Principios de mecatrónica
Clase 1-4

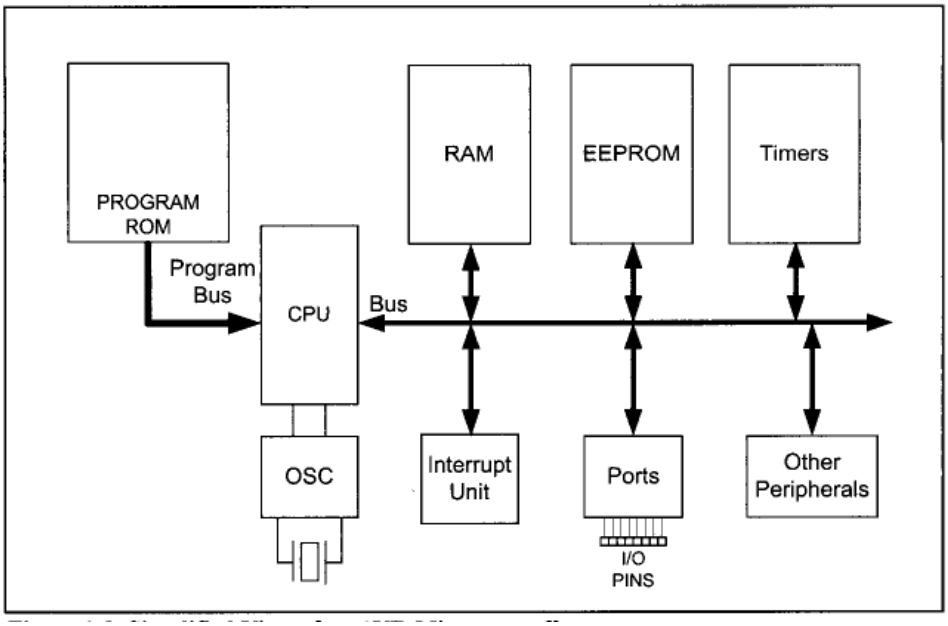
SDI-11561-004, SDI-11561-002

Microcontrolador AVR- de ATMEL

Principales características

- Arquitectura RISC (Reduced instruction set computing) de 8 btis
  - i) RISC tiene una medida fija de instrucción. En una CISC, la instrucción puede ser de 1, 2 o hasta 3 bytes. Esto implica que la medida de la instrucción de entrada nunca es conocida. El AVR usa instrucciones con 2 bytes o 4.
  - ii) Comparado con CISC, todas las arquitectural RISC tienen al menos 32 registros.
  - iii) RISC tiene un conjunto pequeño de instrucciones, las cuales son básicas, tales como ADD, SUB, MUL, LOAD, STORE, AND, OR, EOR, CALL, JUMP, y son limitadas a nivel de progamación de alto nivel.
  - iv) RISC tiene buses separados para datos y código.
  - v) RISC usa una arquitectura de cargar/almacenar. En CISC, los datos pueden ser manipulados mientras todavía están en memoria.
- Ejecución e instrucciones se realizan en un solo ciclo de reloj.

Componentes de un microcontralador



1.- Puertos I/O digitales

i) interruptores

ii) sensores de encendido-apagado

iii) A/D o D/A externos

iv) pantallas digitales

v) actuadores de encendido–apagado

2.- Comunicación serial

i) EEPROM externa

ii) otros microcontroladores

iii) computadora huésped

3.- A/D

i) sensores analógicos

ii) potenciómetros

III) monitoreo de voltajes

4.- D/A

I) actuadores analógicos

II) amplificadores

III) pantallas analógicas

5.- 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.

6.) 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. Asimismo, la memoria RAM puede ser reescrita y leída constantemente. La RAM, que básicamente es una pieza de hardware, funciona almacenando las instrucciones que deberá ejecutar el microprocesador a cada instante. De ahí la importancia de su velocidad.

