Problema 1.- Considere la siguiente situación inicial, donde se indica en hexadecimal el contenido de ciertos registros y direcciones de memoria

Dirección	***Contenido***
$200	$08
$201	$03
$202	$01
$203	$00

Registro	***Contenido***
R0	$00
R1	$FF
YH	$02
YL	$01

Indique el valor que tendrá el registro R0 después de ejecutar la instrucción indicada en cada uno de estos casos:

(a) MOV R0,R1
(b) MOVW R0,Y
(c) LD R0,Y+
(d) LD R0,-Y
(e) LDD R0,Y+2

Problema 2.- Para cada una de las siguientes instrucciones describa la operación que realiza e indique los registros internos y palabras de memoria que intervienen. Proponga para ellos unos valores iniciales concretos y determine qué valores alcanzarán finalmente.

(a) STS $100,R16
(b) ST -X,R2
(c) STD Z+10,R16
(d) LPM R19,Z+
(e) IN R4,PORTB
(f) OUT DDRC,R0

Problema 3.- Para cada una de las siguientes instrucciones describa la operación que realiza e indique los registros internos y palabras de memoria que intervienen. Proponga para ellos unos valores iniciales concretos y determine qué valores alcanzarán finalmente.

(a) PUSH R4
(b) SPM
(c) POP R3
(d) LDS R19,$202
(e) LPM
(f) MOVW X,Y

Problema 4.- Describa las siguientes instrucciones del ATMega328P

(a) SBI P,K
(b) INC Rd
(c) CP Rd,Rr
(d) RCALL etiqueta
(e) BREQ etiqueta
(f) ROL Rd
(g) NOP
(h) SLEEP
(i) WDR
(j) CPSE Rd,Rr

Problema 5.- Sean A y B dos números sin signo de un byte, almacenados en las direcciones $0100 y $0101 respectivamente. Escriba un fragmento de programa que obtenga la suma de ambos números y almacene el resultado (16 bits) en las dos siguientes posiciones de memoria.

Problema 6.- Sean A y B dos números sin signo de un byte, almacenados en las direcciones $0100 y $0101 respectivamente. Escriba un fragmento de programa que obtenga la multiplicación de ambos números y almacene el resultado (16 bits) en las dos siguientes posiciones de memoria.

Problema 7.- Escriba un fragmento de programa que sume tres números sin signo de 1 byte, almacenados consecutivamente a partir de la posición $0100 de la SRAM, y guarde el resultado (16bits) a partir de la dirección $0200.

Problema 8.- Escriba un fragmento de programa que sume tres números con signo de 1 byte, almacenados consecutivamente a partir de la posición $0100 de la SRAM, y guarde el resultado (16bits) a partir de la dirección $0200

Problema 9.- A partir de la dirección $0100 de la SRAM se encuentra almacenado un mensaje de comunicaciones que se ha recibido por el puerto serie. La longitud del mensaje viene indicado en el primer byte de dicho mensaje (máx. 255 bytes). El último byte del mensaje es una suma de comprobación (CHECKSUM) del mensaje completo. Programe una subrutina que devuelva en R0=$00 si

el mensaje recibido es correcto, o R0=$FF si se ha detectado un error (suma de comprobación incorrecta).

Problema 10.- Escriba un programa que traslade una tabla de 16 bytes almacenada en la dirección $100 a la posición $200.

Problema 11.-

(a) Escriba una subrutina que traslade una tabla de datos de una posición a otra de memoria. El número de datos a trasladar se indica en R16, y los registros X e Y indican las direcciones fuente y destino respectivamente.

(b) Resuelva el Problema 10.- utilizando la subrutina del apartado a.

Problema 12.- Escriba un fragmento de programa cargue en R0 el elemento menor de una tabla de 16 números sin signo de 1 byte almacenados a partir de la dirección $0100.

Problema 13.- Escriba una subrutina que devuelva en R0 el elemento menor de una tabla de números con signo de 1 byte. La dirección de comienzo de la tabla se indica en el registro X, y el número de datos en R16. Utilice esta subrutina para resolver el Problema 12.- .

Problema 14.- Escriba un programa que escriba a partir de la dirección $0100, todos los números pares existentes entre el 0 y el 255.

Problema 15.- Una tabla con 100 datos de 1 byte con signo está almacenada a partir de la dirección $0100. Escriba un programa que almacene en R0 el número de datos positivos que hay en dicha tabla, y en R1 el número de datos negativos.

Problema 16.- Una tabla con 16 datos de 1 byte está almacenada a partir de la dirección $0100. Escriba un programa que invierta el orden de la misma.

Problema 17.- Escriba una subrutina que invierta el orden de datos de una tabla. El número de datos (de 1 byte) se indica en R16, y la dirección de comienzo de la tabla en X.

Problema 18.- Se tiene una tabla de 10 números con signo de 1 byte almacenados a partir de la dirección $0200. Escriba un fragmento de programa que ordene la tabla de mayor a menor (algoritmo de la burbuja).

