1, 0 × 302   ; R1 contiene el dato de la ubicación $302.
ADD    R1, R0       ; suma R0 con R1.
```

Si $\$300 = \alpha$ y $\$302 = \beta$, después del código anterior $R1 = \alpha + \beta$.

Instrucciones para Memoria de datos

- Instrucción STS

```
STS   K, Rd   ; almacena el contenido de Rd en la dirección K.
              ; K es una dirección entre $0000 y $FFFF.
```

STS ordena al CPU a almacenar el byte contenido en Rd en la ubicación K.

- Ejemplo

```
LDI   R20, 0 × 99     ; R20 = 0 × 99.
STS   0 × 200, R20    ; almacena R20 en la dirección 0 × 200.
STS   0 × 201, R20    ; almacena R20 en la dirección 0 × 201.
STS   0 × 210, R20    ; almacena R20 en la dirección 0 × 210.
```

Instrucciones para Memoria de datos

- Explicar las siguientes líneas de instrucción

LDS R30, 0×220 ;

LDS R31, 0×221 ;

ADD R31, R30 ;

STS 0×220, $R31$;

Instrucciones para Memoria de datos

- Determine el contenido de R20, R21 y de la ubicación de memoria 0×120, al ejecutar el siguiente programa.

```
LDI   R20, 5          ;
LDI   R21, 2          ;
ADD   R20, R21        ;
ADD   R20, R21        ;
STS   0 × 120, R20    ;
```

Registros I/O

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

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

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

Instrucciones para Memoria de datos

- Instrucción IN

$$\begin{array}{lll} \text{IN} & \text{Rd, K} & ;\text{carga una ubicación I/0 en Rd.} \\ & & ;0 \leq d \leq 31,\ 0 \leq A \leq 61. \end{array}$$

  IN ordena al CPU a cargar el byte contenido en el registro I/O.

- Ejemplo

$$\text{IN} \quad \text{R20, } 0 \times 16 \quad ;\text{R20} = \text{contenido de la ubicación I/0 } 0 \times 16.$$

- En la instrucción IN se utiliza la designación alterna de los registros I/O conocida como direcciones I/O.
- IN se ejecuta más rápido que LDS. IN ocupa menos espacio de memoria. IN puede utilizar los registros en lugar de las direcciones. Todos los AVR incluyen IN.

Instrucciones para Memoria de datos

- Con IN se puede generar el siguiente tipo de instrucciones

```
IN   R19, PIND   ; PIND se carga en R20.
```

- Ejemplo

```
IN   R1, PIND        ; PIND se carga en R1
IN   R2, PINB        ; PINB se carga en R2
ADD  R1, R2          ; R1 = R1 + R2
STS  0 × 300, R1     ; se guarda R1 en 0 × 300
```

Instrucciones para Memoria de datos

- Instrucción OUT

```
OUT   A, Rd   ; almacena Rd en el registro I/O A.
              ; 0 ≤ d ≤ 31, 0 ≤ A ≤ 63
```

OUT ordena al CPU a almacenar el contenido de Rd en el I/O registro A.

- Ejemplo

```
OUT   PORTD, R10   ; almacena R10 en el registro PORTD.
```

- Ejemplo

```
LDI   R20, 0 × E6   ; R20 = 0 × E6.
OUT   SPL, R20      ; almacena el valor de R20 en SPL.
```

Instrucciones para Memoria de datos

- El siguiente programa obtiene el dato de PINB y lo envía al registro I/O PORTC continuamente.

```
AGAIN:  IN   R16, PINB      ; R20 = PINB
        OUT  PORTC, R16     ; almacena el valor de R16 en PORTC.
        JMP  AGAIN          ; las instrucciones se repiten
                            para siempre.
```

Instrucciones para Memoria de datos

- Instrucción MOV

```
MOV   Rd, Rr   ;Rd = Rr (copia Rr a Rd).
               ;Rd y Rr pueden ser cualquiera de los GPR.
```

- Ejemplo

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

Instrucciones para ALU

- Instrucción INC

```
INC   Rd   ; incrementa Rd una unidad, 0 ≤ d ≤ 31.
```

- Ejemplo

```
LDS   R20, 0 × 430   ; R20 carga el valor contenido en 0 × 430.
INC   R20            ; R20 = R20 + 1.
STS   0 × 430, R20   ; almacena el valor de R20 en 0 × 420.
```

Instrucciones para ALU

- Instrucción SUB

SUB Rd, Rr ; Rd = Rd - Rr.

- Ejemplo

```
LDS   R0, 0x300   ; R0 = contenido de 0x300.
LDI   R16, 0x5    ; R16 = 0x5.
SUB   R0, R16     ; R0 = R0 - R16.
STS   0x320, R0   ; almacena en 0x320 el valor de R0.
```

Instrucciones para ALU

- Instrucción DEC

$$\text{DEC} \quad \text{Rd} \quad ; \text{Rd} = \text{Rd - } 1.$$

Esta instrucción resta una unidad de Rd y la guarda en el registro Rd.

- Ejemplo

$$\begin{array}{lll} \text{LDI} & \text{R30, } 3 & ; \text{R30} = 3. \\ \text{DEC} & \text{R30} & ; \text{R30} = 2. \\ \text{DEC} & \text{R30} & ; \text{R30} = 1. \\ \text{DEC} & \text{R30} & ; \text{R30} = 0. \end{array}$$

Instrucciones para ALU

- Instrucción COM

```
COM   Rd   ; calcula el complemento del valos contenido en Rd.
```

- Ejemplo

```
LDI   R16, 0 × 55      ; R16 = 0 × 55.
OUT   PORTB, R16       ; PORTB = 0 × 55.
COM   R16              ; complemento de R16, R16 = 0 × AA.
OUT   PORTB, R16       ; PORTB = 0 × AA.
```