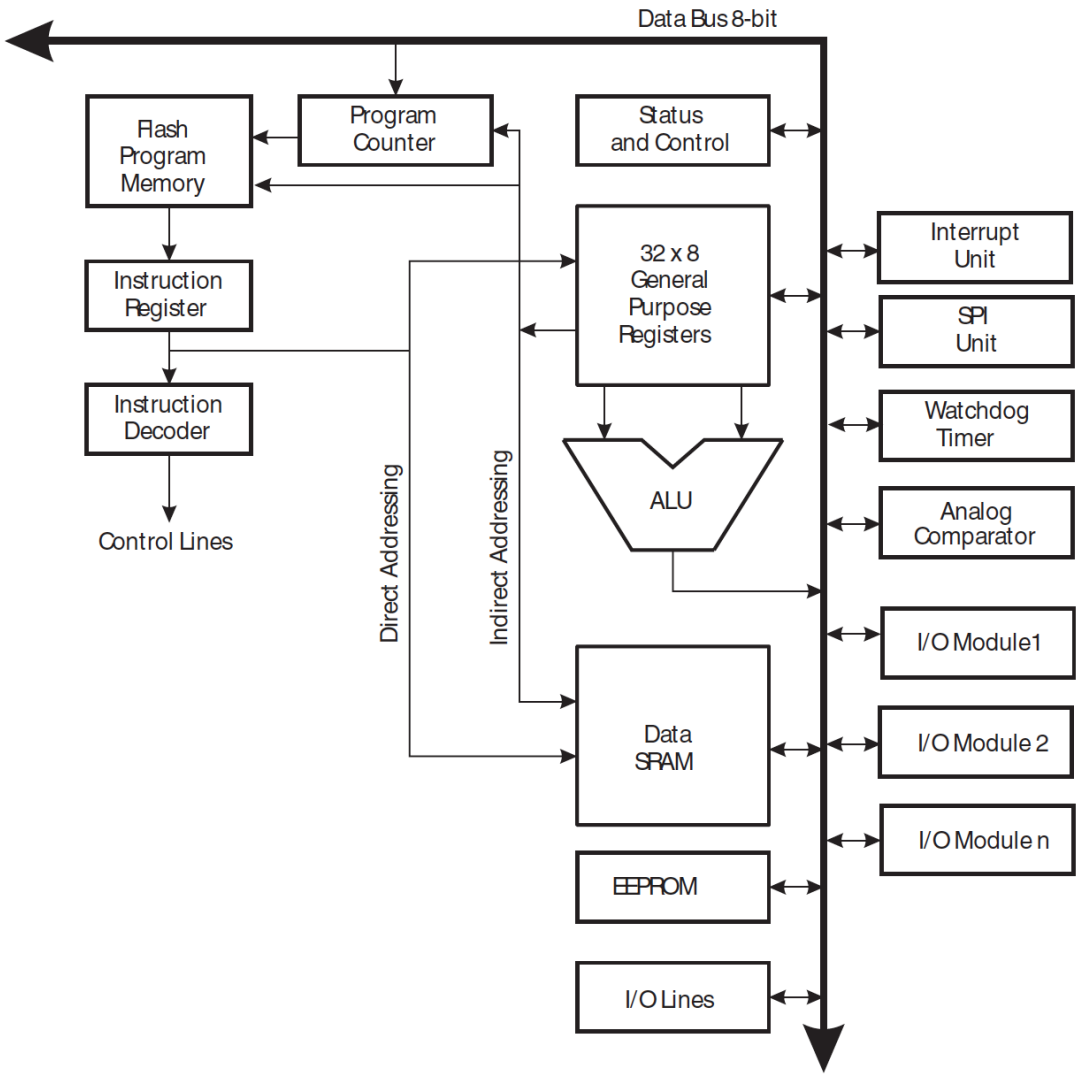
6.) Memoria ROM. La memoria ROM es el medio de almacenamiento de programas o datos que permiten el buen funcionamiento de los ordenadores o dispositivos electrónicos a través de la lectura de la información sin que pueda ser destruida o reprogramable. La memoria ROM es conocida como memoria no volátil 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.

El timer0 AVR es un temporizador–contador de 8 bits, el registro donde se guardan los valores del timer0 AVR es el registro temporizador contador representado por **TCNT0**, cuando es utilizado como temporizador, sus valores aumentaran de uno en uno entre 0 y 255 con cada ciclo de reloj, por ejemplo si el oscilador con el que está funcionando el microcontrolador AVR es de 1MHz, entonces el registro TCNT0 aumentará una unidad en 1μs, si el registro TCNT0 se incrementa en 100 unidades habrán transcurrido 100μs; cuando es utilizado como contador el temporizador AVR ya no aumenta su valor de uno en uno en cada ciclo de reloj, sino que lo hará mediante el flanco de subida o el flanco de bajada de alguna señal que llegue a un pin especial del AVR conectado al timer0 AVR, este pin es identificado como T0,, esto puede variar de acuerdo al microcontrolador avr utilizado, pero siempre se llamará T0.

Núcleo del AVR CPU

El diseño y funcionamiento de una simple computadora como la descrita previamente, forma parte de un microcontrolador, con características como: acceso a memorias de datos e instrucciones, ejecutar cálculos, control de periféricos y manipualación de interrupciones.