Problema 19.- Programe una subrutina que ordene, por el método de la burbuja, una tabla de números con signo de 1 byte. El número de datos se indica con R16, y la dirección de la tabla con el registro X.

Problema 20.- Programe una subrutina que obtenga la media de una tabla de datos de 1 byte. El número de datos se indica con R16, y la dirección de la tabla con el registro X.

Problema 21.- Se desea mover una palabra de 32bits situada en el byte $100 de la memoria SRAM a partir de byte $200 de la memoria SRAM. Indique las distintas formas de hacerlo especificando las instrucciones necesarias.

Problema 22.- Se desea mover una palabra de 32bits situada en el byte $100 de la memoria de programa a partir del byte $200 de la memoria SRAM. Indique las distintas formas de hacerlo especificando las instrucciones necesarias.

Problema 23.- Con las instrucciones del microcontrolador ATMega328P escriba el programa que implemente la instrucción de alto nivel:

```
FOR i=<valor_inicial> TO <valor_final> DO
        cálculo1
siguiente_instrucción_tras_el_bucle
```

Donde *cálculo1* es una secuencia de operaciones que no es necesario indicar y, *valor_inicial* y *valor_final* son dos variables con los límites del bucle FOR.

Problema 24.- Indique los valores que tomarían los pines del microcontrolador ATMega368P tras la ejecución de cada una de las siguientes instrucciones. El contenido de la posición de memoria SRAM $202 es $0A

```
.DEF    TMP=R16
LDI     TMP,$FF
OUT     DDRB,TMP
EOR     TMP,TMP
OUT     DDRC,TMP
LDS     TMP,$202
OUT     PORTB,R16
SBI     PORTB,6
CBI     PINB,2
IN      R0,PINC
```

Problema 25.- El procesador ATMega328P tiene el reloj funcionando a una frecuencia de 1Mhz. Se dispone de un dispositivo que envía 1000 valores de tamaño byte con cierta frecuencia. El procesador atiende los datos mediante interrupciones, siendo la rutina de interrupción la siguiente:

```
RECIBE_BYTE: PUSH  R0
             LDS   R0,UDR0
             ST    Y+,R0
             POP   R0
             RETI
```

Indique cada cuantos microsegundos puede enviar el dispositivo un byte sin que se pierda ninguno.

Problema 26.- En una posición de memoria $DATO existe un número binario entero positivo N. Se pretende obtener el valor SUM=N+(N-1)+(N-2)+ ... +2+1 y almacenarlo a partir de dirección de memoria $SUMA.

(a) Obtenga una subrutina en ensamblador que calcule $SUMA siendo tanto los datos como el resultado de tamaño byte. Suponga que no existe desbordamiento.

(b) Repita el apartado anterior si los datos son de tamaño Word

(c) Indique las modificaciones a realizar en los apartados anteriores si se quiere contemplar el desbordamiento.

Problema 27.- Realice una subrutina para sumar dos vectores de 10 números enteros de 32 bits que se almacenan en posiciones consecutivas de memoria. La dirección de memoria en que se encuentra el primero de ellos es el contenido del registro de direcciones X y, la del segundo está en Y. Almacene el resultado a partir de Z, suponiendo que no existen problemas de desbordamiento.

Problema 28.- Realice un programa que permita trasponer una matriz cuadrada de dimensión 5. Sus componentes son tamaño palabra (W) y la forma de almacenamiento es la mostrada en la figura.

a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}
a_{31}	a_{32}	a_{33}	a_{34}	a_{35}
a_{41}	a_{42}	a_{43}	a_{44}	a_{45}
a_{51}	a_{52}	a_{53}	a_{54}	a_{55}

a_{11} = [M($A)]
a_{12} = [M($A + 2)]
a_{13} = [M($A + 4)]
a_{14} = [M($A + 6)]
a_{15} = [M($A + 8)]
...
...

Problema 29.- A partir de la dirección de memoria $DIR de un sistema basado en un microcontrolador ATMega328P se dispone de una tabla de 20 elementos con signo de tamaño palabra. Se desea diseñar una subrutina que permita calcular el valor medio del absoluto de los elementos, es decir:

$$res = \frac{|tabla[0]| + |tabla[1]| + .. + |tabla[19]|}{20}$$

donde |x| = x si x>=0 o -x si x<0.

Problema 30.- A partir de la dirección de memoria $DIR de un sistema basado en un microcontrolador ATMega328P se dispone de una tabla de 20 elementos con signo de tamaño palabra. Se desea diseñar una subrutina que permita obtener el índice asociado al primer valor más repetido, y, en el caso de que no exista ninguno, la subrutina devolverá -1.

Problema 31.- A partir de la dirección $160 de la memoria de datos del ATMega328P se dispone de una tabla de 20 bytes sin signo. Se desea desarrollar un programa que permita almacenar en R0 el valor que más se repite en dicha tabla y 0 en el registro R1. En caso de que no se repita ninguno, el programa deberá poner a 1 el contenido del registro R1.