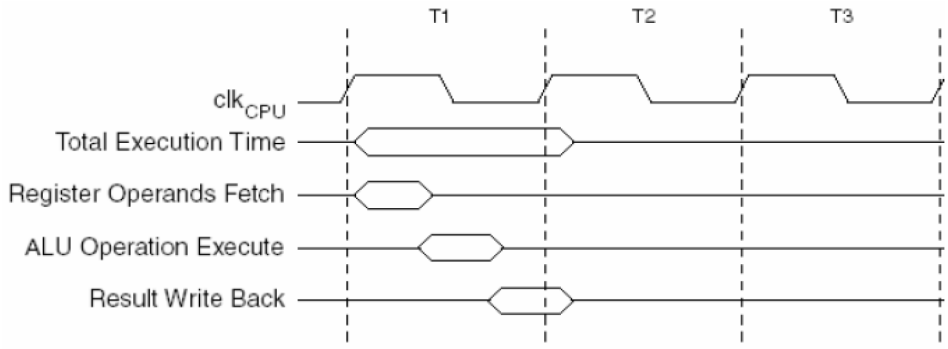
El siguiente diagrama describe el núcleo del microprocesador 2560 el cual forma parte del Arduino Mega utilizado durante el curso.



AVR cpu core

- Después de una operación aritmética, el registro de Estado (Status Register) es actualizado, para reflejar información acerca de la operación.
- La mayoría de instrucciones AVR tienen un formato en una palabra de 16-bits.
- En un ciclo de reloj se pueden leer 2 registros que funcionen como operandos para la ALU, que se realice la operación y el resultado quede disponible para escribirse en uno de esos registros.

- Para la ejecución, la duración del ciclo es suficiente para permitir la lectura de registros, la operación de la ALU y la escritura en el registro destino.
- Tiene un único conjunto de instrucciones y registros.
- Arquitectura Harvard



- Watch dog Timer (WDT) puede ser útil para reiniciar automáticamente el sistema cada vez que se produce un tiempo de espera.

  El restablecimiento del sistema es necesario para evitar fallas del sistema en una situación de falla de hardware o error de programa.

  Uno puede restablecer manualmente el sistema para recuperarse de errores. Pero no siempre es posible restablecer manualmente el sistema, especialmente una vez que se ha implementado. Para superar estos problemas, se necesita un temporizador de vigilancia para restablecer automáticamente el sistema sin intervención humana.
- AVR ATmega16 / ATmega32 tiene tres interrupciones de hardware externas en los pines PD2, PD3 y PB2 que se conocen como INT0, INT1 e INT2, respectivamente. Tras la activación de estas interrupciones, el controlador ATmega se interrumpe en cualquier tarea que esté haciendo y salta para realizar la rutina de servicio de interrupción.

Ejecución de instrucciones

- 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.
- Las instrucciones en la memoria de programa son ejecutadas con una segmentación de dos etapas.
- Mientras una instrucción está siendo ejecutada, la siguiente es capturada de la memoria de programa. Este concepto hace que se produzca una instrucción por cada ciclo de reloj.

Archivo de registros

- El microcontrolador contiene un archivo de registro de propósito general de 32×8 bits, el cual usa un ciclo de reloj.
- En una típica operación en el ALU, dos operandos son la salida del archivo de registro, la operación es ejecutada, y el resultado es almacenado de regreso en el archivo de registro–en un sólo ciclo.
- Apuntadores para el manejo de variables en memoria. los tres últimos pares de registros internos (6 últimos registros) del procesador son usados como punteros de 16 bits al espacio de memoria externa (direccionamiento indirecto), bajo los nombres X, Y y Z.
- El registro Z también puede usarse como apuntador a la memoria de programa.

  Así, los apuntadores pueden "*apuntar*" a (contener la dirección de) cualquier dirección del espacio de RAM. Esto junto con las instrucciones adecuadas conforman el direccionamiento indirecto más potente, muy útil por ejemplo para mover grandes bloques de datos.

- El AVR fue diseñado desde un comienzo para la ejecución eficiente de código C compilado.
- Las instrucciones que operan sobre el archivo de Registros tienen acceso a todos ellos. Las que operan un registro con una constante sólo trabajan con los registros de R17 a R31.
- Cada registro tienen una dirección que le permite ser tratado como cualquier otra localidad de RAM (0x000–0x01F), utilizando instrucciones de Carga (LD) y almacenamiento (ST).

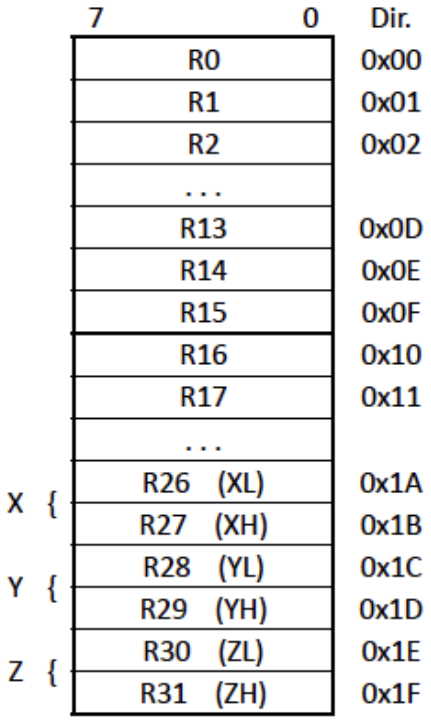


Figura 1: Registro de 32 bits

Mapa de memoria

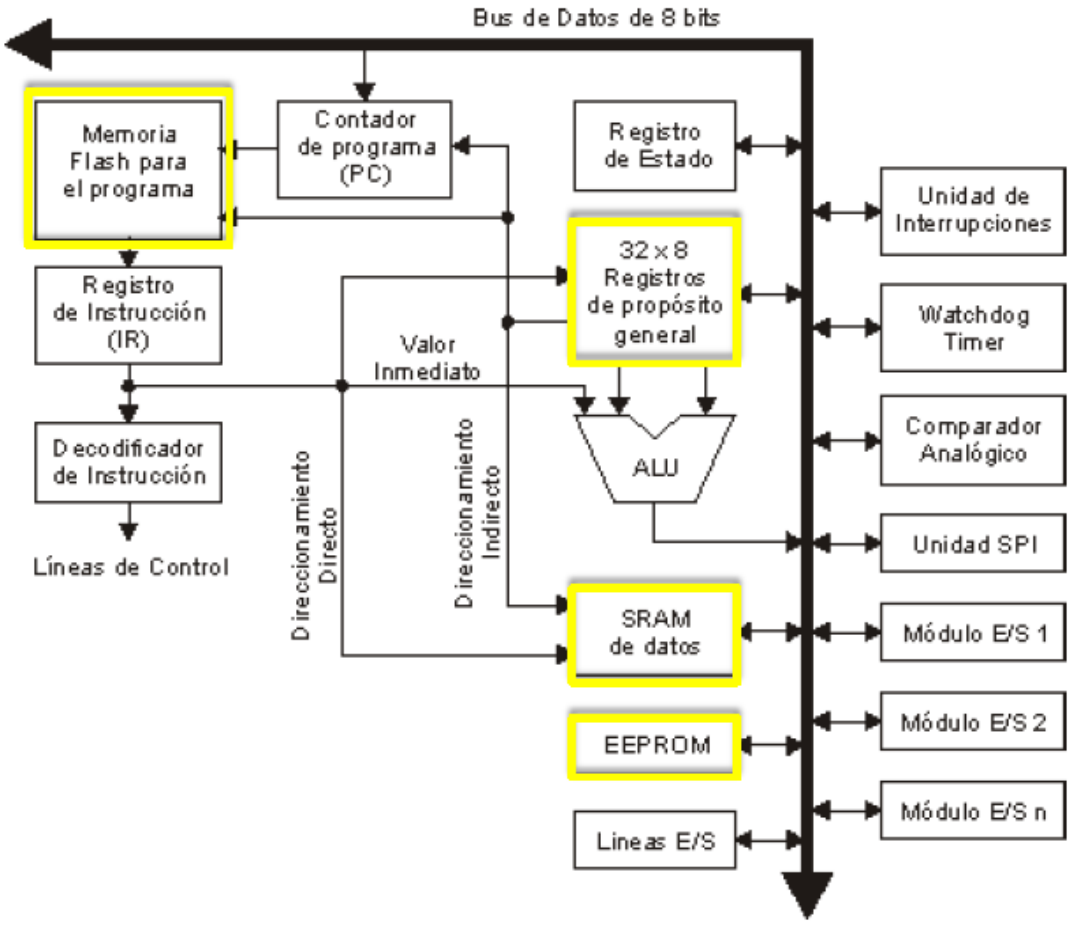


Figura 2: Mapa de memoria

Memoria de programa

- Espacio continuo de memoria Flash cuyo tamaño varia entre procesadores, para el ATMega 8 es de 8 Kbytes (organizados como 4 K x 16 bits) y para el ATMega 16 es de 16Kbytes (organizados como 8 K x 16 bits). Soporta hasta 10,000 ciclos de escritura/borrado.
- La memoria se puede dividir en una sección para aplicación y una sección de arranque, donde podría manejarse un cargador para auto programación.