Problema 32.- En los pines PB7-0 de un microcontrolador ATMega328P se han conectado diodos luminosos cuyos cátodos están a tierra y en el pin PD4 se ha conectado el extremo de un pulsador mecánico (cuyo otro extremo se encuentra conectado a tierra) junto con una resistencia de pull-up. Se desea realizar un programa que permita mostrar a través de los diodos el número de pulsaciones que se generan en el pulsador. La cuenta de las pulsaciones se deberá llevar a cabo por el registro TCNT0 configurado adecuadamente en modo contador para que se incremente cada vez que exista un flanco de bajada (falling edge) en su entrada externa (PD4).

Problema 33.- Programe una rutina que permita convertir un número BCD de ocho dígitos, el cual está almacenado a partir de la dirección NUM_BCD, a binario. El número convertido en binario se almacenará a partir de la dirección NUM_BIN. Represente el diagrama de flujo asociado y dé una breve explicación de las variables que usa el programa.

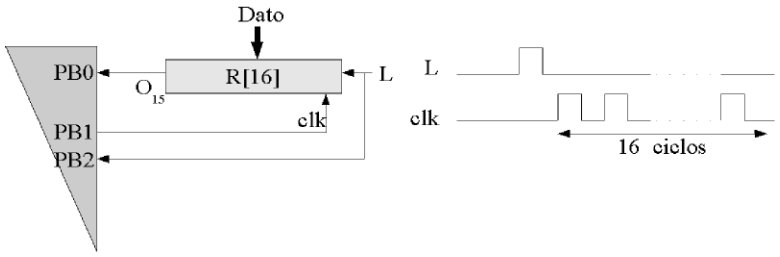
Problema 34.- En los pines del puerto B del microcontrolador ATMega328P se han colocado leds hasta un total de ocho (uno por cada pin del puerto), de modo que un 1 lógico en el pin, enciende el LED y un 0 lógico, lo apaga. En el pin PD2 se ha conectado un pulsador mecánico de modo que si éste está presionado, introduce un 0 lógico en el pin, y un 1 cuando está en reposo. Al inicio, todos los leds estarán apagados y el microcontrolador esperando que la tecla se pulse. Cuando el microcontrolador contabilice dos pulsaciones, todos los leds se pondrán en intermitencia con periodo de 1 segundo (0.5 segundos encendidos y 0.5 segundos apagados).

(a) Diseñe el programa sin usar interrupciones suponiendo que la frecuencia del reloj en de 4Mhz (periodo de 250×10^{-9}s).

(b) Diseñe el programa usando interrupciones para el temporizador y para el pulsador. Supóngase que la frecuencia de reloj es de 4Mhz (periodo de 250×10^{-9}s). Es recomendable el uso de interrupciones para el temporizador 0 y de la entrada externa.

Problema 35.- En la siguiente figura se muestra un registro de desplazamiento de 16 bits cuya salida serie, O_{15} , señal de reloj, clk, y entrada de carga, L, se han conectado a los pines PB0,PB1 y PB2 de un microcontrolador ATMega328P con cristal de 1Mhz. Se desea desarrollar un programa que permita la lectura del dato almacenado en el registro externo y su almacenamiento en los registros R0 y R1 del microcontrolador. En concreto, el registro R0 contendrá el byte más significativo y el R1 el menos significativo. Para poder acceder al dato del registro externo R el microcontrolador debe seguir la siguiente secuencia:

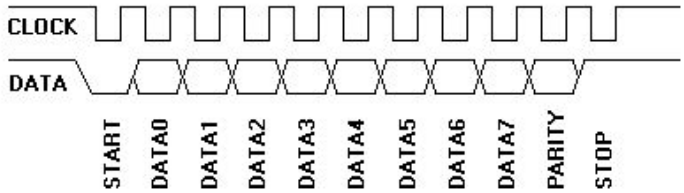
1. De forma externa, la señal L pasará de 0 a 1, lo que implica la carga de un dato en el registro R externo. El microcontrolador escuchará está línea y no tomará ninguna acción hasta que L vuelva a valer 0.
2. Transcurrido el pulso en L, el microcontrolador deberá generar 16 pulsos de reloj, tal y como muestra la figura.
3. En el nivel alto del pulso de reloj, se deberá leer el bit que aparece por la salida O15 y almacenarlo en la posición que corresponda de los registros R0 o R1.
4. Una vez finalizada la lectura de los dieciséis bits, el microcontrolador esperará la activación de un nuevo pulso en L para repetir el proceso.



Problema 36.- Un conocedor del funcionamiento del ATMega328P quiere construir un molinillo espantapájaros para evitar las molestas suciedades que éstos provocan en el balcón de su casa. El molinillo dispone de un motor de corriente continua que se pretende controlar con la salida PB0 del microcontrolador, de forma que, si PB0 es 1 el motor se activa y si es 0, se desactiva. Por otro lado, se dispone de una LDR que permite determinar si es de día o es de noche. La electrónica de adaptación hace que la LDR ponga un 1 en la entrada PB1 del microcontrolador si es de día y 0, si es de noche. Todo el conjunto se alimenta mediante una batería, por lo que es recomendable economizar su consumo. En concreto el sistema debe operar del siguiente modo:

1. Deberá estar en modo “dormido” la mayor parte del tiempo.
2. Deberá “despertarse” por una interrupción del timer0 cada minuto teniendo en cuenta que la señal de reloj es de 1khz (automáticamente el microcontrolador ejecuta la rutina y continúa la ejecución de la instrucción situada detrás de SLEEP).
3. Si es de día, cada 10minutos, se deberá activar el molinillo durante un 1 minuto. Si es de noche, no se procederá a ninguna acción.