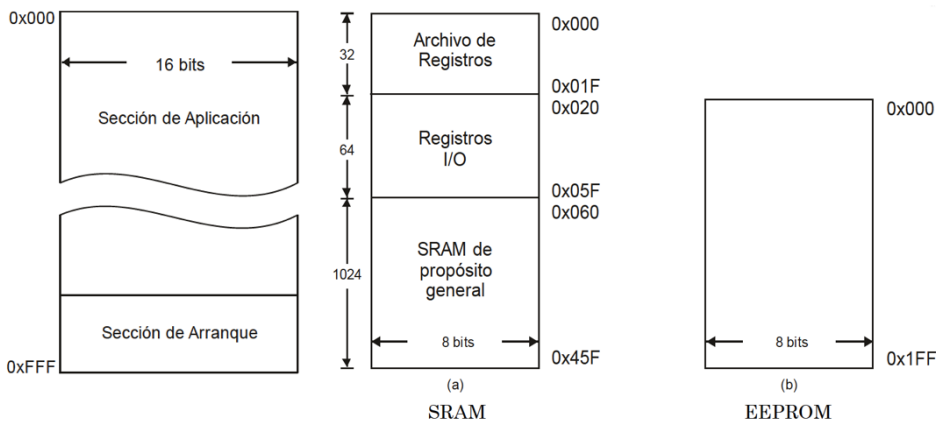


Figura 3: Mapa de memoria

Memoria SRAM de Datos

- Es un espacio de 1120 localidades de 8 bits e incluyen:

  i) Un Archivo de Registros de 32 localidades (registros).

  ii) 64 Registros I / O (Puertos, configuración de recursos, etc).

  iii) 1024 localidades de propósito general. (RAM)

Registros I/O

- Son 64 Registros e incluyen a los Puertos de Entrada/Salida, así como registros para la configuración, el control y el estado de los recursos internos.
- Se tiene acceso a los registros I/O con las instrucciones IN y OUT, intercambiando datos con el Archivo de Registros. Con estas instrucciones deben usarse las direcciones 0x00 - 0x3F.
- Los registros I/O pueden ser tratados como memoria, con las instrucciones LD y ST (direcciones 0x20 - 0x5F).
- Los Registros I/O en el rango 0x00 - 0x1F son accesibles por bit. Para modificarse, con instrucciones SBI y CBI o para evaluarse, a través de las instrucciones SBIS y SBIC.

Conjunto y uso de instrucciones

La sintaxis y operandos utilizados del conjunto de instrucciones que poseen los microcontroladores AVR serán descritos a continuación. El conjunto de instrucciones para estos microcontroladores se puede dividir en el siguiente orden de grupos:

- 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:

[Mnemónico] operando1, operando2

Los Mnemónicos son las instrucciones en lenguaje ensamblador que puede reconocer el microprocesador. 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 es almacenado en operando1.

Nomenclatura del conjunto de instrucciones

Instrucciones Aritméticas y Lógicas

Mnemónico	Operandos	Descripción
ADD	Rd,Rr	Suma sin acarreo
ADC	Rd,Rr	Suma sin acarreo
ADIW	Rd, K	Suma un inmediato a un registro de palabra
SUB	Rd,Rr	Resta sin acarreo
SUBI	Rd,K	Resta un inmediato
SBC	Rd,Rr	Resta con acarreo
SBCI	Rd,K	Resta un inmediato con acarreo
AND	Rd,Rr	AND lógico
ANDI	Rd,K	AND lógico con inmediato
OR	Rd,Rr	OR lógico
ORI	Rd,K	OR lógico con inmediato
EOR	Rd,Rr	OR exclusivo
COM	Rd	Complemento a uno
NEG	Rd	Complemento a dos
SBR	Rd,K	Pon a uno los bits de un registro
CBR	Rd,K	Pon a cero los bits de un registro
INC	Rd	Incrementa al registro
DEC	Rd	Decrementa al registro
TST	Rd	Revisa si es cero o menor
CLR	Rd	Borra el registro
SER	Rd	Carga los bits de un registro
ADIW	Rd+1:Rd,K	Suma un inmediato a un registro de palabra
SBIW	Rd+1:Rd,K	Resta un inmediato a un registro de palabra
MUL	Rd,Rr	Multiplica sin signo
MULS	Rd,Rr	Multiplica con signo
MULSU	Rd,Rr	Multiplica signado con no signado

Instrucciones de desvío (salto)

Mnemónicos	Operandos	Descripción
RJMP	k	Salto relativo
IJMP	Ninguno	Salto indirecto
EIJMP	Ninguno	Salto indirecto extendido
JMP	k	Salto
RCALL	k	Llamada relativa a subrutina
ICALL	Ninguno	Llamada indirecta a subrutina

CALL	k	Llamada a subrutina
RET	Ninguno	Regreso de subrutina
RETI	Ninguno	Regreso de interrupción
CPSE	Rd,Rr	Compare y salta si es igual
CP	Rd,Rr	Compara
CPC	Rd,Rr	Compara con acarreo
CPI	Rd,K	Compara con inmediato
SBRC	Rr,b	Salta si el bit del registro es cero
SBRS	Rr,b	Salta si el bit del registro es uno
SBIC	P,b	Salta si el bit del registro I/O es cero
SBIS	P,b	Salta si el bit del registro I/O es uno
BRBC	s,k	Salta si el bit de SREG es cero
BRBS	s,k	Salta si el bit de SREG es uno
BREQ	k	Salta si es igual
BRNE	k	Salta si es diferente
BRCS	k	Salta si C está a uno
BRCC	k	Salta si C está a cero
BRSH	k	Salta si es mayor o igual
BRLO	k	Salta si es menor
BRMI	k	Salta si es negativo
BRPL	k	Salta si es mayor
BRGE	k	Salta si es mayor o igual (con signo)
BRLT	k	Salta si es menor (con signo)
BRHS	k	Salta si H está a uno

Instrucciones de Transferencia de Datos		
Mnemónicos	**Operandos**	**Descripción**
MOV	Rd,Rr	Copia de registros
MOVW	Rd,Rr	Copia registros de palabra
LDI	Rd,K	Carga de un inmediato
LDS	Rd,k	Carga directa
LD	Rd,X	Carga indirecta
LD	Rd,X+	Carga indirecta con Post-Incremento
LD	Rd,-X	Carga indirecta con Pre-Decremento
LD	Rd,Y	Carga indirecta
LD	Rd,Y+	Carga indirecta con Post-Incremento
LD	Rd,-Y	Carga indirecta con Pre-Decremento
LDD	Rd,Y+q	Carga indirecta con desplazamiento
LD	Rd,Z	Carga indirecta con
LD	Rd,Z+	Carga indirecta con Post-Incremento
LD	Rd,-Z	Carga indirecta con Pre-Decremento
LDD	Rd,Z+q	Carga indirecta con desplazamiento
STS	k,Rr	Almacena directamente

ST	X,Rr	Almacena indirectamente
ST	X+,Rr	Almacena indirectamente con Post-Incremento
ST	-X,Rr	Almacena indirectamente con Pre-Decremento
ST	Y,Rr	Almacena indirectamente
ST	Y+,Rr	Almacena indirectamente con Post-Incremento
ST	-Y,Rr	Almacena indirectamente con Pre-Decremento
ST	Y+q,Rr	Almacena indirectamente con desplazamiento
ST	Z,Rr	Almacena indirectamente
ST	Z+,Rr	Almacena indirectamente con Post-Incremento
ST	-Z,Rr	Almacena indirectamente con Pre-Decremento
ST	Z+q,Rr	Almacena indirectamente con desplazamiento
LPM	Ninguno	Carga memoria de programa
SPM	Ninguno	Almacena memoria de programa
IN	Rd,P	Cargar un registro con un I/O
OUT	P,Rr	Cargar un I/O con un registro
PUSH	Rr	Cargar registro en la pila
POP	Rd	Sacar dato de la pila

Instrucciones de Bit y prueba de Bit

Mnemónicos	Operandos	Descripción
LSL	Rd	Desplazamiento a la izquierda
LSR	Rd	Desplazamiento a la derecha
ROL	Rd	Rotación a la izquierda con acarreo
ROR	Rd	Rotación a la derecha con acarreo
ASR	Rd	Desplazamiento aritmético
SWAP	Rd	Intercambio de nibbles
BSET	s	Poner a uno la bandera
BCLR	s	Poner a cero la bandera
SBI	P,b	Poner a uno el bit de un registro I/O
CBI	P,b	Poner a cero el bit de un registro I/O
BST	Rr,b	Guarda en T el bit de un registro
BLD	Rd,b	Carga T en el bit de un registro
SEC	Ninguno	Poner a uno la bandera C
CLC	Ninguno	Poner a cero la bandera C
SEN	Ninguno	Poner a uno la bandera N
CLN	Ninguno	Poner a cero la bandera N
SEZ	Ninguno	Poner a uno la bandera Z
CLZ	Ninguno	Poner a cero la bandera Z
SEI	Ninguno	Poner a uno la bandera I
CLI	Ninguno	Poner a cero la bandera I
SES	Ninguno	Poner a uno la bandera S
CLN	Ninguno	Poner a cero la bandera N
SEV	Ninguno	Poner a uno la bandera V

Registros y operandos	
Rd	Registro destino (y fuente) del bloque de registros de trabajo
Rr	Registro fuente del bloque de registros de trabajo
R	Resultado después de que una instrucción es ejecutada
K	Dato inmediato (constante)
k	Dirección de memoria
b	Bit de un registro I/O (7-0)
s	Bit en el registro de estado (SREG)
X,Y,Z	Registros de direccionamiento indirecto o registros de palabra (X=R27:R26 Y=R29:R28 Z=R31:R30)
P	Registros del bloque I/O

Banderas del registros de estado (SREG)

C: Bandera de acarreo	S: N⊕V para pruebas de signo
Z: Bandera de resultado cero	H: Bandera de medio acarreo
N: Bandera de resultado negativo	T: Bit transferido por la instrucción BLD o BST
V: Indicador de complemento a dos desbordado	I: Bandera de interrupciones globales

Uso de los archivos de registro–ejemplo

Como hemos mencionado un AVR tiene 32 registros de propósito general (GPR). Ellos pueden ser usados por todas las instrucciones aritméticas y lógicas. Para entender el uso de GRP, nosotros mostraremos en el contexto de dos simples intrusiones: LDI y ADD.