Problema 37.- Se desea simular la pulsación de la tecla ENTER de un teclado de un PC usando un único pulsador externo que se conectará, mediante un microcontrolador ATMega328P, al puerto PS2 de dicho PC. Por sencillez, consideraremos que la comunicación sólo se realizará de forma unidireccional desde el microcontrolador al PC. En ese sentido, el interfaz PS2 consta de dos líneas: DATA y CLOCK. Normalmente esas líneas están a 1 lógico para representar que la transmisión se encuentra en estado de reposo. La siguiente figura representa el protocolo de comunicación a seguir, formado por 1 bit de start, 8 bits de datos, un bit de paridad impar y un bit de stop que siempre está a 1.



El microcontrolador debe generar la señal de reloj cuya duración máxima del nivel máximo o mínimo es de 40us. El PC capturará el bit recibido por la línea DATA en el flanco de bajada. Cuando el usuario pulse el pulsador externo, el microcontrolador enviará el código $5A al interfaz PS2 y cuando libere dicho pulsador, el microcontrolador deberá enviar el código $F0 seguido del $5A..
La señal de reloj es de 2Mhz, los terminales PB3 y PB4 se usarán para generar CLOCK y DATA respectivamente y el pulsador externo se ha conectado al puerto PD0, de forma que cuando está pulsado se introduce un 0 y cuando no, un 1.

(a) Diseñe una rutina que permita configurar los puertos y el timer0 para que que pueda generar desbordamientos cada 40us. No use interrupciones para el temporizador.

(b) Diseñe una rutina que permita generar una ciclo de la señal de reloj. Esto es, usando el timero0, la rutina pondrá el pin de CLOCK a 1 durante 40us y después a 0 durante el mismo tiempo.

(c) Diseñe una rutina que permita calcular el bit de paridad asociado a un byte almacenado en la dirección de memoria DATO ($60 de la SRAM).

(d) Diseñe una rutina que permita transmitir la trama PS2 a partir de las rutinas anteriores y el dato almacenado en la dirección DATO.

(e) Diseñe el programa que permita conectar las rutinas anteriores con los procesos a seguir con la pulsación y liberación de la tecla ENTER externa.

Detalles para resolver los problemas

- El procesador tiene 1K de SRAM
- La memoria SRAM comienza en la dirección 0x100, lo anterior son periféricos
- La carga inmediata con la instrucción LDI solo admite como destino los registros R16..R31
- No se pueden inicializar datos en la memoria SRAM, para ello se incluyen junto con la memoria de programa
- Para leer datos memoria de programa se utiliza la instrucción LPM
- En la memoria se programa no se escribe: Ignorar la instrucción SPM
- Para escribir en periféricos usar las instrucciones IN/OUT aunque se podría hacer con ST/LD
- Las directivas necesarias para hacer los problemas son: .INCLUDE, .EQU y .DEF.

Detalles sobre la programación en ensamblador

- Cuando el procesdor se inicia o resetea PC vale 0.
- En la dirección 0 de la memoria de programa comienzan los vectores de interrupción, pero, si un programa no utiliza interrupciones, el programa puede empezar en la dirección 0. Funciona perfectamente
- Cuando se termine un programa en se debe poner JMP 0 ó FIN: JMP FIN. De lo contrario, el programa en el emulador o micro sigue ejecutandose unos 32000 ciclos (32K de memoria de programa). Termina cuando PC se cicla y comienza de nuevo con PC=0. Se puede comprobar en el simulador.
- Si se van a utilizar las interrupciones solo hay que hacer un código como el de esta plantilla:

```
.ORG 0
                RJMP  RESET  ; Se salta a cada vez que se inicia el micro (PC=0)

.ORG 0x0020     ; La dirección $20 maneja la interrupción de overflow temporizador
                RJMP  INT_TIMER_0 ; Salto a subrutina de manejo de la interrupción

.ORG 0x0034     ; En esta dirección terminan los vectores de interrupción
                MI_PROGRAMA_ASM

INT_TIMER_0:    MI_MANEJO_DE_INTERRUPCION
                ...
                IRET
```

Soluciones propuestas para algunos problemas

Estas soluciones se pueden copiar y pegar en AVRStudio para comprobar el funcionamiento. Para hacer la simulaciones se puede utilizar la cabecera .INCLUDE "M168DEF.INC" que es del microcontrolador ARMega168 que es equivalente