Instrucción LDI

Estado simple, la instrucción LDI copia un dato de 8-bit dentro del GPR. Tiene el siguiente formato.

$LDI\ R_d,\ K$;carga R_d (destinación) con valor inmediato K
;d debe ser entre 16 y 31.

K es un valor de 8-bit que puede ser en decimal de $0-255$, ó en hexadecimal $00-FF$, y R_d es de R_{16} a R_{31} (ninguno de los 16 GRP superiores). La I en LDI significa *inmediato*. La siguiente instrucción carga el registro $R20$ con un valor de $0x25$.

$LDI\ R20,\ 0x25$;carga$R20$ con $0x25$ (R20=0x25).

Remark 1 No es posible cargar valores dentro de los registros $R0$ hasta $R15$ usando la instrucción LDI. Por ejemplo la siguiente instrucción no es valida:

$LDI\ R05,\ 0x75$;instrucción no valida.

Instrucción ADD

La instrucción ADD tiene el siguiente formato:

$ADD\ R_d,\ R_r$;suma R_r a R_d y almacena el resultado en R_d

La instrucción ADD dice al CPU sumar el valor de R_r a R_d y pone el resultado de regreso dentro del registro de R_d. Para sumar dos números tales como $0x25$ y $0x34$, uno puede hacer lo siguiente:

$LDI\ R16,\ 0x25$;carga $0x25$ dentro de $R16$
$LDI\ R17,\ 0x34$;carga $0x34$ dentro de $R17$
$ADD\ R16,\ R17$;suma el valor de $R17$ a $R16$ (R16=R16 +R17).

Cuando programamos los GPRs del microcontrolador AVR con un valor inmediato, debemos notar los siguientes puntos:

- Si queremos presentar un número hexadecimal debemos poner el signo ($) o $0x$ enfrente de él. Si ninguna de estas notaciones son usadas, entonces se considera que el número es decimal.
- Si valores del 0 al F son movidos dentro de los registros de 8 bits, el resto de los bits son asumidos a ser ceros.
- Mover un valor mayor a 255 dentro de los GPR causará un error.

Memoria de datos del AVR

El espacio de datos de memoria esta compuesto de tres partes: Registros de propósito general (GPRs), memoria I/0 and datos de memoria interna (SRAM) y están particionados como se muestra en la siguiente figura.