Error: No se encuentra la fuente de referenciaSolución completa para probar en el simulador AVRStudio. Primero se introducen los datos de la tabla en la memoria del procesador

```
.INCLUDE "M168DEF.INC"
; PROBLEMA 1
; INICIALIZANDO LOS REGISTROS
LDI R16,$0
MOV R0,R16
LDI R16,$FF
MOV R1,R16
LDI YH,$02
LDI YL,$01

;CARGA SE SRAM CON LOS DATOS DE LA TABLA
LDI R16,$08
STS $200,R16
LDI R16,$03
STS $201,R16
LDI R16,$01
```

```
STS $202,R16
LDI R16,$00
STS $203,R16

; PROBLEMA A RESOLVER
MOV  R0,R1  ; R0=$FF
MOVW R0,Y   ; R0=$01 R1=$02
LD   R0,Y+  ; R0=$03 Y=$0202
LD   R0,-Y  ; R0=$03 Y=$0201
LDD  R0,Y+2 ; R0=$00 Y=$0201
```

Problema 21.- y Problema 22.- La solución inicializa primero los datos para poder probarla en el simulador. Los datos de memoria SRAM no se puede inciar por directivas del ensamblador, por tanto, hay que hacer código que escriba numeros.

```
.INCLUDE "m168DEF.INC"
; Problema 4,5. Mover 32 bits desde diferentes memorias
;Inicializando la memorias SRAM y PROG con 12345678

.CSEG
.ORG $0100   ; Datos iniciales en memoria de programa
  .DW $9ABC
  .DW $DEF0

.ORG $0

ldi yh,$01 ;Inicialización de datos de la memoria SRAM
ldi yl,$00
eor r0,r0
inc r0
st  y+,r0
inc r0
st  y+,r0
inc r0
st  y+,r0
inc r0
st  y+,r0

;Movimiento con punteros: x origen y destino

ldi xh,$01 ; moviendo de SRAM a SRAM
ldi xl,$00
ldi yh,$02
ldi yl,$00

ld  r16,x+
st  y+,r16
ld  r16,x+
st  y+,r16
ld  r16,x+
st  y+,r16
ld  r16,x+
st  y+,r16

ldi zh,$01 ; moviendo de PROGRAM a SRAM
ldi zl,$00
ldi yh,$02
ldi yl,$00

lpm r16,z+
st  y+,r16
lpm r16,z+
st  y+,r16
lpm r16,z+
st  y+,r16
lpm r16,z+
st  y+,r16
```

Problema 23.- La solución solo permite bucles con 0<inicio<255 y 0 <fin<255

```
.INCLUDE "m168DEF.INC"
```

```
.DEF inicio=r20
.DEF fin=r21
.DEF i=r22

        ; Se supone que el bucle es i=i+1

        ldi    inicio,0x10
        ldi    fin,0x15

        mov    i,inicio
bucle:  cp     inicio,fin    ; Primero se comprueba por si inicio>fin
        brge   salir_bucle
        inc    inicio
        nop                  ; Instrucciones de calculo1
        nop
        jmp    bucle
salir_bucle:
        nop
        break
```

Problema 24.- Solución paso a paso

```
; PROBLEMA 8

.INCLUDE "m168DEF.INC"
.DEF     TMP=R16

LDI      TMP,$FF   ; Inicialmente estan todos el HI
OUT      DDRB,TMP  ; Configura el puerto B completo como salida
EOR      TMP,TMP   ; Limpia R16 (R16=0)
OUT      DDRC,TMP  ; Configura el puerto C como entrada
LDS      TMP,$202  ; R16=SRAM[$202]
OUT      PINB,R1   ; R1 inicialmente está a cero, todos el puerto B se pone a 0
SBI      PINB,6    ; El pin PB6 se pone a 1
CBI      PINB,2    ; El pin PB2 se pone a 0, aunque ya lo estaba
IN       R0,PINC   ; En R0 se leen los 8 bits del puerto C
```

Problema 25.- Hay que ver cuantos ciclos tarda en ejecutarse cada instrucción en la tercera columna de la tabla de instrucciones. En total suman 11ciclos de reloj, incluyendo RETI. Además, hay que sumar el número de ciclos se tarda en atenderse una interrupción que, según la hoja de características, es de 4 ciclos. Tras esto el PC apunta al vector de interrupción, es decir, ejecuta la instrucción existente en esa dirección de memoria. Habitualmente, la dirección del vector de interrupción tendrá un RJMP SUBRUTINA, la cual tarda otros 3 ciclos más.

Nota: Si el procesador estaba en modo de bajo consumo (sleep mode), entonces a todo esto hay que añadir 4 ciclos más que tarda en despertar.

```
.INCLUDE "m168DEF.INC"

RECIBE_BYTE:    PUSH  R0      ; 1 Ciclo
                LDS   R0,UDR0 ; 2 Ciclos
                ST    Y+,R0   ; 2 Ciclos
                POP   R0      ; 2 Ciclos
                RETI          ; 2 ciclos
```

Finalmente el resultado es: Ciclos= 4 (salto al vector) +3 (salto a la subrutina) + 11 ciclos = 18ciclos. Al funcionar a 1Mhz son 18µseg de tiempo de ejecución.

El dispositivo debe mandar 1 bytes cada 18µseg, durante el envío de 1000 seguidos, el procesador lo único que hace es atender esta interrupción, si hubiera otra que atender se perderían datos.

Problema 29.- Se utilizará la variable intermedia suma_total, de tamaño palabra larga, para almacenar la suma parcial de los veinte números y su posterior división. Una segunda rutina calculará el valor absoluto de un elemento de tamaño palabra, denominado numero.

```
Tabla:          DC.W ....

Suma_total:     DC.L  0
Numero:         DS.W  1
Contador:       DC.B  0       ; Iteraciones del bucle.
Res:            DS.W  1

Rutina:         MOVEA.L  #Tabla,A0    ;Inicializa el registro A0 con el inicio de la tabla.
Bucle:          MOVE.W   (A0)+,numero ;Copia el elemento tabla[i] en numero
                BSR      rutina_abs
                CLR.L    D0
                MOVE.W   Numero,D0
                ADD.L    D0,Suma_total ;Las sumas parciales se deben hacer en tamaño L.
                ADDI.B   #1,Contador
                CMPI.B   #20, Contador
                BNE      Bucle
                MOVE.W   Numero,D0
                DIVU     #20,D0
                MOVE.W   D0,Res
                RTS

rutina_abs:     CMPI.W   #0,numero  ;Compara el número con 0
         BGE    fin_rut_abs
         NEG.W  numero
fin_rut_abs:RTS
```

Problema 31.- Para el acceso a los datos de la tabla utilizamos los punteros X, Y y para la tabla de contadores que situamos a continuación, Z.

```
.INCLUDE "m168DEF.INC"

.EQU     N=10

.DEF     temporal=R16
.DEF     i=R17
.DEF     numero=R18
.DEF     mayor=R19
.DEF     posicion=R0
.DEF     resultado=R1

.DSEG
Tabla:          .BYTE N       ;Tabla de N números con signo de tamaño palabra.
Contadores:     .BYTE  N      ;Tabla de N contadores (uno por cada elemento)

.EQU     tope=Tabla+N

.CSEG

Subrutina:
         ldi    temporal,Contadores
         mov    zl,temporal
         clr    zh                    ;registro Z->Contadores
         clr    i
Puesta1contadores:
         ldi    temporal,1
         st     Z+,temporal
         inc    i
         cpi    i,N
         brne   Puesta1contadores


         ldi    temporal,Tabla
         mov    xl,temporal
         clr    xh                    ;El registro X apunta al inicio de Tabla
         ldi    temporal,Contadores
         mov    zl,temporal
         clr    zh                    ;registro Z-> Contadores

Buclei:  ld     numero,X+
```

```
         mov    yl,xl
         clr    yh                  ;Registro Y = X

Buclej:  ld     temporal,Y+
         cp     temporal,numero
         brne   sigue
         ld     temporal,Z
         inc    temporal
         st     Z,temporal          ;Incrementa el contador asociado
sigue:   cpi    yl,tope
         brne   Buclej
         adiw   zl,1
         cpi    xl,tope-1
         brne   Buclei

BuscaMayor:
         ldi    temporal,Contadores
         mov    zl,temporal
         clr    zh                  ;registro Z-> Contadores
         ldi    mayor,1
         ldi    temporal,0xff
         clr    resultado           ;Puesta a 0 del registro R1
buclemayor:
         ld     numero,Z+
         inc    temporal
         cp     mayor,numero
         brcc   sigue2
         mov    mayor,numero
         mov    posicion,temporal
sigue2:  cpi    zl,Contadores+N
         brne   buclemayor
         ldi    temporal,1
         cp     mayor,temporal
         brne   final
         mov    resultado,temporal
final:   ret
```

Problema 32.- Los diodos cuyos cátodos están a tierra se iluminan cuando hay un 1 en el pin del puerto y se apagan cuando existe un 0.

```
.include “2313def.inc”
.def temp=r16
.org 0

               ldi temp,$FF
               out ddrb, temp       ;Configura el puerto b como salida. No es
                                    ;necesario configura el puerto PD4 como
                                    ;entrada porque por defecto ya lo está.
               Ldi temp,6
               out tccr0,temp       ; Configura el Timer0 en modo contador y en
                                    ; flancos de bajada
               clr temp
               out tcnt0,temp       ; Puesta a cero del contador
Bucle:   in temp,tcnt0
               out portb,temp
               rjmp bucle
```

Error: No se encuentra la fuente de referencia

```
confi_es:
         ldi    temporal,$1f
         out    ddrb,temporal  ;Configura PB0..PB4 como salidas
         clr    temporal
         out    ddrd,temporal  ;Configura PD como entradas
         ret
```

Error: No se encuentra la fuente de referencia

```
confi_uart:
         ldi    temporal,$1d
         out    ucr,temporal  ;Configura rx y tx, transmisión del 9 bit que es un 1
         ldi    temporal,12
         out    ubrr,temporal ;Configura 4800bps a fclk de 1Mhz
         ret
```