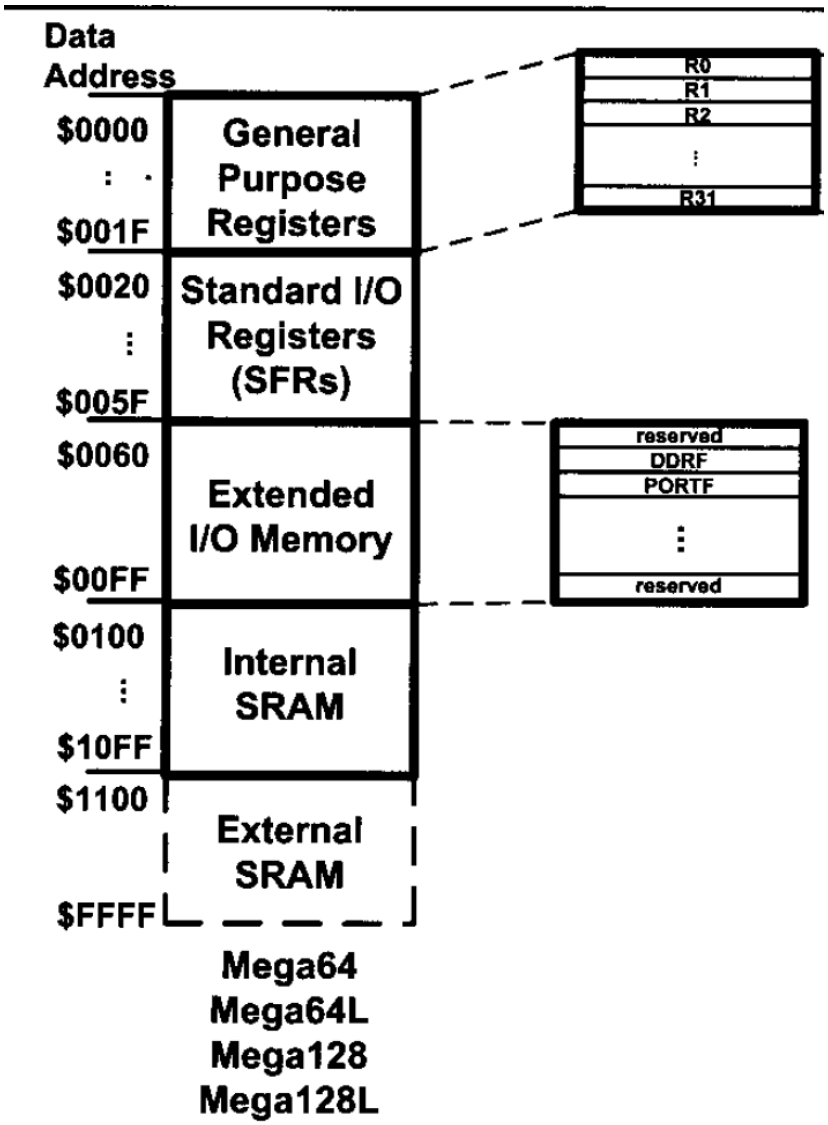


Figura 4: Espacio de memoria de datos

La memoria de I/O es dedicada para funciones especificas como el registro de status, timers, comunicación serial, puertos de I/0, ADC, y algunos otros. La función de cada memoria I/O es definidas por el diseñadorya que se puede usar para controlar el microcontrolador o los periféricos. Todos los AVR tienen al menos 64 bytes (de 8 bits cada registro) de

localidades de memoria I/O.
Los datos internos de la SRAM son ampliamente usados para almacenar dato y parámetros para los programadores de AVR y compiladores C. Se puede acceder a cada localidad de la SRAM directamente por su dirección de memoria. Es decir, se puede usar para almacenar datos en el CPU usando los puertos I/O y seriales.

Usando instrucciones con la memoria de datos

Las instrucciones que hemos usado han sido para trabajar con datos inmediatos constantes nombrados K y el GPRs como su área de destino. Ahora, nosotros mostramos como acceder a los datos de la memoria.

Instrucción LDS (LoaD direct from dataSpace)

$LDS\ R_d,\ K$;carga R_d con el contenido de localización K $(0 \leq d \leq 31)$

;K es una dirección entre \$0000 hasta $\$FFFF$.

La instrucción LSD llama al CPU para cargar (copiar) un byte de una dirección en los datos de memoria a el GPR. Después esta instrucción es ejecutada. El GPR tendrá el mismo valor como el del dato de memoria. La localización en los datos de memoria pueden ser de cualquier parte del espacio de datos. Por ejemplo la instrucción

LDS R20, 0x1

copiará el contenido de la localización 1 (en hex) dentro del $R20$.

La siguiente instrucción cargara en $R5$ el contenido de la localización $0x065$. Donde $0x065$ esta localizado en el SRAM.

LDS R5, 0x200 ; carga $R5$ con el contenido de la localización $0x065$

El siguiente programa suma el contenido de la localización $0x061$ a $0x062$. Para realizarlo, primero cargamos $R0$ con el contenido de la localización $0x061$ y $R1$ con el contenido de la localización $0x062$, entonces sumamos $R0$ a $R1$

$LDS\ R0,\ 0x061$;$R0$ es igual a el contenido de la localización $0x061$
$LDS\ R1,\ 0x062$;$R1$ es igual a el contenido de la localización $0x062$
$ADD\ R1,\ R0$;suma R0 a R1.

Instrucción STS (Store direct to data Space)

$STS\ K,\ R_r$;almacena el registro dentro de la localización K
;K es una dirección entre \$0000 hasta $\$FFFF$.

La instrucción STS llama al CPU para almacenar el contenido de le GPR a una localización de registro en el espacio de memoria. Después de ejecutar esta instrucción la localización en el espacio de datos tendrá el mismo valor como el GRP. La localización puede ser en cualquier parte del espacio de memoria. Por ejemplo la instrucción

STS 0x1 R10 copia el contenido de $R10$ dentro de la localización 1.

La siguiente instrucción almacena el contenido de $R25$ a la localización $0x400$. La cual se encuentra en la SRAM.

$STS\ 0x400,\ R25$;guarda $R25$ en el espacio de datos con localizació $0X400$.

El siguiente programa primero carga el registro $R16$ con el valor dee $0x55$, entonces mueve esté valor a los registros I/O de los puertos B, C y D. Ya que la dirección de PORTB, PORTC y PORTD son $0x38$, $0x35$, y $0x32$ respectivamente.

$LDI\ R16,\ 0x55$;$R16$=55
$STS\ 0x38,\ R16$;copia $R16$ a el puerto B (PORTB=0x55)
$STS\ 0x35,\ R16$;copia $R16$ a el puerto C (PORTC=0x55)

STS $0x32$, $R16$;copia $R16$ a el puerto D (PORTD=0x55)

Realizar un programa que sume el contenido de las localizaciones 0x220 y 0x221 y lo almacene en 0x221.