Error: No se encuentra la fuente de referencia

```
confi_timer:
         ldi    temporal,5
         out    tccr0,temporal       ;Configura prescaler a 1024
         ldi    temporal,2
         out    timsk,temporal       ;Habilita interrupción del timer0
         ldi    temporal,256-97
         out    tcnt0,temporal       ;Genera desbordamiento cada 100ms aproximadamente
         sei                         ;Habilita la generación de interrupciones.
         ret

int_timer0:
         in     temporal,sreg
         push   temporal
         ldi    temporal,256-97
         out    tcnt0,temporal
         inc    aux
         cpi    aux,50
         brne   fin_int
         clr    aux
         ldi    CINCOSEG,1
fin_int: pop    temporal
         out    sreg,temporal
         reti
```

Error: No se encuentra la fuente de referencia

```
include "2313def.inc"
def  temporal=r16
def  aux=r17
def  CINCOSEG=R18
def  valor=r19

         .ORG   0
         rjmp   reset
         .org   6
         rjmp   int_timer0

         (.... Aquí vienen las rutinas que se han desarrollado en los apartados anteriores)

transmite:
         sbis   USR,UDRE
         rjmp   transmite            ;Esperamos a que el UDR se quede vacío.
         out    UDR,valor
         ret

reset:   ldi    temporal, $df
         out    spl,temporal
         clr    CINCOSEG
         rcall  confi_uart
         rcall  confi_es
         rcall  confi_timer

bucle:   sbis   USR,RXC              ;Espera a que se reciba un dato procedente del PC
         rjmp   bucle
         in     temporal,UDR         ;Leemos el dato
         cpi    temporal,0           ;y comprobamos que es 0
         brne   bucle
                                     ;Inicio de la producción
         sbi    PORTB,0              ;Arrancamos la máquina M0
         ldi    valor,0
         rcall  transmite
esp_m0:  sbis   PIND,2               ;Esperamos a que se active el sensor S0
         rjmp   esp_m0
```

```
         cbi    PORTB,0
         sbi    portb,1              ;Arrancamos la máquina M1
         ldi    valor,1
         rcall  transmite
esp_m1:  sbis   PIND,3               ;Esperamos a que se active el sensor S1
         rjmp   esp_m1
         cbi    PORTB,1
         sbi    portb,2              ;Arrancamos la máquina M2
         ldi    valor,2
         rcall  transmite
esp_m2:  sbis   PIND,4               ;Esperamos a que se active el sensor S2
         rjmp   esp_m2
         cbi    PORTB,2
         sbi    portb,3              ;Arrancamos la máquina M3
         ldi    valor,3
         rcall  transmite
esp_m3:  sbis   PIND,5               ;Esperamos a que se active el sensor S1
         rjmp   esp_m3
         cbi    PORTB,3
         sbi    portb,4              ;Arrancamos la máquina M4
         ldi    valor,4
         rcall  transmite
         clr    CINCOSEG
esp_m4:  cpi    CINCOSEG,1
         brne   esp_m4
         cbi    PORTB,4
         rjmp   bucle
```

Problema 35.-

```
.include "2313def.inc"

.def     temp=r16
.def     maskh=r17
.def     maskl=r18
.def     cont=r19

main:    ldi    temp,0x2
         out    ddrb,temp     ;pone el pin PB2 como salida.
         cbi    portb,1       ;pone a 0 la salida de reloj clk
bucle:   ldi    maskh,$80
         clr    maskl
         ldi    cont,16
_1:      sbis   pinb,2
         rjmp   _1            ;espera a que se produzca un pulso en L
_2:      sbic   pinb,2
         rjmp   _2            ;espera a que L       se ponga a 0
buclein:sbi     portb,1       ;pone a 1 la señal de reloj
         sbis   pinb,0        ;lee el bit de entrada si es 1 activamos bit en ri
         rjmp   salto         ;sino saltamos a desplazar el contenido de mask
         or     r0,maskh      ;hacemos or de r0,r1 con maskh,maskl
         or     r1,maskl
salto:   lsr    maskh         ;desplazamos a la derecha el contenido de mask
         ror    maskl
         dec    cont
         cbi    portb,1       ;puesta a 0 de clk
         cpi    cont,0
         brne   buclein
         rjmp   bucle
```

Problema 36.-

```
.include "2313def.inc"

.def     temp= r16   ;Uso temporal
.def     min = r17   ;Contabiliza el número de minutos transcurridos

         .org   0
```

```
         rjmp   main

         .org   6
         rjmp   int_timer0

main:    ldi    temp,$df
         out    spl,temp      ;inicializa el puntero de pila

         ldi    temp,1
         out    ddrb,temp     ;configura PB0 como salida (aquí está el motor)
                              ;y PB1 como entrada (aquí está la LDR)

         ldi    temp,$20
         out    mcucr,temp    ;Habilita el modo dormido y, dentro de éste, el Idle Mode
                              ;que permite el funcionamiento de los temporizadores
         ldi    temp,$4
         out    tccr0,temp    ;Escoge un prescaler de 256 para el timer0
         ldi    temp,256-234;
         out    tcnt0,temp    ;Introduce 234 cuentas en el timer 0 antes de que
                              ;éste genere una interrupción. Esto, unido a que la
                              ;frecuencia de reloj es de 1Khz, provoca que se genere un
                              ;periodo de 1ms * 234 *256
                              ; = 59,9 segundos o, prácticamente, 1minuto.
         ldi    temp,2
         out    tifr,temp     ;pone a 0 el bit TOV0
         out    timsk,temp    ;Habilita interrupciones del timer0
         sei                  ;Habilita interrupciones

         ldi    min,9         ;inicialización de la variable minutos

bucle:   sleep                ;Paso a modo bajo consumo.
         rjmp   bucle


int_timer0:
         inc    min           ;La variable minuto se incrementa de 0 a 9.
         cpi    min,10
         brne   sigue
         clr    min
sigue:   sbi    pinb,1        ;Comprueba si es de día o de noche
         rcall  rutina_motor
         reti

rutina_motor:
         cpi    min,1
         brc    activa_motor
apaga_motor:
         cbi    portb,0
         ret
activa_motor:
         sbi    portb,0
         ret
```

Problema 37.- (a) Diseñe una rutina que permita configurar los puertos y el timer0 para que que pueda generar desbordamientos cada 40us. No use interrupciones para el temporizador.

```
conf_es:
         cbi    ddrd,0        ;PORTD[0] entrada
         sbi    ddrb,3
         sbi    ddrb,4        ;PORTB[4:3] salida
         ldi    temp,1
         out    tccr0,temp    ;TCCR0 = 1, prescaler =1
         ldi    temp,256-80
         out    tcnt0,temp    ;n=80-> PRESCALER X N X TCK = 40us
         ldi    temp,2
         out    tifr,temp     ;Puesta a 0 del flag de overflow
         ret
```

Problema 37.- (b) Diseñe una rutina que permita generar una ciclo de la señal de reloj. Esto es, usando el timer0, la rutina pondrá el pin de CLOCK a 1 durante 40μs y después a 0 durante el mismo tiempo.

```
clock:   sbi    PORTD,4       ;Puesta a 1 de CLOCK
         ldi    temp,256-80
         out    tcnt0,temp    ;Recargamos el contador
         ldi    temp,2
         out    tifr,temp     ;Puesta a 0 del flag
esp1:   in     temp,tifr
         sbrs   temp,1
         rjmp   esp1
         cbi    PORTD,4       ;Puesta a 0 de CLOCK
         ldi    temp,256-80
         out    tcnt0,temp    ;Recargamos el contador
         ldi    temp,2
         out    tifr,temp     ;Puesta a 0 del flag
esp2:    in     temp,tifr
         sbrs   temp,1
         rjmp   esp2
         ret
```

Problema 37.- (c) Diseñe una rutina que permita calcular el bit de paridad asociado a un byte almacenado en la dirección de memoria DATO ($60 de la SRAM).

```
oddparity:
         lds    temp,DATO
         clr    cont
         clr    contador
buoddp:
         lsr    temp
         brcc   sigue
         inc    cont
sigue:
         inc    contador
         cpi    contador,8
         brne   buoddp
         inc    cont
         andi   cont,1
         sts    parity,cont
         ret

         .DSEG
         .ORG $60
DATO:    .BYTE 1
parity:  .byte 1
```

Problema 37.- (d) Diseñe una rutina que permita transmitir la trama PS2 a partir de las rutinas anteriores y el dato almacenado en la dirección DATO.

```
tx_ps2:
         rcall  oddparity
         lds    temp,DATO
         cbi    PORTD,3       ;bit de start
         rcall  clock
         clr    contador
bu_txps2:
         lsr    temp
         brcs   s1txps2
         cbi    PORTD,3       ;Dato a 0
         rjmp   clocktxps2
s1txps2:
         sbi    PORTD,3       ;Dato a 1
clocktxps2:
         rcall  clock
         inc    contador
         cpi    contador,8
         brne   bu_txps2
```

```
         lds    temp,parity
         cpi    temp,0
         brne   s2txps2
         cbi    PORTD,3       ; Paridad a 0
         rcall  clock
         rjmp   stoptxps2
s2txps2:
         sbi    PORTD,3       ; Paridad a 1
         rcall  clock
stoptxps2:
         sbi    PORTD,3       ;Bit de stop
         rcall  clock
         ret
```

Problema 37.- (e) Diseñe el programa que permita conectar las rutinas anteriores con los procesos a seguir con la pulsación y liberación de la tecla ENTER externa.

```
.include "2313def.inc"

.def temp=r16;
.def contador=r17;
.def cont=r18;

         .org 0

main:
          ldi   temp,$df
          out   spl,temp
          rcall conf_es

loop:
         sbic   PIND,0
         rjmp   loop
         ldi    temp,0x5a
         sts    DATO,temp
         rcall  tx_ps2
loop1:    sbis  PIND,0
         rjmp   loop1
         ldi    temp,0xf0
         sts    DATO,temp
         rcall  tx_ps2
         ldi    temp,0x5a
         sts    DATO,temp
         rcall  tx_ps2
         rjmp   loop
```